Semantic Data Integration in Manufacturing Design with a Case Study of Structural Analysis

A thesis presented to

the faculty of

the Russ College of Engineering and Technology of Ohio University

in partial fulfillment

of the requirements for the degree

Master of Science

Arkopaul Sarkar

August 2014

This thesis titled

Semantic Data Integration in Manufacturing Design with a Case Study of Structural Analysis

by

ARKOPAUL SARKAR

has been approved for

the Department of Industrial and Systems Engineering

and the Russ College of Engineering and Technology by

David Koonce

Associate Professor of Industrial and Systems Engineering

Dennis Irwin

Dean, Russ College of Engineering and Technology

Abstract

SARKAR, ARKOPAUL, M.S., August 2014, Industrial and Systems Engineering

Semantic Data Integration in Manufacturing Design with a Case Study of Structural Analysis

Director ofThesis: David Koonce

Product designers produce the design of the product to satisfy the product specifications by applying his own design intent. Due to the lack of semantic capability of the modern Computer aided Design (CAD) applications and standard formats (STEP) to store design data, the design intent of the designer is lost and only geometrical information remain in the design data (non-semantics aware design). Unavailability of the design purpose or the significance of the design in the design data makes It hard to integrate manufacturing design data with other applications of Computer Integrated Manufacturing (CIM). For this reason, it is important to interpret functional properties of design data. In this thesis, a structural analysis process is developed, which uses a novel algorithm from computational geometry to extract Degrees of Freedoms of each part in an assembly. These information is used to recognize translational properties of the parts. Along with it, functional design of a conceptual system, called SIDOS, is presented, which compares the expected behaviors, derived from the functional behaviors, which are in turn extracted from a non-semantics aware design, with the expected behaviors of a semantics-aware template design from the same product family to identify similar components between them and annotates the components of the non-semantics aware design with the matching semantics from the template design.

Dedication

"To be realistically engaged with science means appreciating doubt and uncertainty as the necessary precursor to knowledge and illumination. We must learn to traffic in the unknown, be comfortable with uncertainty, and take pleasure in mystery. While searching for knowledge we must abide by ignorance for an indefinite period. Above all, we need to know how to ask a good question."

-- Stuart Firestein

Table of Contents

List of Tables

List of Figures

Chapter I Introduction

Since the first industrial revolution, men have been invested in manufacturing machines. The first industrial revolution saw the steam power and efficient waterpower, while the second industrial revolution witnessed the introduction of electrical energy and the modern production line. In the twenty-first century, the introduction of computing power in manufacturing paved the way of automation in production lines. In the last forty years, due to the advent of primitive process controllers to advanced super computers and currently cloud computing, we have seen a rapid incline in technological innovations. Robotics and advanced Computer Numerical Control (CNC) technologies are increasingly automating the manufacturing industries. 3D printing technology and distributed computing has given birth to rapid prototyping, which is being widely accepted for its flexibility in designing new products and its effectiveness in reducing overall design cost. Additionally, Computer Aided Design technologies are getting increasingly sophisticated and have already become the principal means of designing and prototyping new products.

Organizations in product design and modeling require constant translations of product design data between formats during all stages of product life cycle. Integration of data gives flexibility and agility by increasing efficacy and productivity of modern decentralized systems [1]. Almost in every process of manufacturing, design data is needed to calculate automated process steps and other information; such as, Manufacturing process planning including machine and tool selection, tool path calculation, sequencing and scheduling [2]. Advanced simulation engines use process

plans data to calculate production costs, raw materials, procurement planning, storage and warehousing [3]. Thus, product design data can be called the gateway for all other manufacturing process information that needs to be interpreted by every other manufacturing process automation software such as computer-aided manufacturing (CAM), computer-aided process planning (CAPP) and enterprise resource planning (ERP). All of these applications are a part of the Computer-integrated manufacturing (CIM) system. In next section, a brief discussion on design data integration in CIM is presented.

A. Data Integration In Different Phases Of Computer-Integrated Manufacturing

Manufacturing of a product starts with a design idea. Computer-Aided Design (CAD) technologies not only help designers capture the product idea into the product design, but also use it in editing, analyzing, documenting and optimizing the design. CAD application packages have evolved to be more resourceful since they replace manual drafting. Sophisticated analysis and optimization tools like feature-based modeling, pattern analysis, quality control, tool path detection, collision detection, and, finite element analysis help designers to be more productive and accurate. Both persistent and transient information produced by CAD applications are usually captured in native data formats specific to the CAD application used. However, many modern CAD applications support neutral formats such as Initial Graphics Exchange Specification (IGES), Standard for Exchange of Product model data (STEP), and Standard Tessellation

Language (STL). These neutral formats not only help CAD applications to interpret a design developed in a different CAD system, but they are used to feed design data to other manufacturing process applications.

The use of rapid prototyping has escalated over the last few years through the advancement of 3D printing technology and agile manufacturing practice [4]. Rapid prototyping is an integral part of CIM, where product design is 3D-printed and tested by different automated analysis systems and the results are fed back to CAD for quick modification. Rapid prototyping requires seamless transfer of product design data to 3D printers. Regular 3D printers available in the market use design data in STL format.

The most important part of computer-integrated manufacturing is the use of computers in the manufacturing processes. Automation of machining processes is accomplished through CNC machines, NC programming, and smarter CAM applications. Mechatronic systems manage different stages of manufacturing from semi-finished parts to finished parts as well as automation of product assembly. Product design data is not only used by intelligent CNC applications to compute cutting paths, parameters and generate other machine control information, but are also used in choosing correct machines, manufacturing cells or tools to perform required operations. Most of the small scale CAD and CAM integration is achieved through neutral formats such as STEP, IGES, and STL. Sometimes, direct communication links using Object Linking and Embedding (OLE), an object oriented communication system are also used between CAD and CAM application [5]. Advanced CAD and CAM systems sometimes publish

Application Programming Interfaces (API), which are a set of functions with pre-defined data structures to send or receive data between applications [6].

CAD and CAM systems are two distinct technologies with different application domains. Computer-aided Process Plan (CAPP) makes the bridge between CAD and CAM functionally. The function of CAPP is to analyze product design data and prepares manufacturing process plans. Manufacturing process plan governs the stages of a product manufacturing from raw material to finished product, followed by assembly and packaging. Automated process planning systems can decide optimized process routes, task schedules, instruction sets for machining operations, quality control measurements, shop-floor management, and safety strategies [7]. In this way CAPP systems use both design data produced by CAD and machining data produced by CAM.

Modern day CAPP systems are tightly integrated with enterprise resource portals that track, maintain and archive various process data. They are further used in automation of various non-manufacturing activities such as, product lifecycle, inventory management, procurement, sales and distribution, supply and demand management, and transport. In this way modern manufacturing runs on a number of automated manufacturing applications that share manufacturing process data amongst themselves in various formats. Product design data are interpreted in different ways by this manufacturing automation software. Complete automation and integration of various CIM application demand intelligent interpretation of product design data. This can only be achieved if design data is capable of storing the contextual significance or the functional purpose of the design.

B.Problem Statement

In spite of the availability of numerous of neutral formats and data translation methods, functional properties of the product cannot be interpreted easily from product design data. The reason of such problem is the unavailability of Semantics in the data formats. Semantics can be defined as a token or sign, which can accurately describe the purpose and contextual significance of data (“The word Semantic is derived from the Ancient Greek word σημαντικός (semantikos), "related to meaning, significant", from σημαίνω (semaino), "to signify, to indicate", which is from σῆμα (sema), "sign, mark, token".” [8]). Current methodologies of storing design data comprise of geometry, topology, assembly and materials of 3D product designs. The information derived from these data is not enough to identify the contextual significance of the design.

The structures and standards used in STEP and related ISO10303 models, are restricted to individual domains. Lack of explicit semantics in the entity definitions found in these domains makes automated mapping of structures from two different domains extremely challenging. The data stored by those structures are not sharable across different domains [9].

Data modelers engaged in manual data integration, often struggle due to lack of explicit contextual meaning in data models used in different domains. Experts from different domains define the same concepts in disparate terms and vice versa. In his book, Robert M. Thacker describes this situation as “islands of competency” where experts in

different fields of manufacturing perceive different processes of manufacturing according to their own individual vision [10].

Many different algorithms including heuristics, meta-heuristics, and optimization try to interpret design data in terms of other systems in CIM. For example, feature recognition systems extract machining features from design data to feed into CAM systems. However, implementation of these algorithms is often complex in nature and have often proved not to be cost-effective in practical scenario [11]. It is apparent that the presence of explicit semantics in design data can provide meaning to information and can also be used directly to interpret design information, obviating the need for more complex algorithms.

Semantics have been used in manufacturing data integration in some recent research. These research efforts are focused on building a standardized global data model as a reference since it represents different manufacturing data models. For example, an object oriented data model called Integrated Manufacturing Planning Model (IMPM) is developed for a rule based manufacturing planning system called IMPlanner [12]. In other research, upper level manufacturing ontologies and models are used to interpret design data stored in a semantically enhanced STEP standard called OntoSTEP [13]. However, most of the CAD systems namely AutoCAD by Autodesk, CATIA by Dassault Systèmes, SolidWorks by Dassault Systèmes SolidWorks Corp., and NX by Siemens, that are prevalent in the market, do not have capability to semantically tag design data. Some of the recent advancements of these applications offer an optional feature to tag data with semantics from global semantics repository [14]. However these features are

rarely used in the commercial manufacturing designs. Therefore it can be concluded that almost every design data produced in the current world is unaware of semantics.

C.Proposed Solution

In order to define the semantics of a product design data, the functional properties of the product design needs to be interpreted. Product design data is comprised of single or multiple parts, assembled in a certain way to achieve the function of the product. Many different research efforts have developed various methods to interpret the functional properties of the product design. In this research, a novel algorithm is devised to interpret translational properties of the parts of a product assembly.

In this algorithm, joints formed by two or more parts of a product assembly are detected by cuboids generated from Boolean intersection of the bounding boxes. They are calculated from the 3D shapes of the parts. In the next step, parts forming the joint are selected pairwise and cross-sectioned by 3D axial planes. Each cross-section generates two polygons that are analyzed by a sweep line in order to determine the clearance of each polygon with respect to other. These clearances calculated at different axial planes, are aggregated to define translational degrees of freedom (DOFs) of one part with respect to other. Once DOFs of each part in the assembly is determined, translational properties of the whole assembly can be analyzed by plotting DOFs of every part in a truth table. In Chapter III line algorithm and joint detection strategy are formally presented along with a detailed discussion on how these algorithms can be used to recognize translational

properties of any design. In Chapter IV algorithms are applied on two demo assembly designs and the results are discussed.

Functional properties extracted from the design data can be used to define the semantics of the product. In this thesis, architecture and design of a system called SIDOS is presented. The function of SIDOS is to annotate a non-semantics aware product design with correct semantics. It is done by translating the functional properties extracted the non-semantics aware design with the help of a product specification ontology.

The proposed system is functional on two levels. At one end of the proposed system, a schema analyzer is responsible for extracting structural behaviors of different components of the design. And on the other end, an ontology equipped with domain knowledge of that particular product will supply contextual information based on the structural behaviors extracted from the design. An alignment algorithm compares the contextual information, supplied by the ontology with the semantics found in a semantically tagged template model, from the same product family. This way the alignment algorithm can map non-semantics components of a design to components of a semantically mapped prototype, based on their contextual similarity. Upon successful matching, the non-semantics component can be annotated with the tag of the matched component of the template design.

As the proposed system tries to integrate design data models through the semantic knowledge of the components of the models, the system is named as “Semantic Integration of Design models though Ontology (System)”, abbreviated SIDOS. The

principle and conceptual architecture of SIDOS system is presented in Chapter V. The development of SIDOS system is kept out of the scope of this thesis.

Chapter II Literature Review

In first place, this chapter talks about current data formats and standards to store and communicate design data. The chapter also discusses the reasons behind why these data formats and standards do not provide enough information to interpret the contextual significance of design. Next in succession, reviews of various feature recognition strategies are presented. These feature recognition strategies are alternative ways to extract purpose and significance of design data by identifying machining features in a design. Recent researches on interpretation of functional properties of the design data from the pure geometrical data are also discussed. These researches are collectively called as structural analysis.

A. Integration Through Neutral Data Formats

In the latter half of twentieth century, transferring complex 3D design data across various manufacturing departments became a major challenge in manufacturing industries. In 1999, National Institute of Standards (NIST) estimated that data incompatibility causes the loss of 90 billion dollars each year to the automotive industries alone[15]. Previous standards in data format of manufacturing designs led the way to more widely accepted International Standard Organization (ISO) standards. Some well-known standards among them are SET from France, VDAFS from Germany, Initial Graphics Exchange Specification (IGES), used in USA, the PDES from the PDES

consortium, DXF from Autodesk, Interface Description Language (IDL), and CDIF from Electronic Industries Association.

STEP, a series of protocols and standards presented by ISO 10303, defines neutral formats for storing data that are generated by different manufacturing software. A schema modeling language called EXPRESS governs the STEP syntax and paradigm. STEP-file formats (.stp), STEP-XML formats and a package of application protocol interface called SDAI. SDAI helps access and communication of the STEP formatted design data by computer middleware. Though ISO initially planned to design a monolithic format to standardize the entire manufacturing data, the task soon appeared to be too big to cover in a single protocol [16]. Thus, several application protocols (APs) were released to cover different applications and domains of manufacturing industries. One fundamental problem of the STEP standard is rooted in the modularization of the standards in overlapping protocols. Although each application protocol contains a definite scope and expectation, different application protocols overlap each other in definition and terms. This is more evident in the domain specific application protocols. It is observed that different definitions in different application protocols define the same kind of products, product structures, and geometry. The introduction of *Application Interpreted Constructs* (AIC) solved this problem primarily in the geometrical domain, however, implementation of an AIC in other application domains is limited [17].

Though STEP standards are constantly evolving, it will take years to fully implement definite standards for every manufacturing application and domain [18]. Evolving STEP standards and application protocols cause major problems in

interoperability of data across industries because of a lack of consistency in development. Application protocols, that are open for extension and modification, often contain localized definitions of manufacturing design and processes, which are not standardized across industries. These entity definitions often lack in capability of capturing function and behavior of intended product model [9].

STEP standards are used in further research to implement different knowledge based, algorithm based, and global schema based integration model. These models extend STEP protocols to store extra information, which are used to control activities across different manufacturing processes.

B. Extraction Of Machining Features Through Feature Recognition

Feature recognition is the first step to identify the context of any part. Several reasons can be mentioned to establish this concept. A stock is transformed in a part when different features are machined on that stock. Therefore features give any part its structural behaviors. Machining information, calculated by the CAM system is based on features and other manufacturing planning information, such as, scheduling, operation, and cost data, that can be calculated from machining information. Therefore, if the features in any part can be analyzed then all other information can be linked seamlessly. In this field, many logic based system such as, syntactic pattern recognition, state transition diagram, automata, graph based approaches, expert systems, and volumetric decomposition methods, have been developed [19].

In spite of widespread use of a form feature library in CAD designs, the end product does not always remember the features used in the model. Automated feature recognition systems solve two problems in this area. First, they implement different algorithms to identify features in the part model; secondly of all, they represent the feature either by topological relationship in B-REP representation or by Boolean operations in CSG modeling.

In spite of various efforts in automatic feature recognition (AFR), created using automata, syntactic analysis and state transition diagram, the oldest and most successful AFR systems are built with help of rule-based systems. Expert Systems, used in research conducted by Babic et al. for CAD and CAPP integration, use a set of production rules that are applied to a set of facts generated from the IGES representation of the part. Facts composed of information on face and edges of a part are exposed to rules, that check base surface, adjacent faces, edge loop and boundary faces to extract features [19].

A series of research on AFR used a graph-based approach. In this approach a graph is created with face and edges of the B-REP model as nodes and arcs joining them with attribute 0 if nodes are in concave adjacency relation or 1 if they are in convex adjacency relation. This graph, called attributed adjacency graph (AAG) can be queried for a similar pattern to match a feature from a feature database [20]. AAG is extended with various other attributes to form Multiple Attribute Adjacency Graph and many other improved parsing algorithms are investigated by number of researchers [19].

A number of researchers concentrated on a reductive approach, called the volumetric decomposition method, to identify features of a model. Some of them

decomposed the convex hull of the part into smaller machine able blocks, called maximal features. These maximal features, acting as intermediate features, are the differences between stock and body. They give the correct definition of features in the part model [21]. In cell-based volumetric decomposition, the maximal features are decomposed into smaller blocks packed together. A number of blocks, which can be machined by a single tool path is aggregated to form a feature [22].

Sommerville et al. proposed a "viewer centered approach", in which light rays produced from a view point casted different projections by getting interfered by different faces of the part. An algorithm combines these projections and able to identify different orthogonal features [23].

C. Recent Research On Structural Analysis

In recent years, various research interests focused on analyzing a complete manufacturing design to extract its functional properties. These research mainly use reconstruction through merging point clouds, triangulation, segmentation, surface reconstruction, sewing, and blending, identification of constraints in an assembly, and kinematic analysis, all of which can be classified as reverse engineering [24]. Shape analysis can extract information, which is not limited to only geometrical interpretation. Shape recognition of conventional objects is a separate discipline from feature recognition. One fundamental difference between the feature recognition and the shape recognition is that, feature recognition depends on analyzing CAD designs, taking into account all

inherent design information and concentrate on identifying individual template features, whereas, shape recognition starts from the point cloud objects, which are analyzed by studying forms and structures without depending on CAD specific data. Shape recognition concentrates on reconstruction and identification of the contextual meaning of its shape.

One significant limitation of product design is that it never captures the information of upright positioning of 3D models. Determining up-right orientation of any 3D object is necessary for further analyzing static positions and kinematic properties of parts in assembly. Fu et al. analyzed the functional geometrical properties of a 3D model to determine the up-right position of the 3D model. Functional geometric properties are chosen because of their unique contribution to the stability of the object on ground. Candidate sets are formed with coplanar points on the convex hull of part, and features influencing maximum static stability, parallelism, symmetry, and visibility. An assessment function built with this candidate set is exposed to training through random forest classifier and an SVM classifier. With the help of assessment functions, this experiment found more than 90% success in predicting the upright position of an independent set of man-made designs with the help of the assessment function [25].

Symmetry of the design is important information in finding the significance and meaning of the design. Symmetry can be found in both manmade and natural objects. Symmetry gives evidence of pattern in the design. Mitra and his associates researched on a novel algorithm, which can detect partial or complete symmetry of 3D design. The method is based on the observation that any shape coincides with its symmetric pair when it has undergone a specific set of Euclidean transformations. In their research they

detected all symmetric subpart pairs in any 3D model, and then performed statistical sample analysis to select few symmetric pairs, which can validate the overall symmetry of the model [26].

Aesthetics is a significant part of the product design. In today's competitive market, satisfying the functionality is not the only deciding factor in the commercial success of product. Therefore, every product design is full of complex features, which may not contribute in the functionality of the model but can help in beautification of the model and reduce material consumption. However these aesthetic features do not contribute much information in the functional analysis but increases the complexity of analyzer algorithms. Mehra et al. investigated a method of abstracting complex man made design in a simplistic model preserving all functional features. In this research, first an envelope is calculated for the parts of the design. This envelope is constructed by a closed manifold surface around the body of the part. Next, a set of edges is selected based on various analyses performed on the manifold. This defines the contour of the 3D Model, producing the abstraction of the model. Although this abstraction process doesn't have capability to extract any functional property from the design, but this process is beneficial in reducing the complexity of the design before exposing it to other structural analysis.

In any assembly, every component has its own position data. Position data is comprised of translational and rotational matrices that define the component's relative position to the base component, or a reference point in primary three-dimensional Euclidean space. However, this transformational information is purely mathematical and cannot be used in contextually describing part relationships. Lee and Andrews proposed a

method to determine two special relationships, called ‘fits’ and ‘against’ which can define orientation of one part with respect to any other part in an assembly. A new structure is introduced, which can hold this two relationships and give the representation of any part in an assembly more meaning than just geometrical information [27].

Assemblies found in most product designs are made of movable parts along with static parts. Motions in parts are important hints in finding the functional significance of any part in the assembly. After identifying the orientation and relative position of parts in the model, analysis can be performed on how the parts interact with each other, how joints are created by interactions of their individual degrees of freedom, ultimately creating a joint co-ordinate system where motion of each part can be defined by vectors [28]. Mitra et al. defines interactions among parts with help of a graph and classifies edges with different interaction property. In the end, they implement forward kinematics to calculate relative speed of one part with respect to another [29].

The research described above can help in developing algorithms in the future to extract more functional properties from a manufacturing design. In this thesis, investigation is conducted on analyzing static positioning information from the assembly. Static positioning of each part in an assembly is determined by the degrees of freedom (DOF). Analysis of DOF is described in Chapter III.

D. Integration Through Schema Translation

The advent of relational databases to store manufacturing data in an entity relationship model raised the issue of semantic incompatibility and data inconsistency between different data schemas. Some early data integration systems concentrated on mutual interoperability, data translation strategy and partial integration. In a seminal research paper, Reddy et al. proposed a four-tier architecture for integrating disparate local data schema into a homogenized global schema [30]. Schema in each level of this architecture is an aggregate of the lower level schemas. Object equivalence classes, property equivalence classes and other related concepts in local schemas are used to perform these aggregations. This proposed architecture resolves the conflicts in naming, scaling, and type in data integration of heterogeneous schema.

This aggregation architecture is used in number of research to integrate local manufacturing definitions into a global model. It served as a reference to obtain contextual meaning of the definition to eliminate semantic incompatibility among local definitions. Koonce proposed a model, called the Unified Data Meta Model (UDMM), which contains shared entities, relationships and constrains among them, and properties, which are commonly shared among different manufacturing processes, tools, and methods [31]. Local process schema share common properties in the UDMM and specialize by having its local properties outside of the UDMM. UDMM acts as a neutral format, and an intelligent interface translates data from local manufacturing model to UDMM format by referring to a knowledge base, which contains every relationship

between the local schema entities and UDMM entities. Integrated Manufacturing Design Environment (IMDE) is an integration environment, which uses the UDMM as a virtual schema to translate different manufacturing process data. This virtual neutral format is the union of the intersection of the data properties extracted from local schema belonging to different functional domains of manufacturing [32].

The UDMM architecture is a novel idea to store all semantically similar entities from different manufacturing processes. It can also store common data among different processes. However, it doesn't provide a mechanism to semantically interpret specialized properties of processes, which are solely owned by individual processes. IMDE system uses intelligent interfaces to translate these local properties but the translation process is non-generic and based on the relationship among different localized entities pre-defined in the translator. Therefore, precision of the translation depends on the quality of the translator and an attached knowledge base.

Chapter III Structural Analysis

Structural analysis of a manufacturing design is defined as the process of extracting static and kinematic properties of a part in an assembly, by analyzing the pure geometrical properties of the part, machining features and interaction with other parts. These static and kinematic properties can be used to recognize the functional properties of a product design. A list of possible static and kinematic properties, which can be extracted from any part through structural analysis, is presented in Table 1. Although, this set of static and kinematic properties can be part of the set of structural behaviors possible to extract from a part, it cannot be claimed that this list is exhaustive.

Much of a manufacturing assembly design is comprised of different parts. The functionality of the assembly is determined by the careful orientation and positioning of the parts in the assembly by the designer. Designer makes a design to meet functional specifications that meets his design intent. However, in data exchange environment the design intent is usually not carried in the design data format, therefore functional design information is lost and only geometry remains in the design data. The degree of freedom (DOF) of a mechanical system can be viewed as the minimum number of coordinates required to specify a configuration of the system. DOFs of any part in an assembly are fundamental properties of the part, which can be used to calculate other functional properties.

In this thesis, a novel algorithm is proposed to calculate the degrees of freedom (DOF) of any part in an assembly. A prototype of structural analyzer capable of calculating the DOFs of a part in an assembly is also developed based on this algorithm.

The theoretical model of this algorithm, technical implementation and testing of the prototype are described in coming sections.

Table 1: Different behaviors to be analyzed by structural analyzer

Category	**Behaviors**	**Sub-Behavior**
Degrees of Freedom	Translation	Moving up and down (heaving)
		Moving left and right (swaying)
		Moving forward and backward (surging)
	Rotation	Tilting forward and backward (pitching)
		Turning left and right (yawing)
		Tilting side to side (rolling)
Interconnection	Static position	Vector position to other shapes or axis or frame of reference
		Bounding another part (Containment)
		Part mating
	Physical relations	Holding
		Resting
		Supporting
		Channel
		Slot
Dependency	Motion transformation	Rotation of one part due to rotation of another
		Translation of one part due to translation of another
		Rotation of one part due to translation of another
		Translation of one part due to Rotation of another

A. Degrees Of Freedom Calculation

In computational geometry, the sweep plane algorithm [33] is a popular method for efficiently searching line intersections, polygon intersections and Voronoi diagram. The sweep plane algorithm can be used to find the relative position of two or more polygons. The cross-section of any part generates a polygon. For any curved surface, the cross-sectioned will generate a curve edged shape. A curve edge can be approximated by series of straight lines, which can convert the shape into a polygon. For example, a circle can be approximated as a cyclic polygon as in Figure 1.

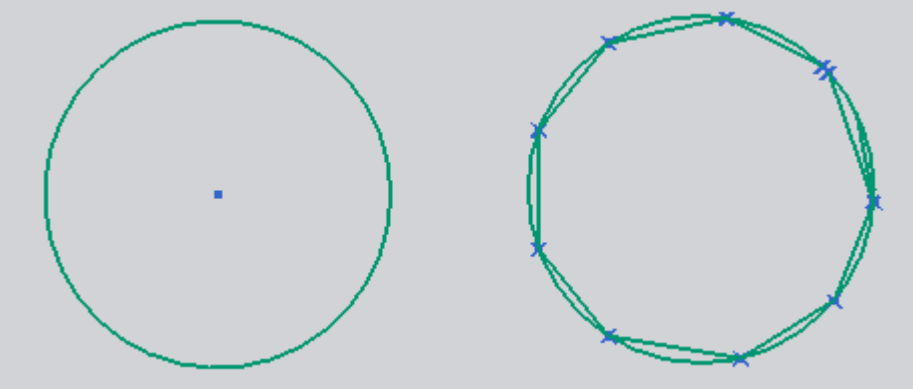


Figure 1: Circle approximated as cyclic polygon

Similarly, any two parts, when sectioned by the same plane, will generate two polygons. The sweep plane algorithm, applied to these two polygons, calculates the clearance of one polygon with respect to other. These clearances are collectively defined as amount of freedoms of part. Any two parts assembly can have cross-sections parallel to XY plane, YZ plane or XZ plane. Each cross-section will be swiped in two directions; e.g.

for XY cross-section sweep line will sweep the plane in X and Y directions. Each sweep line explores amount of freedom that both of these parts have in the opposite direction. Notations for identifying amount of freedom at different planes are detailed in Table 2.

Table 2: Cross-sections taken in different planes and corresponding DOFs

Cross-Section	**Sweep Line**	**Amount of freedom**	**DOF explored**
X-Y	X	$+Ly^{XY}$ and $-Ly^{XY}$	$+Y^{XY}$ and $-Y^{XY}$
	Y	$+Lx^{XY}$ and $-Lx^{XY}$	$+X^{XY}$ and $-X^{XY}$
Y-Z	Y	$+Lz^{YZ}$ and $-Lz^{YZ}$	$+Z^{YZ}$ and $-Z^{YZ}$
	Z	$+Ly^{YZ}$ and $-Ly^{YZ}$	$+Y^{YZ}$ and $-Y^{YZ}$
Z-X	Z	$+Lx^{ZX}$ and $-Lx^{ZX}$	$+X^{ZX}$ and $-X^{ZX}$
	X	$+Lz^{ZX}$ and $-Lz^{ZX}$	$+Z^{ZX}$ and $-Z^{ZX}$

In Figure 2, a two part assembly is used to explain the above mentioned DOF and amount of freedom. This two part assembly is constituted of one block (Part B) with a slot and a slide (Part A) which is placed in the slot.

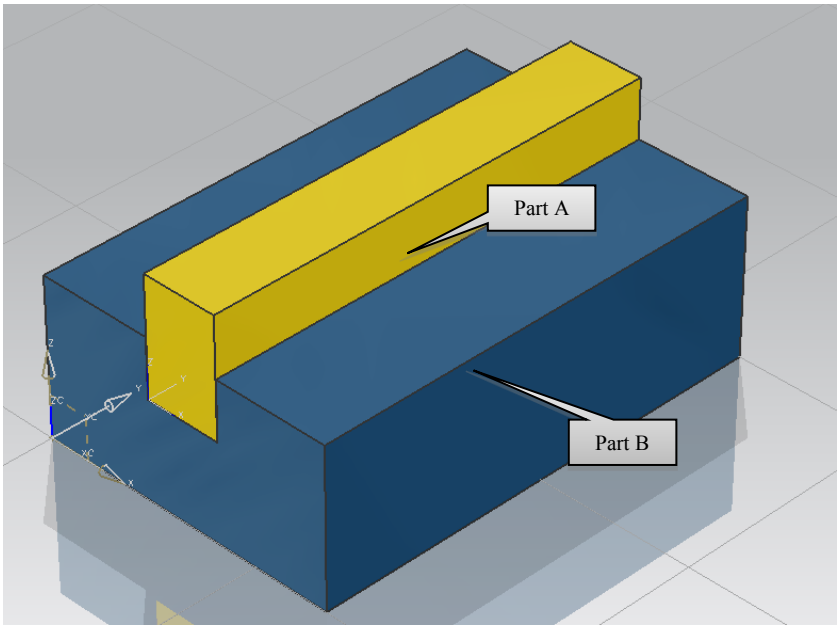


Figure 2: Two Part assembly

When both of the part in the above mentioned assembly is cut by a single cross sectional plane, two polygons are derived. In Figure 3, a cross sectional plane aligned to XY co-ordinate is placed at a certain distance from origin in positive Z direction. Sweep plane algorithm applied on this pair of polygons can derive the following values for the amount of freedom.

$$+Ly^{XY} = \infty \ , -Ly^{XY} = \infty \ , +Lx^{XY} = 0 \ , -Lx^{XY} = 0 \qquad (1)$$

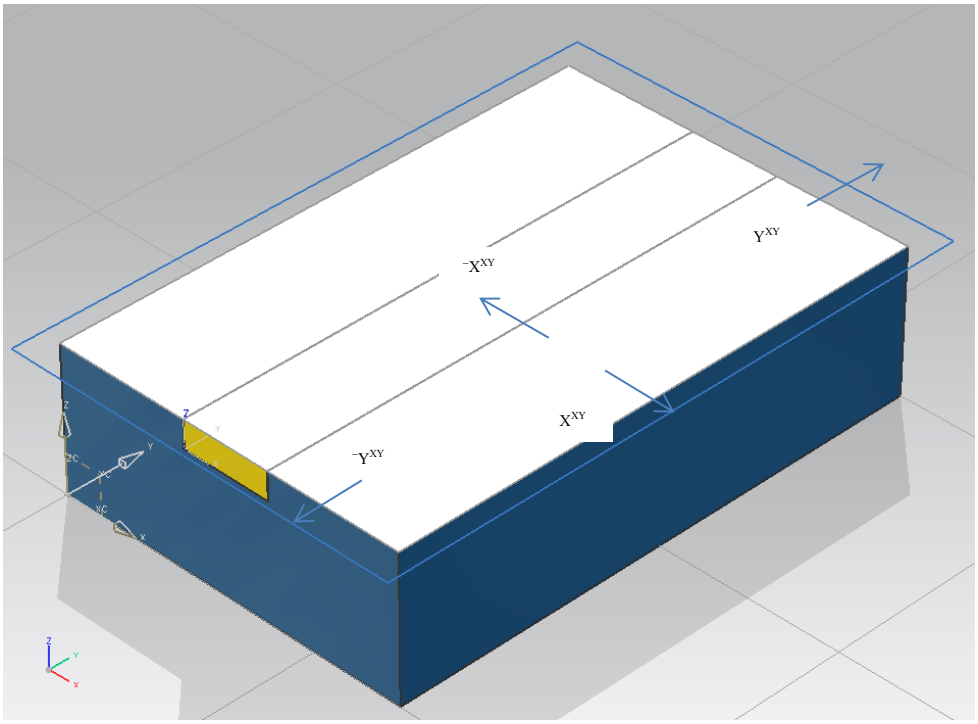


Figure 3: XY plane cross-section and DOF of Part A w.r.t Part B

In Figure 4, a cross sectional plane aligned to YZ co-ordinate, is placed at a certain distance from origin in positive X direction.. Sweep plane algorithm applied on this pair of polygons can derive the following values for the amount of freedom.

$$+Ly^{YZ} = 0 \ , -Ly^{YZ} = 0 \ , +Lz^{YZ} = \infty \ , -Lz^{YZ} = 0 \qquad (2)$$

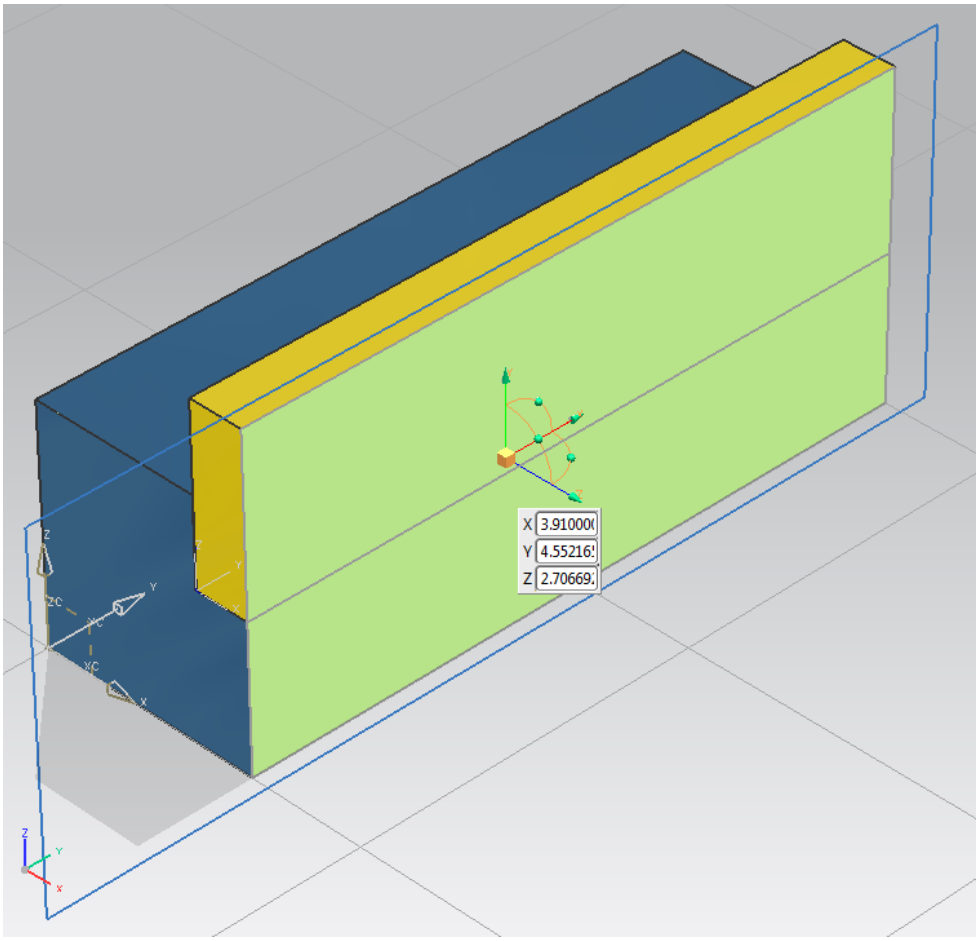


Figure 4: YZ plane cross-section and DOF of Part A w.r.t Part B

In Figure 5, a cross sectional plane aligned to ZX co-ordinate, is placed at a certain distance from origin in positive Y direction.. Sweep plane algorithm applied on this pair of polygons can derive the following values for the amount of freedom.

$$+Lz^{ZX} = \infty \ , -Lz^{ZX} = 0 \ , +Lx^{ZX} = 0 \ , -Lx^{ZX} = 0 \qquad (3)$$

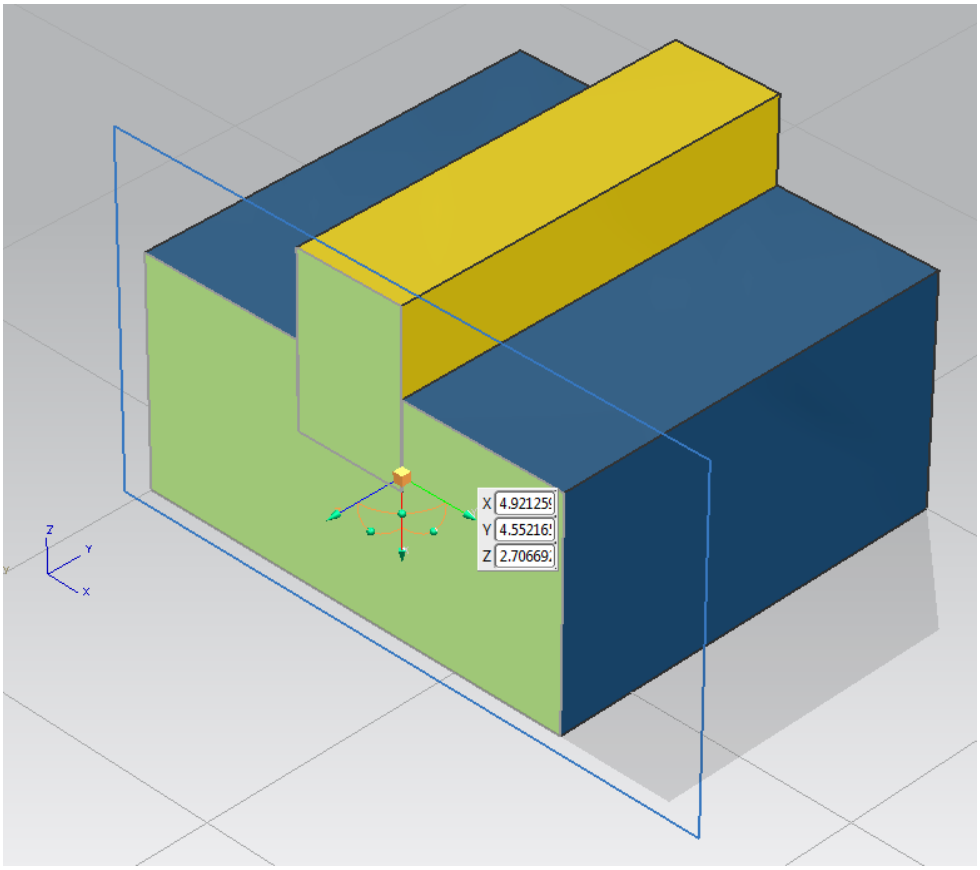


Figure 5: ZX Plane cross-section and DOF of Part A w.r.t part B

Total amount of freedom of each part in the assembly in one direction is the minimum of the amount of freedoms derived from two different cutting planes.

$$ {}^{+}L_x = \min({}^{+}L_x^{XY}, {}^{+}L_x^{ZX}),\ {}^{-}L_x = \min({}^{-}L_x^{XY}, {}^{-}L_x^{ZX}) \quad (4) $$

$$ {}^{+}L_y = \min({}^{+}L_y^{XY}, {}^{+}L_y^{YZ}),\ {}^{-}L_y = \min({}^{-}L_y^{XY}, {}^{-}L_y^{YZ}) \quad (5) $$

$$ {}^{+}L_z = \min({}^{+}L_z^{YZ}, {}^{+}L_z^{ZX}),\ {}^{-}L_z = \min({}^{-}L_z^{YZ}, {}^{-}L_z^{ZX}) \quad (6) $$

Total DOF of part A with respect to part B is the summation of DOF in positive and negative directions.

$$DOF^{AB}_{x} = {}^{+}L_x + {}^{-}L_x \quad (7)$$

$$DOF^{AB}_{y} = {}^{+}L_y + {}^{-}L_y \quad (8)$$

$$DOF^{AB}_{z} = {}^{+}L_z + {}^{-}L_z \quad (9)$$

This total DOFs equals to the total length of the freedom of part A with respect to part B in each of three direction. However, to measure the parts freedom in relation to other part can be sufficiently expressed by Boolean value (0 = free to move, 1 = cannot move). In that case, it is sufficient to say

$$DOF^{AB}_{x} = 0 \text{ if } {}^{+}L_x + {}^{-}L_x > 0 \text{ and } DOF^{AB}_{x} = 0 \text{ if } {}^{+}L_x + {}^{-}L_x = 0 \quad (10)$$

$$DOF^{AB}_{y} = 0 \text{ if } {}^{+}L_y + {}^{-}L_y > 0 \text{ and } DOF^{AB}_{y} = 0 \text{ if } {}^{+}L_y + {}^{-}L_y = 0 \quad (11)$$

$$DOF^{AB}_{z} = 0 \text{ if } {}^{+}L_z + {}^{-}L_z > 0 \text{ and } DOF^{AB}_{z} = 0 \text{ if } {}^{+}L_z + {}^{-}L_z = 0 \quad (12)$$

This Boolean values can be plotted in a truth table shown in Table 3, where DOF each part in relaition to every other part in all three direction can be analyzed. This truth table can be analyzed by manual observation that might result in different functional properties of the parts. Also, this truth table can be used to analyze effect of movement of one part on the corresponding interacting parts. Even the effect of one part on other non-

interacting parts can also be analyzed, by applying transitive property, which is explained next.

Table 3: Truth table of BlockSlide assembly shown in Figure 2

	Block51						**Slide50**					
Direction	X+	X-	Y+	Y-	Z+	Z-	X+	X-	Y+	Y-	Z+	Z-
Block51							1	1	0	0	1	0
Slide50	1	1	0	0	0	1						

Both of the parts in the assembly, shown above, is aligned to the 3D axis. Most of the CAD assembly has a base part, which is aligned to the 3D axis. However, if a CAD design is not aligned to the 3D axis, the assembly would first required to be rotated to align with 3D axis. This alignment process needs visual evaluation of the design. But it is also possible to automate this process by aligning each surface of the parts of the assembly before taking cross-section. This obviously increases the complexity of the execution. In future, optimization methods need to be investigated thoroughly to align the assembly.

The same method of calculating DOF in pair of parts, explained above, can be used to calculate pairwise DOFs of the parts in assembly consisting of more than two parts. For any assembly with more than one part, each pair of interacting parts (Parts forming a joint) is exposed to sweep plane algorithm and the DOFs and corresponding amount of freedom are calculated. To illustrate this concept cross-section of a three parts assembly in XY plane is displayed in Figure 6. The assembly is comprised of Part A, B and C. It can be observed that there are two joints in the assembly, formed by interacting parts A and B, and A and C. These two joints are marked by dashed line. It is to be noted

that these two joints are cross-sectioned individually taking the parts in pairs. The DOFs explained earlier can be plotted in the truth table, which is shown in Table 4. In this table DOFs in +Z and –Z directions are not included because only the cross-section in XY plane is analyzed in this example.

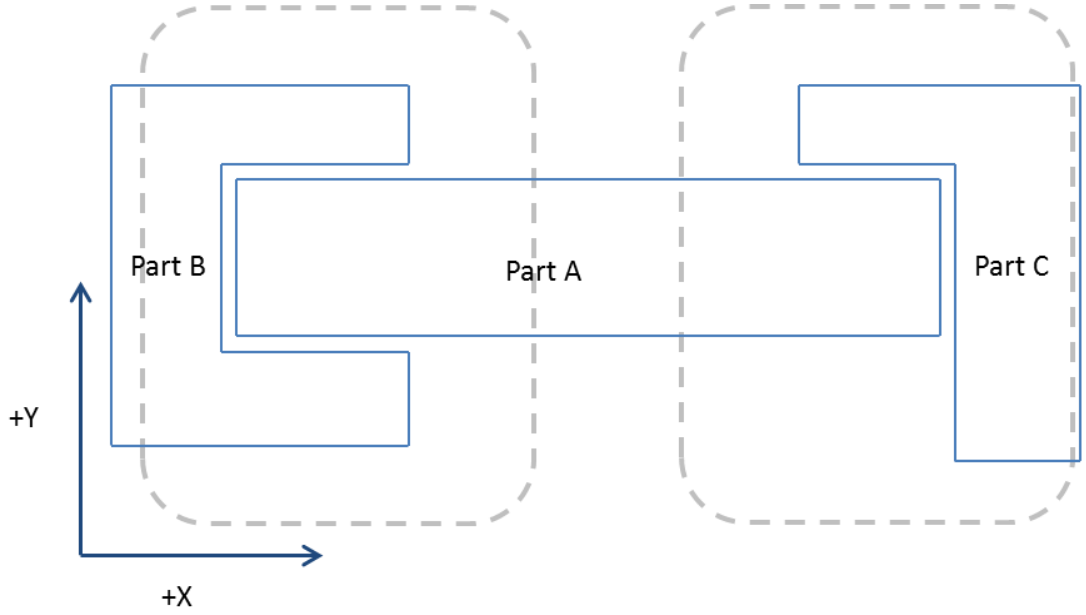


Figure 6: Cross-sections of a three-part assembly generated by XY plane

It is to be noted in Table 4 that DOFs between Part B and C cannot be calculated by sweep plane algorithm because they are non-interacting parts and cannot form joints. However, by applying transitive property. We can decipher some of the motion related functional properties of the part. As we can see from Table 4 that Part A and B cannot move towards each other. Therefore, if Part B is moved in +X direction, it will cause same amount of movement in Part A in +X direction. Similarly, if Part A is moved towards X+

direction it will cause same amount of movement in Part C in +X direction. Therefore we can conclude that if Part B is moved towards +X direction, it will cause same amount of movement in Part C towards +X direction. More example of this transitive property can be found in Chapter IVB.2)

Table 4: Truth table for three parts assembly from Figure 6

	Part A		**Part B**		**Part C**	
	X+	Y+	X+	Y+	X+	Y+
Part A			0	1	1	1
Part B	1	1				
Part C	0	0				

B. Implementation of Sweep Plane Algorithm

The sweep plane algorithm for detecting the degrees of freedom for a cross section is implemented in Java. This algorithm can be applied to a pair of polygons, derived by cross-sectioning two interacting parts. One iteration of this algorithm can detect the amount of freedom of one polygon (Pa) with respect to another polygon (Pub) in one direction. First, every corner points of P1 are stored in a priority queue (Q). This queue stores every point as event for the sweep line where it will stop while traversing the polygon. Events can be of three types: segment start point; segment ending point, and intersection point. Events in the queue is sorted by *compareTo* method comparing them by x first and then y .Two events at same x=x' are sorted lexicographically (x : y).

1) Algorithm to find DOF of Part A w.r.t Part B in XY plane in X direction(S)

Input. Two polygons Pa(part A cross section) and Pb(Part B cross section) (convex or concave)

Output. DOFs in +X , -X direction

1. *Initialize an empty event queue Q. Next, insert the vertices of Pa into Q; Q is sorted by Y-intercept.*
2. *Initialize sweep line at Y=initial sweep line position(min Y intercept of all events in Q)*
3. *While Q is not empty*
    i. *do Determine next event point p in Q and move sweep line to p*
    ii. *delete the event point p from Q*
    iii. *HandleEventPoint(p) (Function with the following operations)*
        a) *Find the intersection points of sweep line and any segment of Pb in +X direction from p.*
        b) *Store the minimum of all distances from p to all intersection points as* $^{+}L_x(p)$
        c) *Find the intersection points of sweep line and any segment of Pb in -X direction from p.*
        d) *Store the minimum of all distances from p to all intersection points as* $^{-}L_x(p)$

4. *Store* ${}^{+}L_x^{a} = min({}^{+}L_x(p1), {}^{+}L_x(p2), {}^{+}L_x(p3), \ldots, {}^{+}L_x(pN))$ *where N is the number of vertices of P1*
5. *Repeat the* steps from 1 to 4 by making *P1 as partA and P2 as partB.*
6. ${}^{+}L_x^{XY} = \min({}^{+}L_x^{a}, {}^{-}L_x^{b})$ and ${}^{-}L_x^{XY} = \min({}^{-}L_x^{a}, {}^{+}L_x^{b})$

The algorithm above can be repeated by initializing the sweep line at X= initial sweep line position to get ${}^{+}L_y^{XY}$ and ${}^{-}L_y^{XY}$. This algorithm is illustrate with following walkthrough.

In Figure 7, Pa is swept by the sweep line, stopping at every event at points p1, p2, p3, and p4 of Pa. At every event, the sweep line also intersect some of the segments of Pb. The minimum distance from the event point to these intersection points are saved. When the sweep line reaches p4, the minimum of distances in X+ and X- direction saved during the sweep is considered as the amount of freedom for Pa with respect to Pb, denoted by ${}^{+}L_x^{a}$ and ${}^{-}L_x^{a}$ respectively.

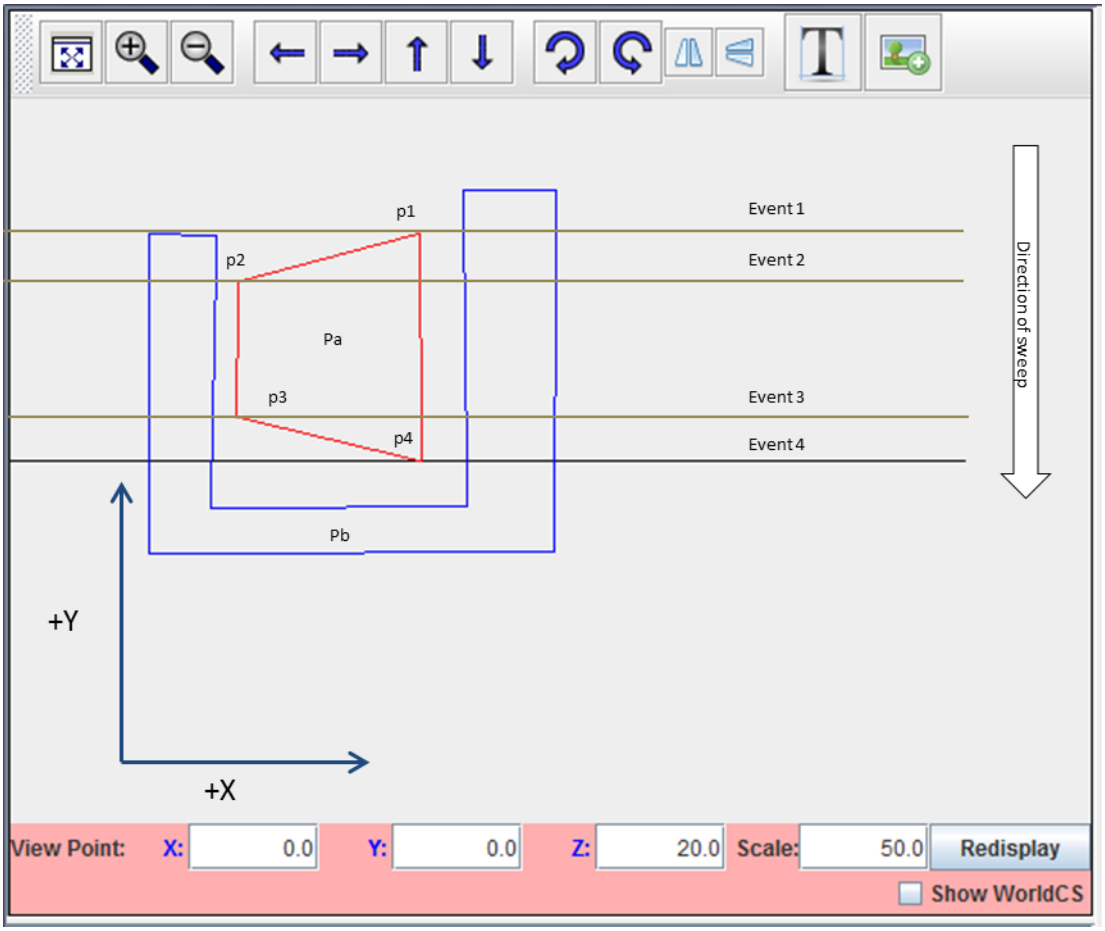


Figure 7: Sweep plane algorithm executed on pair of cross-sections

C.Detection Of Joints

The algorithm presented in the last section for calculating degrees of freedom needs to be applied on a pair of cross-sections taken from two different parts on same plane. This also explains that degrees of freedom of any one part can be calculated only in relation to another part. Therefore, it can be stated that the degrees of freedom calculation is only meaningful for its interacting parts. For example, part A and part B are non-

interacting parts in the assembly displayed in Figure 8. Two interacting parts interact with each other through the joint(s) formed by them. Therefore the cutting plane is positioned in such a way that it can take cross-sections of parts at their joints. Detecting joints in an assembly is achieved by determining intersection of bounding boxes generated by two interacting solid bodies. This is explained below.

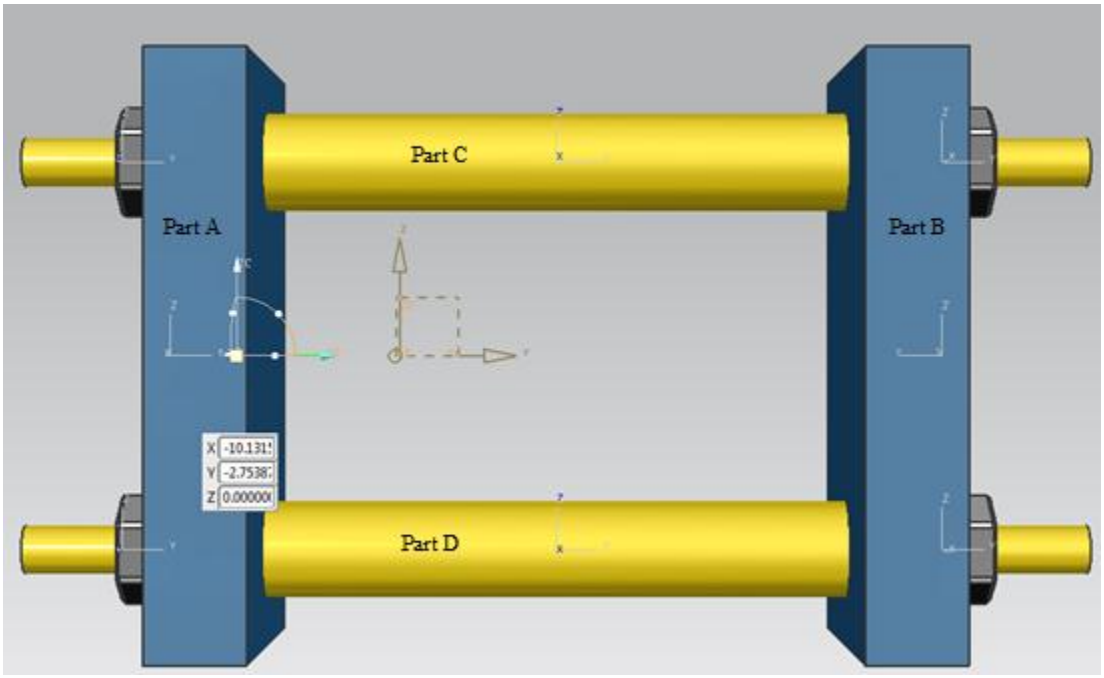


Figure 8: Demonstration of interacting and non-interacting parts of an assembly

In this simple algorithm proposed below, bounding boxes for each part in the assembly is calculated first. Bounding boxes are placed in 3D space at their relative position same as the position of the part in the assembly. Next all possible intersection of bounding boxes are calculated. As all bounding boxes are axis aligned bounding box (AABB), it can be contended that the intersections of two bounding boxes will be a

cuboid. This axis aligned cuboid can be treated as bounding box of the joint between two parts, taking part in the intersection. On the other hand, any pair of parts, who's AABBs are non-intersecting, can be treated as non-interacting parts and no DOF check is needed to be performed.

In Figure 9, the AABBs are calculated for the two parts assembly displayed on the left side. In the middle of the figure, AABBs are plotted in OpenSCAD [34], preserving location and orientation of the corresponding part in the assembly. On the right, the cuboid generated as a result of these two AABBs is shown. This cuboid can represent the region of the joint formed by two parts in the 3D space.

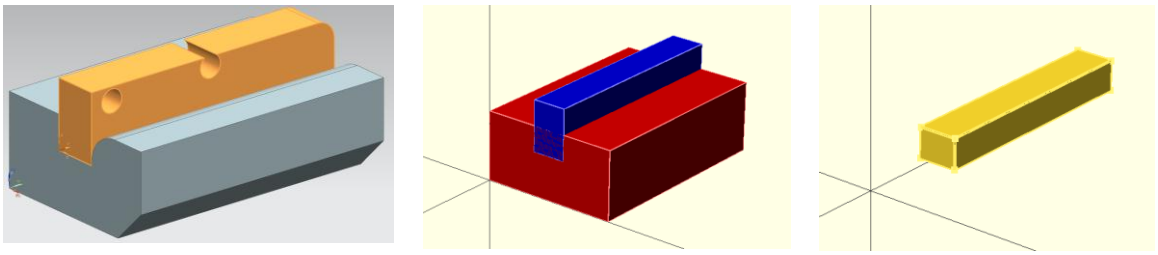


Figure 9: Identification of joints by intersection of bounding boxes

There are several ways to calculate the AABB of a 3D shape. For a polyhedral shape, AABB can be calculated simply by sorting all points found in the B-REP model of the shape by each coordinate (x, y, z), then taking the maximum and minimum coordinates, then making two points (one with maximum coordinates and another with minimum coordinates) as two opposite corners of the AABB. This two opposite corner points are sufficient to define any cuboid in a 3D space. For non-prismatic part with curved surface, the calculation of AABB is more challenging. For a curved surface, the

maximum or minimum of the surface equation obtained by differentiation in each principal direction should be added to the list of points and can be used to find the peak of the curved surface in a given direction. Most of the modern CAD tools are capable of calculating the AABB of parts of a design (Figure 10), and many algorithms exists both in the literature and practice. In this thesis, experiments are conducted on prismatic parts only. Therefore bounding boxes are calculate by extracting points from the B-REP of the part and finding maximum and minimum points after sorting them by their X, Y and Z coordinate position.

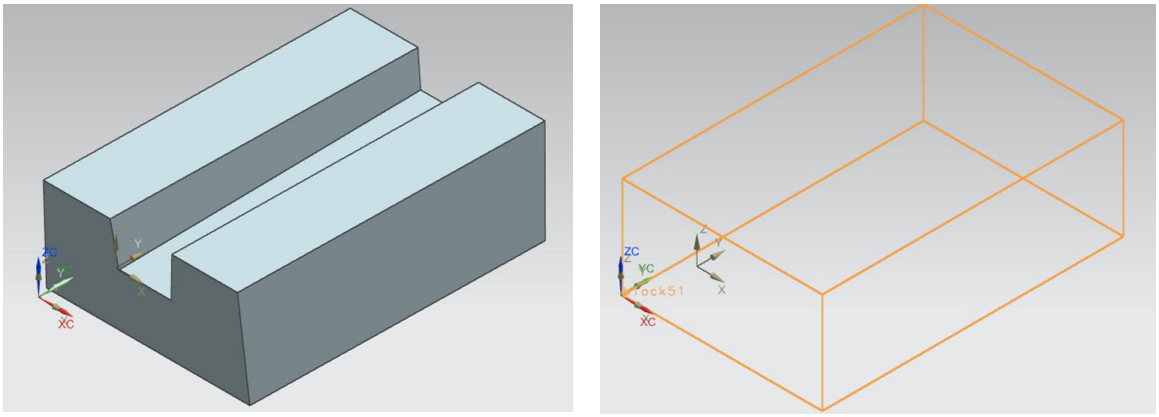


Figure 10: AABB calculated in CAD application

Chapter IV Experiments and Results

Two CAD models, which are used to apply the structural analysis algorithms, are introduced in this chapter. The first section upholds a detailed discussion on various programs and interfaces developed to collect data and implement the algorithms.

Two CAD models are created for the testing algorithms of structural analysis. The first model, called the BlockSlide, is shown in Figure 11. This model is basically a slider in a slot created in a block structure. This model has a large tolerance and non-parallel faces. The slider, being not connected to the block in a fixed joint, can slide in the slot. However, looking at the model, any designer can tell that the slider is free to slide through the slot of the block in Y-Y direction. This is the structural behavior specific to this assembly and can be extracted by structural analysis.

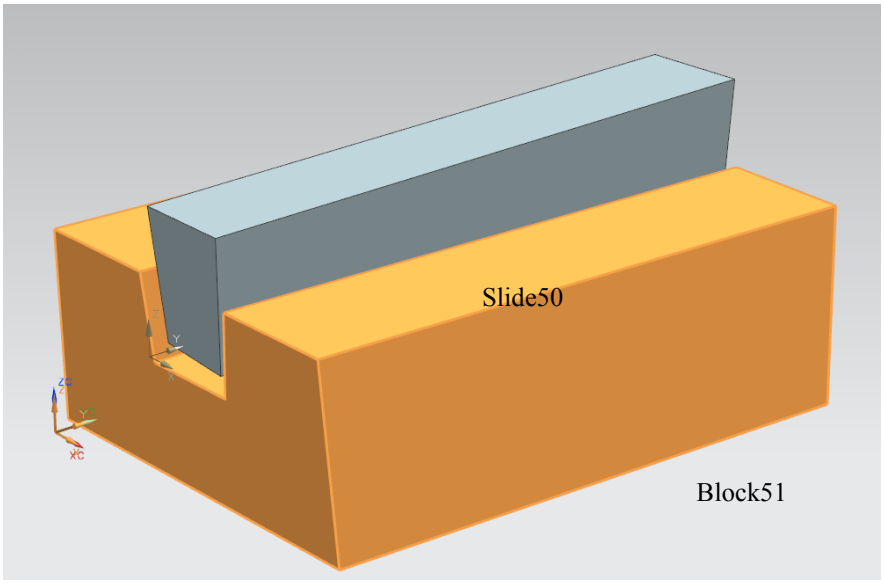


Figure 11: BlockSlide assembly model view

The second model is a four-piece assembly which is displayed in Figure 12: PlateRod assembly model view. This model is much more complex than the BlockSlide model with two plates connected by two rods. The plates have through slots where the rod's tips are inserted. A fastener, that is applied on both tips of the rod can fix the plate with the rods but in this experiment they are omitted for simplicity. When looked closely into the joints formed by plates and rods, it can be seen that the slot is bigger on the side where the rod entered in, and smaller on the other side with an edge inside.

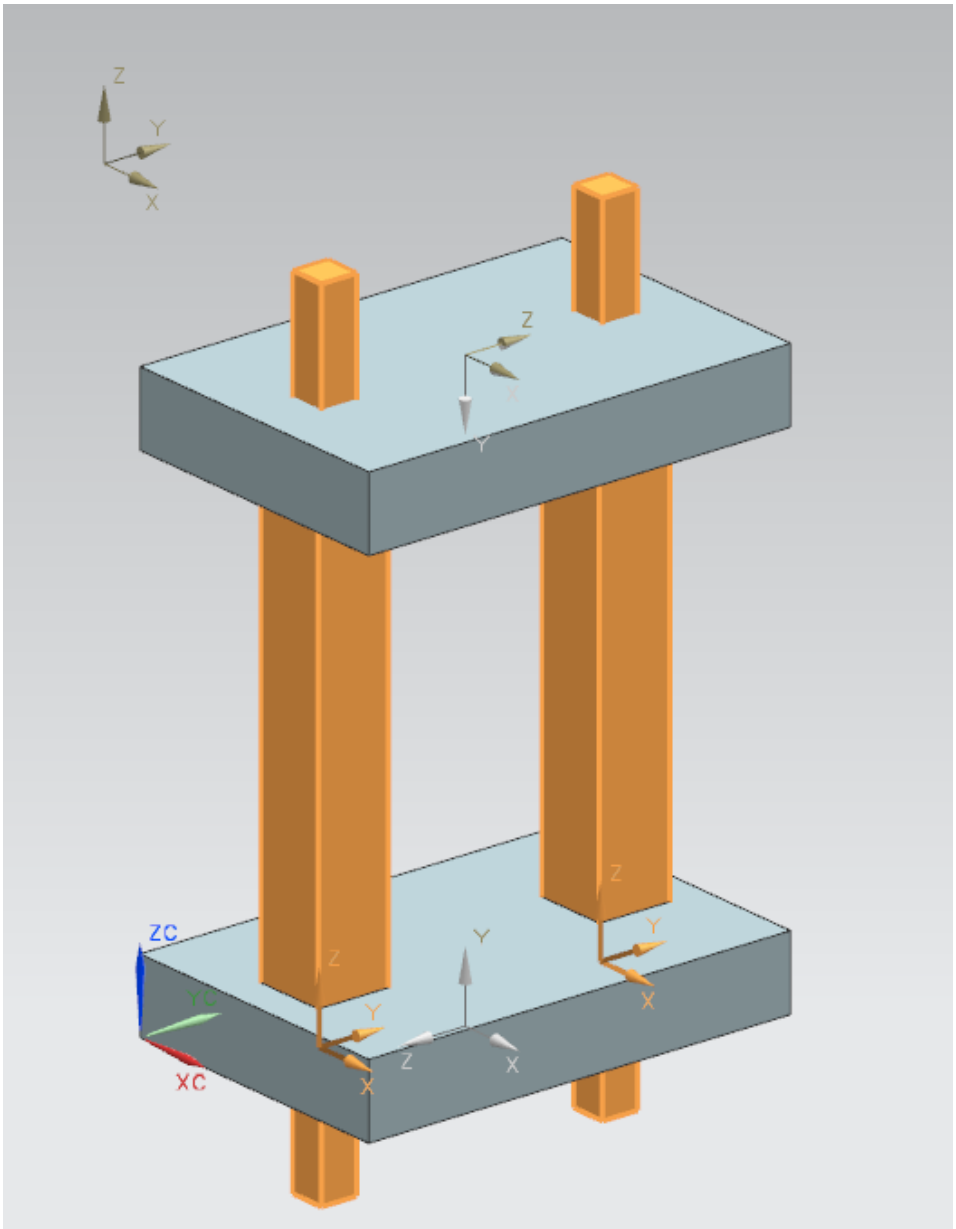


Figure 12: PlateRod assembly model view

As the rods also have tapered tips, the plates cannot be brought closer to each other. This is shown in Figure 13: PlateRod joints in dimmed edge view. This is the structural behavior of this assembly, which can be observed by any designer. This model

is exposed to structural analysis to investigate whether this structural behavior can be extracted with the help of proposed algorithms.

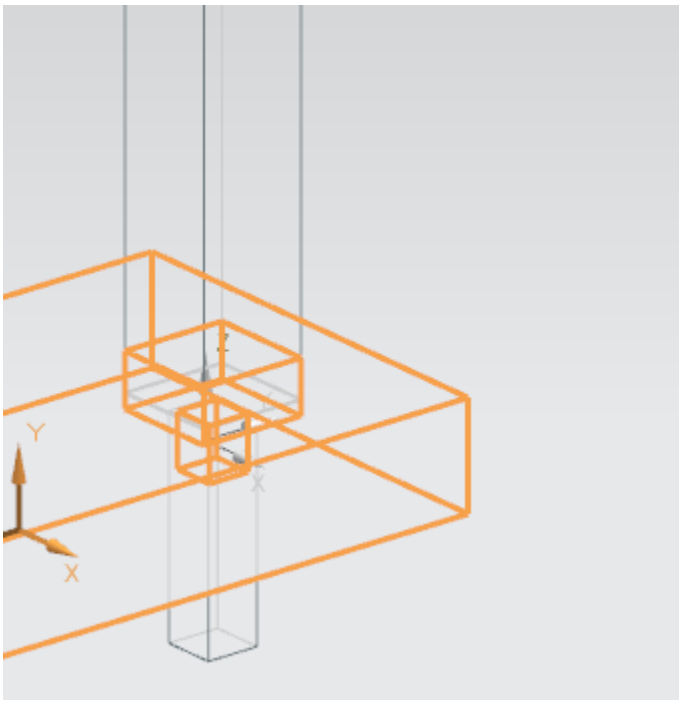


Figure 13: PlateRod joints in dimmed edge view

A. Experiment Tools and Setup

The degrees of freedom calculation described in Chapter IIIA is implemented by a java-testing tool, named TestBed, developed specially for this purpose. TestBed is a versatile tool, which can accommodate different algorithms by implementing an interface called algorithm. The class hierarchy diagram of PolygonDOF, which extends algorithm and implements sweep is explained in the next section and shown in Figure 14. Before a pair of cross-sections is analyzed in this tool, the joints in the assembly should be

identified and then cross-section should be performed in three different axes at a pre-defined position as described on Chapter IIIB. UGS NX from Siemens [35] is a popular and powerful CAD software, which also publish a set of APIs to access its core functionality through third party programs. These APIs are used partially in a set of java and vb.Net programs to calculate bounding boxes for the parts to identify joints and perform cross-section for them. These operations are elaborated later.

1) SweepLine Algorithm

In this model, every line of the cross-section is represented by the class LineSegment. The SweepLine class represents the sweeping line, which can be instantiated at desired position to sweep the cross-section. The two part cross-sectional faces are stored in two different collections of line segments in PolyDOF class. One part is checked for DOF with respect to the other. All end points of the line segments making up the part, is stored in a priority queue of type Event. Sweep Line moves from one event to another, checking whether the SweepLine can intersect any other LineSegment from the other part.

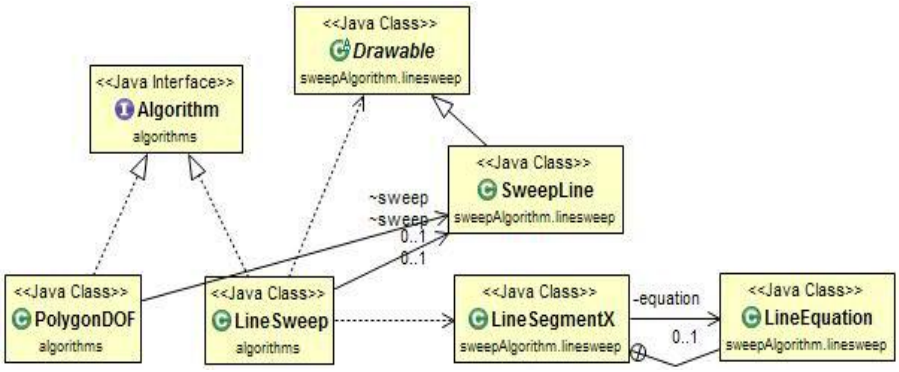


Figure 14: Sweep Line algorithm implementation

2) Bounding Box calculation and cross-section extraction

Bounding box of individual parts are calculated by sorting all points found in the B-REP model of the shape by each coordinate (x, y, z), then taking the maximum and minimum coordinates, then making two points (one with maximum coordinates and another with minimum coordinates) as two opposite corner of the AABB. NXOpen Java library published by UGS NX, has APIs to load an assembly, work on each part separately and perform cross-sections. Session interface, one member of this library, has a number of APIs to open an assembly and access every part. A class called NXConnectAssembly uses this APIs to load a part, browse its components and select any part from the assembly. A separate class called CurveGenerator is developed which uses various NX APIs belonging to interface IntersecctionCurveBuilder found in the NXOpen library, to perform a cross-section in a particular axis at a given distance from origin.

The bounding box algorithm also uses the TestBed mentioned in the last section. A new panel (see Figure 15) is added to the TestBed which has provision to open a .prt file

containing the assembly design. It lists all its component and calculates the ABB for any part selected from the list.

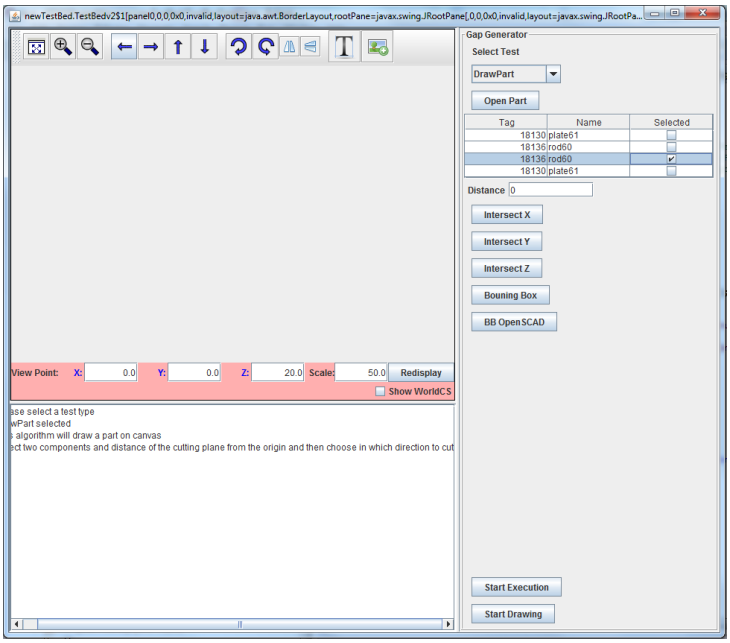


Figure 15: ToolBox panel for bounding box calculation

Various other tools such as Microsoft Excel macros and NX journals are used in this experiments. It is worth to mention about NX journaling, which can record any operation through the UGS NX CAD modeler and churns out a vb.Net code. This code can be enhanced with user specific codes and run to perform operations on the design on NX CAD modeler. A demonstration of how NX journal is used to extract cross-sectional line segments from a part is shown in Figure 16. Please see appendix for code snippets and other information.

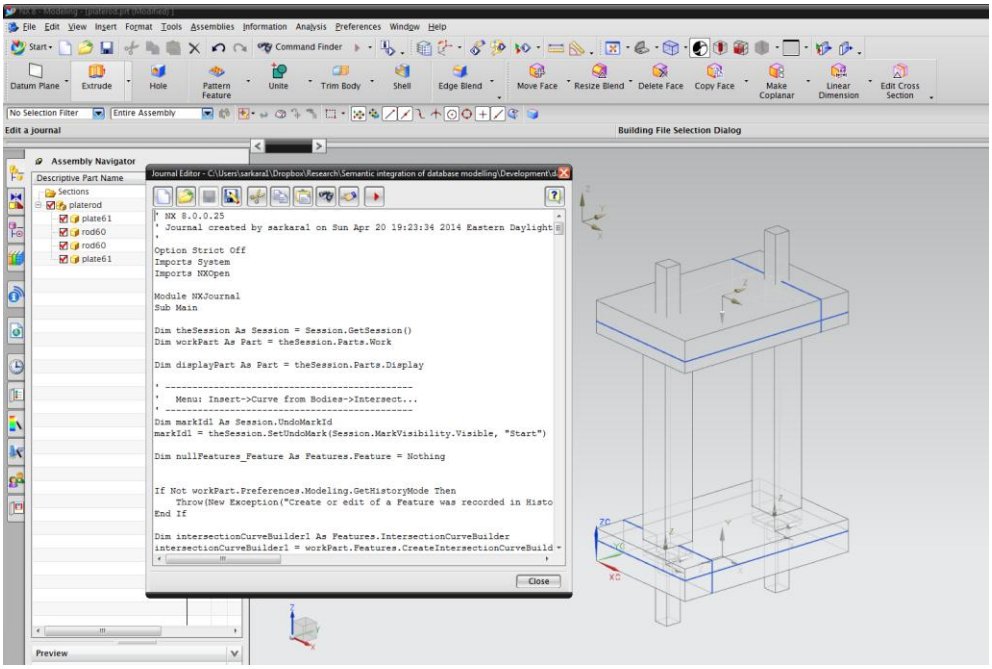


Figure 16: NX Journaling to perform intersection on bodies

Another tool used in the experiment is OpenSCAD, which is an open source CAD compiler of script based modeling [34]. OpenSCAD is made on free libraries; such as, QT for GUI, CGAL for CSG, OpenCSG and OpenGL for CSG rendering. OpenSCAD has a set of script which can be used to design 3D models and perform CSG operations on them. OpenSCAD script library is provided in Figure 36[34]. OpenSCAD scripts are used in the result section to express ABBs and transformations.

B. Results and Discussions

In this section, testing results from the structural analysis experiments performed on two demo designs are recorded. For each design loaded as a .prt file, first the ABB for

each part in the assembly is calculated and joints in the assembly are found by performing Boolean intersection on the bounding boxes. Next cross-sectioning is performed on a point within the range of the ABB surrounding the joint, that are consecutively on pair of parts forming the joint in three different axial planes. The cross-sections extracted are further analyzed with the help of the sweeping line algorithm to determine the DOF for each part with respect to another. Finally, the results from the sweep line algorithm are recorded in the truth table. This truth table is then analyzed through manual observation to interpret the expected behaviors of the design. However, it should be noted that this interpretation should be performed by querying a suitable reference ontology.

1) Result of experiments performed on BlockSlide.prt

ABBs of two parts of BlockSlide assembly are computed separately and the results are recorded in Table 5: ABB Calculation on BlockSlide Assembly. In this table, ABBs and their corresponding transformation is expressed in OpenSCAD scripting format. The ABBs are expressed in the format *cube([X length, Y Length, Z Length], center = true)* and the transformation is expressed in format *translate([x, y, z])* or *rotate([x, y, z]).*

Table 5: ABB Calculation on BlockSlide Assembly

Part name	**ABB calculation**	**ABB Position**
Block51	cube([200,300,100], false)	translate([0,0,0])
Slide50	cube([49.50,300.00,74.00], false)	translate([75.50,0.00,62.50])

In Figure 17, the left side of the picture shows two ABBs modeled together in OpenSCAD and the picture on the right side shows the resultant joint ABB created from the intersection performed on two ABBs.

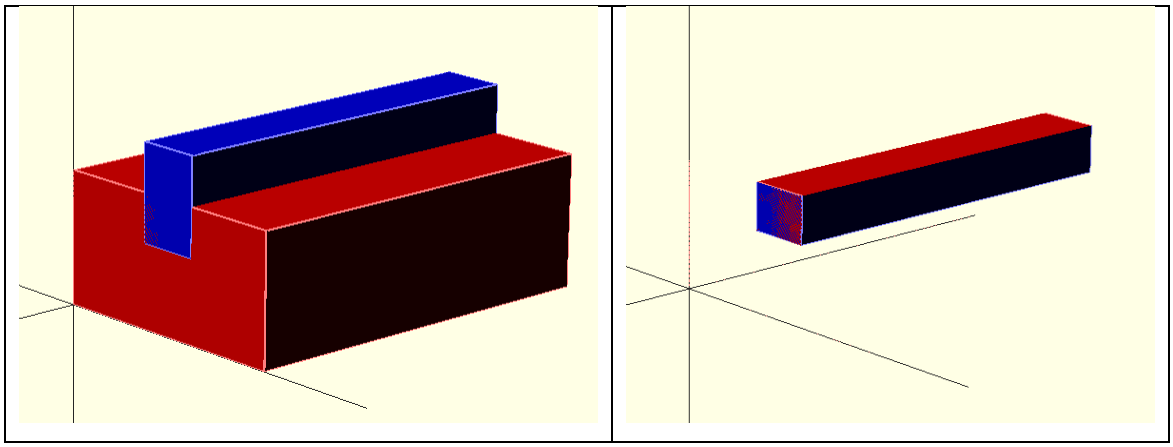


Figure 17: Boolean intersection performed on the ABBs of two parts of BlockSlide assembly

Next, three figures (Figure 18, Figure 19, and Figure 20) show the application of the sweeping line algorithm on three cross-sections extracted from the assemblies.

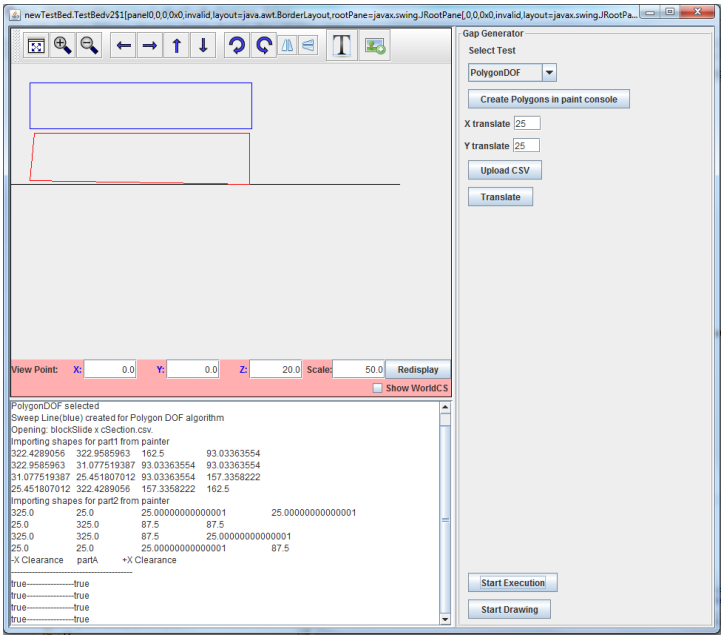


Figure 18: Calculation of DOF of Slide w.r.t. Block in +Y and –Y direction

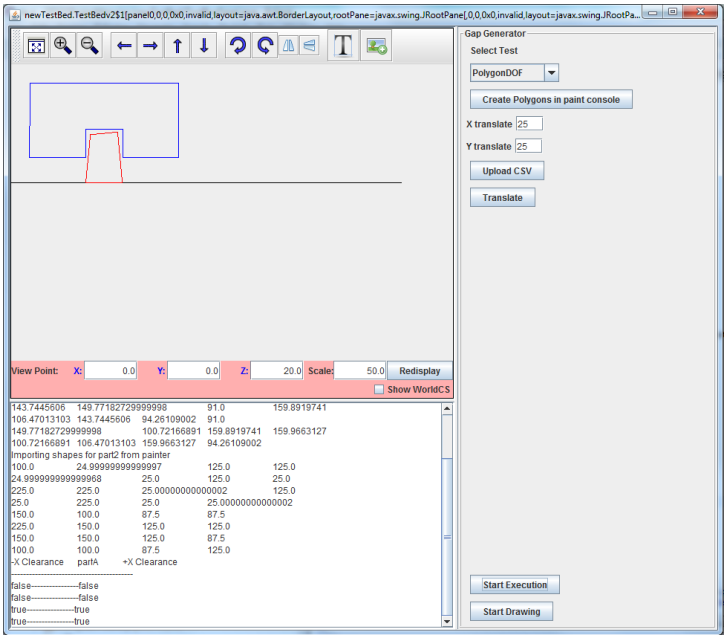


Figure 19: Calculation of DOF of Slide w.r.t. Block in +X and –X direction

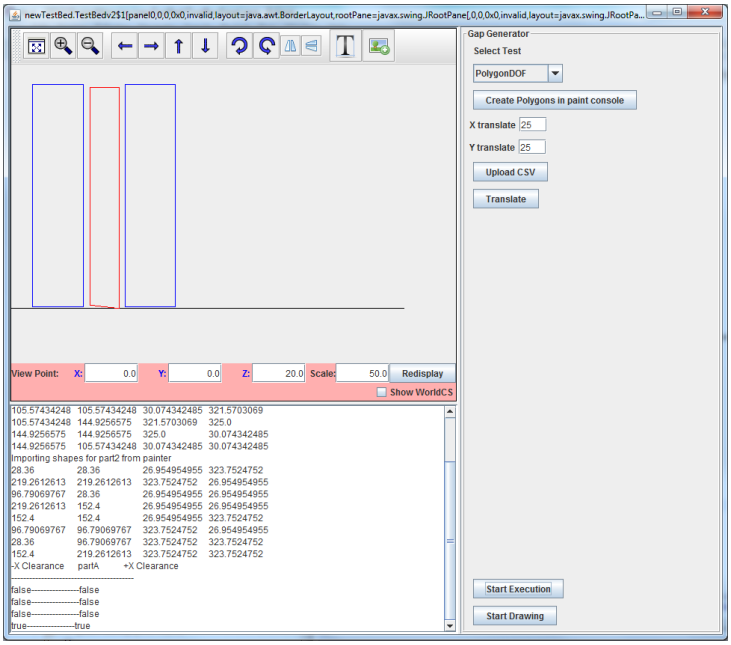


Figure 20: Calculation of DOF of Slide w.r.t. Block in +Z and –Z direction

In the Table 6, the DOF of each part with respect to another part is captured with 1 and 0 value, where 0 stands for complete clearance (infinite degrees of freedom) and 1 stands for no movement (obstruction).

From the truth table, it can be depicted that, both components have no freedom in X-X direction. However, they are both free in Y-Y direction. This clearly shows that the slide can move along Y-Y direction through the slot in the block or the block can move in Y-Y direction keeping the slider fixed. Interestingly, the block is free towards –Z and the slide is free in +Z. This can be deciphered as the block and slide are able to move away from each other but cannot come closer in the opposite direction. These observations are explained in Figure 21, in which white arrows shows no freedom and black arrows show

infinite freedom. As a whole, these observation may form the set of expected behaviors derived from the part.

Table 6: Truth Table for BlockSlide Assembly

	Block51						**Slide50**					
Direction	X+	X-	Y+	Y-	Z+	Z-	X+	X-	Y+	Y-	Z+	Z-
Block51							1	1	0	0	1	0
Slide50	1	1	0	0	0	1						

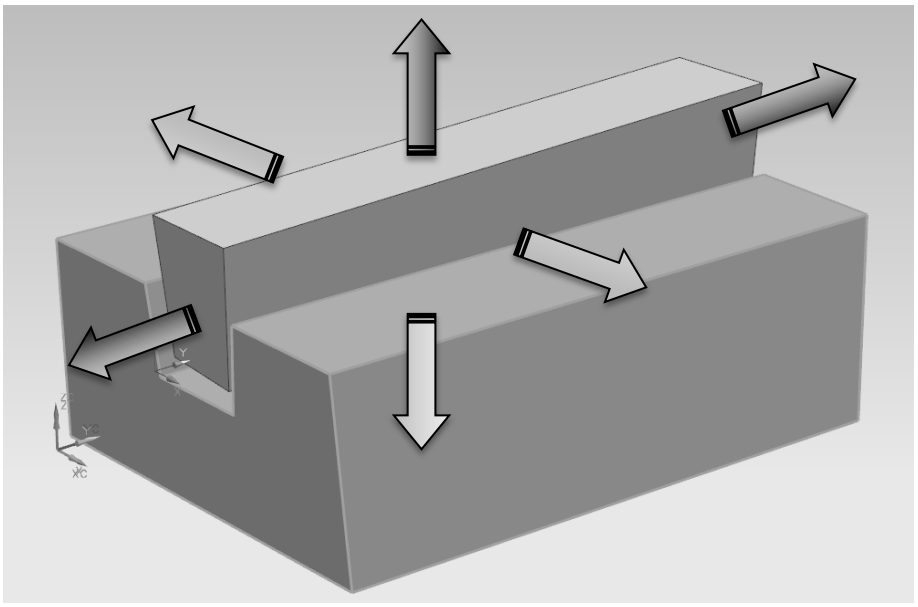


Figure 21: DOF analyzed on BlockSlide assembly

2) Result of experiments performed on PlateRod.prt

ABBs of two parts of BlockSlide assembly are computed separately and the results are recorded in Table 7. In this table, the ABBs and their corresponding transformations

are expressed in OpenSCAD scripting format.

Table 7: ABB Calculation on PlateRod Assembly

Part name	**ABB calculation**	**ABB Position**
Plate61a	cube([100, 25, 150],false)	rotate([-90,0,0]) translate([-37.5,-50,-12.5])
Rod60b	cube([25,25,275],false)	translate([0,0,0])
Plate61a	cube([100, 25, 150],false)	rotate([-90,0,0]) translate([-37.5,-250,-12.5])
Rod60b	cube([25,25,275],false)	translate([0,100,0])

In Figure 22, the left side picture shows four ABBs modeled together in OpenSCAD and the picture on the right side shows the resultant joint ABBs created from the intersection performed on four ABBs.

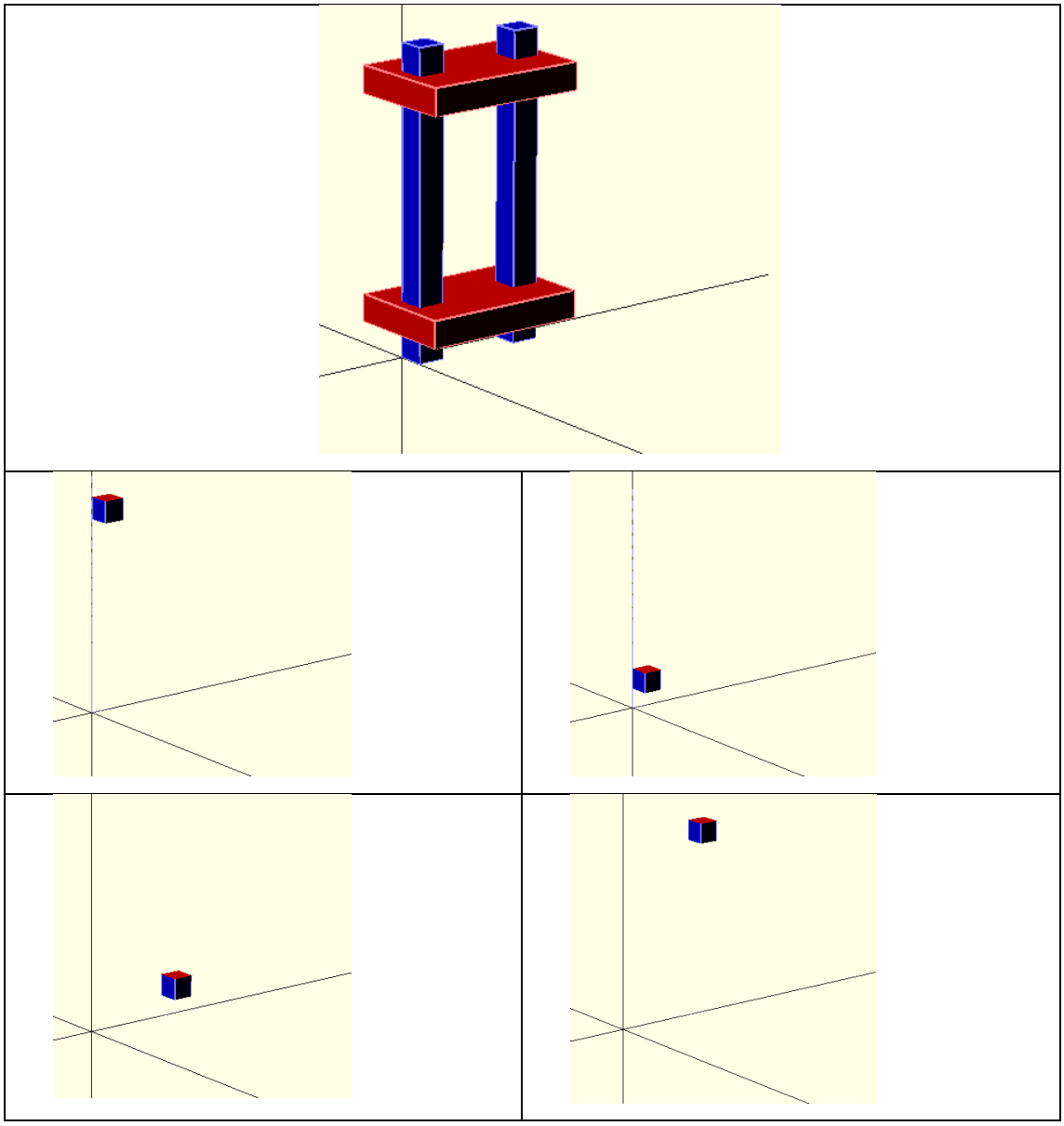


Figure 22: Boolean intersection performed on the ABBs of two parts of PlateRod assembly

In Figure 23 application of sweeping line algorithm in X-X direction for one of the joint is displayed. These cross-sections are taken on one of the plate-rod joints by cutting

with YX plane. It can be noticed the red cross-section from the rod fitting tightly in one of the slots of the plate.

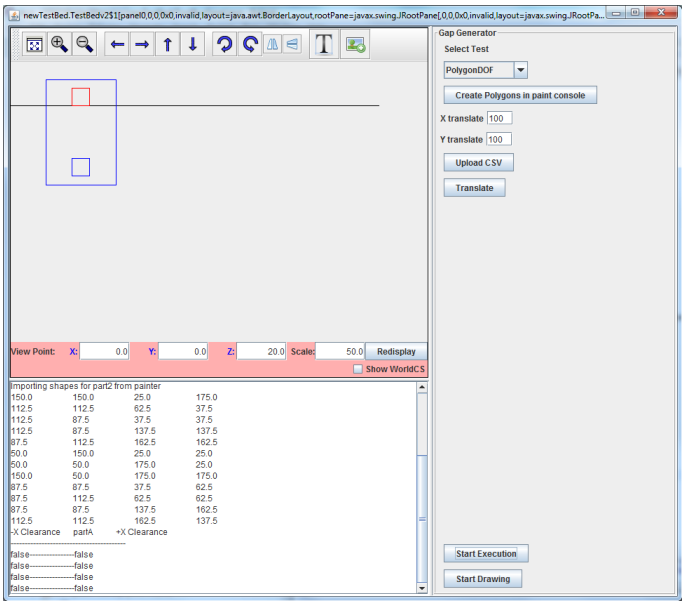


Figure 23: Calculation of DOF of Rod w.r.t Plate for one of the joint in +X and –X direction

In the following tables, the DOF of each part with respect to another part is captured with 1 and 0 values, where 0 stands for complete clearance (infinite degrees of freedom) and 1 stands for no movement (obstruction). Two tables (Table 8 and Table 9) shown are part of one table, which is broken down to respect space.

A set of expected behavior can be interpreted from the truth table, which shows that Plate61a is free in positive Z direction with reference to both rods. On the other hand, Plate61b is free in negative Z direction with respect to both the rods. It is to be noted that Rod60a is free in negative Z direction with respect to Plate61a but not the other way. The

same part is free in positive Z direction with respect to Plate61b but not the other way. These two freedoms of Rod60a negate each other and we can conclude that Rod60a has no DOF. This holds true for Rod60b, too. Therefore, in this assembly, only plates can move away from each other but cannot come closer than a certain distance. This is marked with black arrows in Figure 24.

Table 8: Truth table for PlateRod assembly (A)

	Plate61a						**Rod60a**					
Direction	**X+**	**X-**	**Y+**	**Y-**	**Z+**	**Z-**	**X+**	**X-**	**Y+**	**Y-**	**Z+**	**Z-**
Plate61a							1	1	1	1	0	1
Rod60a	1	1	1	1	1	0						
Plate61b							1	1	1	1	1	0
Rod60b	1	1	1	1	0	1						

Table 9: Truth table for PlateRod assembly (B)

	Plate61b						**Rod60b**					
Direction	**X+**	**X-**	**Y+**	**Y-**	**Z+**	**Z-**	**X+**	**X-**	**Y+**	**Y-**	**Z+**	**Z-**
Plate61a							1	1	1	1	0	1
Rod60a	1	1	1	1	0	1						
Plate61b							1	1	1	1	1	0
Rod60b	1	1	1	1	1	0						

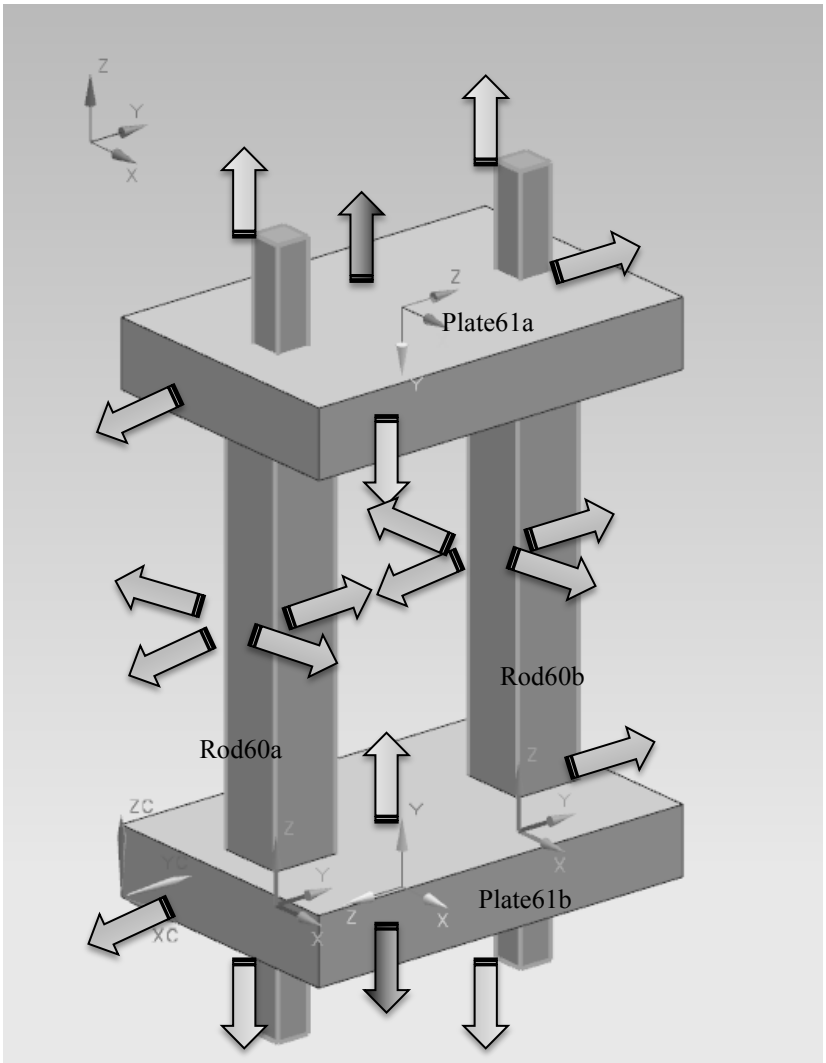


Figure 24: DOF analyzed for PlateRod assembly.

Chapter V Functional Design Of The SIDOS System

It has been shown in the last two chapters that functional properties of design can be extracted from pure geometrical design data. As a proof of concept, translational properties of each part of a product assembly are recognized through some innovative algorithms. Functional properties, extracted from the design need to be stored in the design data as semantics. Semantically enabled design data is very easy to integrate with heterogeneous application of computer-integrated manufacturing.

In this research, a novel design is proposed, which can be used to contextually interpret a non-semantics aware design. This system is based on the Function-behavior-structure model, proposed by John S. Gero in his seminal work on product design paradigms [36]. Example of the design principle of the proposed system is shown in Figure 25. At one end of the proposed system, a schema analyzer will extract structural behaviors of different components of the design. On the other end, an ontology equipped with domain knowledge of that particular product will supply contextual information based on the structural behaviors extracted from the design. An alignment algorithm compares the contextual information, supplied by the ontology with the semantics found in a semantically tagged template model, which should belong to the same product family. This way, the alignment algorithm can map non-semantics components of a design to components of a semantically mapped prototype, based on their contextual similarity. Upon successful matching, the non-semantics component can be tagged with the similar tag of the matched component of the template design.

As the proposed system tries to integrate design data models through the semantic knowledge of the components of the models, the system is named as "Semantic Integration of Design models though Ontology (System)", abbreviated SIDOS.

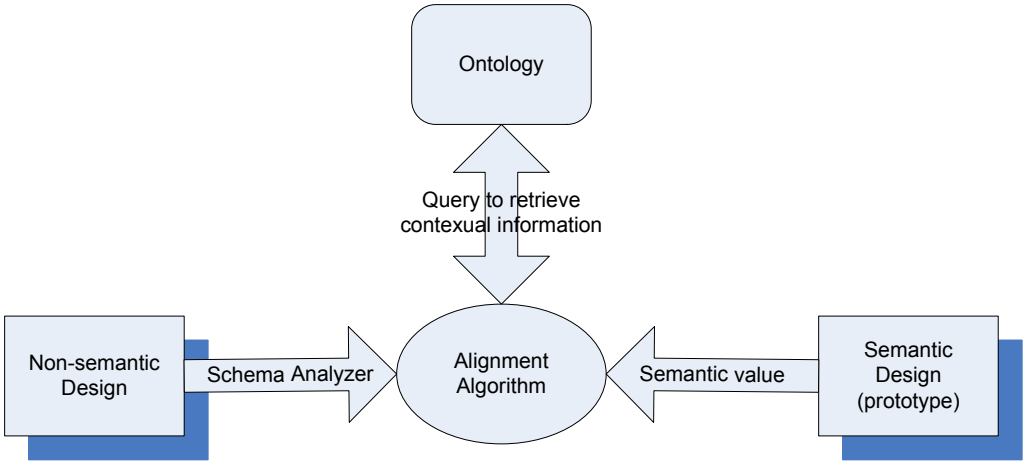
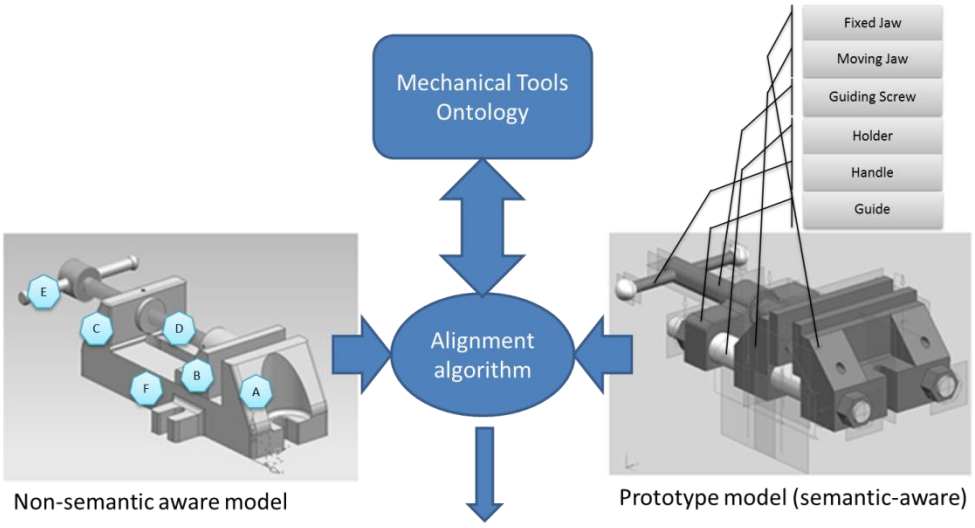


Figure 25: Overview of the proposed system

In Figure 26, a case is presented to illustrate the goal of the proposed system further. In this figure, a non-semantics aware design of a mechanical vise is shown on the left. Different components of the vise are tagged with alphabets from A to F. A template design of a vise is shown on the right. This vise is already semantically tagged and semantic meanings of those components are listed above the design. The template vise is made of a fixed jaw, a moving jaw, a holder, a guiding screw, one pair of holders, and a handle to rotate the guiding screw. A schema analyzer extracts the structural behaviors of different components of the non-semantics aware vise design. Ontology from domain of mechanical tools (a hypothetical ontology which stores knowledge and definitions of various mechanical tools) provides contextual meaning of these components, based on

their corresponding structural behaviors. The alignment algorithm matches components from the non-semantics aware model to components of semantics aware template model by their contextual similarity. From the probable matching result presented in the table at the bottom of the Figure 26, it can be observed, that component A from the vise design is semantically similar to the component “Fixed Jaw” from the template model.



A	Fixed Jaw
B	Moving Jaw
C	Guide
D	Guiding Screw
E	Handle
F	Holder

Figure 26: Semantic interpretation of mechanical vise by the proposed system

A. Principle Of The SIDOS System

Gero (1990) defined the designing process as “goal oriented” and “operates within a context which depends on the designer’s perception of the context” [36]. The function-behavior-structure model (Figure 27) described by Gero represents the model of designing activity. This model is described below. According to Gero, in a design process, a set of Functions (F) is transformed into a set of Design descriptions (D). For example, designer of a mechanical vise designs the apparatus in such a way that it can perform at least few basic functions of the vise, like holding or clamping a work piece. In this case, the design document is purported to be a CAD design. All elements described in that design document with their inter relationships are called the Structure (S). A structure is a set of 3D solids and their inter relationships comprised a manufacturing design. According to Gero, though structure is the physical representation of the Function, they cannot be translated to each other. However, a set of behaviors called structural behaviors (Bs) can be extracted from structure. A set of behaviors like, sliding of one jaw to another, rotation of guiding screws, fixed placement of one jaw and guiding block, joining of different parts etc can be derived from the dimension, placement and orientation of different 3D solids in a vise design. The set of functions pre-defined for the design can also dictate a set of behaviors, which are called Expected Behaviors (Be). In a mechanical design, the technical specifications of that particular design can be called as Expected Behaviors. During the design process, the designer constantly compares Be and Bs and changes S so that Be and Bs are matched.

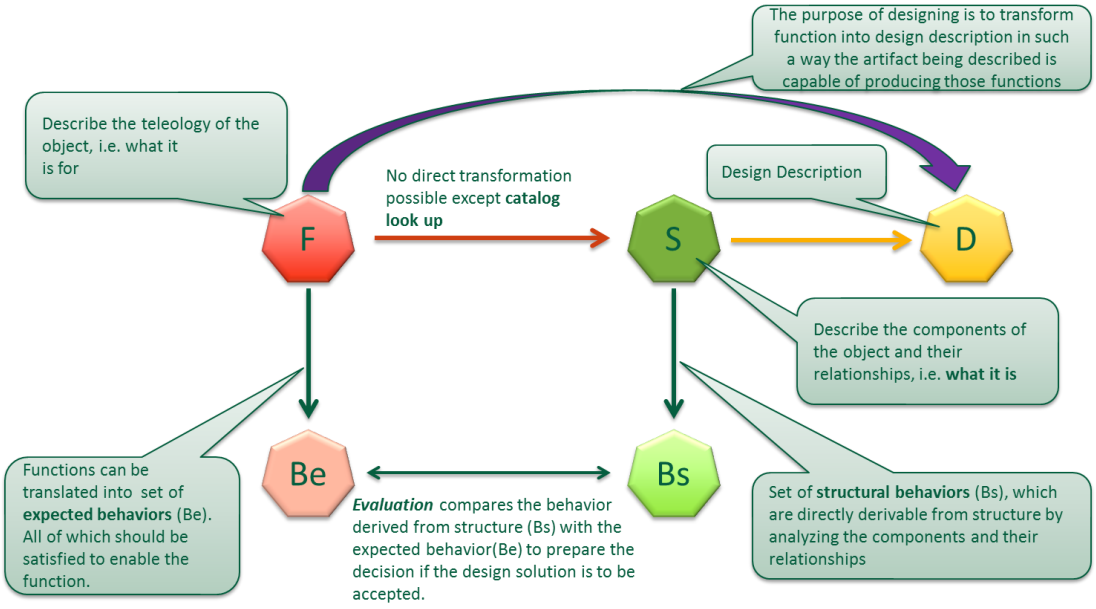


Figure 27: FBS framework [36]

The most important concept in FBS model is that, functions and structures can only be compared through their expected behaviors and structural behaviors. Therefore, it is necessary to extract structural behaviors from the non-semantics aware design as a first step to understand the design purpose. This set of structural behaviors can be reduced to a set of expected behavior, by referencing suitable domain Ontology. Now these expected behaviors can be compared with expected behaviors derived from the semantics-aware standard model and if they are comparable, semantic tag can be assigned to the semantic-aware part to the non-semantics aware part. It is to be noted that ontology used in this process holds the functions of the design, which is in other word, specification of the product.

Let's consider D is a non-semantics aware design and D_X is the semantics-aware CAD design used as standard. Most of the CAD design data format (STEP, IGES, STL

etc.) can be translated into geometrical information, which can be a B-REP or CSG representing pure 3D geometry of the design. Therefore, D can be translated into S.

$$D \rightarrow S \quad (13)$$

S can be analyzed by applying various structural analysis algorithms to the pure 3D geometrical information, denominated by S. Therefore a set of structural behaviors B_S can be analyzed from the structure S.

$$S \rightarrow B_S \quad (14)$$

Product specification Ontology can be used to store all specifications for the product family, of which the participating schemas belong to. This ontology should have artifacts denoting specified forms of the product and functions as classes. Forms and functions should be linked with properties and axioms. From a validated ontology like this, DL query with parameters from B_S can extract a set of expected behavior B_e from the ontology.

$$Bs \rightarrow B_e \quad (15)$$

On the other hand, semantics-aware design data Dx is already embedded with semantic information, which directly correlates to the product specification Ontology used in this process, therefore, Dx can be directly interpreted in a set of expected behaviors B_{eX}.

$$Dx \rightarrow B_{eX} \quad (16)$$

An alignment algorithm can be applied to Be and B_{eX} to find closest pair of sets, which corresponds to a certain design feature of D and Dx respectively. Any design feature of D, which has a closely matching set of expected behaviors with a particular

design feature of Dx, can be annotated with the same semantics used for the design feature in Dx.

B. How Design Data Can Be Aware Of Semantics?

Semantics in manufacturing designs fill the lack of semantic awareness of traditional geometrical modeling. Traditional logical schemas are limited to the terms of representation, rather than meaning. These terms of representations do not talk about the real world context. In a semantic data model, contexts are separately specified in the schema by the data modelers and later reused by users [37]. In terms of data integration, sematic data modeling opens up a possibility of global queries based on semantics, rather than representation. In recent times, semantic data modeling is still a conceptual data modeling practice.

In the SIDOS system, markers are conceptualized as keywords or numerical tokens, which can be embedded as meta-tags with each member (entity or attribute) in a semantically modeled schema. A semantic library acts as a repository of all semantic markers used in different schema. Therefore, semantic markers act as a unique identifier, which refers to a set of expected behaviors described in a semantic library. Several modern modeling technologies can be leveraged during practical implementation of the concept of semantic markers. It is to be noted; that the technical design of the ontology will decide which portion of the data can be annotated.

STEP standards like AP203, AP214, and AP239 can be extended to insert contextual information in the product model data. In these semantically enhanced STEP standards, form, function and behavior of a mechanical part can be captured. In a PDM schema, entities like product_definition_context or application_protocol_definition can be extended in new entities with attributes for capturing links to the semantic library. A mapper program can parse the extended entity information and extract the contextual information (Form, Function, and Behavior) of a part by referring the semantic library. Design data in CAD applications are saved in proprietary structure based on XML standard. Since XML is an open standard, adding new attribute or a child node in the product information node can be achieved by extending these schemas.

Research on the semantic web can contribute in designing semantic markers for manufacturing product design. Developed by W3C, Resource Description Schema (RDF) provides description to the resource. Web Ontology Language (OWL) is a family of languages, which help in developing Ontologies to store knowledge in a special entity-relationship based framework. For manufacturing, there are many existing ontologies capable of storing domain specific product design knowledge inter related to other manufacturing processes. They are described in the next section. In a semantic web, extendible HTML (XHTML) contains semantic annotation to attach semantic meaning to the web information. This same concept can be applied to manufacturing design data. A part data can be embedded with RDF syntax. The example in Figure 29 is using a XML type data representation.

```
<Part rdf: about="http://manufacturing.org/tool/holder/vise/fixedjaw/typeA">
        <Body>
        .....
        </Body>
</Part>
```

Figure 28: Semantic is embedded in XML document in RDF syntax.

C. Design Of Ontology As A Semantic Library

Ontology in information science is defined as the formal representation of the knowledge with hierarchical concepts of the domain that are interlinked by properties, relationships and types. The fundamental reason for choosing an ontology as a semantic library for the proposed system is that it can define any term in a domain through an axiom, which can be used as a standard definition of the term. Semantic markers can refer to these axioms and provide definition of any design feature. A domain specific ontology suitable for the proposed system should have the following two criteria.

1. The ontology should contain knowledge of the functions of products. Functions are product specifications, which are granulized in terms of design components and parts of the design, joints and machining features of each body.
2. The ontology should be linked to the upper level manufacturing Ontology, which in turn should govern other manufacturing domain specific lower level ontologies. In this way, other manufacturing operations can also be connected,

by linking to the product specification Ontology through the semantics embedded in the design data.

A limitation in building a manufacturing ontology as a semantic library is the difficulty in gaining consensus among data modelers. Each community and organization has its own perspective of the real world, which influences the structure of any knowledge library. Therefore a standardized body like the W3 consortium could envisage the global semantic library to be adopted for data modeling. Even a large enterprise can define its own library, which can be used for all data models it creates. However, the effectiveness of the semantic library in integration of heterogeneous data models is dependent on the size and authority of the enterprise. A large enterprise can influence many smaller organizations to adopt its proprietary library.

The question of standardization leads to the idea of using a dynamic ontology rather than a static one. A dynamic ontology is capable of absorbing new concepts in its hierarchy. Data modelers in lieu of the standard can create new concepts. These new concepts are included in the global ontology. This concept is similar to developers writing macros and reusing them from a library. The benefit of a dynamic ontology is that it grows constantly and expands its coverage on multiple domains. This concept can be illustrated mathematically from the equation below, where S covers the set of concepts from domains a and b.

$$S = \{m\_a, m\_b\} \quad (17)$$

In the equation below, a new domain c with a new set of concepts mc is added.

$$S = \{m\_a, m\_b\} \cup \{m\_c\} \quad (18)$$

Upper level ontologies contain generic definitions and concepts of the domain. Lower level ontologies extend generic concepts with localized concepts. Lower level ontologies can define specialized knowledge, that are often domain specific [38]. Multiple levels of ontology with internal compatibility can be implemented in place of single upper level ontology, to increase the flexibility in local level integration. With selective domain filters, this type of pyramidal family of ontologies can accumulate and channel all new updates in the library to the apex library with selective domain filters. In Figure 29, a probable structure of a pyramidal family of Ontologies is displayed. In this structure,

- A generic engineering upper level ontology standardizes generic terms and definitions, mostly from the engineering discipline. This ontology should be chosen from a widely used standardized upper level ontology. This level also refers different cross principle concepts, which help in linking manufacturing to other industries.
- Core manufacturing concepts are captured in a manufacturing specific upper ontology. Several manufacturing data model described in Chapter V can be used as a framework for modeling the fundamental manufacturing knowledge.
- Different industries have different manufacturing practices, which can be captured in different sub-domains specific to different industries.
- An organized specific lower level ontology can capture its generic product structure, processes, and other planning parameters.

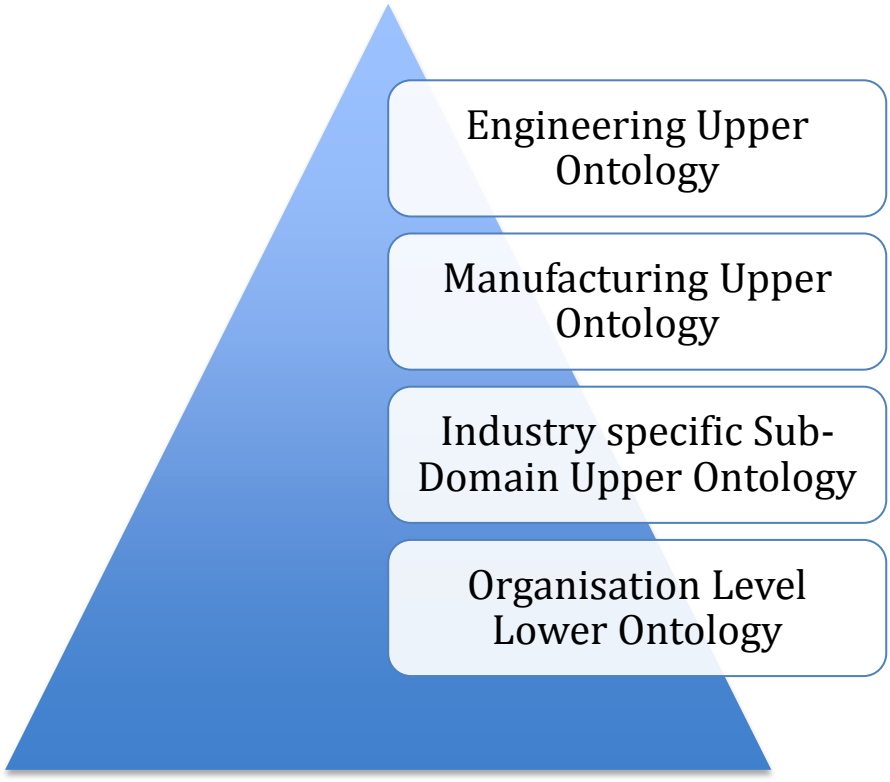


Figure 29: Pyramidal family of Ontologies

Many different manufacturing data models are researched on and developed in past decade, which can be leveraged to create manufacturing upper level ontologies. Object-oriented modeling is a technique, in which objects oriented models are able to capture the subjective reality with its principles, consisting of physical objects, abstraction of concepts, inheritance to derive subtypes from generic types, and polymorphism to capture subtle differences in concepts. For these reasons, a number of research focused on representing product design and standardizing manufacturing planning concepts in object-oriented models. Eventually, these standard models of

developing manufacturing ontologies proved to be the most efficient method of capturing semantics of a domain. Some of those researches are described below.

An object oriented model is developed for a rule based manufacturing planning system called IMPlanner [39], to capture process planning procedures and necessary data. The model representation includes feature model, process model, and necessary portions of part model, tool and machine model, which are required for process planning [40]. In this model, product design data is analyzed and represented as a collection of features. Every feature is represented by a concrete entity that inherits one generic feature type. Features are defined by their characteristics, which are stored as attributes of the feature class. This model also includes relationships between different features in order to represent feature precedence network, which constrains machining sequence for a specific part. Every feature is linked to possible machining processes with a may-be-machined-by relationship, in order to represent manufacturing knowledge. Processes are implemented using class hierarchy similar to features. As described in Figure 30, The IMPlanner model is thus quite flexible to store alternate machining plans at the same time. Additional models are also associated with machining processes to capture additional resources and to provide operational parameters for processing time, and cost estimation. This object-oriented model is used in several manufacturing planning prototype applications.

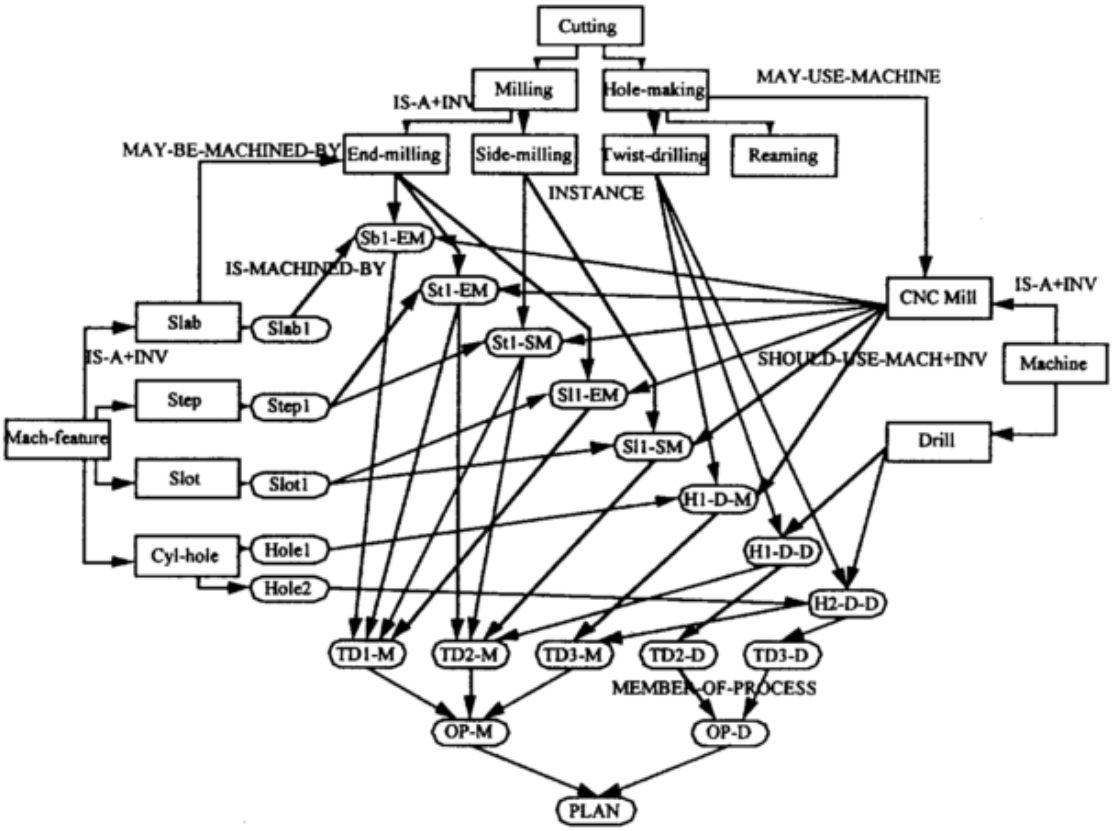


Figure 30: IMPlanner manufacturing planning model [40]

Core principles of Core Product Model (CPM) and its extension CPM2 [41] closely matches with the Gero's FBS structure described in Figure 31. At the heart of CPM lies the main entity called an artifact. An artifact can represent any component, part, assembly or sub-assembly of a product design. An artifact has three fundamental properties namely, function, form and behavior. Function is similar to intended behavior in the FBS model. This abstract concept can be extended to numerous part or assembly behaviors possible. Form can be compared to structure, which is in turn represented by geometry and material. Geometry and material are responsible for giving an artifact the

ability to perform a particular function. But the way the artifact operates in reality is captured by Behavior that can be extended by observed behavior and evaluated behavior. It is worth to note that the purpose of the structural analysis presented later in this document is to compute evaluated behaviors. As CPM is itself not an ontology but rather a conceptual data model, other upper level ontologies and data models evolved from CPM; such as, Open Assembly Model (OAM) to represent product assembly and Product Semantic Representation Language (PSRL) for capturing formal product information. This model, being open and expandable, can incorporate an upper level ontology that acts as a global library for semantics related to product design data. Any organization can extend this ontology to match their standards and conventions. Various product families within the organization can have their own lower level ontologies, which are as subject specific libraries for semantic integration of product designs.

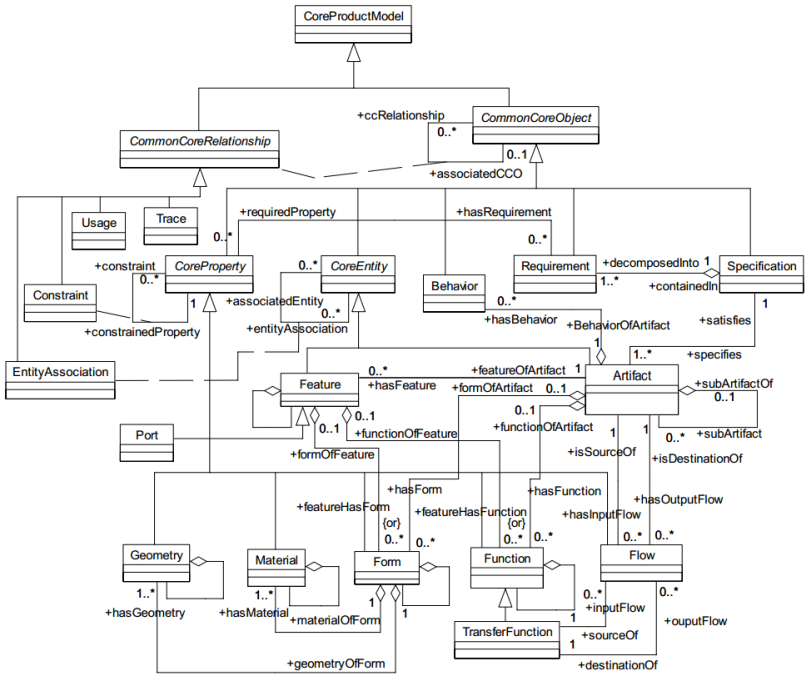


Figure 31: Core Product Model [42]

In the context of this research, assembly is extremely important for any mechanical design. Assembly and sub-assembly of any design follow a different set of principles than individual part design. Open Assembly model (OAM) (Figure 32), an extended model from CPM, defines assembly model, sub-assemblies and its relationship with parts [43]. Artifact class from CPM is extended as Assembly Association in OAM. Assembly Association aggregates different types of ArtifactAssociation classes. ArtifactAssociation class captures relationships among two or more parts in the assembly, or relations between one part and reference objects such as, space, start point and end point. ArtifactAsoociation class is further extended by PositionOrientation, RelativeMotion and Connection classes. The first two classes capture static and kinematic

properties of a part in an assembly. Connection class defines different types of connections possible among different sub-assemblies and parts and is extended by MovableConection, FixedConection and IntermittentConnection classes. Another class, ParametricAssemblyConstraint captures the types of connections present among parts of an assembly. These types are derived from standards defined in ISO 10303-108; such as, Parallel, ParallelWithDimension, SurfaceDistanceWithDimension, AngleWithDimension, Perpendicular, Incidence, Coaxial, Tangent, and FixedComponent.

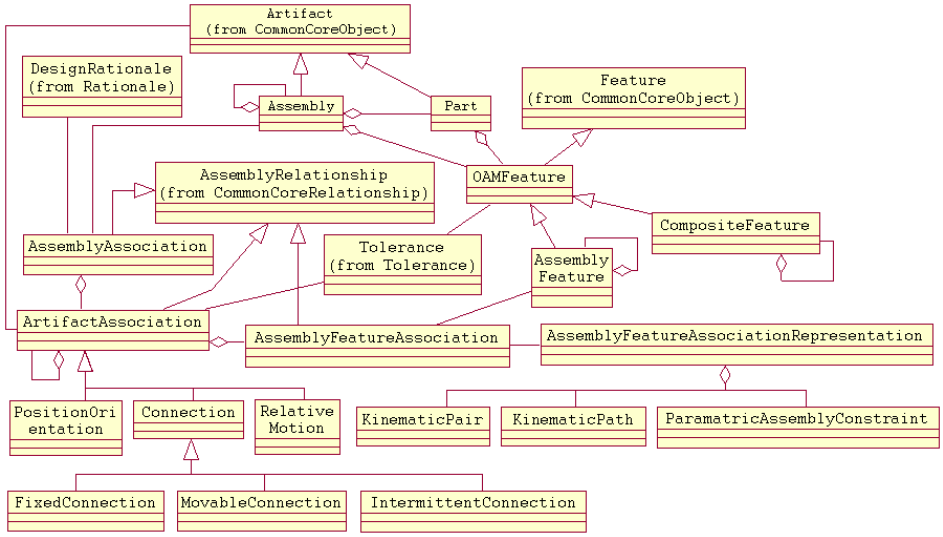


Figure 32: Open Assembly Model [43]

MASON (Manufacturing Semantics Ontology) (Figure 33) ontology proposed by Lemaignan et al. implements generalized manufacturing concepts as upper ontology [44]. This upper ontology guides specific ontologies to be integrated with the same formal concepts. Different entities of the upper level ontology have individual properties and

relationships, which defines the dependency among entities. Various lower level ontologies are included in MASON for different domains of manufacturing, such as, part model, operations and resources.

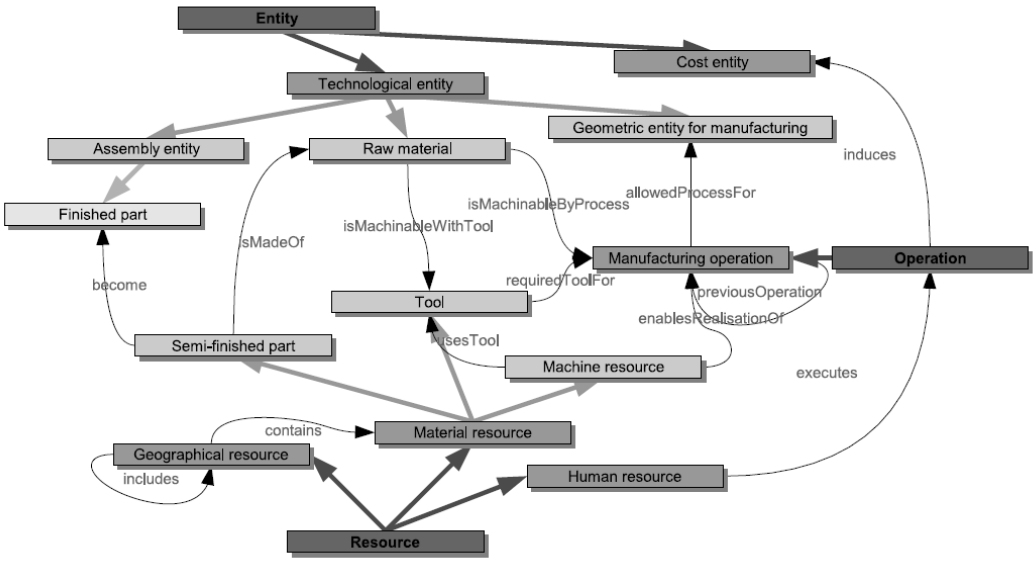


Figure 33: Upper Ontology of MASON [44]

D. Alignment Algorithm

Any manufacturing design can be either a single body part or an assembly made of multiple parts. A single part is a 3D solid body with one or many design features. As explained in Figure 34, these design features give the part a number of structural features. In this way every part of an assembly has their own set of structural behaviors. These sets of structural behaviors can be translated to individual sets of expected behaviors. On the other hand, the template design used for semantic alignment already contains semantic

markers, that can link individual parts of the design to its function defined by the ontology. Therefore every part of the semantics-aware design can directly provide its function expressed in set of expected behaviors. Alignment algorithm compares two sets of expected behaviors, namely, semantics-aware and semantics-unaware parts and it calculates a pair-wise mapping factor. In the end of the process, pairs of best mapping power are considered aligned and non-semantics part is annotated with the same semantic marker of its semantic counterpart. The alignment algorithm is expressed in Figure 34 below and detailed in the following bullet points.

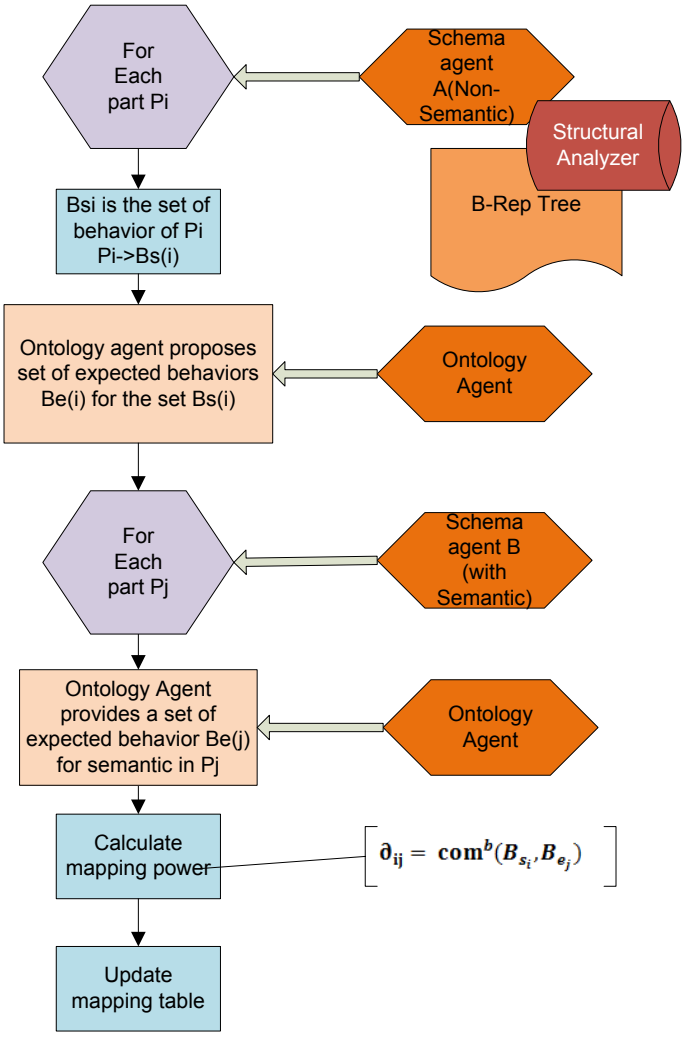


Figure 34: Flowchart of alignment algorithm

- Structure Si of Each component of the non-semantics aware design D is analyzed to extract their individual set of structural behaviors $\{\boldsymbol{B_{si}}\}$.

$\forall \boldsymbol{S_i} \in \mathbf{D};\ \{\boldsymbol{B_s}\}_{\boldsymbol{i}} = \boldsymbol{SA}(\{\boldsymbol{S_i}\})$ **, where SA() is a function which can perform structural analysis and i is the id assigned to the part/component** $\{\boldsymbol{B_s}\}$ (19)

- Ontology provides a set of expected behavior $\{\boldsymbol{B_e}\}$ for each set of structural behavior

$\forall\{\boldsymbol{B_s}\}_{\boldsymbol{i}} \in \{\{\boldsymbol{B_s}\}\};\ \{\boldsymbol{B_e}\}_{\boldsymbol{i}} = \boldsymbol{Q}(\{\boldsymbol{B_s}\}_{\boldsymbol{i}})$ **, where Q() is a query function performed on**

the Ontology

(20)

- Structure Sxi of Each component of the semantics-aware design Dx can supply the semantic marker, which can be directly mapped to a set of expected behavior $\{\boldsymbol{B_{exi}}\}$

  $\forall \boldsymbol{S_{xj}} \in \mathbf{Dx};\ \{\boldsymbol{B_{ex}}\}_{\boldsymbol{j}} = \boldsymbol{S}(\{\boldsymbol{S_{xj}}\})$ **, where S() is a function which can map a set of expected behaviors from semantics and j is the id assigned to the part/component** (21)

- For each pair of sets of expected behaviors $\{\boldsymbol{B_e}\}$ and $\{\boldsymbol{B_{ex}}\}$ mapping power $\boldsymbol{\eth}$ between corresponding parts/component Si and Sxi is calculated by the following function.

$$\forall\{\boldsymbol{B_e}\}_{\boldsymbol{ij}} \in \{\{\boldsymbol{B_e}\}\};\ \forall\{\boldsymbol{B_{ex}}\}_{\boldsymbol{j}} \in \{\{\boldsymbol{B_{ex}}\}\};\ \boldsymbol{\eth_{ij}} = \mathbf{com}^{\boldsymbol{b}}(\{\boldsymbol{B_e}\}_{\boldsymbol{i}}, \{\boldsymbol{B_{ex}}\}_{\boldsymbol{j}}) \quad (22)$$

$$\mathbf{where\ com}^{\boldsymbol{b}}(\{\boldsymbol{B_e}\}_{\boldsymbol{i}}, \{\boldsymbol{B_{ex}}\}_{\boldsymbol{j}}) = \frac{\{\boldsymbol{B_e}\}_{\boldsymbol{i}} \cap \{\boldsymbol{B_{ex}}\}_{\boldsymbol{j}}}{\{\boldsymbol{B_e}\}_{\boldsymbol{i}} \cup \{\boldsymbol{B_{ex}}\}_{\boldsymbol{j}}} \quad (23)$$

- Finally, semantics of part Sxj of semantics-aware design can be assigned to part Si of non-semantics aware design if $\mathbf{com}^{\boldsymbol{b}}(\{\boldsymbol{B_e}\}_{\boldsymbol{i}}, \{\boldsymbol{B_{ex}}\}_{\boldsymbol{j}})$ is highest in the combination.

$$\forall \boldsymbol{S_i} \in \mathbf{D};\ \forall \boldsymbol{S_{xj}} \in \mathbf{Dx}, \forall \boldsymbol{S_{xj`}} \in \mathbf{Dx};$$

$$\mathbf{com}^{\boldsymbol{b}}(\{\boldsymbol{B_e}\}_{\boldsymbol{i}}, \{\boldsymbol{B_{ex}}\}_{\boldsymbol{j}}) > \mathbf{com}^{\boldsymbol{b}}(\{\boldsymbol{B_e}\}_{\boldsymbol{i}}, \{\boldsymbol{B_{ex}}\}_{\boldsymbol{j`}}) \ \mathbf{where\ j} \neq \mathbf{j`} \quad (24)$$

In order to explain the algorithm with a working example, the non-semantics aware vise model and semantics aware template design, shown in Figure 26, are used. In the following example, the structural and expected behaviors derived from the designs are purely observational. Also as this algorithm is not implemented yet, and no Ontology is available, these structural and expected behaviors are compiled from common

knowledge and manual deduction. The following table lists expected behaviors of the components of the semantics-aware vise design.

Table 10: Expected behaviors derived from the semantics-aware vise design

Component	Expected Behaviors
Fixed Jaw	1. No DOF w.r.t any other parts 2. Should form joint/fixture with holder
Moving Jaw	1. Translational DOF w.r.t. holder and guide 2. No DOF w.r.t. guiding screw 3. Should form joint with guiding screw
Guiding Screw	1. Rotational DOF w.r.t. guide, moving jaw 2. No DOF w.r.t. handle 3. Rotation translate into translation of moving jaw
Holder	1. No DOF w.r.t. guide and fixed jaw 2. Forms joint or fixture with guide and fixed jaw
Handle	1. No DOF with guiding screw 2. Forms joint or fixture with guiding screw 3. Rotation translate into rotation of guiding screw
Guide	1. Rotational DOF w.r.t. guiding screw 2. No DOF w.r.t. holder 3. Form joint with guiding screw

Different components of the non-semantics aware vise design are marked in Figure 26. These components can be structurally analyzed and a set of structural behaviors can be extracted. These structural behaviors, when translated to expected

behavior, can be compared with the expected behaviors listed in the table below Figure 26 . For example,

- Component A is fixed with component F and has no DOF with respect to any other components. Therefore this component is most similar to the Fixed jaw.
- Component B can move in one direction only with respect to component C but it is fixed with Component D. If Component C is similar to Guide and Component D is similar to guiding screw, it can be concluded that Component B is most similar to the Moving Jaw.

Component D is fixed with component E, but can rotate. Also, this rotation causes component B to slide. This expected behavior is very unique and only the guiding screw has the similar behavior. Therefore, component D is similar to guiding screw.

Chapter VI Conclusion and Future Work

The purpose of this thesis is to investigate on semantic enabled manufacturing design data. It has been proceed that the significance or the purpose of the design is lost in the design data, as manufacturing design data doesn't contain semantics. In Chapter III, a novel algorithm is proposed to extract the translation properties from the pure geometrical data of the design. This translational property is one of the various functional properties, which can be extracted from the design data.

In Chapter IV a detailed investigation of the structural analysis that are performed on two non-semantics aware designs are presented. A novel algorithm, based on computational geometry is proposed, which used a sweep line to calculate the amount of clearance between different parts of the assembly design. These clearances are used to deduce the DOFs of individual parts of the design. This algorithm is implemented in Java and two demo design assemblies are exposed to it to demonstrate its capability. In chapter 5, results of the analysis are described. The following observations can be made from the structural analysis performed.

- Sweeping line algorithm, proposed here to detect DOFs of the parts in an assembly, is used to detect translational DOFs of the parts.
- When translational DOFs are represented in a truth table, various expected behaviors of the part can be analyzed. In Chapter IVB, various expected behaviors ought to be found in the truth tables are discussed. However, in reality, these expected behaviors should be interpreted by the Ontology.

- In these experiments, a .prt file containing the design data is used as a data source. This type of file can be directly opened in the Unigraphics NX CAD software. However, design data can be represented in both other application specific and neutral formats. Many different Schema translator may be developed for handling different data formats.
- Many other structural behaviors (few of them mentioned in Table 1) can be extracted from geometric data of any manufacturing design. Though the algorithm proposed, can only extract translational DOFs, these experiments successfully demonstrate that different computational geometrical methods can be used to extract structural behaviors from pure geometrical data. Further investigations on them are required to develop other algorithms such as,
    - Rotational sweep line algorithm can be extended from the translational sweep line algorithm to detect interference of different wheels in assemblies; such as, gears, pulley and cranks.
    - Bounded box algorithms can be used to detect the location and orientation of the part in the assembly.
    - Connected graphs with nodes as joints and arc as parts of the assembly can be used to analyze motion transfer and kinematic properties of the assembly.

The SIDOS system, proposed in Chapter V, is a dynamic system can be used to detect the semantic properties of a non-semantics aware part. One reason to make this system distributed by implementing Multi-agent system is that SIDOS is a multi-domain system. As explained earlier, the SIDOS system needs sophisticated algorithms from

computational geometry to extract structural behaviors from pure geometry, also requires domain specific knowledge to build the reference ontology. This multi-disciplinary expertise can be captured easily by a distributed system, since plugging in a new algorithm or different reference ontology or a new schema translator becomes easy. In this sense, the architecture of the SIDOS system proposed in this thesis is quite generic. Several extensions could be built in the future addressing areas of the system through detailed investigation. Some of them are listed below, which as a whole can define the future direction for the research on semantic integration of manufacturing design data.

- Due to the complexity of the development of the reference ontology, a product family with a specific lower level ontology should be built as a prototype. It will holistically test the efficacy of the proposed SIDOS system before diving into developing a complete upper and lower level ontological family. An existing product family, with a well-defined product specific manufacturing plan should be selected from any manufacturing industry. It will help to choose designs for testing SIDOS system and also investigate on the semantic integration of design data with other manufacturing processes. Therefore, collaboration with a semi-automated manufacturing company will be extremely beneficial in future research. On the other hand, domain specific knowledge should come from the industry expertise and should be modeled in the reference ontology by knowledgeable engineers.
- IMPlanner, developed by Sormaz et al. is a distributed manufacturing process planner which uses an object-oriented manufacturing process model for

integration [45]. Recently this system was upgraded to a Market-based coalition system by implementing JADE agent framework [46]. The object-oriented planning model was also converted to an upper level ontology by translating the Object-oriented model with java beans. This ontology is very similar to the Mason Ontology [44]. In future this upper level ontology can be used as a global reference to integrate product family specific lower level ontology, which is discussed in the last point. Furthermore, when this research will be incorporated in the multi-agent society of IMPlanner system, IMPlanner system can utilize the semantics embedded in the product design. It can decide on necessary operations and processing by analyzing the contextual information with help of the semantics embedded in the design data.

- Though SIDOS system is a continuous development, after the basic architecture is built and optimized, global body of standards such as STEPTOOLs Inc, W3 consortium, NIST, or ISO can envision the development of global ontology and domain specific upper ontologies as standard ontologies to be adopted by manufacturing industries. A global upper level ontology will open the door to the semantic integration of various processes of computer integrated manufacturing (CIM); such as, CAD, CAM, CAPP, CAQ, CAE, PPC, and even ERP. Use of semantics in CIM will facilitate seamless data integration among heterogeneous manufacturing processes. Even industries operating on different principles and standards will be able to exchange data without manual intervention. This will not only increase productivity and cut down time, and cost of new product

development but also make manufacturing processes much more intelligent. Seamless integration of data from various sources will help CIM to analyze large amount of data and take situation specific intelligent decision in process planning, job scheduling, quality control, safety and security. It will also provide accurate report and feedback on various operation specific and forecasting-specific measures. This way, semantic integration of manufacturing design data can transform the current computer-integrated manufacturing system into a truly automated system.

References

[1] J. Lee, K. Siau, and S. Hong, "Enterprise integration with ERP and EAI," *Commun. ACM*, vol. 46, no. 2, pp. 54–60, Feb. 2003.

[2] D. N. Šormaz, J. Arumugam, R. S. Harihara, C. Patel, and N. Neerukonda, "Integration of product design, process planning, scheduling, and FMS control using XML data representation," *Robot. Comput. Integr. Manuf.*, vol. 26, no. 6, pp. 583–595, Dec. 2010.

[3] J. S. Smith, "Survey on the use of simulation for manufacturing system design and operation," *J. Manuf. Syst.*, vol. 22, no. 2, pp. 157–171, Jan. 2003.

[4] S. Denning, "Transformational Leadership In Agile Manufacturing: Wikispeed," *Forbes*, 2012.

[5] M. D. Fortenbery, C. M. Stubbs, D. J. Payannet, and R. . Patience, "Ole for design and modeling," S Patent 6,198,487, 2001.

[6] S. Rahimić and E. Šunje, "Design of complex part in CAD-CAE-CAM systems using object oriented method," in *15th International Research/Expert Conference*, 2001, pp. 249–252.

[7] W. Eversheim and J. Schneewind, "Computer-aided process planning—State of the art and future development," *Robot. Comput. Integr. Manuf.*, vol. 10, no. 1–2, pp. 65–70, Jan. 1993.

[8] Unknown, "Semantics," *Wikipedia.org*, 2014. [Online]. Available: http://en.wikipedia.org/wiki/Semantics

[9] R. Barbau, S. Krima, S. Rachuri, A. Narayanan, X. Fiorentini, S. Foufou, and R. D. Sriram, "OntoSTEP: Enriching product model data using ontologies," *Comput. Des.*, vol. 44, no. 6, pp. 575–590, Jun. 2012.

[10] R. M. Thacker, R. E. King, and C. A. Ploskonka, *A new CIM model :a blueprint for a computer-integrated manufacturing enterprise*. Society of Manufacturing Engineers, Publications Development Dept., Reference Publications Division (Dearborn, Mich.), 1989, p. 92.

[11] C. M. Chituc and F. J. Restivo, "Challenges and trends in distributed manufacturing systems: are wise engineering systems the ultimate answer," *Second Int. Symp. Eng. Syst. MIT*, 2009.

[12] D. N. Sormaz, J. Arumugam, and C. Ganduri, *Process Planning and Scheduling for Distributed Manufacturing*. London: Springer London, 2007, pp. 61–90.

[13] S. Krima, B. Barbau, X. Fiorentini, R. Sudarsan, and R. Sriram, "OntoSTEP: OWL-DL Ontology for STEP." National Institute of Standard and Technology, 2009.

[14] D. Systems, "EXALEAD CloudView." [Online]. Available: http://www.3ds.com/products-services/exalead/products/exalead-cloudview/semantic-factory/

[15] S. B. Brunnermeier and S. A. Martin, "Interoperability costs in the US automotive supply chain," *Supply Chain Manag. An Int. J.*, vol. 7, no. 2, pp. 71–82, 2002.

[16] P. R. Wilson, "EXPRESS-I Language Reference Manual," 1994.

[17] iso.org,"ISO10303," *http://www.iso.org/iso/home/store/catalogue_tc/catalogue_detail.htm?csnumber=55257*.[Online].Available: http://www.iso.org/iso/home/store/catalogue_tc/catalogue_detail.htm?csnumber=55257. [Accessed: 01-Apr-2014].

[18] STEPTools, "STEP Application Protocols," *steptools.com*. [Online]. Available: http://www.steptools.com/library/standard/step_2.html. [Accessed: 01-Jun-2014].

[19] B. Babic, N. Nesic, and Z. Miljkovic, "A review of automated feature recognition with rule-based pattern recognition," *Comput. Ind.*, vol. 59, no. 4, pp. 321–337, Apr. 2008.

[20] O. Owodunni and S. Hinduja, "Evaluation of existing and new feature recognition algorithms: Part 1: theory and implementation," *Proc. Inst. Mech. Eng. Part B J. Eng. Manuf.*, vol. 216, no. 6, pp. 839–851, Jan. 2002.

[21] Y. Woo and H. Sakurai, "Recognition of maximal features by volume decomposition," *Comput. Des.*, vol. 34, no. 3, pp. 195–207, Mar. 2002.

[22] H. Sakurai, "Volume decomposition and feature recognition: part 1—polyhedral objects," *Comput. Des.*, vol. 27, no. 11, pp. 833–843, Nov. 1995.

[23] M. G. L. Sommerville, D. E. R. Clark, and J. R. Corney, "Viewer-centered geometric feature recognition," *J. Intell. Manuf.*, vol. 12, no. 4, pp. 359–375, 2001.

[24] P. Benkő, R. R. Martin, and T. Várady, "Algorithms for reverse engineering boundary representation models," *Comput. Des.*, vol. 33, no. 11, pp. 839–851, Sep. 2001.

[25] H. Fu, D. Cohen-Or, G. Dror, and A. Sheffer, “Upright orientation of man-made objects,” in *ACM SIGGRAPH 2008 papers on - SIGGRAPH ’08*, 2008, vol. 1, no. 212, p. 1.

[26] N. J. Mitra, L. J. Guibas, and M. Pauly, “Partial and approximate symmetry detection for 3D geometry,” in *ACM SIGGRAPH 2006 Papers on - SIGGRAPH ’06*, 2006, p. 560.

[27] K. Lee and G. Andrews, “Inference of the positions of components in an assembly: part 2,” *Comput. Des.*, vol. 17, no. 1, pp. 20–24, Jan. 1985.

[28] S. Kim and K. Lee, “An assembly modelling system for dynamic and kinematic analysis,” *Comput. Des.*, vol. 21, no. 1, pp. 2–12, Jan. 1989.

[29] N. J. Mitra, Y.-L. Yang, D.-M. Yan, W. Li, and M. Agrawala, “Illustrating how mechanical assemblies work,” *ACM Trans. Graph.*, vol. 29, no. 4, p. 1, Jul. 2010.

[30] M. P. Reddy, B. E. Prasad, P. G. Reddy, and A. Gupta, “A methodology for integration of heterogeneous databases,” *Knowl. Data ...*, vol. 6, no. 6, pp. 920–933, 1994.

[31] D. A. Koonce, “Manufacturing systems engineering and design: An intelligent, multi-model, integration architecture,” *Int. J. Comput. Integr. Manuf.*, vol. 9, no. 6, pp. 443–453, Jan. 1996.

[32] D. Dhamija, D. A. Koonce, and R. P. Judd, “Development of a unified data meta-model for CAD-CAPP-MRP-NC verification integration,” *Comput. Ind. Eng.*, vol. 33, no. 1–2, pp. 19–22, Oct. 1997.

[33] M. I. Shamos and D. Hoey, *Geometric intersection problems*, vol. 17. IEEE, 1976, pp. 208–215.

[34] M. Kintel and C. Wolf, "OpenSCAD." GNU General Public License, p. GNU General Public License, 2014.

[35] R. Shih, *Parametric Modeling with UGS NX5*. Schroff Development Corporation, 2007.

[36] J. Gero, "Design prototypes: a knowledge representation schema for design," *AI Mag.*, 1990.

[37] T. Adams, J. Dullea, P. Clark, S. Sripada, and T. Barrett, "Semantic Integration of Heterogeneous Information Sources Using a Knowledge-Based System," in *In Proc 5th Int Conf on CS and Informatics (CS&I'2000*, 2000.

[38] J. Euzenat and P. Shvaiko, *Ontology Matching*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2007.

[39] D. N. Sormaz, J. Arumugam, and S. Rajaraman, "Integrative process plan model and representation for intelligent distributed manufacturing planning," *International Journal of Production Research*, vol. 42, no. 17. pp. 3397–3417, 2004.

[40] D. Sormaz and B. Khoshnevis, "Process Planning Knowledge Representation using an Object-oriented Data Model," *Int. J. Comput. Integr. Manuf.*, vol. 10, no. 1–4, pp. 92–104, 1997.

[41] S. J. Fenves, S. Foufou, C. Bock, and R. D. Sriram, "CPM2: A Core Model for Product Data," *J. Comput. Inf. Sci. Eng.*, vol. 8, no. 1, p. 014501, 2008.

[42] S. J. Fenves, S. Foufou, C. Bock, and R. D. Sriram, "CPM2: A Core Model for Product Data," *J. Comput. Inf. Sci. Eng.*, vol. 8, no. 1, p. 014501, 2008.

[43] M. M. Baysal, U. Roy, R. Sudarsan, R. D. Sriram, and K. W. Lyons, "The Open Assembly Model for the Exchange of Assembly and Tolerance Information: Overview and Example," *Vol. 4 24th Comput. Inf. Eng. Conf.*, vol. 2004, pp. 759–770, 2004.

[44] S. Lemaignan, A. Siadat, J.-Y. Dantan, and A. Semenenko, "MASON: A Proposal For An Ontology Of Manufacturing Domain," in *IEEE Workshop on Distributed Intelligent Systems: Collective Intelligence and Its Applications (DIS'06)*, 2006, pp. 195–200.

[45] D. N. Sormaz and B. Khoshnevis, "Generation of alternative process plans in integrated manufacturing systems," *J. Intell. Manuf.*, vol. 14, no. 6, pp. 509–526, Dec. 2003.

[46] D. Sormaz and A. Sarkar, "Distributed Integration of Design and Planning Activities in Manufacturing using Intelligent Agents," in *TMCE*, 2014.

Appendix A: Technical Architecture of SIDOS System

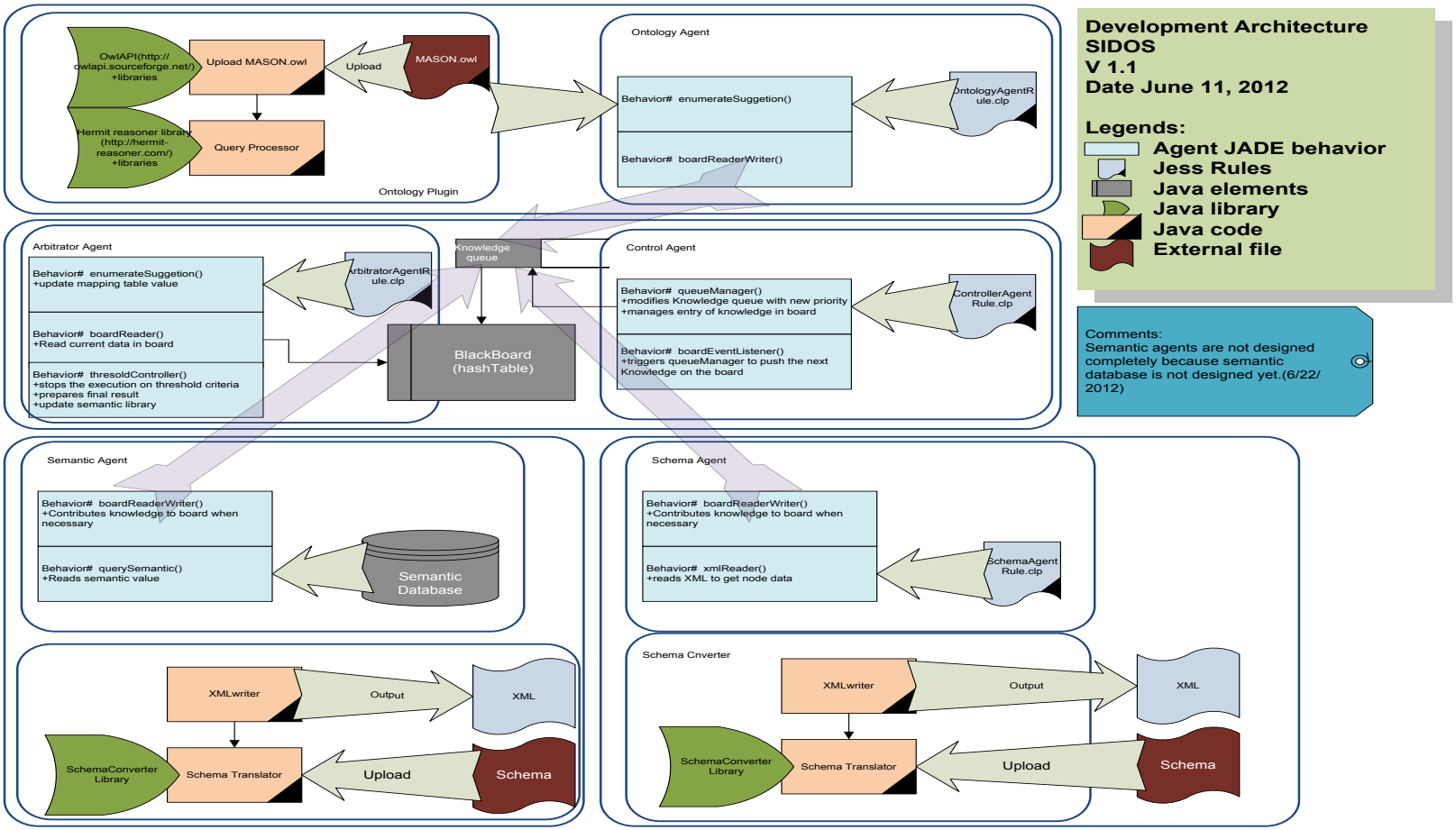


Figure 35: Multi-agent based blackboard architecture implemented by Jade framework for SIDOS system

Appendix B: Code Snippets

NXConnect.Java

```
public class NxConnect implements NXconnectInterface{

	private Session session;
	private static int BINDTIMEOUT = 0;
	Part workPart;

	public NxConnect() throws Exception {
		session = (Session) SessionFactory.get("Session");



	}

	public NxConnect(String serverName, String host, int port)
throws Exception{
		session = lookupServer(serverName, host,
port).session();
		PolicyFileLocator policyFile = new
PolicyFileLocator();
		System.setProperty("java.security.policy",
policyFile.getLocationOfPolicyFile(true));
	}

	public Session getSession() {
		return session;
	}

	public Part openPart(String filePath) throws
```

```
RemoteException, NXException{
            session.parts().openBaseDisplay(filePath);
            workPart = session.parts().work();
            return workPart;
      }


      public XPart getWorkPart() {
            return new XPart(workPart, null);
      }

      /** Looks up the server in the RMI registry */
      private    static    NXRemoteServer    lookupServer(String
serverName, String host, int port) throws Exception
      {
            NXRemoteServer server = null;
            Registry r = LocateRegistry.getRegistry(host);
            String name =  "//" +
                        host +//":"+ port +
                        "/"+serverName;
            System.out.println("Looking up name of server");
            int time = 0;
            if (System.getSecurityManager() == null) {
                  System.setSecurityManager(new
RMISecurityManager(){

                        @Override
                        public void checkConnect(String arg0, int
arg1, Object arg2) {
```

```
                    }

                    @Override
                    public void checkConnect(String arg0, int
arg1) {

                    }

               });
          }
          // Look up the server.  Keep trying until it is found
or
          // the amount of time we have tried exceeds the
amount specified
          //             in             the             property
nxexamples.remoting.rmilookuptimeout
          do
          {
               try
               {
                    server                                    =
(NXRemoteServer)Naming.lookup(name);
               }
               catch ( NotBoundException e )
               {
                    time += 1000;
                    if ( time > BINDTIMEOUT )
                         throw e;
```

```
                    Thread.sleep(1000);
                }
                catch ( ConnectException e )
                {
                    time += 1000;
                    if ( time > BINDTIMEOUT )
                        throw e;
                    Thread.sleep(1000);
                }

            }
            while(server == null);
            System.out.println("Name of server found");
            return server;
        }

        public boolean savePart(String path){
            try {
                PartSaveStatus status = workPart.saveAs(path);
                System.out.println(status);
                status.dispose();
            } catch (RemoteException | NXException e) {
                return false;
            }
            return true;
        }


        public static void main(String[] args) {
```

```
          ArrayList<Face> faceCol = new ArrayList<Face>();

          try {
                NxConnect connect = new NxConnect();

                //Get Part
                Session session = connect.getSession();

     session.parts().openBaseDisplay("C:\\Users\\sarkara1\\Drop
box\\Core
Java\\matrix2\\SIDOSProject\\workpart\\toycar_axle_mod1.prt");

                Part workPart = session.parts().work();
                System.out.println("Working       Part       "+
workPart.journalIdentifier());



                LineCollection lines = workPart.lines();
                LineCollection.Iterator lineitr;

                for(lineitr=lines.iterator();
lineitr.hasNext();){
                      Line line = (Line) lineitr.next();

     System.out.println(line.name()+"["+line.endPoint()+"]");
                }
```

```
                //get Assembly
                ComponentAssembly          assembly          =
workPart.componentAssembly();
                BodyCollection bodyList = workPart.bodies();
                BodyCollection.Iterator itr;
                for (itr = bodyList.iterator(); itr.hasNext();)
                {

                      Body body = (Body) itr.next();
                      System.out.println("\tBody: "+body);
                      Face faceArray[] = body.getFaces();
                      for         (int         inx=0;         inx
<(int)faceArray.length; ++inx)
                      {
                            Face face = faceArray[inx];
                            faceCol.add(face);
                            System.out.println("\t\tFace
:"+face);
                            System.out.println("\t\tFace    Type
:"+face.solidFaceType());


                            Edge[] edgeArray = face.getEdges();
                            for(int          inx1=0;          inx1
<edgeArray.length; inx1++){
                                  Edge edge = edgeArray[inx1];

                                  System.out.println("\t\t\tEdge
```

```
:"+edge);
                                    System.out.println("\t\t\tEdge
Type :"+edge.solidEdgeType());
                                    Edge.VerticesData   vertices   =
edge.getVertices();
                                    System.out.println("\t\t\t\tP1
:"+vertices.vertex1.x+","+

     vertices.vertex1.y+","+
                                                vertices.vertex1.z);
                                    System.out.println("\t\t\t\tP2
:"+vertices.vertex2.x+","+

     vertices.vertex2.y+","+
                                                vertices.vertex2.z);

                              }

                         }
                }


                //start building intersectionbuilder
                Face[] faces = new Face[faceCol.size()];
                for(int i=0; i<faceCol.size(); i++){
                     faces[i] = faceCol.get(i);
                }
                DynamicSectionBuilder     sectionBuilder     =
workPart.dynamicSections().createSectionBuilder(workPart.modeli
```

```
ngViews().workView());

                //set up section builder
                sectionBuilder.setSeriesSpacing(0.01);
                sectionBuilder.setOffset(20.88);
                NXObject object = sectionBuilder.commit();
                //              NXObject[]   commitedObjects   =
sectionBuilder.getCommittedObjects();
                sectionBuilder.destroy();
                PartSaveStatus             status             =
workPart.saveAs("C:\\Users\\sarkara1\\Dropbox\\Core
Java\\matrix2\\SIDOSProject\\workpart\\toycar_body_mod1.prt");
                System.out.println(status);
                status.dispose();

          } catch (NXException | RemoteException e) {
                // TODO Auto-generated catch block
                e.printStackTrace();
          } catch (Exception e) {
                // TODO Auto-generated catch block
                e.printStackTrace();
          }

     }

     public Part getPart() {
          // TODO Auto-generated method stub
          return workPart;
     }
```

```
}
```

SectionCurveGenerator.Java

```
public class SectionCurveGenerator {

	NXconnectInterface connect;
	XPart workPart;
	MethodType axis;
	public NXObject[] lines;

	public static void main(String[] args) {
		try {
			NxConnectAssembly	connect1	=	new
NxConnectAssembly();
			Part			part			=
connect1.openPart("C:\\Users\\sarkara1\\Dropbox\\Core
Java\\matrix2\\SIDOSProject\\parts\\holder_bar.prt");
			XPart xpart = null;
			for(XPart xp:connect1.getParts()){

	if(xp.getComponent().displayName().equals("holder1"))
					xpart = xp;
			}

			SectionCurveGenerator	curveGen	=	new
SectionCurveGenerator(connect1, xpart, MethodType.FIXED_Z);
```

```
                curveGen.performOperation(7);
                for(LineSegment
ls:xpart.getSectionPolygon(MethodType.FIXED_Z)){
                      System.out.println(ls);
                }
     connect1.savePart("C:\\Users\\sarkara1\\Dropbox\\Core
Java\\matrix2\\SIDOSProject\\workpart\\toycar_axle_mod1.prt");

           } catch (Exception e) {
                // TODO Auto-generated catch block
                e.printStackTrace();
           }
     }

     public  SectionCurveGenerator(NXconnectInterface  connect1,
XPart workPart, MethodType axis) throws Exception{
           connect = connect1;
           this.workPart = workPart;
           PartLoadStatus                status                =
connect1.getSession().parts().setWorkComponent(workPart.getComp
onent());
           this.axis = axis;
     }

     public boolean performOperation(double distance){
           Part part = workPart.getPart();
           try{
                System.out.println("Trying  to  intersect  part
"+workPart.getComponent().displayName()+"  at  "+distance+"  from
```

```
origin along "+axis.toString());
                DatumPlane datumPlane = createDatumPlane(part,
axis, distance);

                IntersectionCurveBuilder     curveBuilder     =
part.features().createIntersectionCurveBuilder(null);

     curveBuilder.curveFitData().setTolerance(0.0254);
                BodyCollection bodyList = part.bodies();
                BodyCollection.Iterator itr;
                SelectionIntentRule[]    intentRules    =    new
SelectionIntentRule[1] ;
                Face[] faceArray = null;
                Face[] boundaryFaces = null;
                for (itr = bodyList.iterator(); itr.hasNext();)
                {
                      Body body = (Body) itr.next();
                      faceArray = body.getFaces();
                }
                FaceDumbRule          facetangentRule          =
part.scRuleFactory().createRuleFaceDumb(faceArray);
                intentRules[0] = facetangentRule;

     curveBuilder.firstFace().replaceRules(intentRules, false);
                curveBuilder.firstSet().add(faceArray);

                curveBuilder.secondFace().replaceRules(new
SelectionIntentRule[]
```

```
     {part.scRuleFactory().createRuleFaceDatum(new
DatumPlane[]{datumPlane})}, false);
                curveBuilder.secondSet().add(datumPlane);
                IntersectionCurve      intersectionCurve      =
(IntersectionCurve) curveBuilder.commit();
                if(intersectionCurve != null){
                     workPart.setcSectionCurves(axis,
intersectionCurve.getEntities(), distance);
                     lines = intersectionCurve.getEntities();
                     return true;
                }
                curveBuilder.destroy();
           }
           catch (Exception e) {
                return false;
           }
           return true;
     }

     private     Part     getWorkPart(XPart     xPart)     throws
RemoteException, NXException{

     connect.getSession().parts().setWorkComponent(xPart.getCom
ponent());
           return connect.getSession().parts().work();
     }
```

```
     public DatumPlane createDatumPlane(Part workPart,
MethodType axis, double distance) throws RemoteException,
NXException{

          DatumPlaneBuilder planeBuilder =
workPart.features().createDatumPlaneBuilder(null);
          Plane plane = planeBuilder.getPlane();
          plane.setMethod(axis);
          if (axis == MethodType.FIXED_X)
               plane.setOrigin(new Point3d(distance, 0.0,
0.0));
          if (axis == MethodType.FIXED_Y)
               plane.setOrigin(new Point3d(0.0, distance,
0.0));
          if (axis == MethodType.FIXED_Z)
               plane.setOrigin(new Point3d(0.0, 0.0,
distance));

          plane.setAlternate(PlaneTypes.AlternateType.ONE);
          DatumPlaneFeature feature = (DatumPlaneFeature)
planeBuilder.commitFeature();
          DatumPlane datumPlane = feature.datumPlane();
          planeBuilder.destroy();
          return datumPlane;
     }

}
```

Joint1_Plate_X.vb.Java

```
' NX 8.0.0.25
' Journal created by sarkara1 on Sun Apr 20 19:29:50 2014
Eastern Daylight Time
'
Option Strict Off
Imports System
Imports NXOpen

Module NXJournal
Sub Main

Dim theSession As Session = Session.GetSession()
Dim workPart As Part = theSession.Parts.Work

Dim displayPart As Part = theSession.Parts.Display

' ----------------------------------------------
'   Menu: Insert->Curve from Bodies->Intersect...
' ----------------------------------------------
Dim markId1 As Session.UndoMarkId
markId1                                                              =
theSession.SetUndoMark(Session.MarkVisibility.Visible, "Start")

Dim nullFeatures_Feature As Features.Feature = Nothing



If Not workPart.Preferences.Modeling.GetHistoryMode Then
    Throw(New Exception("Create or edit of a Feature was
recorded in History Mode but playback is in History-Free
```

```
Mode."))
End If

Dim                 intersectionCurveBuilder1                 As
Features.IntersectionCurveBuilder
intersectionCurveBuilder1                                      =
workPart.Features.CreateIntersectionCurveBuilder(nullFeatures_F
eature)

Dim origin1 As Point3d = New Point3d(0.0, 0.0, 0.0)
Dim normal1 As Vector3d = New Vector3d(0.0, 0.0, 1.0)
Dim plane1 As Plane
plane1    =    workPart.Planes.CreatePlane(origin1,    normal1,
SmartObject.UpdateOption.WithinModeling)

Dim            unit1            As            Unit            =
CType(workPart.UnitCollection.FindObject("MilliMeter"), Unit)

Dim expression1 As Expression
expression1                                                    =
workPart.Expressions.CreateSystemExpressionWithUnits("0",
unit1)

Dim expression2 As Expression
expression2                                                    =
workPart.Expressions.CreateSystemExpressionWithUnits("0",
unit1)

Dim expression3 As Expression
```

```
expression3                                                    =
workPart.Expressions.CreateSystemExpressionWithUnits("0",
unit1)

Dim expression4 As Expression
expression4                                                    =
workPart.Expressions.CreateSystemExpressionWithUnits("0",
unit1)

Dim expression5 As Expression
expression5                                                    =
workPart.Expressions.CreateSystemExpressionWithUnits("0",
unit1)

Dim origin2 As Point3d = New Point3d(0.0, 0.0, 0.0)
Dim normal2 As Vector3d = New Vector3d(0.0, 0.0, 1.0)
Dim plane2 As Plane
plane2    =    workPart.Planes.CreatePlane(origin2,    normal2,
SmartObject.UpdateOption.WithinModeling)

Dim expression6 As Expression
expression6                                                    =
workPart.Expressions.CreateSystemExpressionWithUnits("0",
unit1)

Dim expression7 As Expression
expression7                                                    =
workPart.Expressions.CreateSystemExpressionWithUnits("0",
unit1)
```

```
Dim expression8 As Expression
expression8                                                    =
workPart.Expressions.CreateSystemExpressionWithUnits("0",
unit1)

Dim expression9 As Expression
expression9                                                    =
workPart.Expressions.CreateSystemExpressionWithUnits("0",
unit1)

Dim expression10 As Expression
expression10                                                   =
workPart.Expressions.CreateSystemExpressionWithUnits("0",
unit1)

intersectionCurveBuilder1.CurveFitData.Tolerance = 0.0254

theSession.SetUndoMarkName(markId1,     "Intersection     Curve
Dialog")

plane2.SetMethod(PlaneTypes.MethodType.FixedZ)

Dim geom1(-1) As NXObject
plane2.SetGeometry(geom1)

Dim origin3 As Point3d = New Point3d(-50.0, 16.0, 75.0)
plane2.Origin = origin3
```

```
Dim matrix1 As Matrix3x3
matrix1.Xx = 1.0
matrix1.Xy = 0.0
matrix1.Xz = 0.0
matrix1.Yx = 0.0
matrix1.Yy = 0.0
matrix1.Yz = -1.0
matrix1.Zx = 0.0
matrix1.Zy = 1.0
matrix1.Zz = 0.0
plane2.Matrix = matrix1

plane2.SetAlternate(PlaneTypes.AlternateType.One)

plane2.Evaluate()

plane2.SetMethod(PlaneTypes.MethodType.FixedZ)

Dim faces1(23) As Face
Dim          brep1          As          Features.Brep          =
CType(workPart.Features.FindObject("UNPARAMETERIZED_FEATURE(1)"
), Features.Brep)

Dim   face1   As   Face   =   CType(brep1.FindObject("FACE   8
{(50.0000000000001,12.5,0) UNPARAMETERIZED_FEATURE(1)}"), Face)

faces1(0) = face1
Dim   face2   As   Face   =   CType(brep1.FindObject("FACE   15
{(5.0000000000001,6.25,-50)       UNPARAMETERIZED_FEATURE(1)}"),
```

```
Face)

faces1(1) = face2
Dim face3 As Face = CType(brep1.FindObject("FACE 10
{(0,6.25,45.0000000000001) UNPARAMETERIZED_FEATURE(1)}"), Face)

faces1(2) = face3
Dim face4 As Face = CType(brep1.FindObject("FACE 19
{(12.5000000000001,18.75,-50) UNPARAMETERIZED_FEATURE(1)}"),
Face)

faces1(3) = face4
Dim face5 As Face = CType(brep1.FindObject("FACE 14 {(0,6.25,-
45.0000000000001) UNPARAMETERIZED_FEATURE(1)}"), Face)

faces1(4) = face5
Dim face6 As Face = CType(brep1.FindObject("FACE 4 {(0,0,0)
UNPARAMETERIZED_FEATURE(1)}"), Face)

faces1(5) = face6
Dim face7 As Face = CType(brep1.FindObject("FACE 12
{(0,6.25,55) UNPARAMETERIZED_FEATURE(1)}"), Face)

faces1(6) = face7
Dim face8 As Face = CType(brep1.FindObject("FACE 20
{(0.0000000000001,18.75,-62.5) UNPARAMETERIZED_FEATURE(1)}"),
Face)

faces1(7) = face8
```

```
Dim face9 As Face = CType(brep1.FindObject("FACE 22
{(0.0000000000001,18.75,37.5) UNPARAMETERIZED_FEATURE(1)}"),
Face)

faces1(8) = face9
Dim face10 As Face = CType(brep1.FindObject("FACE 24
{(0.0000000000001,18.75,62.5) UNPARAMETERIZED_FEATURE(1)}"),
Face)

faces1(9) = face10
Dim face11 As Face = CType(brep1.FindObject("FACE 7
{(0.0000000000001,12.5,-75) UNPARAMETERIZED_FEATURE(1)}"),
Face)

faces1(10) = face11
Dim face12 As Face = CType(brep1.FindObject("FACE 1 {(0,25,0)
UNPARAMETERIZED_FEATURE(1)}"), Face)

faces1(11) = face12
Dim face13 As Face = CType(brep1.FindObject("FACE 2
{(0.0000000000001,12.5,-50) UNPARAMETERIZED_FEATURE(1)}"),
Face)

faces1(12) = face13
Dim face14 As Face = CType(brep1.FindObject("FACE 16 {(0,6.25,-
55) UNPARAMETERIZED_FEATURE(1)}"), Face)

faces1(13) = face14
Dim face15 As Face = CType(brep1.FindObject("FACE 6 {(-
```

```
50,12.5,0) UNPARAMETERIZED_FEATURE(1)}"), Face)

faces1(14) = face15
Dim face16 As Face = CType(brep1.FindObject("FACE 13 {(-
5,6.25,-50.0000000000001) UNPARAMETERIZED_FEATURE(1)}"), Face)

faces1(15) = face16
Dim face17 As Face = CType(brep1.FindObject("FACE 5
{(0,12.5,75) UNPARAMETERIZED_FEATURE(1)}"), Face)

faces1(16) = face17
Dim face18 As Face = CType(brep1.FindObject("FACE 17 {(-
12.4999999999999,18.75,-50)       UNPARAMETERIZED_FEATURE(1)}"),
Face)

faces1(17) = face18
Dim face19 As Face = CType(brep1.FindObject("FACE 3
{(0.0000000000001,12.5,50) UNPARAMETERIZED_FEATURE(1)}"), Face)

faces1(18) = face19
Dim face20 As Face = CType(brep1.FindObject("FACE 18
{(0.0000000000001,18.75,-37.5)    UNPARAMETERIZED_FEATURE(1)}"),
Face)

faces1(19) = face20
Dim face21 As Face = CType(brep1.FindObject("FACE 9 {(-
5,6.25,50.0000000000001) UNPARAMETERIZED_FEATURE(1)}"), Face)

faces1(20) = face21
```

```
Dim face22 As Face = CType(brep1.FindObject("FACE 11
{(5.0000000000001,6.25,50) UNPARAMETERIZED_FEATURE(1)}"), Face)

faces1(21) = face22
Dim face23 As Face = CType(brep1.FindObject("FACE 21 {(-
12.4999999999999,18.75,50) UNPARAMETERIZED_FEATURE(1)}"), Face)

faces1(22) = face23
Dim face24 As Face = CType(brep1.FindObject("FACE 23
{(12.5000000000001,18.75,50)      UNPARAMETERIZED_FEATURE(1)}"),
Face)

faces1(23) = face24
Dim faceDumbRule1 As FaceDumbRule
faceDumbRule1                                                    =
workPart.ScRuleFactory.CreateRuleFaceDumb(faces1)

Dim rules1(0) As SelectionIntentRule
rules1(0) = faceDumbRule1
intersectionCurveBuilder1.FirstFace.ReplaceRules(rules1, False)

Dim objects1(23) As TaggedObject
objects1(0) = face1
objects1(1) = face2
objects1(2) = face3
objects1(3) = face4
objects1(4) = face5
objects1(5) = face6
objects1(6) = face7
```

```
objects1(7) = face8
objects1(8) = face9
objects1(9) = face10
objects1(10) = face11
objects1(11) = face12
objects1(12) = face13
objects1(13) = face14
objects1(14) = face15
objects1(15) = face16
objects1(16) = face17
objects1(17) = face18
objects1(18) = face19
objects1(19) = face20
objects1(20) = face21
objects1(21) = face22
objects1(22) = face23
objects1(23) = face24
Dim added1 As Boolean
added1 = intersectionCurveBuilder1.FirstSet.Add(objects1)

Dim geom2(-1) As NXObject
plane2.SetGeometry(geom2)

plane2.SetMethod(PlaneTypes.MethodType.FixedX)

Dim geom3(-1) As NXObject
plane2.SetGeometry(geom3)

Dim origin4 As Point3d = New Point3d(0.0, 0.0, 75.0)
```

```
plane2.Origin = origin4

Dim matrix2 As Matrix3x3
matrix2.Xx = 0.0
matrix2.Xy = 0.0
matrix2.Xz = -1.0
matrix2.Yx = 0.0
matrix2.Yy = 1.0
matrix2.Yz = 0.0
matrix2.Zx = 1.0
matrix2.Zy = -0.0
matrix2.Zz = 0.0
plane2.Matrix = matrix2

plane2.SetAlternate(PlaneTypes.AlternateType.One)

plane2.Evaluate()

plane2.SetMethod(PlaneTypes.MethodType.FixedX)

intersectionCurveBuilder1.SecondPlane = plane2

Dim markId2 As Session.UndoMarkId
markId2                                                              =
theSession.SetUndoMark(Session.MarkVisibility.Invisible,
"Intersection Curve")

Dim curve As Features.IntersectionCurve
        curve    =    CType(intersectionCurveBuilder1.Commit(),
```

```
Features.IntersectionCurve)

        Dim lines As NXObject() = curve.GetEntities()
        Dim s As String = ""
        If lines.Length > 0 Then
            For Each line As NXObject In lines
                If TypeOf line Is Line Then
                    Dim l As Line = CType(line, Line)
                    s = s + l.StartPoint.x.ToString() + "," +
l.StartPoint.y.ToString() + "," + l.StartPoint.z.ToString() +
"," + l.EndPoint.x.ToString() + "," + l.EndPoint.y.ToString() +
"," + l.EndPoint.z.ToString() + Environment.NewLine
                End If
            Next

My.Computer.FileSystem.WriteAllText("C:\Users\sarkara1\Document
s\test.csv", s, True)
        Else

My.Computer.FileSystem.WriteAllText("C:\Users\sarkara1\Document
s\test.txt", "No Intersection Curve Found", True)
        End If

theSession.DeleteUndoMark(markId2, Nothing)

theSession.SetUndoMarkName(markId1, "Intersection Curve")

intersectionCurveBuilder1.Destroy()
```

```
Try
  ' Expression is still in use.
  workPart.Expressions.Delete(expression4)
Catch ex As NXException
  ex.AssertErrorCode(1050029)
End Try

Try
  ' Expression is still in use.
  workPart.Expressions.Delete(expression9)
Catch ex As NXException
  ex.AssertErrorCode(1050029)
End Try

Try
  ' Expression is still in use.
  workPart.Expressions.Delete(expression1)
Catch ex As NXException
  ex.AssertErrorCode(1050029)
End Try

plane1.DestroyPlane()

Try
  ' Expression is still in use.
  workPart.Expressions.Delete(expression6)
Catch ex As NXException
  ex.AssertErrorCode(1050029)
End Try
```

```
workPart.Expressions.Delete(expression3)

workPart.Expressions.Delete(expression5)

workPart.Expressions.Delete(expression8)

workPart.Expressions.Delete(expression10)

workPart.Expressions.Delete(expression2)

workPart.Expressions.Delete(expression7)

Dim markId3 As Session.UndoMarkId
markId3                                                    =
theSession.SetUndoMark(Session.MarkVisibility.Visible, "Start")

Dim                 intersectionCurveBuilder2                 As
Features.IntersectionCurveBuilder
intersectionCurveBuilder2                                   =
workPart.Features.CreateIntersectionCurveBuilder(nullFeatures_F
eature)

Dim origin5 As Point3d = New Point3d(0.0, 0.0, 0.0)
Dim normal3 As Vector3d = New Vector3d(0.0, 0.0, 1.0)
Dim plane3 As Plane
plane3    =    workPart.Planes.CreatePlane(origin5,    normal3,
SmartObject.UpdateOption.WithinModeling)
```

```
Dim expression11 As Expression
expression11                                                    =
workPart.Expressions.CreateSystemExpressionWithUnits("0",
unit1)

Dim expression12 As Expression
expression12                                                    =
workPart.Expressions.CreateSystemExpressionWithUnits("0",
unit1)

Dim expression13 As Expression
expression13                                                    =
workPart.Expressions.CreateSystemExpressionWithUnits("0",
unit1)

Dim expression14 As Expression
expression14                                                    =
workPart.Expressions.CreateSystemExpressionWithUnits("0",
unit1)

Dim expression15 As Expression
expression15                                                    =
workPart.Expressions.CreateSystemExpressionWithUnits("0",
unit1)

Dim origin6 As Point3d = New Point3d(0.0, 0.0, 0.0)
Dim normal4 As Vector3d = New Vector3d(0.0, 0.0, 1.0)
Dim plane4 As Plane
plane4    =    workPart.Planes.CreatePlane(origin6,    normal4,
```

```
SmartObject.UpdateOption.WithinModeling)

Dim expression16 As Expression
expression16                                                    =
workPart.Expressions.CreateSystemExpressionWithUnits("0",
unit1)

Dim expression17 As Expression
expression17                                                    =
workPart.Expressions.CreateSystemExpressionWithUnits("0",
unit1)

Dim expression18 As Expression
expression18                                                    =
workPart.Expressions.CreateSystemExpressionWithUnits("0",
unit1)

Dim expression19 As Expression
expression19                                                    =
workPart.Expressions.CreateSystemExpressionWithUnits("0",
unit1)

Dim expression20 As Expression
expression20                                                    =
workPart.Expressions.CreateSystemExpressionWithUnits("0",
unit1)

intersectionCurveBuilder2.CurveFitData.Tolerance = 0.0254
```

```
theSession.SetUndoMarkName(markId3, "Intersection Curve
Dialog")

plane4.SetMethod(PlaneTypes.MethodType.FixedX)

Dim geom4(-1) As NXObject
plane4.SetGeometry(geom4)

Dim origin7 As Point3d = New Point3d(0.0, 0.0, 75.0)
plane4.Origin = origin7

Dim matrix3 As Matrix3x3
matrix3.Xx = 0.0
matrix3.Xy = 0.0
matrix3.Xz = -1.0
matrix3.Yx = 0.0
matrix3.Yy = 1.0
matrix3.Yz = 0.0
matrix3.Zx = 1.0
matrix3.Zy = -0.0
matrix3.Zz = 0.0
plane4.Matrix = matrix3

plane4.SetAlternate(PlaneTypes.AlternateType.One)

plane4.Evaluate()

plane4.SetMethod(PlaneTypes.MethodType.FixedX)
```

```
' ------------------------------------------------
'   Dialog Begin Intersection Curve
' ------------------------------------------------
intersectionCurveBuilder2.Destroy()

Try
  ' Expression is still in use.
  workPart.Expressions.Delete(expression14)
Catch ex As NXException
  ex.AssertErrorCode(1050029)
End Try

Try
  ' Expression is still in use.
  workPart.Expressions.Delete(expression19)
Catch ex As NXException
  ex.AssertErrorCode(1050029)
End Try

Try
  ' Expression is still in use.
  workPart.Expressions.Delete(expression11)
Catch ex As NXException
  ex.AssertErrorCode(1050029)
End Try

plane3.DestroyPlane()

Try
```

```
  ' Expression is still in use.
  workPart.Expressions.Delete(expression16)
Catch ex As NXException
  ex.AssertErrorCode(1050029)
End Try

plane4.DestroyPlane()

workPart.Expressions.Delete(expression13)

workPart.Expressions.Delete(expression15)

workPart.Expressions.Delete(expression18)

workPart.Expressions.Delete(expression20)

workPart.Expressions.Delete(expression12)

workPart.Expressions.Delete(expression17)

theSession.UndoToMark(markId3, Nothing)

theSession.DeleteUndoMark(markId3, Nothing)

' ------------------------------------------------
'   Menu: Tools->Journal->Stop Recording
' ------------------------------------------------

End Sub
```

```
End Module
```

Appendix C: Testing Assemblies Part214 File

BlockSlide.stp

```
ISO-10303-21;
HEADER;
/* Generated by software containing ST-Developer
 * from STEP Tools, Inc. (www.steptools.com)
 */
/* OPTION: using custom schema-name function */

FILE_DESCRIPTION(
/* description */ (''),
/* implementation_level */ '2;1');

FILE_NAME(
/* name */ 'blockslide3.stp',
/* time_stamp */ '2014-04-22T15:59:50-04:00',
/* author */ (''),
/* organization */ (''),
/* preprocessor_version */ 'ST-DEVELOPER v14',
/* originating_system */ 'SIEMENS PLM Software NX 8.0',
/* authorisation */ '');

FILE_SCHEMA (('AUTOMOTIVE_DESIGN { 1 0 10303 214 3 1 1 1 }'));
ENDSEC;

DATA;
#10=PROPERTY_DEFINITION_REPRESENTATION(#14,#12);
#11=PROPERTY_DEFINITION_REPRESENTATION(#15,#13);
```

```
#12=REPRESENTATION('',(#16),#567);
#13=REPRESENTATION('',(#17),#567);
#14=PROPERTY_DEFINITION('pmi validation property','',#31);
#15=PROPERTY_DEFINITION('pmi validation property','',#31);
#16=VALUE_REPRESENTATION_ITEM('number of
annotations',COUNT_MEASURE(0.));
#17=VALUE_REPRESENTATION_ITEM('number of
views',COUNT_MEASURE(0.));
#18=CONTEXT_DEPENDENT_SHAPE_REPRESENTATION(#22,#32);
#19=CONTEXT_DEPENDENT_SHAPE_REPRESENTATION(#23,#33);
#20=NEXT_ASSEMBLY_USAGE_OCCURRENCE('1001','
','BLOCK',#36,#35,$);
#21=NEXT_ASSEMBLY_USAGE_OCCURRENCE('1002','
','SLIDE',#36,#34,$);
#22=(
REPRESENTATION_RELATIONSHIP(' ',' ',#55,#56)
REPRESENTATION_RELATIONSHIP_WITH_TRANSFORMATION(#24)
SHAPE_REPRESENTATION_RELATIONSHIP()
);
#23=(
REPRESENTATION_RELATIONSHIP(' ',' ',#54,#56)
REPRESENTATION_RELATIONSHIP_WITH_TRANSFORMATION(#25)
SHAPE_REPRESENTATION_RELATIONSHIP()
);
#24=ITEM_DEFINED_TRANSFORMATION(' ',' ',#380,#398);
#25=ITEM_DEFINED_TRANSFORMATION(' ',' ',#380,#399);
#26=SHAPE_DEFINITION_REPRESENTATION(#29,#54);
#27=SHAPE_DEFINITION_REPRESENTATION(#30,#55);
#28=SHAPE_DEFINITION_REPRESENTATION(#31,#56);
```

```
#29=PRODUCT_DEFINITION_SHAPE('','',#34);
#30=PRODUCT_DEFINITION_SHAPE('','',#35);
#31=PRODUCT_DEFINITION_SHAPE('','',#36);
#32=PRODUCT_DEFINITION_SHAPE(' ','NAUO PRDDFN',#20);
#33=PRODUCT_DEFINITION_SHAPE(' ','NAUO PRDDFN',#21);
#34=PRODUCT_DEFINITION(' ','',#40,#37);
#35=PRODUCT_DEFINITION(' ','',#41,#38);
#36=PRODUCT_DEFINITION(' ','',#42,#39);
#37=PRODUCT_DEFINITION_CONTEXT('part definition',#53,'design');
#38=PRODUCT_DEFINITION_CONTEXT('part definition',#53,'design');
#39=PRODUCT_DEFINITION_CONTEXT('part definition',#53,'design');
#40=PRODUCT_DEFINITION_FORMATION_WITH_SPECIFIED_SOURCE(
' ',' ',#46,
 .NOT_KNOWN.);
#41=PRODUCT_DEFINITION_FORMATION_WITH_SPECIFIED_SOURCE(
' ',' ',#47,
 .NOT_KNOWN.);
#42=PRODUCT_DEFINITION_FORMATION_WITH_SPECIFIED_SOURCE(
' ',' ',#48,
 .NOT_KNOWN.);
#43=PRODUCT_RELATED_PRODUCT_CATEGORY('part','',(#46));
#44=PRODUCT_RELATED_PRODUCT_CATEGORY('part','',(#47));
#45=PRODUCT_RELATED_PRODUCT_CATEGORY('part','',(#48));
#46=PRODUCT('slide','slide',' ',(#49));
#47=PRODUCT('block','block',' ',(#50));
#48=PRODUCT('blockslide','blockslide',' ',(#51));
#49=PRODUCT_CONTEXT(' ',#53,'mechanical');
#50=PRODUCT_CONTEXT(' ',#53,'mechanical');
#51=PRODUCT_CONTEXT(' ',#53,'mechanical');
```

```
#52=APPLICATION_PROTOCOL_DEFINITION('international standard',
'automotive_design',2010,#53);
#53=APPLICATION_CONTEXT(
'core data for automotive mechanical design processes');
#54=SHAPE_REPRESENTATION('slide50-None',(#380,#73,#57),#569);
#55=SHAPE_REPRESENTATION('block51-None',(#380,#72),#568);
#56=SHAPE_REPRESENTATION('blockslide3-
None',(#380,#398,#399,#381),#567);
#57=GEOMETRIC_CURVE_SET('None',(#58,#59,#60,#61));
#58=TRIMMED_CURVE('',#296,(PARAMETER_VALUE(0.)),(PARAMETER_
VALUE(1.)),.T.,
 .PARAMETER.);
#59=TRIMMED_CURVE('',#297,(PARAMETER_VALUE(0.)),(PARAMETER_
VALUE(1.)),.T.,
 .PARAMETER.);
#60=TRIMMED_CURVE('',#298,(PARAMETER_VALUE(0.)),(PARAMETER_
VALUE(1.)),.T.,
 .PARAMETER.);
#61=TRIMMED_CURVE('',#299,(PARAMETER_VALUE(0.)),(PARAMETER_
VALUE(1.)),.T.,
 .PARAMETER.);
#62=SURFACE_STYLE_USAGE(.BOTH.,#64);
#63=SURFACE_STYLE_USAGE(.BOTH.,#65);
#64=SURFACE_SIDE_STYLE('',(#66));
#65=SURFACE_SIDE_STYLE('',(#67));
#66=SURFACE_STYLE_FILL_AREA(#68);
#67=SURFACE_STYLE_FILL_AREA(#69);
#68=FILL_AREA_STYLE('',(#70));
#69=FILL_AREA_STYLE('',(#71));
```

```
#70=FILL_AREA_STYLE_COLOUR('',#373);
#71=FILL_AREA_STYLE_COLOUR('',#373);
#72=MANIFOLD_SOLID_BREP('',#74);
#73=MANIFOLD_SOLID_BREP('',#75);
#74=CLOSED_SHELL('',(#76,#77,#78,#79,#80,#81,#82,#83,#84,#85));
#75=CLOSED_SHELL('',(#86,#87,#88,#89,#90,#91));
#76=ADVANCED_FACE('',(#108),#92,.F.);
#77=ADVANCED_FACE('',(#109),#93,.F.);
#78=ADVANCED_FACE('',(#110),#94,.F.);
#79=ADVANCED_FACE('',(#111),#95,.F.);
#80=ADVANCED_FACE('',(#112),#96,.F.);
#81=ADVANCED_FACE('',(#113),#97,.F.);
#82=ADVANCED_FACE('',(#114),#98,.T.);
#83=ADVANCED_FACE('',(#115),#99,.F.);
#84=ADVANCED_FACE('',(#116),#100,.F.);
#85=ADVANCED_FACE('',(#117),#101,.T.);
#86=ADVANCED_FACE('',(#118),#102,.T.);
#87=ADVANCED_FACE('',(#119),#103,.T.);
#88=ADVANCED_FACE('',(#120),#104,.T.);
#89=ADVANCED_FACE('',(#121),#105,.T.);
#90=ADVANCED_FACE('',(#122),#106,.T.);
#91=ADVANCED_FACE('',(#123),#107,.T.);
#92=PLANE('',#382);
#93=PLANE('',#383);
#94=PLANE('',#384);
#95=PLANE('',#385);
#96=PLANE('',#386);
#97=PLANE('',#387);
#98=PLANE('',#388);
```

```
#99=PLANE('',#389);
#100=PLANE('',#390);
#101=PLANE('',#391);
#102=PLANE('',#392);
#103=PLANE('',#393);
#104=PLANE('',#394);
#105=PLANE('',#395);
#106=PLANE('',#396);
#107=PLANE('',#397);
#108=FACE_OUTER_BOUND('',#124,.T.);
#109=FACE_OUTER_BOUND('',#125,.T.);
#110=FACE_OUTER_BOUND('',#126,.T.);
#111=FACE_OUTER_BOUND('',#127,.T.);
#112=FACE_OUTER_BOUND('',#128,.T.);
#113=FACE_OUTER_BOUND('',#129,.T.);
#114=FACE_OUTER_BOUND('',#130,.T.);
#115=FACE_OUTER_BOUND('',#131,.T.);
#116=FACE_OUTER_BOUND('',#132,.T.);
#117=FACE_OUTER_BOUND('',#133,.T.);
#118=FACE_OUTER_BOUND('',#134,.T.);
#119=FACE_OUTER_BOUND('',#135,.T.);
#120=FACE_OUTER_BOUND('',#136,.T.);
#121=FACE_OUTER_BOUND('',#137,.T.);
#122=FACE_OUTER_BOUND('',#138,.T.);
#123=FACE_OUTER_BOUND('',#139,.T.);
#124=EDGE_LOOP('',(#140,#141,#142,#143));
#125=EDGE_LOOP('',(#144,#145,#146,#147));
#126=EDGE_LOOP('',(#148,#149,#150,#151));
#127=EDGE_LOOP('',(#152,#153,#154,#155));
```

```
#128=EDGE_LOOP('',(#156,#157,#158,#159));
#129=EDGE_LOOP('',(#160,#161,#162,#163,#164,#165,#166,#167));
#130=EDGE_LOOP('',(#168,#169,#170,#171,#172,#173,#174,#175));
#131=EDGE_LOOP('',(#176,#177,#178,#179));
#132=EDGE_LOOP('',(#180,#181,#182,#183));
#133=EDGE_LOOP('',(#184,#185,#186,#187));
#134=EDGE_LOOP('',(#188,#189,#190,#191));
#135=EDGE_LOOP('',(#192,#193,#194,#195));
#136=EDGE_LOOP('',(#196,#197,#198,#199));
#137=EDGE_LOOP('',(#200,#201,#202,#203));
#138=EDGE_LOOP('',(#204,#205,#206,#207));
#139=EDGE_LOOP('',(#208,#209,#210,#211));
#140=ORIENTED_EDGE('',*,*,#236,.T.);
#141=ORIENTED_EDGE('',*,*,#237,.T.);
#142=ORIENTED_EDGE('',*,*,#238,.T.);
#143=ORIENTED_EDGE('',*,*,#239,.T.);
#144=ORIENTED_EDGE('',*,*,#240,.T.);
#145=ORIENTED_EDGE('',*,*,#241,.T.);
#146=ORIENTED_EDGE('',*,*,#242,.T.);
#147=ORIENTED_EDGE('',*,*,#243,.T.);
#148=ORIENTED_EDGE('',*,*,#243,.F.);
#149=ORIENTED_EDGE('',*,*,#244,.T.);
#150=ORIENTED_EDGE('',*,*,#239,.F.);
#151=ORIENTED_EDGE('',*,*,#245,.T.);
#152=ORIENTED_EDGE('',*,*,#246,.T.);
#153=ORIENTED_EDGE('',*,*,#247,.T.);
#154=ORIENTED_EDGE('',*,*,#248,.T.);
#155=ORIENTED_EDGE('',*,*,#249,.T.);
#156=ORIENTED_EDGE('',*,*,#250,.T.);
```

```
#157=ORIENTED_EDGE('',*,*,#251,.T.);
#158=ORIENTED_EDGE('',*,*,#252,.T.);
#159=ORIENTED_EDGE('',*,*,#253,.T.);
#160=ORIENTED_EDGE('',*,*,#248,.F.);
#161=ORIENTED_EDGE('',*,*,#254,.T.);
#162=ORIENTED_EDGE('',*,*,#250,.F.);
#163=ORIENTED_EDGE('',*,*,#255,.T.);
#164=ORIENTED_EDGE('',*,*,#240,.F.);
#165=ORIENTED_EDGE('',*,*,#245,.F.);
#166=ORIENTED_EDGE('',*,*,#238,.F.);
#167=ORIENTED_EDGE('',*,*,#256,.T.);
#168=ORIENTED_EDGE('',*,*,#252,.F.);
#169=ORIENTED_EDGE('',*,*,#257,.T.);
#170=ORIENTED_EDGE('',*,*,#246,.F.);
#171=ORIENTED_EDGE('',*,*,#258,.T.);
#172=ORIENTED_EDGE('',*,*,#236,.F.);
#173=ORIENTED_EDGE('',*,*,#244,.F.);
#174=ORIENTED_EDGE('',*,*,#242,.F.);
#175=ORIENTED_EDGE('',*,*,#259,.T.);
#176=ORIENTED_EDGE('',*,*,#237,.F.);
#177=ORIENTED_EDGE('',*,*,#258,.F.);
#178=ORIENTED_EDGE('',*,*,#249,.F.);
#179=ORIENTED_EDGE('',*,*,#256,.F.);
#180=ORIENTED_EDGE('',*,*,#253,.F.);
#181=ORIENTED_EDGE('',*,*,#259,.F.);
#182=ORIENTED_EDGE('',*,*,#241,.F.);
#183=ORIENTED_EDGE('',*,*,#255,.F.);
#184=ORIENTED_EDGE('',*,*,#251,.F.);
#185=ORIENTED_EDGE('',*,*,#254,.F.);
```

```
#186=ORIENTED_EDGE('',*,*,#247,.F.);
#187=ORIENTED_EDGE('',*,*,#257,.F.);
#188=ORIENTED_EDGE('',*,*,#260,.T.);
#189=ORIENTED_EDGE('',*,*,#261,.T.);
#190=ORIENTED_EDGE('',*,*,#262,.T.);
#191=ORIENTED_EDGE('',*,*,#263,.T.);
#192=ORIENTED_EDGE('',*,*,#264,.F.);
#193=ORIENTED_EDGE('',*,*,#265,.T.);
#194=ORIENTED_EDGE('',*,*,#260,.F.);
#195=ORIENTED_EDGE('',*,*,#266,.T.);
#196=ORIENTED_EDGE('',*,*,#267,.F.);
#197=ORIENTED_EDGE('',*,*,#268,.T.);
#198=ORIENTED_EDGE('',*,*,#264,.T.);
#199=ORIENTED_EDGE('',*,*,#269,.F.);
#200=ORIENTED_EDGE('',*,*,#262,.F.);
#201=ORIENTED_EDGE('',*,*,#270,.T.);
#202=ORIENTED_EDGE('',*,*,#267,.T.);
#203=ORIENTED_EDGE('',*,*,#271,.T.);
#204=ORIENTED_EDGE('',*,*,#271,.F.);
#205=ORIENTED_EDGE('',*,*,#269,.T.);
#206=ORIENTED_EDGE('',*,*,#266,.F.);
#207=ORIENTED_EDGE('',*,*,#263,.F.);
#208=ORIENTED_EDGE('',*,*,#261,.F.);
#209=ORIENTED_EDGE('',*,*,#265,.F.);
#210=ORIENTED_EDGE('',*,*,#268,.F.);
#211=ORIENTED_EDGE('',*,*,#270,.F.);
#212=VERTEX_POINT('',#483);
#213=VERTEX_POINT('',#484);
#214=VERTEX_POINT('',#486);
```

```
#215=VERTEX_POINT('',#488);
#216=VERTEX_POINT('',#492);
#217=VERTEX_POINT('',#493);
#218=VERTEX_POINT('',#495);
#219=VERTEX_POINT('',#497);
#220=VERTEX_POINT('',#504);
#221=VERTEX_POINT('',#505);
#222=VERTEX_POINT('',#507);
#223=VERTEX_POINT('',#509);
#224=VERTEX_POINT('',#513);
#225=VERTEX_POINT('',#514);
#226=VERTEX_POINT('',#516);
#227=VERTEX_POINT('',#518);
#228=VERTEX_POINT('',#537);
#229=VERTEX_POINT('',#538);
#230=VERTEX_POINT('',#540);
#231=VERTEX_POINT('',#542);
#232=VERTEX_POINT('',#546);
#233=VERTEX_POINT('',#547);
#234=VERTEX_POINT('',#552);
#235=VERTEX_POINT('',#553);
#236=EDGE_CURVE('',#212,#213,#272,.T.);
#237=EDGE_CURVE('',#213,#214,#273,.T.);
#238=EDGE_CURVE('',#214,#215,#274,.T.);
#239=EDGE_CURVE('',#215,#212,#275,.T.);
#240=EDGE_CURVE('',#216,#217,#276,.T.);
#241=EDGE_CURVE('',#217,#218,#277,.T.);
#242=EDGE_CURVE('',#218,#219,#278,.T.);
#243=EDGE_CURVE('',#219,#216,#279,.T.);
```

```
#244=EDGE_CURVE('',#219,#212,#280,.T.);
#245=EDGE_CURVE('',#215,#216,#281,.T.);
#246=EDGE_CURVE('',#220,#221,#282,.T.);
#247=EDGE_CURVE('',#221,#222,#283,.T.);
#248=EDGE_CURVE('',#222,#223,#284,.T.);
#249=EDGE_CURVE('',#223,#220,#285,.T.);
#250=EDGE_CURVE('',#224,#225,#286,.T.);
#251=EDGE_CURVE('',#225,#226,#287,.T.);
#252=EDGE_CURVE('',#226,#227,#288,.T.);
#253=EDGE_CURVE('',#227,#224,#289,.T.);
#254=EDGE_CURVE('',#222,#225,#290,.T.);
#255=EDGE_CURVE('',#224,#217,#291,.T.);
#256=EDGE_CURVE('',#214,#223,#292,.T.);
#257=EDGE_CURVE('',#226,#221,#293,.T.);
#258=EDGE_CURVE('',#220,#213,#294,.T.);
#259=EDGE_CURVE('',#218,#227,#295,.T.);
#260=EDGE_CURVE('',#228,#229,#300,.T.);
#261=EDGE_CURVE('',#229,#230,#301,.T.);
#262=EDGE_CURVE('',#230,#231,#302,.T.);
#263=EDGE_CURVE('',#231,#228,#303,.T.);
#264=EDGE_CURVE('',#232,#233,#304,.T.);
#265=EDGE_CURVE('',#232,#229,#305,.T.);
#266=EDGE_CURVE('',#228,#233,#306,.T.);
#267=EDGE_CURVE('',#234,#235,#307,.T.);
#268=EDGE_CURVE('',#234,#232,#308,.T.);
#269=EDGE_CURVE('',#235,#233,#309,.T.);
#270=EDGE_CURVE('',#230,#234,#310,.T.);
#271=EDGE_CURVE('',#235,#231,#311,.T.);
#272=LINE('',#482,#312);
```

```
#273=LINE('',#485,#313);
#274=LINE('',#487,#314);
#275=LINE('',#489,#315);
#276=LINE('',#491,#316);
#277=LINE('',#494,#317);
#278=LINE('',#496,#318);
#279=LINE('',#498,#319);
#280=LINE('',#500,#320);
#281=LINE('',#501,#321);
#282=LINE('',#503,#322);
#283=LINE('',#506,#323);
#284=LINE('',#508,#324);
#285=LINE('',#510,#325);
#286=LINE('',#512,#326);
#287=LINE('',#515,#327);
#288=LINE('',#517,#328);
#289=LINE('',#519,#329);
#290=LINE('',#521,#330);
#291=LINE('',#522,#331);
#292=LINE('',#523,#332);
#293=LINE('',#525,#333);
#294=LINE('',#526,#334);
#295=LINE('',#527,#335);
#296=LINE('',#532,#336);
#297=LINE('',#533,#337);
#298=LINE('',#534,#338);
#299=LINE('',#535,#339);
#300=LINE('',#536,#340);
#301=LINE('',#539,#341);
```

```
#302=LINE('',#541,#342);
#303=LINE('',#543,#343);
#304=LINE('',#545,#344);
#305=LINE('',#548,#345);
#306=LINE('',#549,#346);
#307=LINE('',#551,#347);
#308=LINE('',#554,#348);
#309=LINE('',#555,#349);
#310=LINE('',#557,#350);
#311=LINE('',#558,#351);
#312=VECTOR('',#404,1.);
#313=VECTOR('',#405,1.);
#314=VECTOR('',#406,1.);
#315=VECTOR('',#407,1.);
#316=VECTOR('',#410,1.);
#317=VECTOR('',#411,1.);
#318=VECTOR('',#412,1.);
#319=VECTOR('',#413,1.);
#320=VECTOR('',#416,1.);
#321=VECTOR('',#417,1.);
#322=VECTOR('',#420,1.);
#323=VECTOR('',#421,1.);
#324=VECTOR('',#422,1.);
#325=VECTOR('',#423,1.);
#326=VECTOR('',#426,1.);
#327=VECTOR('',#427,1.);
#328=VECTOR('',#428,1.);
#329=VECTOR('',#429,1.);
#330=VECTOR('',#432,1.);
```

```
#331=VECTOR('',#433,1.);
#332=VECTOR('',#434,1.);
#333=VECTOR('',#437,1.);
#334=VECTOR('',#438,1.);
#335=VECTOR('',#439,1.);
#336=VECTOR('',#448,65.9562059856757);
#337=VECTOR('',#449,37.4168117888178);
#338=VECTOR('',#450,49.0502147178955);
#339=VECTOR('',#451,69.1551302348579);
#340=VECTOR('',#452,1.);
#341=VECTOR('',#453,1.);
#342=VECTOR('',#454,1.);
#343=VECTOR('',#455,1.);
#344=VECTOR('',#458,1.);
#345=VECTOR('',#459,1.);
#346=VECTOR('',#460,1.);
#347=VECTOR('',#463,1.);
#348=VECTOR('',#464,1.);
#349=VECTOR('',#465,1.);
#350=VECTOR('',#468,1.);
#351=VECTOR('',#469,1.);
#352=PRESENTATION_LAYER_ASSIGNMENT('1','Layer
1',(#381,#72,#58,#59,#60,#61,
#73));
#353=STYLED_ITEM('',(#360),#381);
#354=STYLED_ITEM('',(#361),#72);
#355=STYLED_ITEM('',(#362),#58);
#356=STYLED_ITEM('',(#363),#59);
#357=STYLED_ITEM('',(#364),#60);
```

```
#358=STYLED_ITEM('',(#365),#61);
#359=STYLED_ITEM('',(#366),#73);
#360=PRESENTATION_STYLE_ASSIGNMENT((#367));
#361=PRESENTATION_STYLE_ASSIGNMENT((#62));
#362=PRESENTATION_STYLE_ASSIGNMENT((#368));
#363=PRESENTATION_STYLE_ASSIGNMENT((#369));
#364=PRESENTATION_STYLE_ASSIGNMENT((#370));
#365=PRESENTATION_STYLE_ASSIGNMENT((#371));
#366=PRESENTATION_STYLE_ASSIGNMENT((#63));
#367=CURVE_STYLE('',#375,POSITIVE_LENGTH_MEASURE(0.1763888
88888889),#372);
#368=CURVE_STYLE('',#376,POSITIVE_LENGTH_MEASURE(0.3527777
77777778),#374);
#369=CURVE_STYLE('',#377,POSITIVE_LENGTH_MEASURE(0.3527777
77777778),#374);
#370=CURVE_STYLE('',#378,POSITIVE_LENGTH_MEASURE(0.3527777
77777778),#374);
#371=CURVE_STYLE('',#379,POSITIVE_LENGTH_MEASURE(0.3527777
77777778),#374);
#372=COLOUR_RGB('Strong
Stone',0.576470588235294,0.545082780193 79,0.392156862745098);
#373=COLOUR_RGB('Medium
Steel',0.596063172350652,0.666666666666667,0.686259250782025);
#374=COLOUR_RGB('Medium Royal',0.2,0.4,0.8);
#375=DRAUGHTING_PRE_DEFINED_CURVE_FONT('continuous');
#376=DRAUGHTING_PRE_DEFINED_CURVE_FONT('continuous');
#377=DRAUGHTING_PRE_DEFINED_CURVE_FONT('continuous');
#378=DRAUGHTING_PRE_DEFINED_CURVE_FONT('continuous');
#379=DRAUGHTING_PRE_DEFINED_CURVE_FONT('continuous');
```

```
#380=AXIS2_PLACEMENT_3D('',#480,#400,#401);
#381=AXIS2_PLACEMENT_3D('',#481,#402,#403);
#382=AXIS2_PLACEMENT_3D('',#490,#408,#409);
#383=AXIS2_PLACEMENT_3D('',#499,#414,#415);
#384=AXIS2_PLACEMENT_3D('',#502,#418,#419);
#385=AXIS2_PLACEMENT_3D('',#511,#424,#425);
#386=AXIS2_PLACEMENT_3D('',#520,#430,#431);
#387=AXIS2_PLACEMENT_3D('',#524,#435,#436);
#388=AXIS2_PLACEMENT_3D('',#528,#440,#441);
#389=AXIS2_PLACEMENT_3D('',#529,#442,#443);
#390=AXIS2_PLACEMENT_3D('',#530,#444,#445);
#391=AXIS2_PLACEMENT_3D('',#531,#446,#447);
#392=AXIS2_PLACEMENT_3D('',#544,#456,#457);
#393=AXIS2_PLACEMENT_3D('',#550,#461,#462);
#394=AXIS2_PLACEMENT_3D('',#556,#466,#467);
#395=AXIS2_PLACEMENT_3D('',#559,#470,#471);
#396=AXIS2_PLACEMENT_3D('',#560,#472,#473);
#397=AXIS2_PLACEMENT_3D('',#561,#474,#475);
#398=AXIS2_PLACEMENT_3D('',#562,#476,#477);
#399=AXIS2_PLACEMENT_3D('',#563,#478,#479);
#400=DIRECTION('',(0.,0.,1.));
#401=DIRECTION('',(1.,0.,0.));
#402=DIRECTION('',(0.,0.,1.));
#403=DIRECTION('',(1.,0.,0.));
#404=DIRECTION('',(-
0.137957456800183,0.0587343627961175,0.988695107067978));
#405=DIRECTION('',(-0.00232345693428625,-
0.999858655535086,0.0166514413623847));
#406=DIRECTION('',(0.137927266803465,0.0623364974592235,-
```

```
0.988478745424828));
#407=DIRECTION('',(-
0.00186029654467328,0.999909392761882,0.0133321252368251));
#408=DIRECTION('',(-0.990404895365489,0.,-0.138196031911464));
#409=DIRECTION('',(0.,-1.,0.));
#410=DIRECTION('',(0.0795881247520423,-
0.0627383866288622,0.994851559400527));
#411=DIRECTION('',(-0.00133211772273761,0.999860467244364,-
0.0166514715342202));
#412=DIRECTION('',(-0.0796057706982702,-0.0591131960630719,-
0.995072133728376));
#413=DIRECTION('',(-0.00106657125783944,-0.999910554224474,-
0.013332140722993));
#414=DIRECTION('',(0.996815278536125,0.,-0.07974522228289));
#415=DIRECTION('',(0.,-1.,0.));
#416=DIRECTION('',(-1.,-1.04947916305964E-017,-
2.77695686711364E-016));
#417=DIRECTION('',(1.,-1.11226905297626E-
017,2.77407453615892E-016));
#418=DIRECTION('',(2.77531087822582E-016,0.0133321483061494,-
0.999911122961207));
#419=DIRECTION('',(-1.,0.,-2.77555756156289E-016));
#420=DIRECTION('',(-0.0319273987723256,-0.0592711610872384,-
0.99773121163518));
#421=DIRECTION('',(0.000531644748049991,-
0.999861838322973,0.0166138983765623));
#422=DIRECTION('',(0.0319202836699709,-
0.0629059644396953,0.997508864686595));
#423=DIRECTION('',(-0.000532847486229514,0.999861212447006,-
```

```
0.0166514839446724));
#424=DIRECTION('',(0.999488392880756,0.,-0.0319836285721841));
#425=DIRECTION('',(0.,-1.,0.));
#426=DIRECTION('',(0.0628137506916133,0.0628137506916133,-
0.996046618109867));
#427=DIRECTION('',(0.00104772289405661,0.999861430840519,-
0.0166138916057547));
#428=DIRECTION('',(-
0.0628277109839773,0.0591842369382303,0.99626798846021));
#429=DIRECTION('',(-0.00105009315220412,-
0.999860803119031,0.0166514771278081));
#430=DIRECTION('',(-0.998017436499479,0.,-0.0629380365360033));
#431=DIRECTION('',(0.,-1.,0.));
#432=DIRECTION('',(1.,-1.53534473834867E-
033,1.01033360929657E-016));
#433=DIRECTION('',(-1.,-6.38448196337505E-
018,2.06281632432954E-019));
#434=DIRECTION('',(-1.,5.30835363538334E-018,-
1.85208682862164E-016));
#435=DIRECTION('',(-6.35884136154596E-
018,0.998017436499479,0.0629380365360033));
#436=DIRECTION('',(1.,0.,9.71445146547012E-017));
#437=DIRECTION('',(-1.,0.,-1.01033360929657E-016));
#438=DIRECTION('',(1.,4.99032533710111E-018,1.85037170770859E-
016));
#439=DIRECTION('',(1.,-6.00198183740537E-018,1.6872534936295E-
037));
#440=DIRECTION('',(5.99141909231366E-018,0.998240123782801,-
0.0593013934920476));
```

```
#441=DIRECTION('',(-1.,0.,-1.01033360929657E-016));
#442=DIRECTION('',(1.85094612513574E-016,-0.0166514863085712,-
0.999861354390555));
#443=DIRECTION('',(-1.,-4.98732999343332E-018,-
1.80411241501588E-016));
#444=DIRECTION('',(-9.99419183898486E-020,-0.0166514863085712,-
0.999861354390555));
#445=DIRECTION('',(-1.,6.00376953013848E-018,0.));
#446=DIRECTION('',(1.0101941626745E-016,-0.0166139007244899,-
0.999861979626547));
#447=DIRECTION('',(1.,0.,1.01033360929657E-016));
#448=DIRECTION('',(0.0871557427476582,0.,-0.996194698091746));
#449=DIRECTION('',(0.996194698091745,0.,-0.0871557427476581));
#450=DIRECTION('',(-0.999998851535112,0.,0.0015155620928241));
#451=DIRECTION('',(0.0871557427476578,0.,0.996194698091746));
#452=DIRECTION('',(0.0868265938642476,0.0868265938642476,-
0.992432509138967));
#453=DIRECTION('',(0.996165738604111,0.00762491778797317,-
0.0871532091207886));
#454=DIRECTION('',(0.0868265938642472,-
0.0868265938642476,0.992432509138967));
#455=DIRECTION('',(-0.999998846257634,-
0.00013239308467001,0.00151325988230213));
#456=DIRECTION('',(0.,-0.996194698091746,-0.0871557427476582));
#457=DIRECTION('',(0.,0.0871557427476582,-0.996194698091746));
#458=DIRECTION('',(-0.0871456865253051,-
0.0151904940727351,0.996079754944281));
#459=DIRECTION('',(4.0504227598039E-018,-
1.,1.76411756253482E-018));
```

```
#460=DIRECTION('',(-
0.00152132297606744,0.999847645978987,0.0173888011858433));
#461=DIRECTION('',(-0.996194698091746,-4.18876265480318E-018,-
0.0871557427476582));
#462=DIRECTION('',(-0.0871557427476582,0.,0.996194698091746));
#463=DIRECTION('',(0.0871557427476578,0.,0.996194698091746));
#464=DIRECTION('',(-0.992403876506104,-
0.0871557427476582,0.0868240888334651));
#465=DIRECTION('',(-0.996223440096615,-0.0868265938642476,-
2.79284439827277E-016));
#466=DIRECTION('',(-
0.0868240888334652,0.996194698091745,0.00759612349389592));
#467=DIRECTION('',(-0.996223440096615,-0.0868265938642476,0.));
#468=DIRECTION('',(0.,1.,0.));
#469=DIRECTION('',(-0.00152091971382225,-0.999847726744427,-
0.017384191877292));
#470=DIRECTION('',(0.996194698091746,0.,-0.0871557427476578));
#471=DIRECTION('',(-0.0871557427476578,0.,-0.996194698091746));
#472=DIRECTION('',(0.00151533300568351,-
0.0173864963789149,0.999847695156391));
#473=DIRECTION('',(0.,-0.999848843100563,-0.0173865163406824));
#474=DIRECTION('',(-0.0871557427476581,-2.11042216668041E-018,-
0.996194698091745));
#475=DIRECTION('',(-0.996194698091746,0.,0.0871557427476582));
#476=DIRECTION('',(0.,0.,1.));
#477=DIRECTION('',(1.,0.,0.));
#478=DIRECTION('',(0.,0.,1.));
#479=DIRECTION('',(1.,0.,0.));
#480=CARTESIAN_POINT('',(0.,0.,0.));
```

#481=CARTESIAN_POINT('',(75.5,1.4210854715202E-014,62.5));
#482=CARTESIAN_POINT('',(74.9788202907328,297.395155717807,57.1517879164149));
#483=CARTESIAN_POINT('',(74.236816635788,297.711058263971,62.4694807768529));
#484=CARTESIAN_POINT('',(69.696446597988,299.644087091946,95.008799381086));
#485=CARTESIAN_POINT('',(69.6987725792974,300.645032067726,94.9921298483687));
#486=CARTESIAN_POINT('',(69.0017518818214,0.694485460102542,99.9874448469463));
#487=CARTESIAN_POINT('',(77.5500914706091,4.55792422320328,38.7243444606348));
#488=CARTESIAN_POINT('',(74.7845432013308,3.30802928468555,58.5441070571291));
#489=CARTESIAN_POINT('',(74.7904388460093,0.139120270011757,58.5018549369335));
#490=CARTESIAN_POINT('',(75.,300.,57.));
#491=CARTESIAN_POINT('',(127.50549461001,1.87179478940647,81.3186826251257));
#492=CARTESIAN_POINT('',(125.68352856457,3.30802928468553,58.5441070571291));
#493=CARTESIAN_POINT('',(128.998995587756,0.694485460102472,99.9874448469463));
#494=CARTESIAN_POINT('',(128.600341320731,299.916269610443,95.0042665091431));
#495=CARTESIAN_POINT('',(128.600703950487,299.644087091946,95.008799381086));
#496=CARTESIAN_POINT('',(129.000279549596,299.94080164574,100.

003494369949));
#497=CARTESIAN_POINT('',(125.997558462148,297.711058263971,62.469480776853));
#498=CARTESIAN_POINT('',(125.679857029669,-0.134034685776398,58.4982128708563));
#499=CARTESIAN_POINT('',(129.,300.,100.));
#500=CARTESIAN_POINT('',(-2.04637349479634E-014,297.711058263971,62.4694807768529));
#501=CARTESIAN_POINT('',(-1.62124531755506E-014,3.30802928468556,58.5441070571291));
#502=CARTESIAN_POINT('',(-1.62370117351429E-014,0.,58.5));
#503=CARTESIAN_POINT('',(4.00011241825716,299.940802756666,100.003513070537));
#504=CARTESIAN_POINT('',(3.84028158019474,299.644087091946,95.008799381086));
#505=CARTESIAN_POINT('',(0.806905520301186,294.012819654024,0.215797509411928));
#506=CARTESIAN_POINT('',(0.80460517271514,298.339072804463,0.143911647347991));
#507=CARTESIAN_POINT('',(0.959682789839923,6.68530981732001,4.99008718249747));
#508=CARTESIAN_POINT('',(3.39900005161299,1.87809674513206,81.2187516129062));
#509=CARTESIAN_POINT('',(3.99959823510227,0.694485460102577,99.9874448469463));
#510=CARTESIAN_POINT('',(3.84013628999787,299.916716758333,95.0042590624336));
#511=CARTESIAN_POINT('',(3.99999999999999,300.,100.));
#512=CARTESIAN_POINT('',(201.153314196715,8.84700789040915,-

```
29.2882679764876));
#513=CARTESIAN_POINT('',(193.000791766409,0.694485460102545,9
9.9874448469463));
#514=CARTESIAN_POINT('',(198.991616123626,6.68530981731998,4.9
900871824975));
#515=CARTESIAN_POINT('',(199.299165498107,300.185394015636,0.1
1323281573628));
#516=CARTESIAN_POINT('',(199.292697454361,294.012819654024,0.2
15797509411944));
#517=CARTESIAN_POINT('',(199.975259687035,293.369840238239,-
10.6076893229804));
#518=CARTESIAN_POINT('',(193.314760399391,299.644087091946,95.
008799381086));
#519=CARTESIAN_POINT('',(193.316995088545,301.771877457584,94.
9733635959291));
#520=CARTESIAN_POINT('',(200.,300.,-11.));
#521=CARTESIAN_POINT('',(-5.04165279379714E-
016,6.68530981731999,4.99008718249748));
#522=CARTESIAN_POINT('',(-1.122140868094E-
014,0.694485460102486,99.9874448469463));
#523=CARTESIAN_POINT('',(-9.17139565393105E-
015,0.694485460102487,99.9874448469463));
#524=CARTESIAN_POINT('',(-1.12147030631919E-014,0.,111.));
#525=CARTESIAN_POINT('',(-2.18027476561359E-
017,294.012819654024,0.215797509411923));
#526=CARTESIAN_POINT('',(-1.7608325266632E-
014,299.644087091946,95.008799381086));
#527=CARTESIAN_POINT('',(3.38757082145922E-
017,299.644087091946,95.008799381086));
```

```
#528=CARTESIAN_POINT('',(0.,294.,0.));
#529=CARTESIAN_POINT('',(75.,-0.0594059405942815,100.));
#530=CARTESIAN_POINT('',(200.,-0.0594059405942815,100.));
#531=CARTESIAN_POINT('',(1.07959201061711E-
032,307.,1.09074809265452E-048));
#532=CARTESIAN_POINT('',(0.221668912880384,150.,72.466312731881));
#533=CARTESIAN_POINT('',(5.97013103437948,150.,6.761090022703775));
#534=CARTESIAN_POINT('',(49.2718272983265,150.,72.3919740858096));
#535=CARTESIAN_POINT('',(43.2445605578965,150.,3.50000000000001));
#536=CARTESIAN_POINT('',(0.186586720701157,0.186586720701158,
72.867304023385));
#537=CARTESIAN_POINT('',(0.449218620794389,0.449218620794389,
69.8654076689458));
#538=CARTESIAN_POINT('',(5.97013103437949,5.97013103437949,6.76109002270376));
#539=CARTESIAN_POINT('',(18.6364783054747,6.06708262875108,5.6529282282004));
#540=CARTESIAN_POINT('',(43.2445605578965,6.25543944210357,3.49999999999999));
#541=CARTESIAN_POINT('',(49.3134132792989,0.186586720701157,72.867304023385));
#542=CARTESIAN_POINT('',(49.0443477127424,0.455652287257603,69.7918705247728));
#543=CARTESIAN_POINT('',(24.7421144893581,0.452434835924679,69.8286461617897));
```

```
#544=CARTESIAN_POINT('',(24.75,0.,75.));
#545=CARTESIAN_POINT('',(3.61186866629015,296.315380926626,33.
7161522335988));
#546=CARTESIAN_POINT('',(5.97013103437949,296.726453143195,6.7
6109002270375));
#547=CARTESIAN_POINT('',(0.,295.685790733991,75.));
#548=CARTESIAN_POINT('',(5.97013103437949,147.842895366995,6.7
6109002270375));
#549=CARTESIAN_POINT('',(0.224882526592251,147.887940854478,7
2.4295809590734));
#550=CARTESIAN_POINT('',(0.,147.842895366995,75.));
#551=CARTESIAN_POINT('',(49.5,300.,75.));
#552=CARTESIAN_POINT('',(43.2445605578965,300.,3.5));
#553=CARTESIAN_POINT('',(49.5,300.,75.));
#554=CARTESIAN_POINT('',(43.2445605578965,300.,3.5));
#555=CARTESIAN_POINT('',(49.5,300.,75.));
#556=CARTESIAN_POINT('',(46.2191751177779,300.,37.5));
#557=CARTESIAN_POINT('',(43.2445605578965,150.,3.5));
#558=CARTESIAN_POINT('',(49.2718967822361,150.0456784986,72.39
27682905301));
#559=CARTESIAN_POINT('',(49.5,150.,75.));
#560=CARTESIAN_POINT('',(24.75,297.842895366995,75.));
#561=CARTESIAN_POINT('',(43.2445605578965,153.127719721052,3.5
));
#562=CARTESIAN_POINT('',(0.,0.,0.));
#563=CARTESIAN_POINT('',(75.5,1.4210854715202E-014,62.5));
#564=MECHANICAL_DESIGN_GEOMETRIC_PRESENTATION_REPRESE
NTATION('',(#353),#567);
#565=MECHANICAL_DESIGN_GEOMETRIC_PRESENTATION_REPRESE
```

```
NTATION('',(#354),#568);
#566=MECHANICAL_DESIGN_GEOMETRIC_PRESENTATION_REPRESE
NTATION('',(#355,#356,
#357,#358,#359),#569);
#567=(
GEOMETRIC_REPRESENTATION_CONTEXT(3)
GLOBAL_UNCERTAINTY_ASSIGNED_CONTEXT((#570))
GLOBAL_UNIT_ASSIGNED_CONTEXT((#578,#574,#573))
REPRESENTATION_CONTEXT('blockslide3','TOP_LEVEL_ASSEMBLY_PA
RT')
);
#568=(
GEOMETRIC_REPRESENTATION_CONTEXT(3)
GLOBAL_UNCERTAINTY_ASSIGNED_CONTEXT((#571))
GLOBAL_UNIT_ASSIGNED_CONTEXT((#578,#574,#573))
REPRESENTATION_CONTEXT('block51','COMPONENT_PART')
);
#569=(
GEOMETRIC_REPRESENTATION_CONTEXT(3)
GLOBAL_UNCERTAINTY_ASSIGNED_CONTEXT((#572))
GLOBAL_UNIT_ASSIGNED_CONTEXT((#578,#574,#573))
REPRESENTATION_CONTEXT('slide50','COMPONENT_PART')
);
#570=UNCERTAINTY_MEASURE_WITH_UNIT(LENGTH_MEASURE(2.E-
005),#578,
'DISTANCE_ACCURACY_VALUE','Maximum Tolerance applied to
model');
#571=UNCERTAINTY_MEASURE_WITH_UNIT(LENGTH_MEASURE(2.E-
005),#578,
```

```
'DISTANCE_ACCURACY_VALUE','Maximum Tolerance applied to
model');
#572=UNCERTAINTY_MEASURE_WITH_UNIT(LENGTH_MEASURE(2.E-
005),#578,
'DISTANCE_ACCURACY_VALUE','Maximum Tolerance applied to
model');
#573=(
NAMED_UNIT(*)
SI_UNIT($,.STERADIAN.)
SOLID_ANGLE_UNIT()
);
#574=(
CONVERSION_BASED_UNIT('DEGREE',#576)
NAMED_UNIT(#575)
PLANE_ANGLE_UNIT()
);
#575=DIMENSIONAL_EXPONENTS(0.,0.,0.,0.,0.,0.,0.);
#576=PLANE_ANGLE_MEASURE_WITH_UNIT(PLANE_ANGLE_MEASUR
E(0.0174532925),#577);
#577=(
NAMED_UNIT(*)
PLANE_ANGLE_UNIT()
SI_UNIT($,.RADIAN.)
);
#578=(
LENGTH_UNIT()
NAMED_UNIT(*)
SI_UNIT(.MILLI.,.METRE.)
);
```

```
ENDSEC;
END-ISO-10303-21;
```

PlateRod.stp

```
ISO-10303-21;
HEADER;
/* Generated by software containing ST-Developer
 * from STEP Tools, Inc. (www.steptools.com)
 */
/* OPTION: using custom schema-name function */

FILE_DESCRIPTION(
/* description */ (''),
/* implementation_level */ '2;1');

FILE_NAME(
/* name */ 'platerod.stp',
/* time_stamp */ '2014-04-09T18:34:35-04:00',
/* author */ (''),
/* organization */ (''),
/* preprocessor_version */ 'ST-DEVELOPER v15',
/* originating_system */ 'SIEMENS PLM Software NX 8.5',
/* authorisation */ '');

FILE_SCHEMA (('AUTOMOTIVE_DESIGN { 1 0 10303 214 3 1 1 1 }'));
ENDSEC;

DATA;
```

```
#10=CONTEXT_DEPENDENT_SHAPE_REPRESENTATION(#18,#36);
#11=CONTEXT_DEPENDENT_SHAPE_REPRESENTATION(#19,#37);
#12=CONTEXT_DEPENDENT_SHAPE_REPRESENTATION(#20,#38);
#13=CONTEXT_DEPENDENT_SHAPE_REPRESENTATION(#21,#39);
#14=NEXT_ASSEMBLY_USAGE_OCCURRENCE('1001','
','PLATE',#42,#41,$);
#15=NEXT_ASSEMBLY_USAGE_OCCURRENCE('1002','
','ROD',#42,#40,$);
#16=NEXT_ASSEMBLY_USAGE_OCCURRENCE('1003','
','ROD',#42,#40,$);
#17=NEXT_ASSEMBLY_USAGE_OCCURRENCE('1004','
','PLATE',#42,#41,$);
#18=(
REPRESENTATION_RELATIONSHIP(' ',' ',#61,#62)
REPRESENTATION_RELATIONSHIP_WITH_TRANSFORMATION(#22)
SHAPE_REPRESENTATION_RELATIONSHIP()
);
#19=(
REPRESENTATION_RELATIONSHIP(' ',' ',#60,#62)
REPRESENTATION_RELATIONSHIP_WITH_TRANSFORMATION(#23)
SHAPE_REPRESENTATION_RELATIONSHIP()
);
#20=(
REPRESENTATION_RELATIONSHIP(' ',' ',#60,#62)
REPRESENTATION_RELATIONSHIP_WITH_TRANSFORMATION(#24)
SHAPE_REPRESENTATION_RELATIONSHIP()
);
#21=(
REPRESENTATION_RELATIONSHIP(' ',' ',#61,#62)
```

```
REPRESENTATION_RELATIONSHIP_WITH_TRANSFORMATION(#25)
SHAPE_REPRESENTATION_RELATIONSHIP()
);
#22=ITEM_DEFINED_TRANSFORMATION(' ',' ',#803,#844);
#23=ITEM_DEFINED_TRANSFORMATION(' ',' ',#803,#845);
#24=ITEM_DEFINED_TRANSFORMATION(' ',' ',#803,#846);
#25=ITEM_DEFINED_TRANSFORMATION(' ',' ',#803,#847);
#26=SHAPE_REPRESENTATION_RELATIONSHIP('MODEL',
'relationship between rod-MODEL and rod-MODEL',#60,#28);
#27=SHAPE_REPRESENTATION_RELATIONSHIP('MODEL',
'relationship between plate-MODEL and plate-MODEL',#61,#29);
#28=ADVANCED_BREP_SHAPE_REPRESENTATION('rod-
MODEL',(#80),#1243);
#29=ADVANCED_BREP_SHAPE_REPRESENTATION('plate-
MODEL',(#79),#1242);
#30=SHAPE_DEFINITION_REPRESENTATION(#33,#60);
#31=SHAPE_DEFINITION_REPRESENTATION(#34,#61);
#32=SHAPE_DEFINITION_REPRESENTATION(#35,#62);
#33=PRODUCT_DEFINITION_SHAPE('','',#40);
#34=PRODUCT_DEFINITION_SHAPE('','',#41);
#35=PRODUCT_DEFINITION_SHAPE('','',#42);
#36=PRODUCT_DEFINITION_SHAPE(' ','NAUO PRDDFN',#14);
#37=PRODUCT_DEFINITION_SHAPE(' ','NAUO PRDDFN',#15);
#38=PRODUCT_DEFINITION_SHAPE(' ','NAUO PRDDFN',#16);
#39=PRODUCT_DEFINITION_SHAPE(' ','NAUO PRDDFN',#17);
#40=PRODUCT_DEFINITION(' ','',#46,#43);
#41=PRODUCT_DEFINITION(' ','',#47,#44);
#42=PRODUCT_DEFINITION(' ','',#48,#45);
#43=PRODUCT_DEFINITION_CONTEXT('part definition',#59,'design');
```

```
#44=PRODUCT_DEFINITION_CONTEXT('part definition',#59,'design');
#45=PRODUCT_DEFINITION_CONTEXT('part definition',#59,'design');
#46=PRODUCT_DEFINITION_FORMATION_WITH_SPECIFIED_SOURCE('
',' ',#52,
 .NOT_KNOWN.);
#47=PRODUCT_DEFINITION_FORMATION_WITH_SPECIFIED_SOURCE('
',' ',#53,
 .NOT_KNOWN.);
#48=PRODUCT_DEFINITION_FORMATION_WITH_SPECIFIED_SOURCE('
',' ',#54,
 .NOT_KNOWN.);
#49=PRODUCT_RELATED_PRODUCT_CATEGORY('part','',(#52));
#50=PRODUCT_RELATED_PRODUCT_CATEGORY('part','',(#53));
#51=PRODUCT_RELATED_PRODUCT_CATEGORY('part','',(#54));
#52=PRODUCT('rod','rod',' ',(#55));
#53=PRODUCT('plate','plate',' ',(#56));
#54=PRODUCT('platerod','platerod',' ',(#57));
#55=PRODUCT_CONTEXT(' ',#59,'mechanical');
#56=PRODUCT_CONTEXT(' ',#59,'mechanical');
#57=PRODUCT_CONTEXT(' ',#59,'mechanical');
#58=APPLICATION_PROTOCOL_DEFINITION('international standard',
'automotive_design',2010,#59);
#59=APPLICATION_CONTEXT(
'core data for automotive mechanical design processes');
#60=SHAPE_REPRESENTATION('rod-MODEL',(#803),#1243);
#61=SHAPE_REPRESENTATION('plate-MODEL',(#803),#1242);
#62=SHAPE_REPRESENTATION('platerod-
none',(#803,#844,#845,#846,#847),#1241);
#63=PRESENTATION_LAYER_ASSIGNMENT('1','Layer 1',(#79,#80));
```

```
#64=STYLED_ITEM('',(#66),#79);
#65=STYLED_ITEM('',(#67),#80);
#66=PRESENTATION_STYLE_ASSIGNMENT((#68));
#67=PRESENTATION_STYLE_ASSIGNMENT((#69));
#68=SURFACE_STYLE_USAGE(.BOTH.,#70);
#69=SURFACE_STYLE_USAGE(.BOTH.,#71);
#70=SURFACE_SIDE_STYLE('',(#72));
#71=SURFACE_SIDE_STYLE('',(#73));
#72=SURFACE_STYLE_FILL_AREA(#74);
#73=SURFACE_STYLE_FILL_AREA(#75);
#74=FILL_AREA_STYLE('',(#76));
#75=FILL_AREA_STYLE('',(#77));
#76=FILL_AREA_STYLE_COLOUR('',#78);
#77=FILL_AREA_STYLE_COLOUR('',#78);
#78=COLOUR_RGB('Medium
Steel',0.596063172350652,0.666666666666667,0.686259250782025);
#79=MANIFOLD_SOLID_BREP('',#81);
#80=MANIFOLD_SOLID_BREP('',#82);
#81=CLOSED_SHELL('',(#117,#118,#119,#120,#121,#122,#123,#124,#1
25,#126,
#127,#128,#129,#130,#131,#132,#133,#134,#135,#136,#137,#138,#139
,#140));
#82=CLOSED_SHELL('',(#141,#142,#143,#144,#145,#146,#147,#148,#1
49,#150,
#151,#152,#153,#154,#155,#156));
#83=FACE_OUTER_BOUND('',#221,.T.);
#84=FACE_OUTER_BOUND('',#222,.T.);
#85=FACE_OUTER_BOUND('',#223,.T.);
#86=FACE_OUTER_BOUND('',#224,.T.);
```

```
#87=FACE_OUTER_BOUND('',#225,.T.);
#88=FACE_OUTER_BOUND('',#226,.T.);
#89=FACE_OUTER_BOUND('',#227,.T.);
#90=FACE_OUTER_BOUND('',#228,.T.);
#91=FACE_OUTER_BOUND('',#229,.T.);
#92=FACE_OUTER_BOUND('',#230,.T.);
#93=FACE_OUTER_BOUND('',#231,.T.);
#94=FACE_OUTER_BOUND('',#232,.T.);
#95=FACE_OUTER_BOUND('',#233,.T.);
#96=FACE_OUTER_BOUND('',#234,.T.);
#97=FACE_OUTER_BOUND('',#235,.T.);
#98=FACE_OUTER_BOUND('',#236,.T.);
#99=FACE_OUTER_BOUND('',#237,.T.);
#100=FACE_OUTER_BOUND('',#238,.T.);
#101=FACE_OUTER_BOUND('',#239,.T.);
#102=FACE_OUTER_BOUND('',#240,.T.);
#103=FACE_OUTER_BOUND('',#243,.T.);
#104=FACE_OUTER_BOUND('',#244,.T.);
#105=FACE_OUTER_BOUND('',#245,.T.);
#106=FACE_OUTER_BOUND('',#246,.T.);
#107=FACE_OUTER_BOUND('',#249,.T.);
#108=FACE_OUTER_BOUND('',#250,.T.);
#109=FACE_OUTER_BOUND('',#251,.T.);
#110=FACE_OUTER_BOUND('',#252,.T.);
#111=FACE_OUTER_BOUND('',#253,.T.);
#112=FACE_OUTER_BOUND('',#254,.T.);
#113=FACE_OUTER_BOUND('',#255,.T.);
#114=FACE_OUTER_BOUND('',#256,.T.);
#115=FACE_OUTER_BOUND('',#257,.T.);
```

```
#116=FACE_OUTER_BOUND('',#258,.T.);
#117=ADVANCED_FACE('',(#197,#198,#199),#157,.F.);
#118=ADVANCED_FACE('',(#200,#201),#158,.T.);
#119=ADVANCED_FACE('',(#202,#203),#159,.F.);
#120=ADVANCED_FACE('',(#204,#205,#206),#160,.T.);
#121=ADVANCED_FACE('',(#83),#161,.T.);
#122=ADVANCED_FACE('',(#84),#162,.T.);
#123=ADVANCED_FACE('',(#85),#163,.T.);
#124=ADVANCED_FACE('',(#86),#164,.T.);
#125=ADVANCED_FACE('',(#87),#165,.F.);
#126=ADVANCED_FACE('',(#88),#166,.F.);
#127=ADVANCED_FACE('',(#89),#167,.F.);
#128=ADVANCED_FACE('',(#90),#168,.F.);
#129=ADVANCED_FACE('',(#91),#169,.T.);
#130=ADVANCED_FACE('',(#92),#170,.T.);
#131=ADVANCED_FACE('',(#93),#171,.T.);
#132=ADVANCED_FACE('',(#94),#172,.T.);
#133=ADVANCED_FACE('',(#95),#173,.T.);
#134=ADVANCED_FACE('',(#96),#174,.T.);
#135=ADVANCED_FACE('',(#97),#175,.T.);
#136=ADVANCED_FACE('',(#98),#176,.T.);
#137=ADVANCED_FACE('',(#99),#177,.F.);
#138=ADVANCED_FACE('',(#100),#178,.F.);
#139=ADVANCED_FACE('',(#101),#179,.F.);
#140=ADVANCED_FACE('',(#102),#180,.F.);
#141=ADVANCED_FACE('',(#207,#208),#181,.F.);
#142=ADVANCED_FACE('',(#103),#182,.T.);
#143=ADVANCED_FACE('',(#104),#183,.T.);
#144=ADVANCED_FACE('',(#105),#184,.T.);
```

```
#145=ADVANCED_FACE('',(#106),#185,.T.);
#146=ADVANCED_FACE('',(#209,#210),#186,.T.);
#147=ADVANCED_FACE('',(#107),#187,.F.);
#148=ADVANCED_FACE('',(#108),#188,.F.);
#149=ADVANCED_FACE('',(#109),#189,.F.);
#150=ADVANCED_FACE('',(#110),#190,.F.);
#151=ADVANCED_FACE('',(#111),#191,.T.);
#152=ADVANCED_FACE('',(#112),#192,.F.);
#153=ADVANCED_FACE('',(#113),#193,.F.);
#154=ADVANCED_FACE('',(#114),#194,.F.);
#155=ADVANCED_FACE('',(#115),#195,.F.);
#156=ADVANCED_FACE('',(#116),#196,.T.);
#157=PLANE('',#804);
#158=PLANE('',#805);
#159=PLANE('',#806);
#160=PLANE('',#807);
#161=PLANE('',#808);
#162=PLANE('',#809);
#163=PLANE('',#810);
#164=PLANE('',#811);
#165=PLANE('',#812);
#166=PLANE('',#813);
#167=PLANE('',#814);
#168=PLANE('',#815);
#169=PLANE('',#816);
#170=PLANE('',#817);
#171=PLANE('',#818);
#172=PLANE('',#819);
#173=PLANE('',#820);
```

```
#174=PLANE('',#821);
#175=PLANE('',#822);
#176=PLANE('',#823);
#177=PLANE('',#824);
#178=PLANE('',#825);
#179=PLANE('',#826);
#180=PLANE('',#827);
#181=PLANE('',#828);
#182=PLANE('',#829);
#183=PLANE('',#830);
#184=PLANE('',#831);
#185=PLANE('',#832);
#186=PLANE('',#833);
#187=PLANE('',#834);
#188=PLANE('',#835);
#189=PLANE('',#836);
#190=PLANE('',#837);
#191=PLANE('',#838);
#192=PLANE('',#839);
#193=PLANE('',#840);
#194=PLANE('',#841);
#195=PLANE('',#842);
#196=PLANE('',#843);
#197=FACE_BOUND('',#211,.T.);
#198=FACE_BOUND('',#212,.T.);
#199=FACE_BOUND('',#213,.T.);
#200=FACE_BOUND('',#214,.T.);
#201=FACE_BOUND('',#215,.T.);
#202=FACE_BOUND('',#216,.T.);
```

```
#203=FACE_BOUND('',#217,.T.);
#204=FACE_BOUND('',#218,.T.);
#205=FACE_BOUND('',#219,.T.);
#206=FACE_BOUND('',#220,.T.);
#207=FACE_BOUND('',#241,.T.);
#208=FACE_BOUND('',#242,.T.);
#209=FACE_BOUND('',#247,.T.);
#210=FACE_BOUND('',#248,.T.);
#211=EDGE_LOOP('',(#259,#260,#261,#262));
#212=EDGE_LOOP('',(#263,#264,#265,#266));
#213=EDGE_LOOP('',(#267,#268,#269,#270));
#214=EDGE_LOOP('',(#271,#272,#273,#274));
#215=EDGE_LOOP('',(#275,#276,#277,#278));
#216=EDGE_LOOP('',(#279,#280,#281,#282));
#217=EDGE_LOOP('',(#283,#284,#285,#286));
#218=EDGE_LOOP('',(#287,#288,#289,#290));
#219=EDGE_LOOP('',(#291,#292,#293,#294));
#220=EDGE_LOOP('',(#295,#296,#297,#298));
#221=EDGE_LOOP('',(#299,#300,#301,#302));
#222=EDGE_LOOP('',(#303,#304,#305,#306));
#223=EDGE_LOOP('',(#307,#308,#309,#310));
#224=EDGE_LOOP('',(#311,#312,#313,#314));
#225=EDGE_LOOP('',(#315,#316,#317,#318));
#226=EDGE_LOOP('',(#319,#320,#321,#322));
#227=EDGE_LOOP('',(#323,#324,#325,#326));
#228=EDGE_LOOP('',(#327,#328,#329,#330));
#229=EDGE_LOOP('',(#331,#332,#333,#334));
#230=EDGE_LOOP('',(#335,#336,#337,#338));
#231=EDGE_LOOP('',(#339,#340,#341,#342));
```

```
#232=EDGE_LOOP('',(#343,#344,#345,#346));
#233=EDGE_LOOP('',(#347,#348,#349,#350));
#234=EDGE_LOOP('',(#351,#352,#353,#354));
#235=EDGE_LOOP('',(#355,#356,#357,#358));
#236=EDGE_LOOP('',(#359,#360,#361,#362));
#237=EDGE_LOOP('',(#363,#364,#365,#366));
#238=EDGE_LOOP('',(#367,#368,#369,#370));
#239=EDGE_LOOP('',(#371,#372,#373,#374));
#240=EDGE_LOOP('',(#375,#376,#377,#378));
#241=EDGE_LOOP('',(#379,#380,#381,#382));
#242=EDGE_LOOP('',(#383,#384,#385,#386));
#243=EDGE_LOOP('',(#387,#388,#389,#390));
#244=EDGE_LOOP('',(#391,#392,#393,#394));
#245=EDGE_LOOP('',(#395,#396,#397,#398));
#246=EDGE_LOOP('',(#399,#400,#401,#402));
#247=EDGE_LOOP('',(#403,#404,#405,#406));
#248=EDGE_LOOP('',(#407,#408,#409,#410));
#249=EDGE_LOOP('',(#411,#412,#413,#414));
#250=EDGE_LOOP('',(#415,#416,#417,#418));
#251=EDGE_LOOP('',(#419,#420,#421,#422));
#252=EDGE_LOOP('',(#423,#424,#425,#426));
#253=EDGE_LOOP('',(#427,#428,#429,#430));
#254=EDGE_LOOP('',(#431,#432,#433,#434));
#255=EDGE_LOOP('',(#435,#436,#437,#438));
#256=EDGE_LOOP('',(#439,#440,#441,#442));
#257=EDGE_LOOP('',(#443,#444,#445,#446));
#258=EDGE_LOOP('',(#447,#448,#449,#450));
#259=ORIENTED_EDGE('',*,*,#515,.T.);
#260=ORIENTED_EDGE('',*,*,#516,.T.);
```

```
#261=ORIENTED_EDGE('',*,*,#517,.T.);
#262=ORIENTED_EDGE('',*,*,#518,.T.);
#263=ORIENTED_EDGE('',*,*,#519,.F.);
#264=ORIENTED_EDGE('',*,*,#520,.F.);
#265=ORIENTED_EDGE('',*,*,#521,.F.);
#266=ORIENTED_EDGE('',*,*,#522,.F.);
#267=ORIENTED_EDGE('',*,*,#523,.F.);
#268=ORIENTED_EDGE('',*,*,#524,.F.);
#269=ORIENTED_EDGE('',*,*,#525,.F.);
#270=ORIENTED_EDGE('',*,*,#526,.F.);
#271=ORIENTED_EDGE('',*,*,#527,.T.);
#272=ORIENTED_EDGE('',*,*,#528,.T.);
#273=ORIENTED_EDGE('',*,*,#529,.T.);
#274=ORIENTED_EDGE('',*,*,#530,.T.);
#275=ORIENTED_EDGE('',*,*,#531,.T.);
#276=ORIENTED_EDGE('',*,*,#532,.T.);
#277=ORIENTED_EDGE('',*,*,#533,.T.);
#278=ORIENTED_EDGE('',*,*,#534,.T.);
#279=ORIENTED_EDGE('',*,*,#535,.F.);
#280=ORIENTED_EDGE('',*,*,#536,.F.);
#281=ORIENTED_EDGE('',*,*,#537,.F.);
#282=ORIENTED_EDGE('',*,*,#538,.F.);
#283=ORIENTED_EDGE('',*,*,#539,.T.);
#284=ORIENTED_EDGE('',*,*,#540,.T.);
#285=ORIENTED_EDGE('',*,*,#541,.T.);
#286=ORIENTED_EDGE('',*,*,#542,.T.);
#287=ORIENTED_EDGE('',*,*,#543,.T.);
#288=ORIENTED_EDGE('',*,*,#544,.T.);
#289=ORIENTED_EDGE('',*,*,#545,.T.);
```

```
#290=ORIENTED_EDGE('',*,*,#546,.T.);
#291=ORIENTED_EDGE('',*,*,#547,.F.);
#292=ORIENTED_EDGE('',*,*,#548,.F.);
#293=ORIENTED_EDGE('',*,*,#549,.F.);
#294=ORIENTED_EDGE('',*,*,#550,.F.);
#295=ORIENTED_EDGE('',*,*,#551,.T.);
#296=ORIENTED_EDGE('',*,*,#552,.T.);
#297=ORIENTED_EDGE('',*,*,#553,.T.);
#298=ORIENTED_EDGE('',*,*,#554,.T.);
#299=ORIENTED_EDGE('',*,*,#551,.F.);
#300=ORIENTED_EDGE('',*,*,#555,.F.);
#301=ORIENTED_EDGE('',*,*,#519,.T.);
#302=ORIENTED_EDGE('',*,*,#556,.T.);
#303=ORIENTED_EDGE('',*,*,#552,.F.);
#304=ORIENTED_EDGE('',*,*,#556,.F.);
#305=ORIENTED_EDGE('',*,*,#522,.T.);
#306=ORIENTED_EDGE('',*,*,#557,.T.);
#307=ORIENTED_EDGE('',*,*,#553,.F.);
#308=ORIENTED_EDGE('',*,*,#557,.F.);
#309=ORIENTED_EDGE('',*,*,#521,.T.);
#310=ORIENTED_EDGE('',*,*,#558,.T.);
#311=ORIENTED_EDGE('',*,*,#554,.F.);
#312=ORIENTED_EDGE('',*,*,#558,.F.);
#313=ORIENTED_EDGE('',*,*,#520,.T.);
#314=ORIENTED_EDGE('',*,*,#555,.T.);
#315=ORIENTED_EDGE('',*,*,#559,.F.);
#316=ORIENTED_EDGE('',*,*,#547,.T.);
#317=ORIENTED_EDGE('',*,*,#560,.T.);
#318=ORIENTED_EDGE('',*,*,#540,.F.);
```

```
#319=ORIENTED_EDGE('',*,*,#560,.F.);
#320=ORIENTED_EDGE('',*,*,#550,.T.);
#321=ORIENTED_EDGE('',*,*,#561,.T.);
#322=ORIENTED_EDGE('',*,*,#541,.F.);
#323=ORIENTED_EDGE('',*,*,#561,.F.);
#324=ORIENTED_EDGE('',*,*,#549,.T.);
#325=ORIENTED_EDGE('',*,*,#562,.T.);
#326=ORIENTED_EDGE('',*,*,#542,.F.);
#327=ORIENTED_EDGE('',*,*,#562,.F.);
#328=ORIENTED_EDGE('',*,*,#548,.T.);
#329=ORIENTED_EDGE('',*,*,#559,.T.);
#330=ORIENTED_EDGE('',*,*,#539,.F.);
#331=ORIENTED_EDGE('',*,*,#563,.T.);
#332=ORIENTED_EDGE('',*,*,#543,.F.);
#333=ORIENTED_EDGE('',*,*,#564,.F.);
#334=ORIENTED_EDGE('',*,*,#533,.F.);
#335=ORIENTED_EDGE('',*,*,#565,.T.);
#336=ORIENTED_EDGE('',*,*,#544,.F.);
#337=ORIENTED_EDGE('',*,*,#563,.F.);
#338=ORIENTED_EDGE('',*,*,#532,.F.);
#339=ORIENTED_EDGE('',*,*,#566,.T.);
#340=ORIENTED_EDGE('',*,*,#545,.F.);
#341=ORIENTED_EDGE('',*,*,#565,.F.);
#342=ORIENTED_EDGE('',*,*,#531,.F.);
#343=ORIENTED_EDGE('',*,*,#564,.T.);
#344=ORIENTED_EDGE('',*,*,#546,.F.);
#345=ORIENTED_EDGE('',*,*,#566,.F.);
#346=ORIENTED_EDGE('',*,*,#534,.F.);
#347=ORIENTED_EDGE('',*,*,#523,.T.);
```

```
#348=ORIENTED_EDGE('',*,*,#567,.F.);
#349=ORIENTED_EDGE('',*,*,#530,.F.);
#350=ORIENTED_EDGE('',*,*,#568,.T.);
#351=ORIENTED_EDGE('',*,*,#526,.T.);
#352=ORIENTED_EDGE('',*,*,#569,.F.);
#353=ORIENTED_EDGE('',*,*,#527,.F.);
#354=ORIENTED_EDGE('',*,*,#567,.T.);
#355=ORIENTED_EDGE('',*,*,#525,.T.);
#356=ORIENTED_EDGE('',*,*,#570,.F.);
#357=ORIENTED_EDGE('',*,*,#528,.F.);
#358=ORIENTED_EDGE('',*,*,#569,.T.);
#359=ORIENTED_EDGE('',*,*,#524,.T.);
#360=ORIENTED_EDGE('',*,*,#568,.F.);
#361=ORIENTED_EDGE('',*,*,#529,.F.);
#362=ORIENTED_EDGE('',*,*,#570,.T.);
#363=ORIENTED_EDGE('',*,*,#571,.T.);
#364=ORIENTED_EDGE('',*,*,#538,.T.);
#365=ORIENTED_EDGE('',*,*,#572,.F.);
#366=ORIENTED_EDGE('',*,*,#516,.F.);
#367=ORIENTED_EDGE('',*,*,#572,.T.);
#368=ORIENTED_EDGE('',*,*,#537,.T.);
#369=ORIENTED_EDGE('',*,*,#573,.F.);
#370=ORIENTED_EDGE('',*,*,#517,.F.);
#371=ORIENTED_EDGE('',*,*,#573,.T.);
#372=ORIENTED_EDGE('',*,*,#536,.T.);
#373=ORIENTED_EDGE('',*,*,#574,.F.);
#374=ORIENTED_EDGE('',*,*,#518,.F.);
#375=ORIENTED_EDGE('',*,*,#574,.T.);
#376=ORIENTED_EDGE('',*,*,#535,.T.);
```

```
#377=ORIENTED_EDGE('',*,*,#571,.F.);
#378=ORIENTED_EDGE('',*,*,#515,.F.);
#379=ORIENTED_EDGE('',*,*,#575,.F.);
#380=ORIENTED_EDGE('',*,*,#576,.F.);
#381=ORIENTED_EDGE('',*,*,#577,.F.);
#382=ORIENTED_EDGE('',*,*,#578,.F.);
#383=ORIENTED_EDGE('',*,*,#579,.F.);
#384=ORIENTED_EDGE('',*,*,#580,.F.);
#385=ORIENTED_EDGE('',*,*,#581,.F.);
#386=ORIENTED_EDGE('',*,*,#582,.F.);
#387=ORIENTED_EDGE('',*,*,#583,.F.);
#388=ORIENTED_EDGE('',*,*,#584,.F.);
#389=ORIENTED_EDGE('',*,*,#575,.T.);
#390=ORIENTED_EDGE('',*,*,#585,.T.);
#391=ORIENTED_EDGE('',*,*,#586,.F.);
#392=ORIENTED_EDGE('',*,*,#585,.F.);
#393=ORIENTED_EDGE('',*,*,#578,.T.);
#394=ORIENTED_EDGE('',*,*,#587,.T.);
#395=ORIENTED_EDGE('',*,*,#588,.F.);
#396=ORIENTED_EDGE('',*,*,#587,.F.);
#397=ORIENTED_EDGE('',*,*,#577,.T.);
#398=ORIENTED_EDGE('',*,*,#589,.T.);
#399=ORIENTED_EDGE('',*,*,#590,.F.);
#400=ORIENTED_EDGE('',*,*,#589,.F.);
#401=ORIENTED_EDGE('',*,*,#576,.T.);
#402=ORIENTED_EDGE('',*,*,#584,.T.);
#403=ORIENTED_EDGE('',*,*,#591,.F.);
#404=ORIENTED_EDGE('',*,*,#592,.F.);
#405=ORIENTED_EDGE('',*,*,#593,.F.);
```

```
#406=ORIENTED_EDGE('',*,*,#594,.F.);
#407=ORIENTED_EDGE('',*,*,#583,.T.);
#408=ORIENTED_EDGE('',*,*,#586,.T.);
#409=ORIENTED_EDGE('',*,*,#588,.T.);
#410=ORIENTED_EDGE('',*,*,#590,.T.);
#411=ORIENTED_EDGE('',*,*,#579,.T.);
#412=ORIENTED_EDGE('',*,*,#595,.F.);
#413=ORIENTED_EDGE('',*,*,#596,.F.);
#414=ORIENTED_EDGE('',*,*,#597,.T.);
#415=ORIENTED_EDGE('',*,*,#582,.T.);
#416=ORIENTED_EDGE('',*,*,#598,.F.);
#417=ORIENTED_EDGE('',*,*,#599,.F.);
#418=ORIENTED_EDGE('',*,*,#595,.T.);
#419=ORIENTED_EDGE('',*,*,#581,.T.);
#420=ORIENTED_EDGE('',*,*,#600,.F.);
#421=ORIENTED_EDGE('',*,*,#601,.F.);
#422=ORIENTED_EDGE('',*,*,#598,.T.);
#423=ORIENTED_EDGE('',*,*,#580,.T.);
#424=ORIENTED_EDGE('',*,*,#597,.F.);
#425=ORIENTED_EDGE('',*,*,#602,.F.);
#426=ORIENTED_EDGE('',*,*,#600,.T.);
#427=ORIENTED_EDGE('',*,*,#596,.T.);
#428=ORIENTED_EDGE('',*,*,#599,.T.);
#429=ORIENTED_EDGE('',*,*,#601,.T.);
#430=ORIENTED_EDGE('',*,*,#602,.T.);
#431=ORIENTED_EDGE('',*,*,#591,.T.);
#432=ORIENTED_EDGE('',*,*,#603,.T.);
#433=ORIENTED_EDGE('',*,*,#604,.F.);
#434=ORIENTED_EDGE('',*,*,#605,.F.);
```

```
#435=ORIENTED_EDGE('',*,*,#594,.T.);
#436=ORIENTED_EDGE('',*,*,#606,.T.);
#437=ORIENTED_EDGE('',*,*,#607,.F.);
#438=ORIENTED_EDGE('',*,*,#603,.F.);
#439=ORIENTED_EDGE('',*,*,#593,.T.);
#440=ORIENTED_EDGE('',*,*,#608,.T.);
#441=ORIENTED_EDGE('',*,*,#609,.F.);
#442=ORIENTED_EDGE('',*,*,#606,.F.);
#443=ORIENTED_EDGE('',*,*,#592,.T.);
#444=ORIENTED_EDGE('',*,*,#605,.T.);
#445=ORIENTED_EDGE('',*,*,#610,.F.);
#446=ORIENTED_EDGE('',*,*,#608,.F.);
#447=ORIENTED_EDGE('',*,*,#604,.T.);
#448=ORIENTED_EDGE('',*,*,#607,.T.);
#449=ORIENTED_EDGE('',*,*,#609,.T.);
#450=ORIENTED_EDGE('',*,*,#610,.T.);
#451=VERTEX_POINT('',#1036);
#452=VERTEX_POINT('',#1037);
#453=VERTEX_POINT('',#1039);
#454=VERTEX_POINT('',#1041);
#455=VERTEX_POINT('',#1044);
#456=VERTEX_POINT('',#1045);
#457=VERTEX_POINT('',#1047);
#458=VERTEX_POINT('',#1049);
#459=VERTEX_POINT('',#1052);
#460=VERTEX_POINT('',#1053);
#461=VERTEX_POINT('',#1055);
#462=VERTEX_POINT('',#1057);
#463=VERTEX_POINT('',#1061);
```

```
#464=VERTEX_POINT('',#1062);
#465=VERTEX_POINT('',#1064);
#466=VERTEX_POINT('',#1066);
#467=VERTEX_POINT('',#1069);
#468=VERTEX_POINT('',#1070);
#469=VERTEX_POINT('',#1072);
#470=VERTEX_POINT('',#1074);
#471=VERTEX_POINT('',#1078);
#472=VERTEX_POINT('',#1079);
#473=VERTEX_POINT('',#1081);
#474=VERTEX_POINT('',#1083);
#475=VERTEX_POINT('',#1086);
#476=VERTEX_POINT('',#1087);
#477=VERTEX_POINT('',#1089);
#478=VERTEX_POINT('',#1091);
#479=VERTEX_POINT('',#1095);
#480=VERTEX_POINT('',#1096);
#481=VERTEX_POINT('',#1098);
#482=VERTEX_POINT('',#1100);
#483=VERTEX_POINT('',#1103);
#484=VERTEX_POINT('',#1104);
#485=VERTEX_POINT('',#1106);
#486=VERTEX_POINT('',#1108);
#487=VERTEX_POINT('',#1111);
#488=VERTEX_POINT('',#1112);
#489=VERTEX_POINT('',#1114);
#490=VERTEX_POINT('',#1116);
#491=VERTEX_POINT('',#1160);
#492=VERTEX_POINT('',#1161);
```

```
#493=VERTEX_POINT('',#1163);
#494=VERTEX_POINT('',#1165);
#495=VERTEX_POINT('',#1168);
#496=VERTEX_POINT('',#1169);
#497=VERTEX_POINT('',#1171);
#498=VERTEX_POINT('',#1173);
#499=VERTEX_POINT('',#1177);
#500=VERTEX_POINT('',#1178);
#501=VERTEX_POINT('',#1183);
#502=VERTEX_POINT('',#1187);
#503=VERTEX_POINT('',#1193);
#504=VERTEX_POINT('',#1194);
#505=VERTEX_POINT('',#1196);
#506=VERTEX_POINT('',#1198);
#507=VERTEX_POINT('',#1202);
#508=VERTEX_POINT('',#1204);
#509=VERTEX_POINT('',#1208);
#510=VERTEX_POINT('',#1212);
#511=VERTEX_POINT('',#1219);
#512=VERTEX_POINT('',#1221);
#513=VERTEX_POINT('',#1225);
#514=VERTEX_POINT('',#1229);
#515=EDGE_CURVE('',#451,#452,#611,.T.);
#516=EDGE_CURVE('',#452,#453,#612,.T.);
#517=EDGE_CURVE('',#453,#454,#613,.T.);
#518=EDGE_CURVE('',#454,#451,#614,.T.);
#519=EDGE_CURVE('',#455,#456,#615,.T.);
#520=EDGE_CURVE('',#457,#455,#616,.T.);
#521=EDGE_CURVE('',#458,#457,#617,.T.);
```

```
#522=EDGE_CURVE('',#456,#458,#618,.T.);
#523=EDGE_CURVE('',#459,#460,#619,.T.);
#524=EDGE_CURVE('',#461,#459,#620,.T.);
#525=EDGE_CURVE('',#462,#461,#621,.T.);
#526=EDGE_CURVE('',#460,#462,#622,.T.);
#527=EDGE_CURVE('',#463,#464,#623,.T.);
#528=EDGE_CURVE('',#464,#465,#624,.T.);
#529=EDGE_CURVE('',#465,#466,#625,.T.);
#530=EDGE_CURVE('',#466,#463,#626,.T.);
#531=EDGE_CURVE('',#467,#468,#627,.T.);
#532=EDGE_CURVE('',#468,#469,#628,.T.);
#533=EDGE_CURVE('',#469,#470,#629,.T.);
#534=EDGE_CURVE('',#470,#467,#630,.T.);
#535=EDGE_CURVE('',#471,#472,#631,.T.);
#536=EDGE_CURVE('',#473,#471,#632,.T.);
#537=EDGE_CURVE('',#474,#473,#633,.T.);
#538=EDGE_CURVE('',#472,#474,#634,.T.);
#539=EDGE_CURVE('',#475,#476,#635,.T.);
#540=EDGE_CURVE('',#476,#477,#636,.T.);
#541=EDGE_CURVE('',#477,#478,#637,.T.);
#542=EDGE_CURVE('',#478,#475,#638,.T.);
#543=EDGE_CURVE('',#479,#480,#639,.T.);
#544=EDGE_CURVE('',#480,#481,#640,.T.);
#545=EDGE_CURVE('',#481,#482,#641,.T.);
#546=EDGE_CURVE('',#482,#479,#642,.T.);
#547=EDGE_CURVE('',#483,#484,#643,.T.);
#548=EDGE_CURVE('',#485,#483,#644,.T.);
#549=EDGE_CURVE('',#486,#485,#645,.T.);
#550=EDGE_CURVE('',#484,#486,#646,.T.);
```

```
#551=EDGE_CURVE('',#487,#488,#647,.T.);
#552=EDGE_CURVE('',#488,#489,#648,.T.);
#553=EDGE_CURVE('',#489,#490,#649,.T.);
#554=EDGE_CURVE('',#490,#487,#650,.T.);
#555=EDGE_CURVE('',#455,#487,#651,.T.);
#556=EDGE_CURVE('',#456,#488,#652,.T.);
#557=EDGE_CURVE('',#458,#489,#653,.T.);
#558=EDGE_CURVE('',#457,#490,#654,.T.);
#559=EDGE_CURVE('',#483,#476,#655,.T.);
#560=EDGE_CURVE('',#484,#477,#656,.T.);
#561=EDGE_CURVE('',#486,#478,#657,.T.);
#562=EDGE_CURVE('',#485,#475,#658,.T.);
#563=EDGE_CURVE('',#469,#480,#659,.T.);
#564=EDGE_CURVE('',#470,#479,#660,.T.);
#565=EDGE_CURVE('',#468,#481,#661,.T.);
#566=EDGE_CURVE('',#467,#482,#662,.T.);
#567=EDGE_CURVE('',#463,#460,#663,.T.);
#568=EDGE_CURVE('',#466,#459,#664,.T.);
#569=EDGE_CURVE('',#464,#462,#665,.T.);
#570=EDGE_CURVE('',#465,#461,#666,.T.);
#571=EDGE_CURVE('',#452,#472,#667,.T.);
#572=EDGE_CURVE('',#453,#474,#668,.T.);
#573=EDGE_CURVE('',#454,#473,#669,.T.);
#574=EDGE_CURVE('',#451,#471,#670,.T.);
#575=EDGE_CURVE('',#491,#492,#671,.T.);
#576=EDGE_CURVE('',#493,#491,#672,.T.);
#577=EDGE_CURVE('',#494,#493,#673,.T.);
#578=EDGE_CURVE('',#492,#494,#674,.T.);
#579=EDGE_CURVE('',#495,#496,#675,.T.);
```

```
#580=EDGE_CURVE('',#497,#495,#676,.T.);
#581=EDGE_CURVE('',#498,#497,#677,.T.);
#582=EDGE_CURVE('',#496,#498,#678,.T.);
#583=EDGE_CURVE('',#499,#500,#679,.T.);
#584=EDGE_CURVE('',#491,#499,#680,.T.);
#585=EDGE_CURVE('',#492,#500,#681,.T.);
#586=EDGE_CURVE('',#500,#501,#682,.T.);
#587=EDGE_CURVE('',#494,#501,#683,.T.);
#588=EDGE_CURVE('',#501,#502,#684,.T.);
#589=EDGE_CURVE('',#493,#502,#685,.T.);
#590=EDGE_CURVE('',#502,#499,#686,.T.);
#591=EDGE_CURVE('',#503,#504,#687,.T.);
#592=EDGE_CURVE('',#505,#503,#688,.T.);
#593=EDGE_CURVE('',#506,#505,#689,.T.);
#594=EDGE_CURVE('',#504,#506,#690,.T.);
#595=EDGE_CURVE('',#507,#496,#691,.T.);
#596=EDGE_CURVE('',#508,#507,#692,.T.);
#597=EDGE_CURVE('',#508,#495,#693,.T.);
#598=EDGE_CURVE('',#509,#498,#694,.T.);
#599=EDGE_CURVE('',#507,#509,#695,.T.);
#600=EDGE_CURVE('',#510,#497,#696,.T.);
#601=EDGE_CURVE('',#509,#510,#697,.T.);
#602=EDGE_CURVE('',#510,#508,#698,.T.);
#603=EDGE_CURVE('',#504,#511,#699,.T.);
#604=EDGE_CURVE('',#512,#511,#700,.T.);
#605=EDGE_CURVE('',#503,#512,#701,.T.);
#606=EDGE_CURVE('',#506,#513,#702,.T.);
#607=EDGE_CURVE('',#511,#513,#703,.T.);
#608=EDGE_CURVE('',#505,#514,#704,.T.);
```

```
#609=EDGE_CURVE('',#513,#514,#705,.T.);
#610=EDGE_CURVE('',#514,#512,#706,.T.);
#611=LINE('',#1035,#707);
#612=LINE('',#1038,#708);
#613=LINE('',#1040,#709);
#614=LINE('',#1042,#710);
#615=LINE('',#1043,#711);
#616=LINE('',#1046,#712);
#617=LINE('',#1048,#713);
#618=LINE('',#1050,#714);
#619=LINE('',#1051,#715);
#620=LINE('',#1054,#716);
#621=LINE('',#1056,#717);
#622=LINE('',#1058,#718);
#623=LINE('',#1060,#719);
#624=LINE('',#1063,#720);
#625=LINE('',#1065,#721);
#626=LINE('',#1067,#722);
#627=LINE('',#1068,#723);
#628=LINE('',#1071,#724);
#629=LINE('',#1073,#725);
#630=LINE('',#1075,#726);
#631=LINE('',#1077,#727);
#632=LINE('',#1080,#728);
#633=LINE('',#1082,#729);
#634=LINE('',#1084,#730);
#635=LINE('',#1085,#731);
#636=LINE('',#1088,#732);
#637=LINE('',#1090,#733);
```

```
#638=LINE('',#1092,#734);
#639=LINE('',#1094,#735);
#640=LINE('',#1097,#736);
#641=LINE('',#1099,#737);
#642=LINE('',#1101,#738);
#643=LINE('',#1102,#739);
#644=LINE('',#1105,#740);
#645=LINE('',#1107,#741);
#646=LINE('',#1109,#742);
#647=LINE('',#1110,#743);
#648=LINE('',#1113,#744);
#649=LINE('',#1115,#745);
#650=LINE('',#1117,#746);
#651=LINE('',#1119,#747);
#652=LINE('',#1120,#748);
#653=LINE('',#1122,#749);
#654=LINE('',#1124,#750);
#655=LINE('',#1127,#751);
#656=LINE('',#1128,#752);
#657=LINE('',#1130,#753);
#658=LINE('',#1132,#754);
#659=LINE('',#1135,#755);
#660=LINE('',#1136,#756);
#661=LINE('',#1138,#757);
#662=LINE('',#1140,#758);
#663=LINE('',#1143,#759);
#664=LINE('',#1144,#760);
#665=LINE('',#1146,#761);
#666=LINE('',#1148,#762);
```

```
#667=LINE('',#1151,#763);
#668=LINE('',#1152,#764);
#669=LINE('',#1154,#765);
#670=LINE('',#1156,#766);
#671=LINE('',#1159,#767);
#672=LINE('',#1162,#768);
#673=LINE('',#1164,#769);
#674=LINE('',#1166,#770);
#675=LINE('',#1167,#771);
#676=LINE('',#1170,#772);
#677=LINE('',#1172,#773);
#678=LINE('',#1174,#774);
#679=LINE('',#1176,#775);
#680=LINE('',#1179,#776);
#681=LINE('',#1180,#777);
#682=LINE('',#1182,#778);
#683=LINE('',#1184,#779);
#684=LINE('',#1186,#780);
#685=LINE('',#1188,#781);
#686=LINE('',#1190,#782);
#687=LINE('',#1192,#783);
#688=LINE('',#1195,#784);
#689=LINE('',#1197,#785);
#690=LINE('',#1199,#786);
#691=LINE('',#1201,#787);
#692=LINE('',#1203,#788);
#693=LINE('',#1205,#789);
#694=LINE('',#1207,#790);
#695=LINE('',#1209,#791);
```

```
#696=LINE('',#1211,#792);
#697=LINE('',#1213,#793);
#698=LINE('',#1215,#794);
#699=LINE('',#1218,#795);
#700=LINE('',#1220,#796);
#701=LINE('',#1222,#797);
#702=LINE('',#1224,#798);
#703=LINE('',#1226,#799);
#704=LINE('',#1228,#800);
#705=LINE('',#1230,#801);
#706=LINE('',#1232,#802);
#707=VECTOR('',#850,1.);
#708=VECTOR('',#851,1.);
#709=VECTOR('',#852,1.);
#710=VECTOR('',#853,1.);
#711=VECTOR('',#854,1.);
#712=VECTOR('',#855,1.);
#713=VECTOR('',#856,1.);
#714=VECTOR('',#857,1.);
#715=VECTOR('',#858,1.);
#716=VECTOR('',#859,1.);
#717=VECTOR('',#860,1.);
#718=VECTOR('',#861,1.);
#719=VECTOR('',#864,1.);
#720=VECTOR('',#865,1.);
#721=VECTOR('',#866,1.);
#722=VECTOR('',#867,1.);
#723=VECTOR('',#868,1.);
#724=VECTOR('',#869,1.);
```

```
#725=VECTOR('',#870,1.);
#726=VECTOR('',#871,1.);
#727=VECTOR('',#874,1.);
#728=VECTOR('',#875,1.);
#729=VECTOR('',#876,1.);
#730=VECTOR('',#877,1.);
#731=VECTOR('',#878,1.);
#732=VECTOR('',#879,1.);
#733=VECTOR('',#880,1.);
#734=VECTOR('',#881,1.);
#735=VECTOR('',#884,1.);
#736=VECTOR('',#885,1.);
#737=VECTOR('',#886,1.);
#738=VECTOR('',#887,1.);
#739=VECTOR('',#888,1.);
#740=VECTOR('',#889,1.);
#741=VECTOR('',#890,1.);
#742=VECTOR('',#891,1.);
#743=VECTOR('',#892,1.);
#744=VECTOR('',#893,1.);
#745=VECTOR('',#894,1.);
#746=VECTOR('',#895,1.);
#747=VECTOR('',#898,1.);
#748=VECTOR('',#899,1.);
#749=VECTOR('',#902,1.);
#750=VECTOR('',#905,1.);
#751=VECTOR('',#910,1.);
#752=VECTOR('',#911,1.);
#753=VECTOR('',#914,1.);
```

```
#754=VECTOR('',#917,1.);
#755=VECTOR('',#922,1.);
#756=VECTOR('',#923,1.);
#757=VECTOR('',#926,1.);
#758=VECTOR('',#929,1.);
#759=VECTOR('',#934,1.);
#760=VECTOR('',#935,1.);
#761=VECTOR('',#938,1.);
#762=VECTOR('',#941,1.);
#763=VECTOR('',#946,1.);
#764=VECTOR('',#947,1.);
#765=VECTOR('',#950,1.);
#766=VECTOR('',#953,1.);
#767=VECTOR('',#958,1.);
#768=VECTOR('',#959,1.);
#769=VECTOR('',#960,1.);
#770=VECTOR('',#961,1.);
#771=VECTOR('',#962,1.);
#772=VECTOR('',#963,1.);
#773=VECTOR('',#964,1.);
#774=VECTOR('',#965,1.);
#775=VECTOR('',#968,1.);
#776=VECTOR('',#969,1.);
#777=VECTOR('',#970,1.);
#778=VECTOR('',#973,1.);
#779=VECTOR('',#974,1.);
#780=VECTOR('',#977,1.);
#781=VECTOR('',#978,1.);
#782=VECTOR('',#981,1.);
```

```
#783=VECTOR('',#984,1.);
#784=VECTOR('',#985,1.);
#785=VECTOR('',#986,1.);
#786=VECTOR('',#987,1.);
#787=VECTOR('',#990,1.);
#788=VECTOR('',#991,1.);
#789=VECTOR('',#992,1.);
#790=VECTOR('',#995,1.);
#791=VECTOR('',#996,1.);
#792=VECTOR('',#999,1.);
#793=VECTOR('',#1000,1.);
#794=VECTOR('',#1003,1.);
#795=VECTOR('',#1008,1.);
#796=VECTOR('',#1009,1.);
#797=VECTOR('',#1010,1.);
#798=VECTOR('',#1013,1.);
#799=VECTOR('',#1014,1.);
#800=VECTOR('',#1017,1.);
#801=VECTOR('',#1018,1.);
#802=VECTOR('',#1021,1.);
#803=AXIS2_PLACEMENT_3D('',#1034,#848,#849);
#804=AXIS2_PLACEMENT_3D('',#1059,#862,#863);
#805=AXIS2_PLACEMENT_3D('',#1076,#872,#873);
#806=AXIS2_PLACEMENT_3D('',#1093,#882,#883);
#807=AXIS2_PLACEMENT_3D('',#1118,#896,#897);
#808=AXIS2_PLACEMENT_3D('',#1121,#900,#901);
#809=AXIS2_PLACEMENT_3D('',#1123,#903,#904);
#810=AXIS2_PLACEMENT_3D('',#1125,#906,#907);
#811=AXIS2_PLACEMENT_3D('',#1126,#908,#909);
```

```
#812=AXIS2_PLACEMENT_3D('',#1129,#912,#913);
#813=AXIS2_PLACEMENT_3D('',#1131,#915,#916);
#814=AXIS2_PLACEMENT_3D('',#1133,#918,#919);
#815=AXIS2_PLACEMENT_3D('',#1134,#920,#921);
#816=AXIS2_PLACEMENT_3D('',#1137,#924,#925);
#817=AXIS2_PLACEMENT_3D('',#1139,#927,#928);
#818=AXIS2_PLACEMENT_3D('',#1141,#930,#931);
#819=AXIS2_PLACEMENT_3D('',#1142,#932,#933);
#820=AXIS2_PLACEMENT_3D('',#1145,#936,#937);
#821=AXIS2_PLACEMENT_3D('',#1147,#939,#940);
#822=AXIS2_PLACEMENT_3D('',#1149,#942,#943);
#823=AXIS2_PLACEMENT_3D('',#1150,#944,#945);
#824=AXIS2_PLACEMENT_3D('',#1153,#948,#949);
#825=AXIS2_PLACEMENT_3D('',#1155,#951,#952);
#826=AXIS2_PLACEMENT_3D('',#1157,#954,#955);
#827=AXIS2_PLACEMENT_3D('',#1158,#956,#957);
#828=AXIS2_PLACEMENT_3D('',#1175,#966,#967);
#829=AXIS2_PLACEMENT_3D('',#1181,#971,#972);
#830=AXIS2_PLACEMENT_3D('',#1185,#975,#976);
#831=AXIS2_PLACEMENT_3D('',#1189,#979,#980);
#832=AXIS2_PLACEMENT_3D('',#1191,#982,#983);
#833=AXIS2_PLACEMENT_3D('',#1200,#988,#989);
#834=AXIS2_PLACEMENT_3D('',#1206,#993,#994);
#835=AXIS2_PLACEMENT_3D('',#1210,#997,#998);
#836=AXIS2_PLACEMENT_3D('',#1214,#1001,#1002);
#837=AXIS2_PLACEMENT_3D('',#1216,#1004,#1005);
#838=AXIS2_PLACEMENT_3D('',#1217,#1006,#1007);
#839=AXIS2_PLACEMENT_3D('',#1223,#1011,#1012);
#840=AXIS2_PLACEMENT_3D('',#1227,#1015,#1016);
```

```
#841=AXIS2_PLACEMENT_3D('',#1231,#1019,#1020);
#842=AXIS2_PLACEMENT_3D('',#1233,#1022,#1023);
#843=AXIS2_PLACEMENT_3D('',#1234,#1024,#1025);
#844=AXIS2_PLACEMENT_3D('',#1235,#1026,#1027);
#845=AXIS2_PLACEMENT_3D('',#1236,#1028,#1029);
#846=AXIS2_PLACEMENT_3D('',#1237,#1030,#1031);
#847=AXIS2_PLACEMENT_3D('',#1238,#1032,#1033);
#848=DIRECTION('',(0.,0.,1.));
#849=DIRECTION('',(1.,0.,0.));
#850=DIRECTION('',(-1.,0.,0.));
#851=DIRECTION('',(3.46944695195361E-016,0.,-1.));
#852=DIRECTION('',(1.,0.,2.77555756156289E-016));
#853=DIRECTION('',(-1.38777878078145E-016,0.,1.));
#854=DIRECTION('',(-1.,0.,0.));
#855=DIRECTION('',(-5.55111512312578E-016,0.,1.));
#856=DIRECTION('',(1.,0.,2.77555756156289E-016));
#857=DIRECTION('',(2.31296463463574E-016,0.,-1.));
#858=DIRECTION('',(3.46944695195361E-016,0.,1.));
#859=DIRECTION('',(-1.,0.,0.));
#860=DIRECTION('',(-1.38777878078145E-016,0.,-1.));
#861=DIRECTION('',(1.,0.,-2.77555756156289E-016));
#862=DIRECTION('',(0.,-1.,0.));
#863=DIRECTION('',(0.,0.,-1.));
#864=DIRECTION('',(1.,0.,-2.77555756156289E-016));
#865=DIRECTION('',(-1.38777878078145E-016,0.,-1.));
#866=DIRECTION('',(-1.,0.,0.));
#867=DIRECTION('',(3.46944695195361E-016,0.,1.));
#868=DIRECTION('',(-2.168404344971E-015,0.,1.));
#869=DIRECTION('',(-1.,0.,-1.04083408558608E-014));
```

```
#870=DIRECTION('',(0.,0.,-1.));
#871=DIRECTION('',(1.,0.,1.38777878078144E-015));
#872=DIRECTION('',(0.,1.,0.));
#873=DIRECTION('',(0.,0.,1.));
#874=DIRECTION('',(-1.,0.,0.));
#875=DIRECTION('',(-1.38777878078145E-016,0.,1.));
#876=DIRECTION('',(1.,0.,2.77555756156289E-016));
#877=DIRECTION('',(3.46944695195361E-016,0.,-1.));
#878=DIRECTION('',(-1.,0.,1.38777878078144E-015));
#879=DIRECTION('',(0.,0.,-1.));
#880=DIRECTION('',(1.,0.,-1.04083408558608E-014));
#881=DIRECTION('',(2.168404344971E-015,0.,1.));
#882=DIRECTION('',(0.,-1.,0.));
#883=DIRECTION('',(0.,0.,-1.));
#884=DIRECTION('',(0.,0.,1.));
#885=DIRECTION('',(1.,0.,1.04083408558608E-014));
#886=DIRECTION('',(2.168404344971E-015,0.,-1.));
#887=DIRECTION('',(-1.,0.,-1.38777878078144E-015));
#888=DIRECTION('',(0.,0.,-1.));
#889=DIRECTION('',(-1.,0.,1.38777878078144E-015));
#890=DIRECTION('',(2.168404344971E-015,0.,1.));
#891=DIRECTION('',(1.,0.,-1.04083408558608E-014));
#892=DIRECTION('',(-1.,0.,0.));
#893=DIRECTION('',(2.31296463463574E-016,0.,-1.));
#894=DIRECTION('',(1.,0.,2.77555756156289E-016));
#895=DIRECTION('',(-5.55111512312578E-016,0.,1.));
#896=DIRECTION('',(0.,-1.,0.));
#897=DIRECTION('',(0.,0.,-1.));
#898=DIRECTION('',(0.,-1.,0.));
```

```
#899=DIRECTION('',(0.,-1.,0.));
#900=DIRECTION('',(0.,0.,1.));
#901=DIRECTION('',(1.,0.,0.));
#902=DIRECTION('',(0.,-1.,0.));
#903=DIRECTION('',(-1.,0.,-2.31296463463574E-016));
#904=DIRECTION('',(-2.28983498828939E-016,0.,1.));
#905=DIRECTION('',(0.,-1.,0.));
#906=DIRECTION('',(2.77555756156289E-016,0.,-1.));
#907=DIRECTION('',(-1.,0.,-2.77555756156289E-016));
#908=DIRECTION('',(1.,0.,5.55111512312578E-016));
#909=DIRECTION('',(5.55111512312578E-016,0.,-1.));
#910=DIRECTION('',(0.,1.,0.));
#911=DIRECTION('',(0.,1.,0.));
#912=DIRECTION('',(-1.,0.,0.));
#913=DIRECTION('',(0.,0.,1.));
#914=DIRECTION('',(0.,1.,0.));
#915=DIRECTION('',(-1.04083408558608E-014,0.,-1.));
#916=DIRECTION('',(-1.,0.,1.04083408558608E-014));
#917=DIRECTION('',(0.,1.,0.));
#918=DIRECTION('',(1.,0.,-2.168404344971E-015));
#919=DIRECTION('',(-2.16840434497101E-015,0.,-1.));
#920=DIRECTION('',(1.38777878078144E-015,0.,1.));
#921=DIRECTION('',(1.,0.,-1.38777878078145E-015));
#922=DIRECTION('',(0.,-1.,0.));
#923=DIRECTION('',(0.,-1.,0.));
#924=DIRECTION('',(1.,0.,0.));
#925=DIRECTION('',(0.,0.,-1.));
#926=DIRECTION('',(0.,-1.,0.));
#927=DIRECTION('',(1.04083408558608E-014,0.,-1.));
```

```
#928=DIRECTION('',(-1.,0.,-1.04083408558608E-014));
#929=DIRECTION('',(0.,-1.,0.));
#930=DIRECTION('',(-1.,0.,-2.168404344971E-015));
#931=DIRECTION('',(-2.16840434497101E-015,0.,1.));
#932=DIRECTION('',(-1.38777878078144E-015,0.,1.));
#933=DIRECTION('',(1.,0.,1.38777878078145E-015));
#934=DIRECTION('',(0.,1.,0.));
#935=DIRECTION('',(0.,1.,0.));
#936=DIRECTION('',(1.,0.,-3.46944695195361E-016));
#937=DIRECTION('',(-3.46944695195361E-016,0.,-1.));
#938=DIRECTION('',(0.,1.,0.));
#939=DIRECTION('',(-2.77555756156289E-016,0.,-1.));
#940=DIRECTION('',(-1.,0.,2.77555756156289E-016));
#941=DIRECTION('',(0.,1.,0.));
#942=DIRECTION('',(-1.,0.,1.38777878078145E-016));
#943=DIRECTION('',(1.38777878078145E-016,0.,1.));
#944=DIRECTION('',(0.,0.,1.));
#945=DIRECTION('',(1.,0.,0.));
#946=DIRECTION('',(0.,-1.,0.));
#947=DIRECTION('',(0.,-1.,0.));
#948=DIRECTION('',(-1.,0.,-3.46944695195361E-016));
#949=DIRECTION('',(-3.46944695195361E-016,0.,1.));
#950=DIRECTION('',(0.,-1.,0.));
#951=DIRECTION('',(2.77555756156289E-016,0.,-1.));
#952=DIRECTION('',(-1.,0.,-2.77555756156289E-016));
#953=DIRECTION('',(0.,-1.,0.));
#954=DIRECTION('',(1.,0.,1.38777878078145E-016));
#955=DIRECTION('',(1.38777878078145E-016,0.,-1.));
#956=DIRECTION('',(0.,0.,1.));
```

```
#957=DIRECTION('',(1.,0.,0.));
#958=DIRECTION('',(1.,-1.38777878078145E-016,0.));
#959=DIRECTION('',(-1.38777878078145E-016,1.,0.));
#960=DIRECTION('',(-1.,-1.38777878078145E-016,0.));
#961=DIRECTION('',(5.20417042793042E-017,-1.,0.));
#962=DIRECTION('',(5.55111512312578E-016,-1.,0.));
#963=DIRECTION('',(-1.,0.,0.));
#964=DIRECTION('',(-2.77555756156289E-016,1.,0.));
#965=DIRECTION('',(1.,0.,0.));
#966=DIRECTION('',(0.,0.,-1.));
#967=DIRECTION('',(-1.,0.,0.));
#968=DIRECTION('',(1.,-1.38777878078145E-016,0.));
#969=DIRECTION('',(0.,0.,-1.));
#970=DIRECTION('',(0.,0.,-1.));
#971=DIRECTION('',(1.38777878078145E-016,1.,0.));
#972=DIRECTION('',(-1.,1.38777878078145E-016,0.));
#973=DIRECTION('',(5.20417042793042E-017,-1.,0.));
#974=DIRECTION('',(0.,0.,-1.));
#975=DIRECTION('',(1.,5.20417042793042E-017,0.));
#976=DIRECTION('',(-5.20417042793042E-017,1.,0.));
#977=DIRECTION('',(-1.,-1.38777878078145E-016,0.));
#978=DIRECTION('',(0.,0.,-1.));
#979=DIRECTION('',(1.38777878078145E-016,-1.,0.));
#980=DIRECTION('',(1.,1.38777878078145E-016,0.));
#981=DIRECTION('',(-1.38777878078145E-016,1.,0.));
#982=DIRECTION('',(-1.,-1.38777878078145E-016,0.));
#983=DIRECTION('',(1.38777878078145E-016,-1.,0.));
#984=DIRECTION('',(-5.55111512312578E-016,1.,0.));
#985=DIRECTION('',(-1.,0.,0.));
```

```
#986=DIRECTION('',(2.77555756156289E-016,-1.,0.));
#987=DIRECTION('',(1.,0.,0.));
#988=DIRECTION('',(0.,0.,-1.));
#989=DIRECTION('',(-1.,0.,0.));
#990=DIRECTION('',(0.,0.,-1.));
#991=DIRECTION('',(5.55111512312578E-016,-1.,0.));
#992=DIRECTION('',(0.,0.,-1.));
#993=DIRECTION('',(1.,5.55111512312578E-016,0.));
#994=DIRECTION('',(-5.55111512312578E-016,1.,0.));
#995=DIRECTION('',(0.,0.,-1.));
#996=DIRECTION('',(1.,0.,0.));
#997=DIRECTION('',(0.,1.,0.));
#998=DIRECTION('',(0.,0.,1.));
#999=DIRECTION('',(0.,0.,-1.));
#1000=DIRECTION('',(-2.77555756156289E-016,1.,0.));
#1001=DIRECTION('',(-1.,-2.77555756156289E-016,0.));
#1002=DIRECTION('',(2.77555756156289E-016,-1.,0.));
#1003=DIRECTION('',(-1.,0.,0.));
#1004=DIRECTION('',(0.,-1.,0.));
#1005=DIRECTION('',(0.,0.,-1.));
#1006=DIRECTION('',(0.,0.,1.));
#1007=DIRECTION('',(1.,0.,0.));
#1008=DIRECTION('',(0.,0.,-1.));
#1009=DIRECTION('',(-5.55111512312578E-016,1.,0.));
#1010=DIRECTION('',(0.,0.,-1.));
#1011=DIRECTION('',(1.,5.55111512312578E-016,0.));
#1012=DIRECTION('',(-5.55111512312578E-016,1.,0.));
#1013=DIRECTION('',(0.,0.,-1.));
#1014=DIRECTION('',(1.,0.,0.));
```

```
#1015=DIRECTION('',(0.,-1.,0.));
#1016=DIRECTION('',(0.,0.,-1.));
#1017=DIRECTION('',(0.,0.,-1.));
#1018=DIRECTION('',(2.77555756156289E-016,-1.,0.));
#1019=DIRECTION('',(-1.,-2.77555756156289E-016,0.));
#1020=DIRECTION('',(2.77555756156289E-016,-1.,0.));
#1021=DIRECTION('',(-1.,0.,0.));
#1022=DIRECTION('',(0.,1.,0.));
#1023=DIRECTION('',(0.,0.,1.));
#1024=DIRECTION('',(0.,0.,-1.));
#1025=DIRECTION('',(-1.,0.,0.));
#1026=DIRECTION('',(0.,1.,0.));
#1027=DIRECTION('',(1.,0.,0.));
#1028=DIRECTION('',(0.,0.,1.));
#1029=DIRECTION('',(1.,0.,0.));
#1030=DIRECTION('',(0.,0.,1.));
#1031=DIRECTION('',(1.,0.,0.));
#1032=DIRECTION('',(0.,-1.,0.));
#1033=DIRECTION('',(1.,0.,0.));
#1034=CARTESIAN_POINT('',(0.,0.,0.));
#1035=CARTESIAN_POINT('',(-50.,25.,62.5));
#1036=CARTESIAN_POINT('',(12.5000000000001,25.,62.5));
#1037=CARTESIAN_POINT('',(-12.4999999999999,25.,62.5));
#1038=CARTESIAN_POINT('',(-12.4999999999999,25.,75.));
#1039=CARTESIAN_POINT('',(-12.4999999999999,25.,37.5));
#1040=CARTESIAN_POINT('',(-49.9999999999999,25.,37.5));
#1041=CARTESIAN_POINT('',(12.5000000000001,25.,37.5));
#1042=CARTESIAN_POINT('',(12.5000000000001,25.,75.));
#1043=CARTESIAN_POINT('',(50.,25.,75.));
```

```
#1044=CARTESIAN_POINT('',(50.,25.,75.));
#1045=CARTESIAN_POINT('',(-50.,25.,75.));
#1046=CARTESIAN_POINT('',(50.0000000000001,25.,-75.));
#1047=CARTESIAN_POINT('',(50.0000000000001,25.,-75.));
#1048=CARTESIAN_POINT('',(-49.9999999999999,25.,-75.));
#1049=CARTESIAN_POINT('',(-49.9999999999999,25.,-75.));
#1050=CARTESIAN_POINT('',(-50.,25.,75.));
#1051=CARTESIAN_POINT('',(-12.4999999999999,25.,75.));
#1052=CARTESIAN_POINT('',(-12.4999999999999,25.,-62.5));
#1053=CARTESIAN_POINT('',(-12.4999999999999,25.,-37.5));
#1054=CARTESIAN_POINT('',(-50.,25.,-62.5));
#1055=CARTESIAN_POINT('',(12.5000000000001,25.,-62.5));
#1056=CARTESIAN_POINT('',(12.5000000000001,25.,75.));
#1057=CARTESIAN_POINT('',(12.5000000000001,25.,-37.5));
#1058=CARTESIAN_POINT('',(-50.,25.,-37.5));
#1059=CARTESIAN_POINT('',(-50.,25.,75.));
#1060=CARTESIAN_POINT('',(-12.4999999999999,12.5,-37.5));
#1061=CARTESIAN_POINT('',(-12.4999999999999,12.5,-37.5));
#1062=CARTESIAN_POINT('',(12.5000000000001,12.5,-37.5));
#1063=CARTESIAN_POINT('',(12.5000000000001,12.5,-37.5));
#1064=CARTESIAN_POINT('',(12.5000000000001,12.5,-62.5));
#1065=CARTESIAN_POINT('',(12.5000000000001,12.5,-62.5));
#1066=CARTESIAN_POINT('',(-12.4999999999999,12.5,-62.5));
#1067=CARTESIAN_POINT('',(-12.4999999999999,12.5,-62.5));
#1068=CARTESIAN_POINT('',(5.00000000000005,12.5,-50.));
#1069=CARTESIAN_POINT('',(5.00000000000006,12.5,-55.));
#1070=CARTESIAN_POINT('',(5.00000000000004,12.5,-45.));
#1071=CARTESIAN_POINT('',(-2.36199948489002E-014,12.5,-45.));
#1072=CARTESIAN_POINT('',(-4.99999999999997,12.5,-
```

```
45.0000000000001));
#1073=CARTESIAN_POINT('',(-4.99999999999997,12.5,-50.));
#1074=CARTESIAN_POINT('',(-4.99999999999997,12.5,-55.));
#1075=CARTESIAN_POINT('',(3.53606033343112E-014,12.5,-55.));
#1076=CARTESIAN_POINT('',(2.8421709430404E-014,12.5,-50.));
#1077=CARTESIAN_POINT('',(12.5000000000001,12.5,62.5));
#1078=CARTESIAN_POINT('',(12.5000000000001,12.5,62.5));
#1079=CARTESIAN_POINT('',(-12.4999999999999,12.5,62.5));
#1080=CARTESIAN_POINT('',(12.5000000000001,12.5,37.5));
#1081=CARTESIAN_POINT('',(12.5000000000001,12.5,37.5));
#1082=CARTESIAN_POINT('',(-12.4999999999999,12.5,37.5));
#1083=CARTESIAN_POINT('',(-12.4999999999999,12.5,37.5));
#1084=CARTESIAN_POINT('',(-12.4999999999999,12.5,62.5));
#1085=CARTESIAN_POINT('',(3.53606033343112E-014,12.5,55.));
#1086=CARTESIAN_POINT('',(5.00000000000006,12.5,55.));
#1087=CARTESIAN_POINT('',(-4.99999999999997,12.5,55.));
#1088=CARTESIAN_POINT('',(-4.99999999999997,12.5,50.));
#1089=CARTESIAN_POINT('',(-
4.99999999999997,12.5,45.0000000000001));
#1090=CARTESIAN_POINT('',(-2.36199948489002E-014,12.5,45.));
#1091=CARTESIAN_POINT('',(5.00000000000004,12.5,45.));
#1092=CARTESIAN_POINT('',(5.00000000000005,12.5,50.));
#1093=CARTESIAN_POINT('',(2.8421709430404E-014,12.5,50.));
#1094=CARTESIAN_POINT('',(-4.99999999999997,0.,-55.));
#1095=CARTESIAN_POINT('',(-4.99999999999997,0.,-55.));
#1096=CARTESIAN_POINT('',(-4.99999999999997,0.,-
45.0000000000001));
#1097=CARTESIAN_POINT('',(-4.99999999999997,0.,-
45.0000000000001));
```

```
#1098=CARTESIAN_POINT('',(5.00000000000004,0.,-45.));
#1099=CARTESIAN_POINT('',(5.00000000000004,0.,-45.));
#1100=CARTESIAN_POINT('',(5.00000000000006,0.,-55.));
#1101=CARTESIAN_POINT('',(5.00000000000006,0.,-55.));
#1102=CARTESIAN_POINT('',(-4.99999999999997,0.,55.));
#1103=CARTESIAN_POINT('',(-4.99999999999997,0.,55.));
#1104=CARTESIAN_POINT('',(-
4.99999999999997,0.,45.0000000000001));
#1105=CARTESIAN_POINT('',(5.00000000000006,0.,55.));
#1106=CARTESIAN_POINT('',(5.00000000000006,0.,55.));
#1107=CARTESIAN_POINT('',(5.00000000000004,0.,45.));
#1108=CARTESIAN_POINT('',(5.00000000000004,0.,45.));
#1109=CARTESIAN_POINT('',(-
4.99999999999997,0.,45.0000000000001));
#1110=CARTESIAN_POINT('',(50.,0.,75.));
#1111=CARTESIAN_POINT('',(50.,0.,75.));
#1112=CARTESIAN_POINT('',(-50.,0.,75.));
#1113=CARTESIAN_POINT('',(-50.,0.,75.));
#1114=CARTESIAN_POINT('',(-49.9999999999999,0.,-75.));
#1115=CARTESIAN_POINT('',(-49.9999999999999,0.,-75.));
#1116=CARTESIAN_POINT('',(50.0000000000001,0.,-75.));
#1117=CARTESIAN_POINT('',(50.0000000000001,0.,-75.));
#1118=CARTESIAN_POINT('',(-50.,0.,75.));
#1119=CARTESIAN_POINT('',(50.,25.,75.));
#1120=CARTESIAN_POINT('',(-50.,25.,75.));
#1121=CARTESIAN_POINT('',(50.,25.,75.));
#1122=CARTESIAN_POINT('',(-49.9999999999999,25.,-75.));
#1123=CARTESIAN_POINT('',(-50.,25.,75.));
#1124=CARTESIAN_POINT('',(50.0000000000001,25.,-75.));
```

```
#1125=CARTESIAN_POINT('',(-49.9999999999999,25.,-75.));
#1126=CARTESIAN_POINT('',(50.0000000000001,25.,-75.));
#1127=CARTESIAN_POINT('',(-4.99999999999997,56.,55.));
#1128=CARTESIAN_POINT('',(-
4.99999999999997,56.,45.0000000000001));
#1129=CARTESIAN_POINT('',(-4.99999999999997,56.,55.));
#1130=CARTESIAN_POINT('',(5.00000000000004,56.,45.));
#1131=CARTESIAN_POINT('',(-
4.99999999999997,56.,45.0000000000001));
#1132=CARTESIAN_POINT('',(5.00000000000006,56.,55.));
#1133=CARTESIAN_POINT('',(5.00000000000004,56.,45.));
#1134=CARTESIAN_POINT('',(5.00000000000006,56.,55.));
#1135=CARTESIAN_POINT('',(-4.99999999999997,56.,-
45.0000000000001));
#1136=CARTESIAN_POINT('',(-4.99999999999997,56.,-55.));
#1137=CARTESIAN_POINT('',(-4.99999999999997,56.,-55.));
#1138=CARTESIAN_POINT('',(5.00000000000004,56.,-45.));
#1139=CARTESIAN_POINT('',(-4.99999999999997,56.,-
45.0000000000001));
#1140=CARTESIAN_POINT('',(5.00000000000006,56.,-55.));
#1141=CARTESIAN_POINT('',(5.00000000000004,56.,-45.));
#1142=CARTESIAN_POINT('',(5.00000000000006,56.,-55.));
#1143=CARTESIAN_POINT('',(-12.4999999999999,29.5,-37.5));
#1144=CARTESIAN_POINT('',(-12.4999999999999,29.5,-62.5));
#1145=CARTESIAN_POINT('',(-12.4999999999999,29.5,-62.5));
#1146=CARTESIAN_POINT('',(12.5000000000001,29.5,-37.5));
#1147=CARTESIAN_POINT('',(-12.4999999999999,29.5,-37.5));
#1148=CARTESIAN_POINT('',(12.5000000000001,29.5,-62.5));
#1149=CARTESIAN_POINT('',(12.5000000000001,29.5,-37.5));
```

```
#1150=CARTESIAN_POINT('',(12.5000000000001,29.5,-62.5));
#1151=CARTESIAN_POINT('',(-12.4999999999999,29.5,62.5));
#1152=CARTESIAN_POINT('',(-12.4999999999999,29.5,37.5));
#1153=CARTESIAN_POINT('',(-12.4999999999999,29.5,62.5));
#1154=CARTESIAN_POINT('',(12.5000000000001,29.5,37.5));
#1155=CARTESIAN_POINT('',(-12.4999999999999,29.5,37.5));
#1156=CARTESIAN_POINT('',(12.5000000000001,29.5,62.5));
#1157=CARTESIAN_POINT('',(12.5000000000001,29.5,37.5));
#1158=CARTESIAN_POINT('',(12.5000000000001,29.5,62.5));
#1159=CARTESIAN_POINT('',(-
23.0422426986387,23.029526813094,175.));
#1160=CARTESIAN_POINT('',(-
23.0422426986387,23.029526813094,175.));
#1161=CARTESIAN_POINT('',(1.95775730136131,23.029526813094,175
.));
#1162=CARTESIAN_POINT('',(-23.0422426986387,-
1.970473186906,175.));
#1163=CARTESIAN_POINT('',(-23.0422426986387,-
1.97047318690599,175.));
#1164=CARTESIAN_POINT('',(1.95775730136131,-
1.97047318690599,175.));
#1165=CARTESIAN_POINT('',(1.95775730136131,-
1.97047318690599,175.));
#1166=CARTESIAN_POINT('',(1.95775730136131,23.029526813094,175
.));
#1167=CARTESIAN_POINT('',(-
16.7922426986387,16.779526813094,175.));
#1168=CARTESIAN_POINT('',(-
16.7922426986387,16.779526813094,175.));
```

```
#1169=CARTESIAN_POINT('',(-
16.7922426986387,4.27952681309401,175.));
#1170=CARTESIAN_POINT('',(-
4.2922426986387,16.779526813094,175.));
#1171=CARTESIAN_POINT('',(-
4.2922426986387,16.779526813094,175.));
#1172=CARTESIAN_POINT('',(-
4.29224269863869,4.27952681309401,175.));
#1173=CARTESIAN_POINT('',(-
4.29224269863869,4.27952681309401,175.));
#1174=CARTESIAN_POINT('',(-
16.7922426986387,4.27952681309401,175.));
#1175=CARTESIAN_POINT('',(0.,0.,175.));
#1176=CARTESIAN_POINT('',(-
23.0422426986387,23.029526813094,0.));
#1177=CARTESIAN_POINT('',(-
23.0422426986387,23.029526813094,0.));
#1178=CARTESIAN_POINT('',(1.95775730136131,23.029526813094,0.));
#1179=CARTESIAN_POINT('',(-
23.0422426986387,23.029526813094,175.));
#1180=CARTESIAN_POINT('',(1.95775730136131,23.029526813094,175
.));
#1181=CARTESIAN_POINT('',(-
23.0422426986387,23.029526813094,175.));
#1182=CARTESIAN_POINT('',(1.95775730136131,23.029526813094,0.));
#1183=CARTESIAN_POINT('',(1.95775730136131,-
1.97047318690599,0.));
#1184=CARTESIAN_POINT('',(1.95775730136131,-
1.97047318690599,175.));
```

```
#1185=CARTESIAN_POINT('',(1.95775730136131,23.029526813094,175
.));
#1186=CARTESIAN_POINT('',(1.95775730136131,-
1.97047318690599,0.));
#1187=CARTESIAN_POINT('',(-23.0422426986387,-
1.97047318690599,0.));
#1188=CARTESIAN_POINT('',(-23.0422426986387,-
1.97047318690599,175.));
#1189=CARTESIAN_POINT('',(1.95775730136131,-
1.97047318690599,175.));
#1190=CARTESIAN_POINT('',(-23.0422426986387,-1.970473186906,0.));
#1191=CARTESIAN_POINT('',(-23.0422426986387,-
1.970473186906,175.));
#1192=CARTESIAN_POINT('',(-
16.7922426986387,4.27952681309401,5.6843418860808E-014));
#1193=CARTESIAN_POINT('',(-
16.7922426986387,4.27952681309401,5.6843418860808E-014));
#1194=CARTESIAN_POINT('',(-
16.7922426986387,16.779526813094,5.6843418860808E-014));
#1195=CARTESIAN_POINT('',(-
4.29224269863869,4.27952681309401,5.6843418860808E-014));
#1196=CARTESIAN_POINT('',(-
4.29224269863869,4.279526813094,5.6843418860808E-014));
#1197=CARTESIAN_POINT('',(-
4.2922426986387,16.779526813094,5.6843418860808E-014));
#1198=CARTESIAN_POINT('',(-
4.2922426986387,16.779526813094,5.6843418860808E-014));
#1199=CARTESIAN_POINT('',(-
16.7922426986387,16.779526813094,5.6843418860808E-014));
```

```
#1200=CARTESIAN_POINT('',(0.,0.,0.));
#1201=CARTESIAN_POINT('',(-
16.7922426986387,4.27952681309401,225.));
#1202=CARTESIAN_POINT('',(-
16.7922426986387,4.27952681309401,225.));
#1203=CARTESIAN_POINT('',(-
16.7922426986387,16.779526813094,225.));
#1204=CARTESIAN_POINT('',(-
16.7922426986387,16.779526813094,225.));
#1205=CARTESIAN_POINT('',(-
16.7922426986387,16.779526813094,225.));
#1206=CARTESIAN_POINT('',(-
16.7922426986387,16.779526813094,225.));
#1207=CARTESIAN_POINT('',(-
4.29224269863869,4.27952681309401,225.));
#1208=CARTESIAN_POINT('',(-
4.29224269863869,4.27952681309401,225.));
#1209=CARTESIAN_POINT('',(-
16.7922426986387,4.27952681309401,225.));
#1210=CARTESIAN_POINT('',(-
16.7922426986387,4.27952681309401,225.));
#1211=CARTESIAN_POINT('',(-
4.2922426986387,16.779526813094,225.));
#1212=CARTESIAN_POINT('',(-
4.2922426986387,16.779526813094,225.));
#1213=CARTESIAN_POINT('',(-
4.29224269863869,4.27952681309401,225.));
#1214=CARTESIAN_POINT('',(-
4.29224269863869,4.27952681309401,225.));
```

```
#1215=CARTESIAN_POINT('',(-
4.2922426986387,16.779526813094,225.));
#1216=CARTESIAN_POINT('',(-
4.2922426986387,16.779526813094,225.));
#1217=CARTESIAN_POINT('',(0.,0.,225.));
#1218=CARTESIAN_POINT('',(-16.7922426986387,16.779526813094,-
50.));
#1219=CARTESIAN_POINT('',(-16.7922426986387,16.779526813094,-
50.));
#1220=CARTESIAN_POINT('',(-16.7922426986387,4.27952681309401,-
50.));
#1221=CARTESIAN_POINT('',(-16.7922426986387,4.27952681309401,-
50.));
#1222=CARTESIAN_POINT('',(-16.7922426986387,4.27952681309401,-
50.));
#1223=CARTESIAN_POINT('',(-16.7922426986387,4.27952681309401,-
50.));
#1224=CARTESIAN_POINT('',(-4.2922426986387,16.779526813094,-
50.));
#1225=CARTESIAN_POINT('',(-4.2922426986387,16.779526813094,-
50.));
#1226=CARTESIAN_POINT('',(-16.7922426986387,16.779526813094,-
50.));
#1227=CARTESIAN_POINT('',(-16.7922426986387,16.779526813094,-
50.));
#1228=CARTESIAN_POINT('',(-4.29224269863869,4.279526813094,-
50.));
#1229=CARTESIAN_POINT('',(-4.29224269863869,4.279526813094,-
50.));
```

```
#1230=CARTESIAN_POINT('',(-4.2922426986387,16.779526813094,-
50.));
#1231=CARTESIAN_POINT('',(-4.2922426986387,16.779526813094,-
50.));
#1232=CARTESIAN_POINT('',(-4.29224269863869,4.27952681309401,-
50.));
#1233=CARTESIAN_POINT('',(-4.29224269863869,4.27952681309401,-
50.));
#1234=CARTESIAN_POINT('',(0.,0.,-50.));
#1235=CARTESIAN_POINT('',(81.0470455486706,62.5000007450581,-
47.4082082695946));
#1236=CARTESIAN_POINT('',(91.5892882473094,101.970473931964,-
234.908208269595));
#1237=CARTESIAN_POINT('',(91.5892882473094,1.97047393196404,-
234.908208269595));
#1238=CARTESIAN_POINT('',(81.0470455486706,62.5000007450581,-
247.408208269595));
#1239=MECHANICAL_DESIGN_GEOMETRIC_PRESENTATION_REPRESE
NTATION('',(#64),
#1242);
#1240=MECHANICAL_DESIGN_GEOMETRIC_PRESENTATION_REPRESE
NTATION('',(#65),
#1243);
#1241=(
GEOMETRIC_REPRESENTATION_CONTEXT(3)
GLOBAL_UNCERTAINTY_ASSIGNED_CONTEXT((#1244))
GLOBAL_UNIT_ASSIGNED_CONTEXT((#1252,#1248,#1247))
REPRESENTATION_CONTEXT('platerod','TOP_LEVEL_ASSEMBLY_PART')
);
```

```
#1242=(
GEOMETRIC_REPRESENTATION_CONTEXT(3)
GLOBAL_UNCERTAINTY_ASSIGNED_CONTEXT((#1245))
GLOBAL_UNIT_ASSIGNED_CONTEXT((#1252,#1248,#1247))
REPRESENTATION_CONTEXT('plate','COMPONENT_PART')
);
#1243=(
GEOMETRIC_REPRESENTATION_CONTEXT(3)
GLOBAL_UNCERTAINTY_ASSIGNED_CONTEXT((#1246))
GLOBAL_UNIT_ASSIGNED_CONTEXT((#1252,#1248,#1247))
REPRESENTATION_CONTEXT('rod','COMPONENT_PART')
);
#1244=UNCERTAINTY_MEASURE_WITH_UNIT(LENGTH_MEASURE(2.E-
005),#1252,
'DISTANCE_ACCURACY_VALUE','Maximum Tolerance applied to model');
#1245=UNCERTAINTY_MEASURE_WITH_UNIT(LENGTH_MEASURE(2.E-
005),#1252,
'DISTANCE_ACCURACY_VALUE','Maximum Tolerance applied to model');
#1246=UNCERTAINTY_MEASURE_WITH_UNIT(LENGTH_MEASURE(2.E-
005),#1252,
'DISTANCE_ACCURACY_VALUE','Maximum Tolerance applied to model');
#1247=(
NAMED_UNIT(*)
SI_UNIT($,.STERADIAN.)
SOLID_ANGLE_UNIT()
);
#1248=(
CONVERSION_BASED_UNIT('DEGREE',#1250)
NAMED_UNIT(#1249)
```

```
PLANE_ANGLE_UNIT()
);
#1249=DIMENSIONAL_EXPONENTS(0.,0.,0.,0.,0.,0.);
#1250=PLANE_ANGLE_MEASURE_WITH_UNIT(PLANE_ANGLE_MEASUR
E(0.0174532925),#1251);
#1251=(
NAMED_UNIT(*)
PLANE_ANGLE_UNIT()
SI_UNIT($,.RADIAN.)
);
#1252=(
LENGTH_UNIT()
NAMED_UNIT(*)
SI_UNIT(.MILLI.,.METRE.)
);
ENDSEC;
END-ISO-10303-21;
```

Appendix D: OpenSCAD Scripts

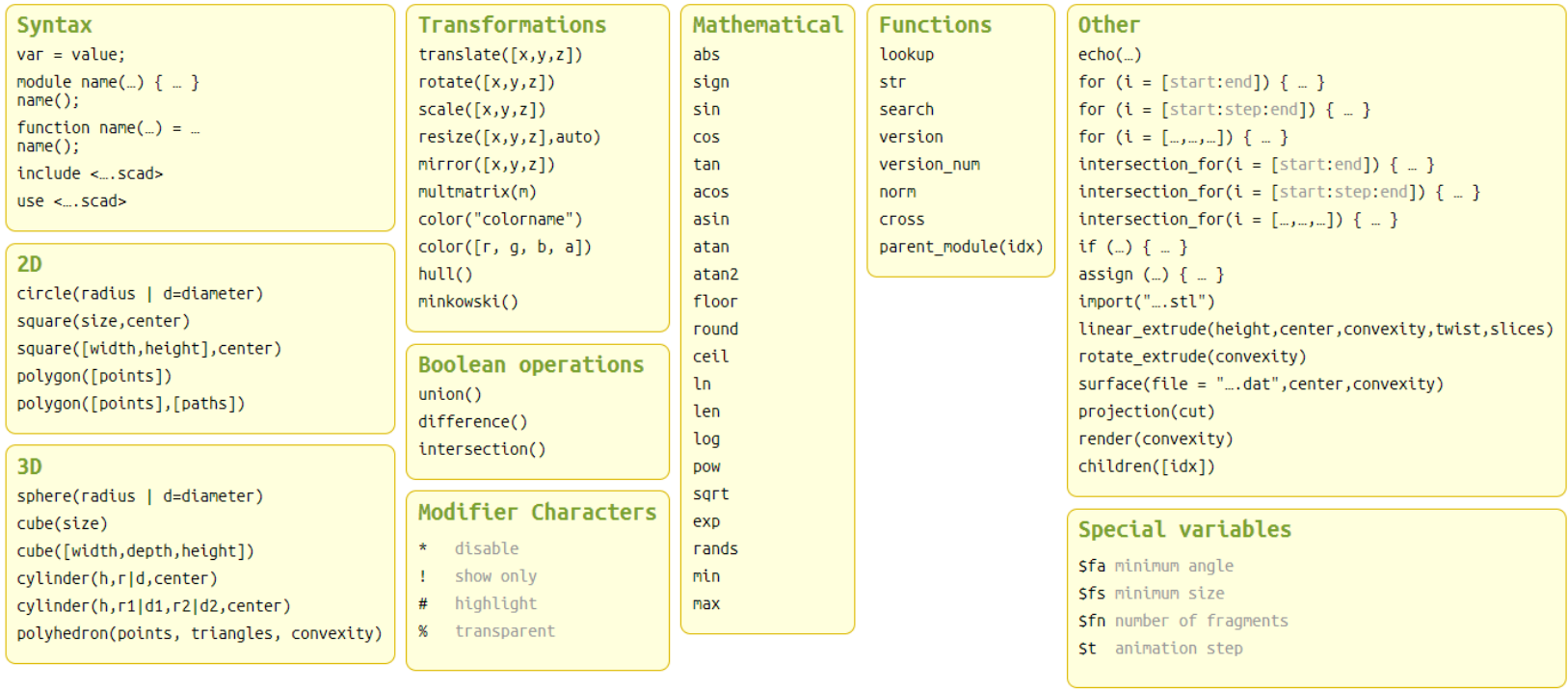


Figure 36: OpenSCAD scripts for 3D modeling [34]

Thesis and Dissertation Services