Interoperability of Geometric Dimension & Tolerance Data between CAD Systems through ISO STEP AP 242

by

Adarsh Venkiteswaran

A Thesis Presented in Partial Fulfillment
of the Requirements for the Degree
Master of Science

Approved July 2016 by the
Graduate Supervisory Committee:

Jami. J. Shah, Chair
Martin Hardwick
Joseph. K. Davidson

ARIZONA STATE UNIVERSITY

August 2016

ABSTRACT

There is very little in the way of prescriptive procedures to guide designers in tolerance specification. This shortcoming motivated the group at Design Automation Lab to automate tolerancing of mechanical assemblies. GD&T data generated by the Auto-Tolerancing software is semantically represented using a neutral Constraint Tolerance Feature (CTF) graph file format that is consistent with the ASME Y14.5 standard and the ISO STEP Part 21 file. The primary objective of this research is to communicate GD&T information from the CTF file to a neutral machine readable format. The latest STEP AP 242 (ISO 10303-242) "Managed model based 3D engineering" aims to support smart manufacturing by capturing semantic Product Manufacturing Information (PMI) within the 3D model and also helping with long-term archiving of the product information. In line with the recommended practices published by CAx Implementor Forum, this research discusses the implementation of CTF to AP 242 translator. The input geometry available in STEP AP 203 format is pre-processed using STEP-NC DLL and 3D InterOp. While the former is initially used to attach persistent IDs to the topological entities in STEP, the latter retains the IDs during translation to ACIS entities for consumption by other modules in the Auto-tolerancing module. The associativity of GD&T available in CTF file to the input geometry is through persistent IDs. C++ libraries used for the translation to STEP AP 242 is provided by StepTools Inc through the STEP-NC DLL. Finally, the output STEP file is tested using available AP 242 readers and shows full conformance with the STEP standard. Using the output AP 242 file, semantic GDT data can now be automatically consumed by downstream applications such as Computer Aided Process Planning (CAPP), Computer Aided Inspection (CAI), Computer Aided Tolerance Systems (CATS) and Coordinate Measuring Machines (CMM).

ACKNOWLEDGMENTS

I would first like to thank my thesis adviser Dr. Jami. J. Shah for providing me with numerous challenges during my graduate life. Dr. Shah has continually shifted goalposts and instilled in me the confidence and patience required to approach open-ended problems.

I would also like to thank Dr. Martin Hardwick for his generous and timely support with the STEP Libraries that would prove to be an integral part of this work. In addition, I would like to acknowledge Dr. Joseph K Davidson, as the second reader of this thesis, and I am gratefully indebted to his valuable comments on this research.

Finally, I must express my profound gratitude to my parents, colleagues at the Design Automation Lab and friends for providing me with unwavering support and continuous encouragement throughout my years of study. This accomplishment would not have been possible without them.

TABLE OF CONTENTS

CHAPTER Page

CHAPTER Page

LIST OF TABLES

LIST OF FIGURES

Figure Page

Chapter 1

INTRODUCTION

1.1 Problem Definition

Nominal size or dimension is a widely used misnomer in the manufacturing industry. Even the latest technologies cannot guarantee components that are manufactured precisely to the nominal dimensions specified by the designer. However, proper functioning of these systems can still be ensured by specifying permissible variations to the nominal sizes of these manufacturing features. The scientific way of determining and representing these variations is known as tolerance specification. Proper tolerances aid manufacturability and assemblability of these components thereby reducing scrap rate. Industries use a broad range of Computer Aided Design(CAD) systems to design these assemblies. With the advent of CAD, 3D models have been widely used in engineering design and development instead of paper drawings. CAD systems have helped improve productivity, meet increasing production demands and reduce product development cycle times. CAD vendors have also developed Product Life-cycle Management (PLM) systems to simplify data management across different disciplines involved in New Product Development (NPD). Though a majority of these CAD/CAM/CAE systems use ACIS, Parasolid, C3D and CGM geometric kernels underneath, they save the data in an application-specific file format that cannot be used for data exchange between heterogeneous CAD systems. Communicating the required information in a standard format that can be consumed by any other CAD system in machine readable form will help determine the manufacturability of these 3D models in a standalone test bench. Most commercial CAD systems support specification of GD&T on CAD models only at a presentation level. The absence of a semantic representation within the model undermines the capability to automate other processes that require these data. Neutral formats such as IGES (Initial Graphics Exchange Specification), SAT or STEP (**ST**andard for the **E**xchange of **P**roduct model data) have been used in the past, but they only transfer geometry between CAD systems. They do not allow tolerance data transfer even though the standards exist, because CAD vendors have not yet implemented translators. Most manufacturing and inspection planning vendors still use paper drawings to set up and review their processes. These drawings are often prone to errors, contain redundant data and may result in misinterpretation of the design intent. To overcome these concerns and also support Small and Medium-sized Enterprises (SMEs) in a high-technology environment, there is a need to develop a product model that represents complex design, reduces interoperability expenses and provides scope for automation. After more than two decades of success with STEP, industries have recognized the need for a unified product data model capable of representing machine-readable manufacturing information. The outcome of this effort is the development and standardization of a digital product model formally known as the ISO 10303-242, titled "Model-Based 3D Engineering" (Feeney, Frechette, and Srinivasan 2015). Under this standard, 3D models can capture rich semantic data which can be useful for automation of various downstream modules, particularly manufacturing. Further, once the translators are available, users will no longer need to maintain the 2D drawings, thereby reducing redundancies and ambiguities in the represented data. However, the success of this entire effort is dependent on the CAD vendor's enthusiasm to develop commercial translators according to the recommended practices (*CAx-IF Recommended Practices*

2014) . Currently, there are no commercial CAD systems that can read or write AP 242 files. Through the implementation of a direct AP203 to AP 242 translator, this thesis aims to capture the GD&T data required for manufacturing applications such as Computer Aided Process Planning (CAPP), Computer Aided Inspection (CAI), Coordinate Measuring Machine (CMM) and Computer Numerical Control (CNC) tool path generation. STEP AP 242 can also capture other Product Manufacturing Information such as surface texture, finish requirements, material specifications, welding symbols, and other annotations present in the model.

1.2 Background

This thesis is part of a bigger project at the Design Automation Lab (DAL), Arizona State University which aims at Automated Tolerancing of Mechanical Assemblies. This section discusses the motivation and scope of this work.

1.2.1 Concept of Automated Tolerancing of Mechanical Assemblies

GD&T concerns all stages of product development: design, part manufacturing, assembly and quality assurance. Designers need to ensure proper functioning and assemblability by specifying the tolerances. Designers also need to perform assembly stack analyses to ensure adequate clearance or interference at critical locations so that parts may be assembled interchangeably. On the other hand, manufacturing is concerned with selection of process plans that meet design requirements through tolerance conversion and manufacturing stack analysis. Lastly, a Quality Assurance

(QA) team checks for manufacturing variations through inspection planning, data collection and conformance assurance.

Various computer aided tools are available today for stack analyses, CMM data reduction and feature fitting. However, these tools are for tolerance analysis and not synthesis. Synthesis involves determining the required clearance ranges at each mating feature. This involves determining the contributors to each such variation, tolerance types needed on each contributor, sequence in which to apply dimensional and geometric controls, and the distribution of tolerance budget across contributors in each stack. These tasks are commonly referred to as tolerance stack analysis, tolerance synthesis, determining datum flow chains and value allocation respectively. After the completion of all these tasks, a statistical tolerance analysis is done to determine the percentage of assemblies that would meet the desired goals. However, if the tolerance schema does not pass, designers need to determine which tolerances to change based on the result of the tolerance analysis through sensitivities and percent contributions of each contributor in each stack.

The ability for a designer to generate a good tolerance scheme may take years of experience and knowledge; it requires deep knowledge not only of the device function, but also of manufacturing processes and practices, as well as the GD&T standards (*ISO 1101, Geometrical product specifications (GPS)* 2012; *ASME Y14.5: Dimensioning and Tolerancing* 2009). This tedious task is often left to “detailers” at the end of the design process and often requires trial and error, particularly for new products. There is very little in the way of prescriptive procedures to guide designers in tolerance synthesis. This shortcoming motivated the group at DAL to automate tolerance synthesis of mechanical assemblies.

1.2.2 Scope of Auto-tolerancing

Two important considerations that determine the scope of this work are:

1. Nominal design of the assembly and all constituent parts has been completed, i.e. the part geometries, nominal dimensions and assembly configuration have been determined.
2. From a purely design point of view, tolerance synthesis is based on two objectives: assemblability and design intent. Only the former can be determined purely from geometry and therefore this work determines the exact tolerances to only ensure mating of two components.

A classification of rigid mechanical assemblies can reveal common functions and structures, allowing for simplified algorithm for automating the tolerance schema generation process. The tolerance schema generated for assemblies in the same class are likely to follow ascertainable patterns based on common structures and functions. The types of joints can be categorized to allow for automatic tolerance schema generation using their mating features and the degrees of freedom allowed. Mating feature geometry will dictate which features will be chosen as datums and which tolerance classes will be assigned.

1.2.3 Assemblability

Assemblability is defined as “the ability to assemble/fit a set of parts in a specified configuration given a nominal geometry and its corresponding tolerances”. This definition of assemblability has a large contribution in automating tolerance allocation. While the goal is still to reduce the production life-cycle cost, the aim of assigning proper tolerances is primarily to reduce the amount of defective parts and ensure that the produced parts fit together and form a functioning assembly. For the purposes of assemblability, the required condition for mating two or more features is that they are not over-constrained. The types of constraints that assemblability looks at are kinematic constraints. This condition is extendable to all type of assemblies that would require clearance in order to be assembled. Constraints will be discussed in more detail in the next chapter. However in general, constraints can be classified as Geometric, Topological and Algebraic. Also, it is important to identify the key features and the direction in which the they need to be controlled. These directions include local or global links for graph searching such as: size dimensions, location and position (Sambhoos, Koc, and Nagi 2009).

1.2.4 Tolerancing of Assemblies

Tolerance values are determined based on the functionality and assemblability of parts. The functionality can be clearly inferred from the CAD model if the types of fits are specified, or if the types of "components off the shelf" (COTS) are listed. In the case of a fastener (floating, fixed or double-fixed), the material, size and type can be factors that the tolerance values are based on. The values may also depend on the external systems or components that are chosen from a library and the type of mechanical interface they have. It is worth mentioning that tolerance values are always a trade-off between manufacturing cost and functionality. Tighter tolerances provide more control but are directly related to higher manufacturing costs; therefore, it is optimal to specify the loosest tolerances that still meet the requirements of the functional aspects of the design. For this reason, the tolerance allocation and analysis process is iterative to balance tolerances;i.e., to tighten tolerances where functional requirements are not met, and to loosen them when functional requirements are exceeded. Also more liberal values can reduce the production and inspection costs.

1.3 Overview of Software Architecture

It is assumed that the assembly and part geometries are created in a CAD system and that data can be output in a neutral B-Rep format (STEP AP203). Before assigning tolerances, the software needs to identify the characteristics of the assembly, such as mating features and geometric constraints. These characteristics are the results of the Pre-processing tasks, as seen in the flowchart in Fig 1.

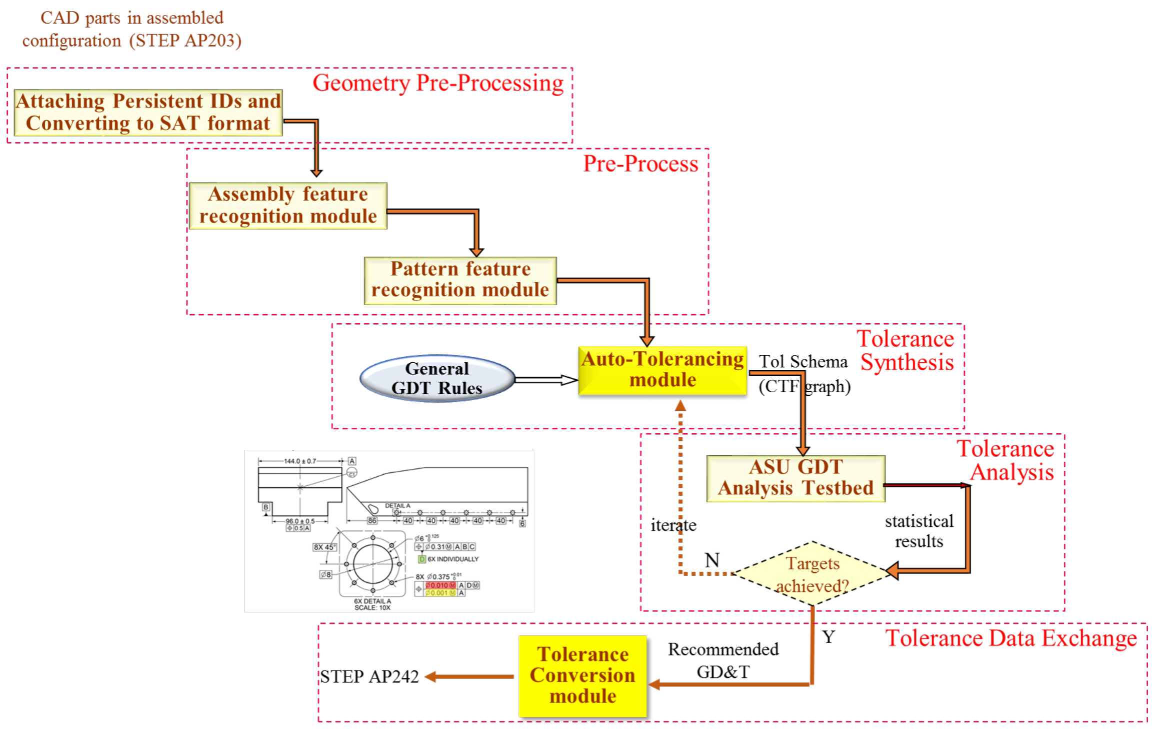


Figure 1: Overview of the Auto-tolerancing toolset

Further, without more detailed analysis, one can begin generating a preliminary tolerance schema by using some general, good practice rules applicable to all mechanical assemblies. Additional platform or artifact specific rules may also be needed depending on component types and their interfaces. The schema is then populated with tolerance values (Tolerance allocation). This is followed by multiple iterations of analysis and modifications, first the values and then the schema itself, if necessary, until tolerance specifications achieve all desired objectives.

1.4 Translator Module

This translator will then output the complete GD&T data generated by the previous modules into a neutral format such as STEP AP 242.ISO 10303-242 defines two ways of defining PMI within the Part 21 file (*CAx-IF Recommended Practices* 2014).

1. Representation: This refers to semantic representation of the PMI information within the appropriate STEP entities. Machine readable GD&T data can be automatically consumed by the receiving system. For instance, after importing the STEP file into another CAD system, PMI information can be viewed and modified just as it can be in the native CAD system. Once translators are available, computer readable software systems like CMMs and CAI teams can automatically consume this data and significantly automate their processes.
2. Presentation: This refers to human-readable way of capturing the tolerances within the CAD model using graphical annotations.

At this stage, the scope of the translator module is limited to the following,

1. STEP Part 21 file contains semantic representation of GD&T.
2. PMI information is limited to 1st order GD&T.
3. Input geometry is available in STEP AP 203 format.
4. Tolerance schema generation and analysis is done in-house and available in the CTF graph file format.

1.5 Approach Overview

At ASU for over 15 years, GD&T information used for tolerance analysis has been semantically represented using a neutral graph data structure format that is consistent with the ASME Y14.5 standard. This neutral representation abstracts the geometry in the form of features and constraints and adds tolerances as attributes. The final output is a Constraint Tolerance Feature (CTF) graph file with a doubly linked list of related entities at each level. This serves as a clean digital interface for semantic representation of GD&T information and is consistent with the ISO STEP Part 21 file. It contains all the tolerances along with the geometric information of the tolerance features. The above model is robust, and, in conjunction with the ASU GD&T test bed, can perform different kinds of tolerance analysis. Also, since this representation does not contain explicit geometry information, industries with restricted sharing of their CAD models can use this representation for tolerance analysis. More detailed explanation of the CTF graph file will be presented in the next chapter.

1.5.1 Attaching Persistent IDs

All the pre-processing tasks such as Assembly Feature Recognition, Pattern Recognition and also Tolerance synthesis are performed using the ACIS geometric kernel. Therefore, to be consistent, the input STEP file entities will need to be translated into ACIS entities before feeding into the above modules. The final output of the Auto-tolerancing module will be a CTF graph file in textual format. Since these data have been abstracted from the input AP 203 file, the primary objective is to identify a method to put this information back into the STEP file. One way to achieve this objective is by attaching persistent IDs to the topological entities in the input step file. The problem of retaining persistent IDs is familiar within the CAD developer community, wherein topological entities lose their IDs when imported into a new CAD system. However, ACIS 3D InterOp retains the persistent IDs during conversion from STEP to SAT, thereby retaining the same IDs in the CTF graph file. Hence, tolerance information present in the final CTF file points to the topological entities in the original STEP file.

1.5.2 Reading Complete Tolerance Information

All the required GD&T information such as nominal size, target feature, tolerance zone, datum feature, datum precedence, and material modifiers are available in the CTF file. The existing CTF parser was modified to read all the above information and populate a Graph data structure. This data structure is a light-weight version of the original CTF Graph data structure since it only contains the entity IDs and nominal sizes of features. The original CTF Graph data structure contains much more information to aid a variety of tolerance analysis methods.

1.5.3 Basic Intrinsic Validation of Tolerance Data

Once the tolerance scheme has been automated and multiple iterations of allocation and re-allocation of values have been done, the system performs basic checks before writing the output AP 242 file. The system checks for the following.

1. Is the entity a feature of size or not? For a feature of size, associate tolerances to the shape_aspect or derived_shape_aspect entity accordingly.
2. Avoid redundant data, e.g. re-use an existing datum instead of redefining it again.
3. Ensure size, dimension and form tolerances do not have datums associated with them.

1.5.4 Writing STEP AP242 Output File

Now that all the tolerance information is available in the Intermediate Graph Data Structure, the geometry is queried and PMI information is embedded into the corresponding topological entities using C++ APIs provided by STEPNC DLL (Hardwick n.d.). The STEP-NC DLL is a .NET API that can be used from C#, Visual Basic, or C++ through Windows Common Language Infrastructure (CLI) calls. The DLL also exports a COM version of API for use with C++ applications that cannot use CLI to call the .NET functions.

1.6 Thesis Organization

The original contribution of this thesis is the development of a translator to export GD&T information from an in-house CTF format to the standard ISO 10303 STEP AP 242 format. Interoperability issues that arise during data exchange among heterogeneous CAD systems is an evolving problem and Chapter 2 discusses more about this issue. Chapter 2 also contains the essential elements of a comprehensive product data model required for data exchange in Mechanical CAD. This thesis describes the translation of GD&T as per the ASME Y 14.5 standard for tolerances, into the ISO STEP AP 242 standard, and hence, Chapter 3 is intended to give the reader an overview of both standards. Chapter 4 contains the different concepts explored during this project and illustrates the implementation by providing a comparison of the entities in CTF vs. STEP along with the required STEP NC DLL C++ APIs. Verification of the output STEP AP 242 file is done by using available conformance checking packages. Various test cases in this regard are shown in Chapter 5. Lastly, Chapter 6 contains description of the Limitations, Future Work, and Conclusion of this thesis.

Chapter 2

INTEROPERABILITY AMONG CAD SYSTEMS

All the CAD systems can support import and export of data in either of the two ways:

1. A direct translator between the two systems
2. Translating to an intermediate neutral format and further to the destination format.

2.1 Direct Translation

Direct translation is resource intensive and the complexity of data translation scales up $O(n^2)$, as there are more data formats involved in the translation process. Whereas, translating to intermediate neutral format is efficient to $O(n)$ and also easily expandable. The entire sequence of steps required for data exchange between two CAD systems A and B through direct translation is shown in Figure 2.

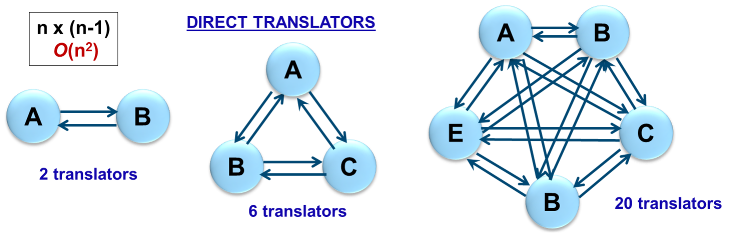


Figure 2: Direct translation between different file formats, where n = no. of CAD systems

2.2 Neutral Format

Neutral formats such as IGES, SAT or STEP have been used in the past to transfer geometry between CAD systems as shown in Figure 3.

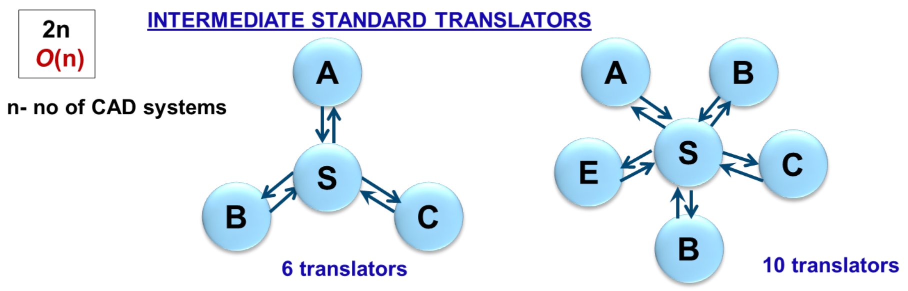


Figure 3: Translation to intermediate neutral file format

Systems A and B might not support the same set of entities during translation because they may have different capabilities, representation structure and also different native formats. During data exchange, one needs to consider all the different interfacing systems and the transformations involved. Pre and post-processor translators as shown in Figure 4 usually do not have a one-one mapping between the entities of the neutral format resulting in information loss.

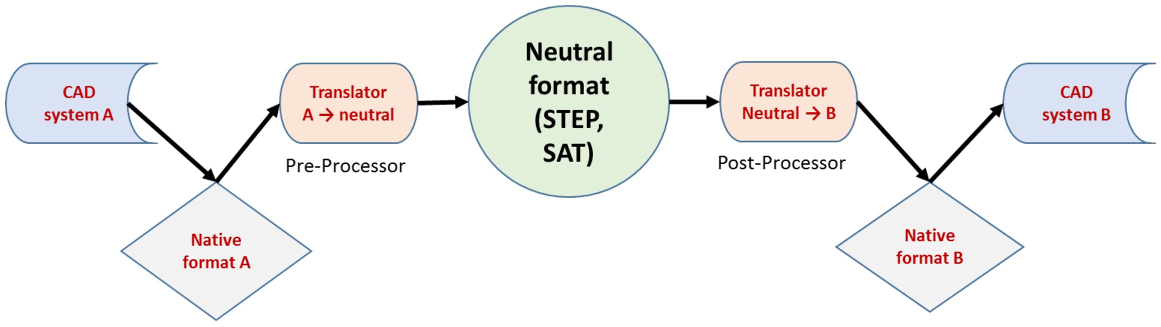


Figure 4: Overview of CAD data exchange procedure

2.3 Essential Elements during Data Exchange

As an overview, we discuss the essential items that constitute a comprehensive digital product data exchange model.

2.3.1 Construction History

Every CAD system stores the sequence of operations performed by the user to model the part. However, during translation to a neutral format like STEP, the entire sequence of operations carried out by the user is lost. Widely used AP 203ed1 and other neutral formats currently do not support procedural modeling and hence, renders the model to be non-editable when imported into a different CAD system. Most of the time, designers spend laborious hours recreating the whole model in the receiving system to account for any subtle changes in the model. ISO 10303-55 provides guidelines for the exchange of construction history through STEP. Most CAD systems differ greatly about the granularity of geometric and topological information represented in their data structure (Kim et al. 2008). CAD vendors need to develop translators capable of mapping this information to appropriate STEP entities to avoid the ambiguities while recreating the model in receiving CAD system.

2.3.2 Parameters

Parameters are dimensions or values associated with a part or a feature. Parameters control the location and geometric properties of the entity such as length, width, height and radius. They can either be values that are explicitly defined by the user such as the radius of a hole or implicit parameters which are attributes associated with the model. During data exchange, desired parameters are calculated from existing parameters through conversion algorithms but often involve numerical errors. e.g., In system A, a circle is defined by its center, radius, start and end points. Whereas in system B, the same circle is defined by its start, end, and interior points. Similarly, certain CAD systems support splines of up to order three whereas if the other system supports higher order splines, the conversion algorithm translates entities from the source system to the nearest compatible entity in the receiving system. ISO 10303 Part 108 together with Part 55 can be used to transfer both implicit and explicit parameters (Kim et al. 2008).

2.3.3 Constraints

Constraints are restrictions applied to the degrees of freedom of geometric entities. Constraints in parametric modelers are classified as (Bettig and Shah 2001):

1. Algebraic: Constraint equations exist between two or more parameters of the model, and the modeler ensures that these relationships are always satisfied.
2. Geometric: Logical relationships between geometric elements such as parallel, tangent and perpendicular needs to be true.
3. Dimensional: Distance or Angle relationships such as length or radius is specified between geometrical entities.

Parametric modeling allows users to drive the design with few key independent parameters. All other variables are related to these independent parameters, referred to as the associativity. Associativity allows automatic propagation of design changes

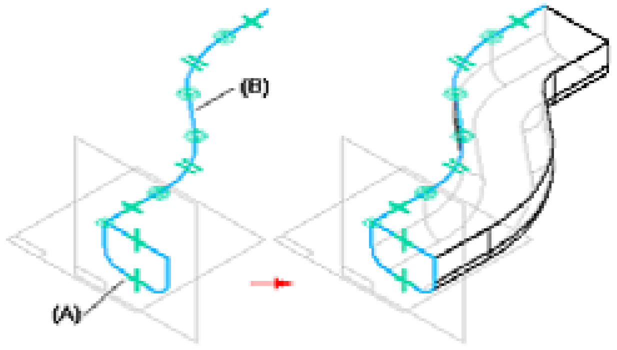


Figure 5: 2D Constraints with 3D history (General Sweep to obtain a 3D solid)

during any modifications to the model. The underlying technology of parametric modelers is constraint solving and involves:

1. List of variables
2. Parameters describing the entities
3. Constraints between the entities
4. Allowable range of each parameter

The technique used in most commercial systems is using 2D constraints with 3D history. The necessary sequence of steps as shown in Figure 5 includes:

1. Define a planar topology (2D Sketch).
2. Specify dimensional, geometric, & algebraic constraints.
3. Solve constraint equations in 2D.
4. Sweep/loft the profile to get 3D solid.

An external constraint solver like the DCM 2D manages 2D constraints in the sketcher. If a change involves parameters that belong to a 2D sketch, the geometric modeler rolls back the history to the point of change and sends the new parameters and constraints to the constraint solver. It then updates the sketch based on the solved values and then rolls forward the history. Again, interoperability problems associated with the exchange of explicit and implicit constraints using ISO 10303-108 has been comprehensively discussed in detail by Kim et al. 2008.

2.3.4 Shape or Geometry

Two of the most widely used STEP Application Protocols for the exchange of geometry are AP203 Configuration Controlled Design and AP214 Core Data for Automotive Mechanical Design Processes. While the former was intended for aerospace, the latter catered to the needs of the automotive industry. Both have remained as default data exchange standards for 3D solid geometric models. Geometry can be exchanged to different degrees using the appropriate Conformance Classes (CC). Ideally, AP 203 has 12 conformance classes, and AP 214 has 20 of them (ISO 2006). Most CAD systems do not implement of all these categories. The most commonly used classes are the ones listed in Table 1. Parameters, constraints, implicit dimensions and history do not transfer through AP 203 ed1. AP 214 includes all capabilities of AP 203 and additionally includes colors, layers, GD&T, and certain forms of design intent. AP 203ed2 has added capabilities similar to AP 214 including parametric information, GD&T and better interoperability with AP214. While dealing with geometric data exchange, the latest AP242 standard has combined capabilities of both AP 203 and AP 214.

Table 1: Conformance Classes in STEP

Geometric model	CC2	CC3	CC4	CC5	CC6
Trimmed surface	x				
Wireframe		x			
Manifold surface rep.			x		
Faceted B-Rep				x	
“advanced” B-Rep					x

2.3.5 Part and Assembly Features

Part features are stereotypical geometric shapes that can be positioned and defined completely with a few set of parameters. They provide a powerful means of capturing design intent, alleviate tedious construction procedures and allow the user to reason at a higher level of abstraction than provided by the geometric and topological entities (Venkataraman, Shah, and Summers 2001). Features on the part which are modeled using parameters can be easily modified utilizing the associativities between the geometric entities and constraints. The key to feature based system is a knowledge base of a set of pre-defined feature library. ISO 10303 has standardized feature based data exchange using the Part 111 definition which has been implemented using the AP 224 Mechanical product definition for process planning using machining features. It consists of a parametric library of machining features consisting of 51 manufacturing features (Kramer et al. 2001). In STEP AP 224, features are stored using the B-rep (Boundary Representation) format. While this representation works well with the machining feature library defined in AP 224, it fails to capture any semantic information associated with assembly features. Many researchers have identified the need for a more comprehensive data structure to enable feature based data exchange.

Assembly features are pairs of part features mating with each other. Assembly features encode mutual constraints on mating features shape, dimensions, position, and orientation. Assembly feature definition consists of both directly specified attributes and derived parameters. Derived parameters refer to implicit constraints already existing in the model, while independent parameters are user defined inputs at the time of assembly.

Commercial CAD still has very limited capability for feature data exchange using a neutral format. This shortcoming is primarily due to the lack of a comprehensive standard for feature definition. This limited view of feature data exchanges curbs the enormous potential of automation in various downstream modules.

2.3.6 GD&T

Computer models for tolerance synthesis and analysis have been a research topic for many years. Shah, Yan, and Zhang 1998 has previously discussed a complete overview of the different types of GD&T models. A brief outline of these GD&T models is discussed below for review.

Attribute models: The basic characteristic of attribute models is that tolerance information is stored as an attribute of either geometric entities or metric relations in CAD systems (Computer Aided Manufacturing-International, Johnson, and Associates 1985; Ranyak and Fridshal 1988; Shah and Miller 1990; Roy and Liu 1993; Roy and Fang 1996; Maeda and Tokuoka 1996; Tsai and Cutkosky 1997). Due to the absence of GD&T semantics in the model structure, this approach cannot perform validation of the information.

Offset models: In this method, the maximal and minimal object volume is obtained by offsetting the object by corresponding amounts on either side of the nominal boundary (Requicha 1983; Jayaraman and Srinivasan 1989). Offset models can only represent a composite tolerance zone; they cannot distinguish between effects of different tolerance types, nor interrelations among tolerance specifications.

Parametric models: Tolerances are modeled as plus/minus variations of dimensional or shape parameters. Parameter values are obtained by solving a set of simultaneous equations representing the constraints (Hillyard and Braid 1978; Krishnan, Eyada, and Ong 1997; Turner 1993). The parametric equations can be used for point-to-point tolerance analysis rather than zone based analysis. However, this method is not consistent with GD&T standards.

Kinematic models: Entities are modeled as "virtual" links and joints. A "kinematic link" is used between a tolerance zone and its datum features (Rivest, Fortin, and Desrochers 1993; Desrochers and Rivière 1997; Gao, Chase, and Magleby 1998). Tolerance analysis is based on vector additions. The first order partial derivative of analyzed dimension with respect to its component dimensions in terms of a transformation matrix is used for tolerance analysis. Both the parametric model and kinematic model can represent all the tolerance classes, but not all the information involved in GD&T can be stored. Datum systems cannot be validated and the analysis is point based rather than zone based.

DoF models treat geometric entities (points, lines, planes) as if they were rigid bodies with degrees of freedom (DoFs) (Zhang 1992; Wu 2002; Kandikjan, Shah, and

Davidson 2001). Geometric relations (angular and linear) are treated as constraints on DoFs. Y14.5 tolerance classes are characterized by how each DoF of each entity is controlled.

Technologically and Topologically Related Surfaces: (TTRS) models bear many similarities to DoF models (Clement, Riviere, and Serre 1996; Desrochers and Maranzana 1996). Later researchers have tried to express Y14.5 tolerance classes in terms of TTRS but this is not entirely achieved. Although mathematically elegant, TTRS models are indifferent to Y14.5 Rule 1, floating zones, effects of bonus and shift, form tolerance, or datum precedence. DoF models facilitate the validation of DRF and tolerance types. The model presented here is therefore based on DoF models.

ASU GD&T Global Model is a hybrid between the DoF model and Attribute model (Wu, Shah, and Davidson 2003).

Analysis Neutral Model was developed as a successor of the ASU GD&T Global Model (Shen, Shah, and Davidson 2008) to overcome the difficulties in performing simulation based tolerance analysis. At ASU, for over 15 years we have been using a neutral representation that contains all data needed for tolerance analysis and is consistent with the ASME Y14.5 standard (Wu, Shah, and Davidson 2003). This neutral representation abstracts the geometry in the form of features and constraints and adds tolerances as attributes. The final output is a Constraint Tolerance Feature (CTF) graph file with doubly linked list of related entities at each level. This representation serves as a clean digital interface for semantic representation of GD&T information and is consistent with the ISO STEP Part 21 file. It contains all the tolerances along with the geometric information of the tolerance features. More details about the CTF will be discussed in the later sections.

There are different commercial neutral file formats such as SAT file or the latest STEP AP 242 standard that can help achieve GD&T interoperability. 3D InterOp from Spatial has capabilities to translate the PMI information from CATIA, ProE and NX CAD packages to SAT format. InterOp reads the geometry supplied in B-Rep format and until R24 version, utilizes a file to file data transfer with a wide choice of translation options. Internally, PMI is represented as meta-data attached via attributes to the topology or other PMI attributes. However, dimensions are slightly different from the above representation. Further, there is no associativity between the graphical and the semantic PMI represented in InterOp. In a recent study published by NIST (Collins 2016), all the gaps in mapping PMI between STEP and ACIS have been comprehensively documented. InterOp does not fully comply with the Y14.5 standard in its representation of the tolerance classes and the non-availability of commercial translators to transfer embedded PMI from SAT, diminishes its scope as a choice for neutral format.

2.3.7 Assembly Structure

Every product model consists of multiple parts that form a hierarchy of sub-systems and components. The user first creates part geometries using solid modeling and further assembles these components using appropriate mating constraints. The parts are be positioned and oriented with respect to each other using transformation matrices. To completely constraint a part within an assembly, various associative conditions such as mating, geometric, parametric, kinematic and structural relations needs to be defined. However, there are no neutral formats capable of capturing all these associative relations.

2.4 ASU Constraint-Tolerance-Feature (CTF) Graph Model

2.4.1 Assembly Features Representation using N-Rep

At ASU DAL, we have been using the N-rep data structure to store both part and assembly features extracted from AP 203 (Murshed, Dixon, and Shah 2009; Murshed et al. 2007; Medichalam, Shah, and D'Souza 2004). The following template defines Part features,

1. Topological Relationships (edge convexity relation).
2. Geometry (face types: planar and cylindrical,).
3. Geometric Relationships (parallel, perpendicular and coaxial)
4. Parameters (typically dimensions defined by geometric relations or attributes).
5. Parametric Relationships (derived parameters or constraints on parameter values).

Assembly feature definition consists of following properties and attributes.

1. Part features that constitute the assembly feature
2. Assembly Parameter Definition
    a) Geometric - parameter defined by two geometric entities directly
    b) Algebraic - parameter defined by other parameters
3. Constraints/ Relations

    a) Geometric - constraint between two geometric entities
    b) Algebraic - constraint between parameters

4. Kinematic Relations – DoFs and motion limits
5. Structural Relations: load point, component directions and magnitude, time functions

All the above information is represented in N-rep format, allowing users to define features on the fly. N-rep defines a uniform set of attributes for all features and illustrates how this template can be extended to define any form feature. Both the in-house Assembly Feature Recognition and Pattern Feature recognition suites use this representation (Vemulapalli et al. 2014). This representation is also used in the development of an Auto Tolerancing system where the authors discuss the implementation of pre-processing algorithms and the benefits of using N-rep format to transfer GD&T information along with the geometry between different modules (Mohan et al. 2014; Haghighi et al. 2015).

2.4.2 CTF Graph structure

An overview of the CTF graph data structure is presented here. For a detailed description, please refer Shen, Shah, and Davidson 2008. CTF file is a textual representation of a graph data structure which stores all the above information as required for various kinds of tolerance analysis. One of the notable advantages of this representation is its exemption of the CAD geometry in the variation analysis. Also, this representation is computationally feasible and desirable to perform dimension variation analysis because it does not vary the features directly on CAD model. Figure

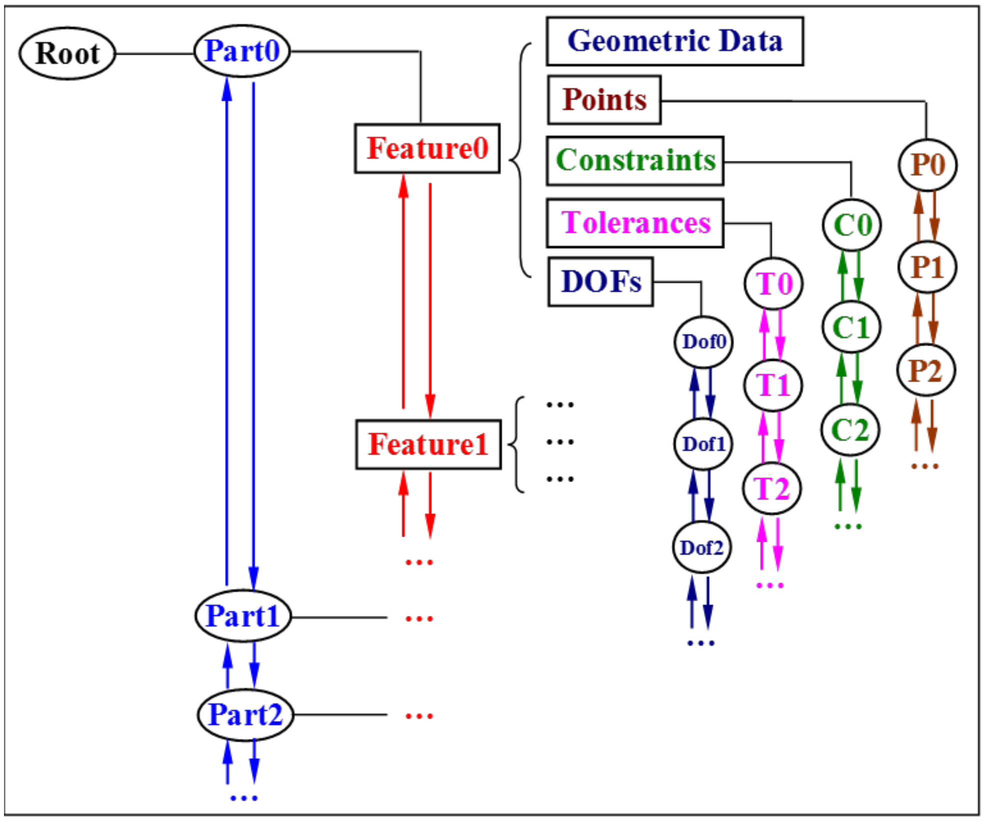


Figure 6: CTF Graph model

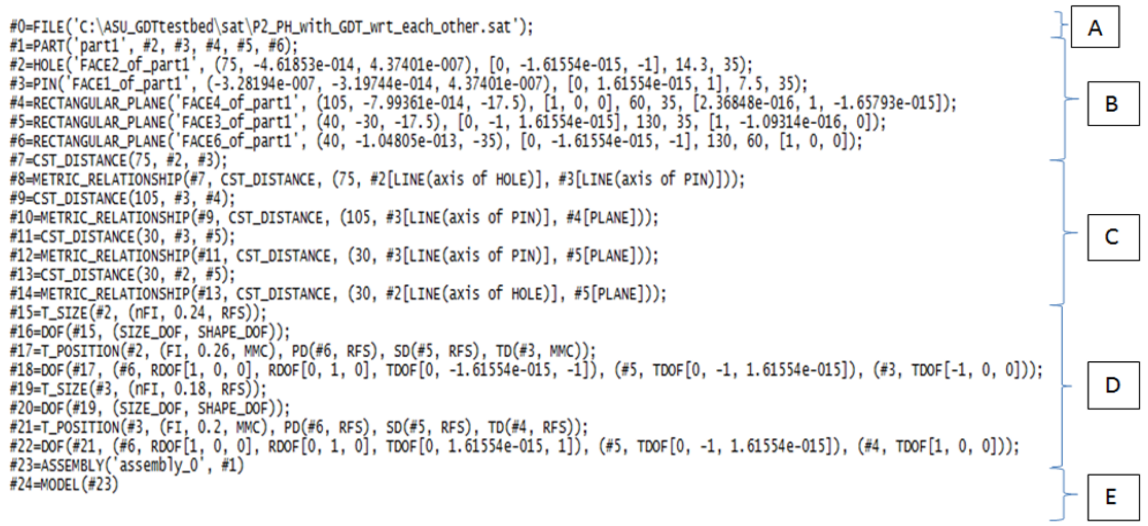
6 is a representation of the global GD&T model and shows the connectedness between GD&T information.

The file structure contains five main sections

1. Sec A: contains the name of the B-rep file.
2. Sec B: contains information about the features in a Part and their data. In case of an assembly, this section will have multiple parts and their feature information.
3. Sec C: contains data about the constraints and metric relations, including the mating conditions.
4. Sec D: tolerance data and DoFs

5. Sec E: assembly hierarchy

The file structure of CTF is shown in Figure 7.



```
#0=FILE('C:\ASU_GDTtestbed\sat\P2_PH_with_GDT_wrt_each_other.sat');
#1=PART('part1', #2, #3, #4, #5, #6);
#2=HOLE('FACE2_of_part1', (75, -4.61853e-014, 4.37401e-007), [0, -1.61554e-015, -1], 14.3, 35);
#3=PIN('FACE1_of_part1', (-3.28194e-007, -3.19744e-014, 4.37401e-007), [0, 1.61554e-015, 1], 7.5, 35);
#4=RECTANGULAR_PLANE('FACE4_of_part1', (105, -7.99361e-014, -17.5), [1, 0, 0], 60, 35, [2.36848e-016, 1, -1.65793e-015]);
#5=RECTANGULAR_PLANE('FACE3_of_part1', (40, -30, -17.5), [0, -1, 1.61554e-015], 130, 35, [1, -1.09314e-016, 0]);
#6=RECTANGULAR_PLANE('FACE6_of_part1', (40, -1.04805e-013, -35), [0, -1.61554e-015, -1], 130, 60, [1, 0, 0]);
#7=CST_DISTANCE(75, #2, #3);
#8=METRIC_RELATIONSHIP(#7, CST_DISTANCE, (75, #2[LINE(axis of HOLE)], #3[LINE(axis of PIN)]));
#9=CST_DISTANCE(105, #3, #4);
#10=METRIC_RELATIONSHIP(#9, CST_DISTANCE, (105, #3[LINE(axis of PIN)], #4[PLANE]));
#11=CST_DISTANCE(30, #3, #5);
#12=METRIC_RELATIONSHIP(#11, CST_DISTANCE, (30, #3[LINE(axis of PIN)], #5[PLANE]));
#13=CST_DISTANCE(30, #2, #5);
#14=METRIC_RELATIONSHIP(#13, CST_DISTANCE, (30, #2[LINE(axis of HOLE)], #5[PLANE]));
#15=T_SIZE(#2, (nFI, 0.24, RFS));
#16=DOF(#15, (SIZE_DOF, SHAPE_DOF));
#17=T_POSITION(#2, (FI, 0.26, MMC), PD(#6, RFS), SD(#5, RFS), TD(#3, MMC));
#18=DOF(#17, (#6, RDOF[1, 0, 0], RDOF[0, 1, 0], TDOF[0, -1.61554e-015, -1]), (#5, TDOF[0, -1, 1.61554e-015]), (#3, TDOF[-1, 0, 0]));
#19=T_SIZE(#3, (nFI, 0.18, RFS));
#20=DOF(#19, (SIZE_DOF, SHAPE_DOF));
#21=T_POSITION(#3, (FI, 0.2, MMC), PD(#6, RFS), SD(#5, RFS), TD(#4, RFS));
#22=DOF(#21, (#6, RDOF[1, 0, 0], RDOF[0, 1, 0], TDOF[0, 1.61554e-015, 1]), (#5, TDOF[0, -1, 1.61554e-015]), (#4, TDOF[1, 0, 0]));
#23=ASSEMBLY('assembly_0', #1)
#24=MODEL(#23)
```

Figure 7: CTF File structure

2.4.2.1 Tolerance Feature Representation- Real vs. Trimmed Features

Features are stereotypical shapes defined by specific topology, geometry, and constraints. For the purpose of tolerance specification, features can be classified as Real and Trimmed. Real features are toleranced surfaces on a part and can be of any type and shape. These real features or abstracted primitives like a pure point feature, an infinite line feature, or an infinite plane feature does not give much information and hence it is necessary to approximate the real features by trimmed features. Trimmed features are defined as the features simplified or abstracted from the real ones with minor cutouts and protrusions suppressed.

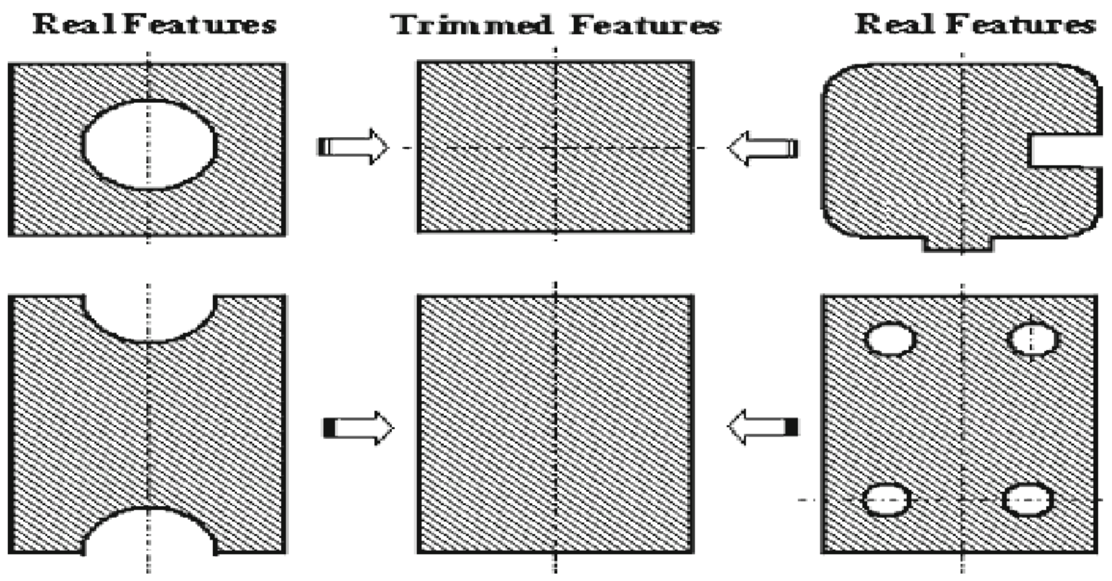


Figure 8: Real vs. Trimmed Features

Consider the example in Figure 9, the planar features can be approximated by an ideal rectangular planar feature. Indeed, these real surfaces, with their cutouts and/or protrusions, would most likely to be manufactured (e.g.milled) at one setup. These minor cutouts and/or protrusions do not affect, the choice of the manufacturing process. Therefore this idealization is not only necessary but also reasonable because its effect on the simulation result is negligible. The CTF-Graph Based Model represents real or trimmed features and not the primitives or their combinations.

2.4.2.2 Constraints and Metric Relationships

A geometric constraint in GD&T corresponds to a basic metric relationship between the primitives. Each metric relationship may be expressed in one or more analytical equations from the analytical geometry. Different metric relationships exist between the primitives, and constrain the DoFs of geometric entities w.r.t. each other (Wu et al. 2003). We use the same representation, i.e. (X_i, Y_i, Z_i) for points, $A_iX + B_iY + C_iZ + D_i = 0$ for planes, and $(X - X_i)/p_i = (Y - Y_i)/q_i = (Z - Z_i)/r_i$ for lines. But for example, when a feature of size is involved, the feature's size must be taken into account; or in order to compute the distance between a point and a circle (i.e. a special plane), the coincident relationship between a point and a plane can be used to check if the point is coincident with the circle defined plane. Geometric constraints may be specified dimensions, mating conditions, or geometric relations, such as perpendicularity, parallelism. For size features, size constraints can be

2.4.2.3 Tolerances

Since a tolerance is used to control the variation of a certain geometric constraint, it depends on the corresponding geometric constraint. In other words, a tolerance cannot exist without its corresponding geometric constraint. For instance, a dimensional plus/minus tolerance has no meaning if the corresponding dimension does not exist. Therefore, constraints and tolerances are combined to obtain both the nominal dimension and allowable variations on the feature by the use of a Graph data structure. This data structure has geometric features (or trimmed features) at its nodes and further parameters, tolerances and constraints as its leaf nodes. The tolerance information in the Model is represented in different classes derived from the tolerance base class which is based on the ASME Y 14.5 standard. The tolerance information includes

1. Geometry
    a) Target Feature and its associated parameters
    b) Datum Reference Frame

        i. Datum Features and associated parameters.
        ii. Datum precedence

2. Target and Datum Feature Material Modifiers
    a) Maximum Material Condition (MMC)
    b) Least Material Condition(LMC)
    c) Regardless of Feature of Size(RFS)
3. Tolerance Zone

Chapter 3

ISO 10303 STEP STANDARD

IGES was one of the earlier formats used for the exchange of product shape data. However, IGES was replaced by STEP and is no longer being enhanced. Also, it is worth noting that IGES and STEP were not designed to be super sets of all CAD systems. With the development of STEP, variety of data exchange formats have been developed under ISO 10303. STEP supports different product types (ships, electronics, mechanical and architectural) and applications (design, analysis, manufacturing and quality assurance). STEP is managed by the ISO/TC 184/SC 4 committee. These standards have an Object Oriented (OO) data representation where the primary mechanism of exchange are classes. These classes are defined as super/sub types, with attributes and rules.

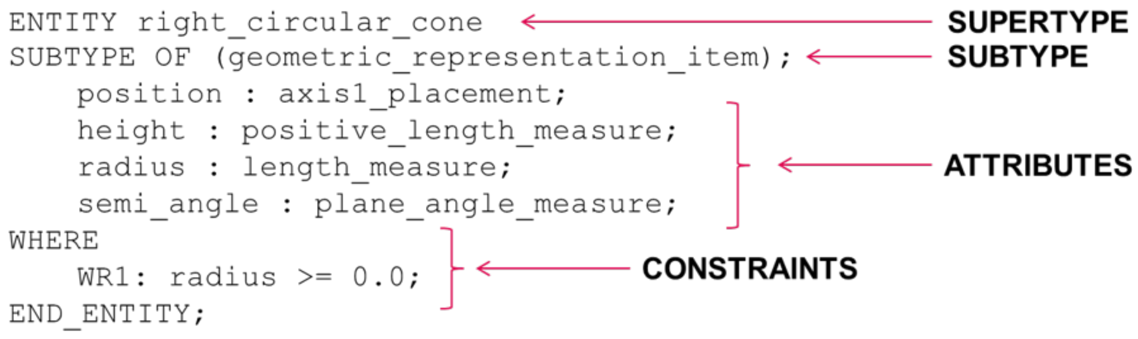
Table 2: STEP Components

No.	Components of STEP	ISO 10303 Part No.
1	Descriptions methods	1x
2	Implementation methods	2x
3	Conformance testing methodology and framework	3x
4	Integrated generic resources	4x-6x
5	Integrated application resources	1xx
6	Application Protocol	2xx
7	ATS - Abstract Test Suite	34
8	AIC - Application Interpreted Constructs	5xx
9	Application Modules	1xxx

As we look back at the developments realized through digital product data exchange, using a standardized product model like STEP has significantly reduced product development time and cost, as well as improved life-cycle support (Kemmerer 1999).Currently, every CAD system allows the user to create part and assembly models. These models contain rich information than just the shape of the product. To aid complete interoperability of part and assembly model elements among heterogeneous CAD systems, ISO has published various standards under STEP IRs and their implementation in different APs. However, complete interoperability of these elements is not fully realized even under the latest AP 242 standard.

3.1 Express Language

EXPRESS is a standard data-modelling language defined in ISO 10303-11 for defining product data. Express captures different data objects, their relationships and validates the populated data structure using constraints and algorithmic rules. EXPRESS-G (Graphical) is a light-weight graphical representation of the data model. Figure 9 shows and example of Express schema.



```
ENTITY right_circular_cone
SUBTYPE OF (geometric_representation_item);
    position : axis1_placement;
    height : positive_length_measure;
    radius : length_measure;
    semi_angle : plane_angle_measure;
WHERE
    WR1: radius >= 0.0;
END_ENTITY;
```

Figure 9: Express Data Model

Figure 10 graphically shows data the data relationship using for Express-G.

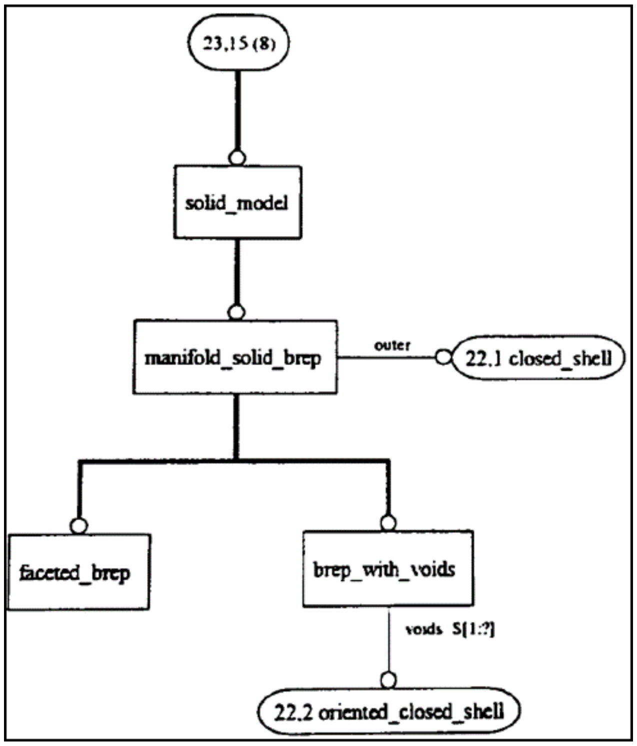


Figure 10: Express-G

3.2 Standard Data Access Interface (SDAI)

STEP defines the set of general SDAI operations in ISO 10303-22 for implementation of the standard. Application Protocols defined using EXPRESS is developed using Application Programming Interfaces(APIs). STEP supports ISO 10303- 23, 24 and 27; defined as C++, C and Java binding to the SDAI respectively.

3.3 Application Protocols (APs)

The STEP standard is build on an integrated architecture framework to enable information sharing across many applications covered by the standard. STEP covers

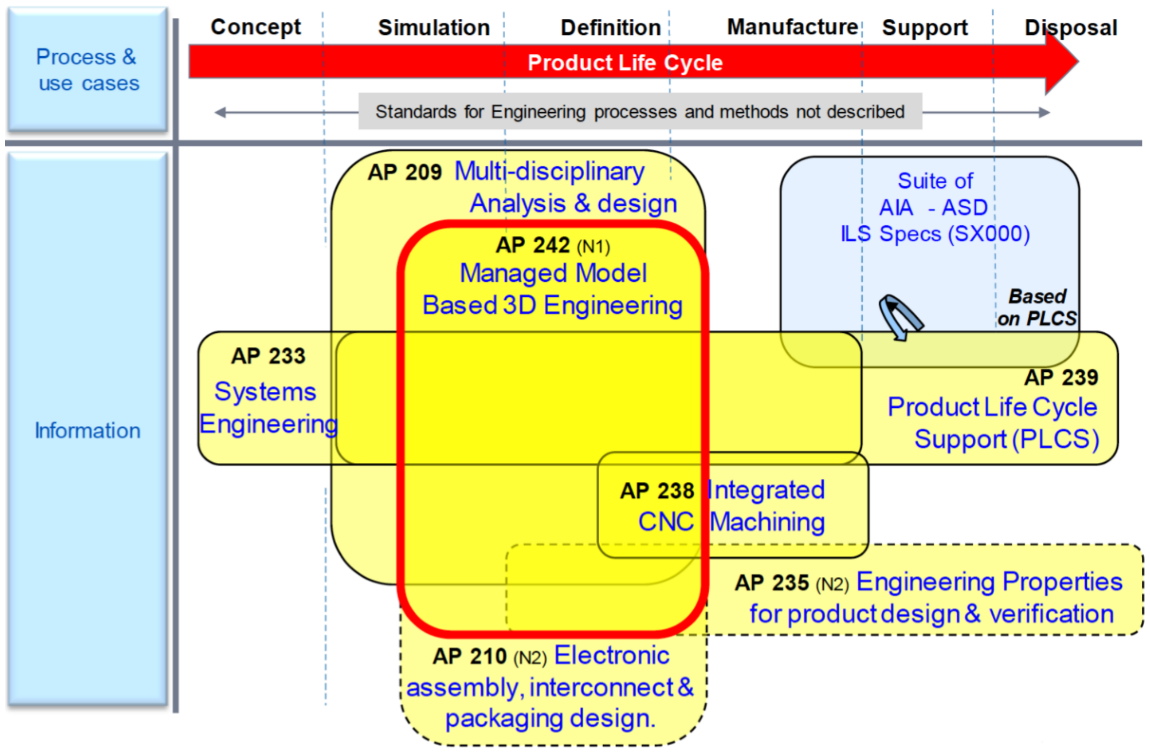

Process & use cases
Concept
Simulation
Definition
Manufacture
Support
Disposal
Product Life Cycle
Standards for Engineering processes and methods not described
Information
AP 209 Multi-disciplinary Analysis & design
AP 242 (N1) Managed Model Based 3D Engineering
AP 233 Systems Engineering
Suite of AIA - ASD ILS Specs (SX000)
Based on PLCS
AP 239 Product Life Cycle Support (PLCS)
AP 238 Integrated CNC Machining
AP 235 (N2) Engineering Properties for product design & verification
AP 210 (N2) Electronic assembly, interconnect & packaging design.

Figure 11: STEP Life Cycle Support

Source: J.-Y. Delaunay,"Background and use of STEP AP 242 in the European Aerospace and Defense Industries," presented at the PDT Europe 2014, Paris, France, 2014.

a wide range of applications and hence breaking down the information into domain specific Application Protocols helps to better manage and implement the standard.

These APs serve as schema for data exchange in a particular product category or purpose. All APs and IRs are written in Object Oriented data modeling language called EXPRESS.Some of the notable APs used in the field of Mechanical Engineering is described in Table 3.

Table 3: Application Protocols in STEP

No.	APs	Application
1	AP 203	Configuration Controlled 3D 2 Designs of Mechanical Parts and Assemblies (Revised by AP 242)
2	AP 209	Multidisciplinary Analysis and Design
3	AP 214	Core Data for Automotive Mechanical Design Processes(Revised by AP 242)
4	AP 223	Exchange of Design and Manufacturing Product Information for Cast parts
5	AP 224	Mechanical Product Definition for Process Planning using Machining Features
6	AP 238	Application Interpreted Model for Computerized Numerical Controllers (CNC)
7	AP 239	Product Life Cycle support
8	AP 240	Process Plans for Machined products
9	AP 242	Managed Model-Based 3D Engineering

3.4 Integrated Resources

APs are derived from lower level schema called Integrated Resources (IR). IRs specify the EXPRESS schemas that are independent of specific implementation and usage. These are further divided into:

1. Integrated Generic Resource: ISO 10303 Part 41 - 61
2. Integrated Application Resource: ISO 10303 Part 101 - 112

Some of the commonly used IRs are as shown in Table 4.

Table 4: Integrated Resources in STEP

No.	Part	Description
1	Part 42	Geometric and Topological representation
2	Part 44	Product structure configuration
3	Part 46	Visual presentation
4	Part 47	Shape Variation Tolerances (Semantic GDT)
5	Part 101	Draughting
6	Part 104	Finite Element Analysis (FEA)
7	Part 110	Computational Fluid Dynamics (CFD) data
8	Part 111	Elements for the procedural modelling of solid shapes (3D CAD)

3.5 Application Interpreted Constructs (AIC)

Application Interpreted Constructs (AICs) were not present in the first release of the STEP standard. They were later introduced to aid specializations in the geometric area to provide better integration between APs. For instance while importing a 3D model, the shape information being exchanged is defined by any of the AICs derived from the Part 42 as shown in Table 5.

Table 5: Application Interpreted Constructs in STEP

No.	Part	Description
1	Part 507	Geometrically bounded surface
2	Part 508	Non-manifold surface
3	Part 509	Manifold surface
4	Part 514	Advanced Boundary Representation (B-Rep)
5	Part 515	Constructive Solid Geometry (CSG)
6	Part 519	Geometric tolerances

Chapter 4

DEVELOPMENT OF THE TRANSLATOR

4.1 Conceptual Design

The primary aim of this research is to develop a translator from STEP AP 203 format to a neutral or proprietary format by attaching the GD&T information to the 3D geometry.

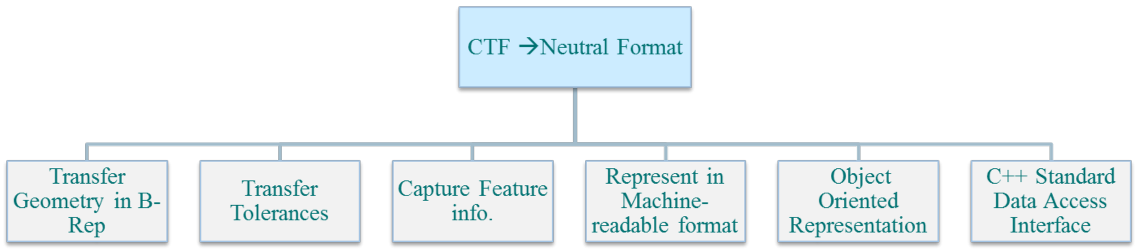

CTF →Neutral Format
Transfer Geometry in B-Rep
Transfer Tolerances
Capture Feature info.
Represent in Machine-readable format
Object Oriented Representation
C++ Standard Data Access Interface

Figure 12: Main functions of the translator

GD&T synthesis, value allocation and analysis is done using an in-house software. Tolerance Features, Constraints, Parameters, Degrees of Freedom and Tolerance Frames is represented using the Constraint Tolerance Feature (CTF) graph file. In particular, CTF graph conforms to the ASME Y14.5 GD&T standard. Similarly, the destination format is also expected to comply with the ASME Y 14.5 standard in its representation. Three concepts were formulated to achieve this goal.

1. Attach GD&T to ACIS SAT file All the preprocessing modules such as feature recognition, pattern recognition and tolerance stack loop detection are written

using ACIS geometric kernels and hence SAT file was a default choice for the destination format. However, since there aren't many systems that can read or write the PMI information available in SAT format, and also due to certain shortcomings of the SAT format as described in NIST Testing for Interoperability in accurately representing tolerance information.

2. Append the STEP AP 203 file with STEP GD&T schema definitions.
   To append tolerances to an AP 203 file, the GD&T needs to be associated with the geometric entities. Achieving this through a brute-force way requires intensive data relationship management hence is avoided.
3. Use STEP NC Libraries from StepTools Inc. to convert to STEP AP 242.
   StepTools has previously developed the STEP-NC libraries which contain GD&T schemas based on all the APs. Additionally, STEP NC libraries can also define device independent tool paths, and CAM independent volume removal features based on the geometric constructs used in the ISO 10303 STEP standard. StepTools has rich experience developing and enhancing STEP libraries for manufacturing applications. Hence this approach was taken to translate GD&T from CTF file to the STEP AP 242 format.

4.2 Implementation Guidelines

Successful development and implementation of ISO standards are dependent on the efforts various organizations. PLM interoperability is a significant step towards enterprise integration and industries have begun to realize huge savings through these standardization of product data model. ProSTEP iViP and PDES Inc. are the two agencies that spearhead the maintenance and implementation of these translators for the widely used STEP standards in the mechanical industry. While the former is the owner of AP 203, the latter manages AP 214. There are also other agencies that promote the development of translators depending on the application of the standard APs. One of the important steps towards combined interoperability testing was the merger of PDES, Inc. and the ProSTEP iViP Association testing forums, also known as the CAx - Implementor Forum (CAx-IF). In this thesis, the primary focus will be on the guidelines put forth by the CAx Implementor Forum for the implementation of a STEP AP 242 translator. The scope of work is limited to definition of GD&T data as Semantic Representation applied to Boundary Representation (B-Rep) solid models. An overview of the complete implementation is shown in 13.

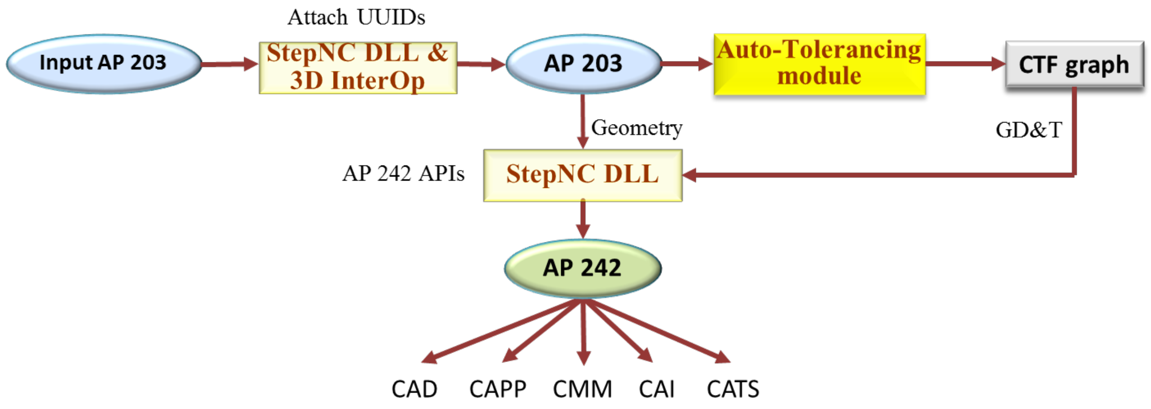


Figure 13: Overview of complete translation process

4.2.1 Pre-processing of input STEP AP 203 file

The input STEP AP 203 will be preprocessed using ACIS, InterOp and StepNC DLL methods as shown in Figure 14. The objective of pre-processing is as follows,

1. Linking CTF Entities to STEP Entities through persistent IDs - The pre-processing modules such as Feature Recognition, Pattern Recognition, Tolerance Synthesis, Tolerance Allocation and Tolerance Analysis are performed independently using the ACIS geometric kernel. Hence, the entities need to have persistent IDs for identification across different modules. To have the same IDs for both the STEP and the ACIS SAT entities, the following pre-processing steps are required.
2. Convert from STEP AP 203 to SAT - All the other modules are written using ACIS geometric kernel and hence the input to these modules should be a SAT file.

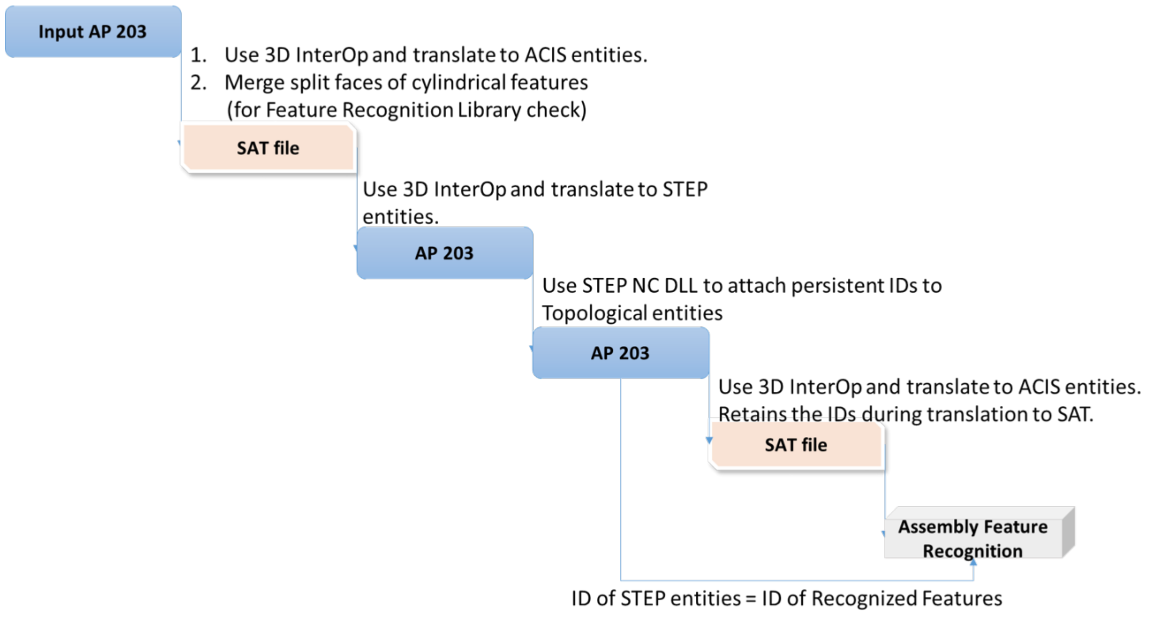

Input AP 203
1. Use 3D InterOp and translate to ACIS entities.
2. Merge split faces of cylindrical features (for Feature Recognition Library check)
SAT file
Use 3D InterOp and translate to STEP entities.
AP 203
Use STEP NC DLL to attach persistent IDs to Topological entities
AP 203
Use 3D InterOp and translate to ACIS entities. Retains the IDs during translation to SAT.
SAT file
Assembly Feature Recognition
ID of STEP entities = ID of Recognized Features

Figure 14: Preprocessing steps to attach persistent IDs to the input AP 203 file

4.2.2 Generating Complete GD&T for the Input Assembly

The Auto-tolerancing software performs Assembly Feature Recognition and Pattern Feature Recognition and extract all the feature parameters. Combining the output of the above modules with some good practise GD&T rules, the Tolerance Schema Generation module creates 1st Order Tolerance schema sans values. The GD&T schema generated is represented using the ASU CTF model. The CTF file sans tolerance values is then transferred to the Tolerance Value Allocation module. At this stage, the Feature Control Frames that were previously generated by the Schema Generation module is populated with values based on the available tolerance budgets. Lastly, 3D statistical variation analysis is carried out to ensure that the assembly variations are within the specified limit for assemblability requirements. The complete GD&T data is now available in the CTF file format.

4.2.3 Reading the CTF file using a Parser

The output of the Auto-tolerancing software is a CTF file that contains all the information such as Features, Parameters, Constraints and Tolerances. Hence, the primary step is to extract all the information from the CTF file and to store it into an intermediate data structure. The intermediate data structure would contain all the data as shown in Figure 15, except that it is a graph data structure. The 3D Assembly at the top forms the parent node and all the constituting Parts become its child nodes. In the CTF data structure, each part contains two types of Features,

1. Machining Features such as Pin, Hole, Tab and Slot features.
2. Trimmed Features such as Plane and Mid-plane.

These Machining Features further contain child nodes with the parameters that define them completely, e.g. Width and Height for a Slot, or Radius and Depth for a Hole. Also, Machining Features that are critical for assemblability between two parts such as a pin-hole pair needs to have a tolerance node associated with them. The tolerance node contains all the information that is required to create a tolerance control frame. It contains pointers to the Target Feature, Tolerance type, Value, Datum Reference Frame and Material Modifiers.

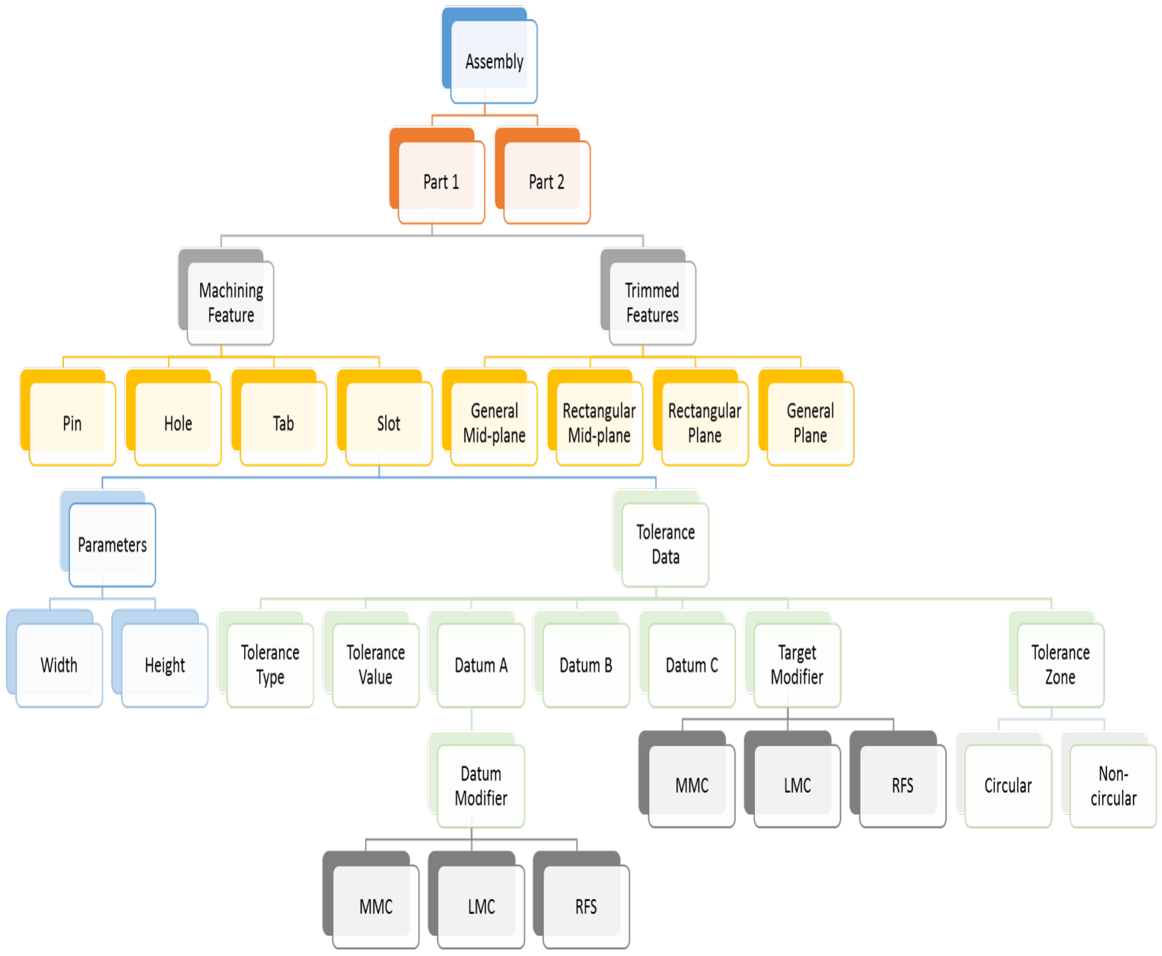


Figure 15: Entities Extracted from CTF file

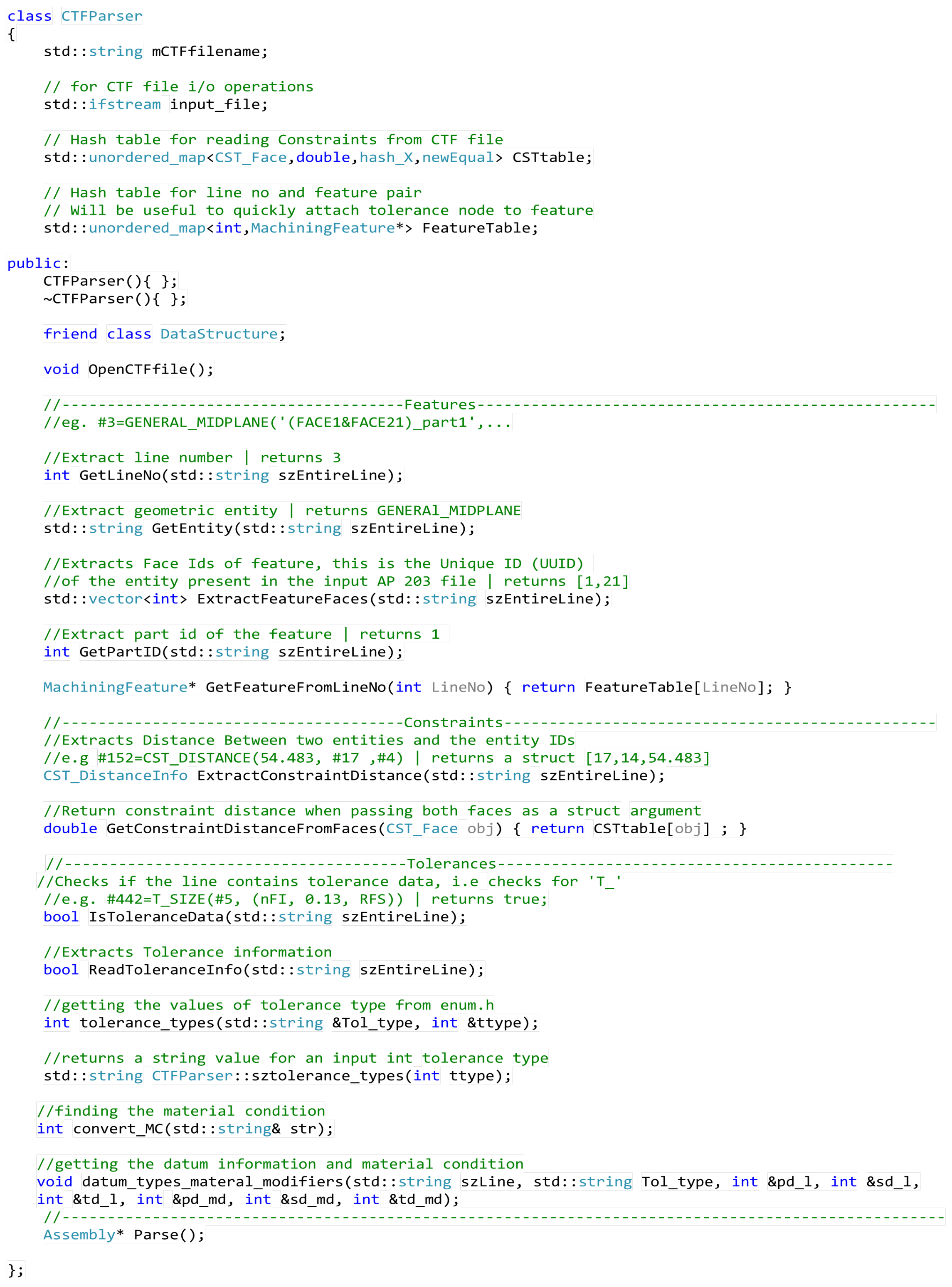
In order to read all the above information from the CTF file and populate the intermediate data structure, a CTF Parser based on the pseudo code shown in shown in Figure 16 is implemented. This parser is a modification of the existing one because it stores only the entity IDs, parameters and tolerance data as compared to the original CTF parser which contains much more information such as the plane normals, axis vector and Degree of Freedoms constrained.

Before we begin parsing, the CTF file name is obtained as a user input with the .ctf extension. However, in the Auto-tolerancing suite, the file name is internally assigned automatically by the previous module. Once the input file is opened, the parser reads through each line and compares the type of entity being read. For instance, if the entity being referenced in the line is a Part, the Part ID is extracted of that part which is a persistent ID assigned by the preprocessing module. Similarly for a Feature, it compares the feature type and extracts all the Face IDs and parameters that defines the feature. This is achieved through the GetFeatureFromLineNo() method which returns a pointer to the Machining Feature being referenced in that line of the CTF file. Some features have multiple faces associated with them and all the associated faces will be read by the parser.

```
Open the CTF File obtained through input from the user
Create empty nodes to store various entities such as Assembly, Part, Feature, Constraints and Tolerances
Parse through the CTF file
    Get Line No
    Get Entity Type
    If Part
        Extract Part IDs
    If Feature
        Compare Feature type
            For Pin or Hole feature
                Extract Face ID of Cylindrical Face
            For Tab or Slot feature
                Extract Face IDs of both the parallel Planar Faces
            For Pattern of Features
                Extract Face IDs of all constituting Features
            For any other features
                Extract all the constituting Face IDs
        Extract Parameters (Nominal Size)
    If Constraints
        Compare Constraint type*
            For Distance Constraint
                Extract the Face IDs of the Features being constrained
                Extract Constraint Value/Nominal Dimension
    If Tolerance
        Extract the complete tolerance frame data
Close the CTF file

*Perpendicularity and Parallelism is not handled in this version of translator
```

Figure 16: CTF Parsing Pseudo code

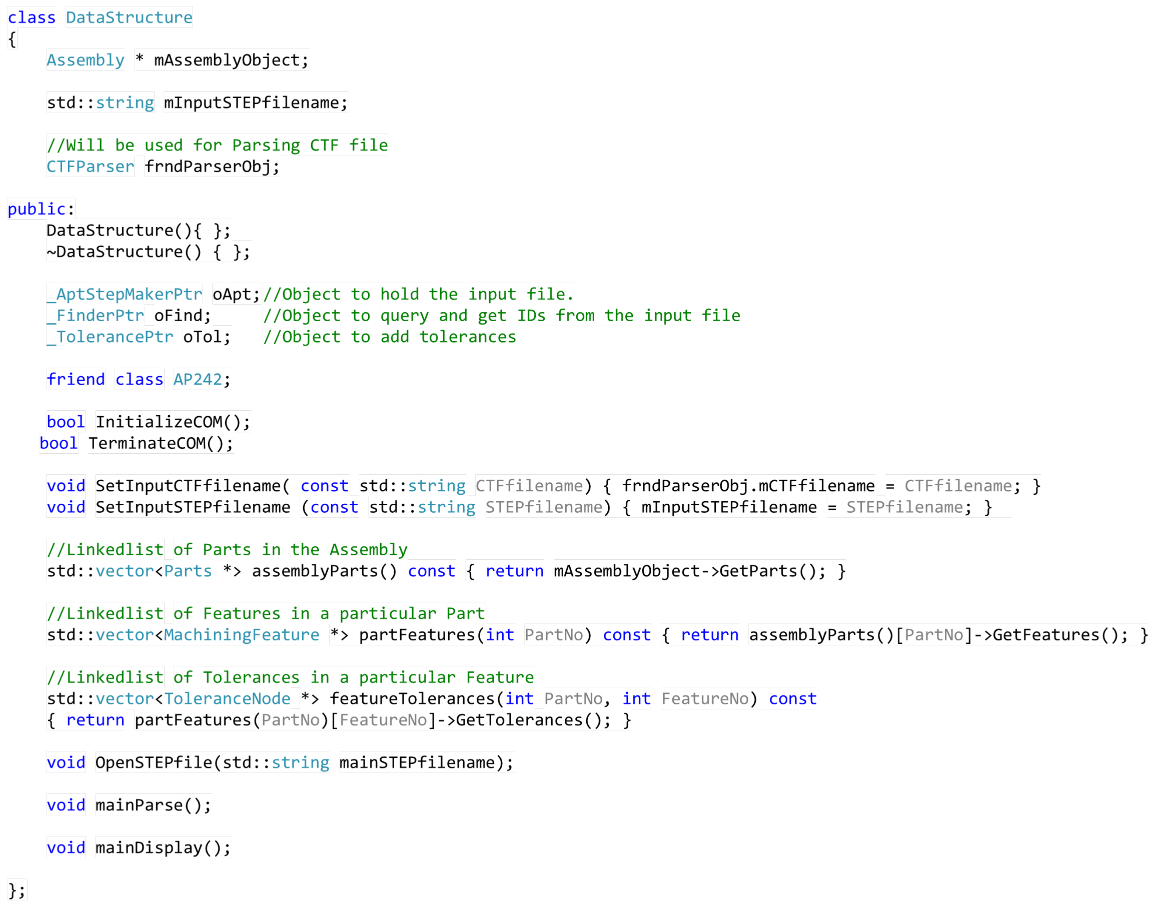
The CTFParser class contains the methods as shown in the code below. Since CTF file has the same structure as a Part 21 file, line numbers act as pointers to specific entity types. CSTtable and FeatureTable are containers of the type std::unordered_map and is used for faster retrieval of entities. CSTtable is used to map the constraint value to the line number whereas FeatureTable is used for mapping feature type to the line number.



```cpp
class CTFParser
{
    std::string mCTFfilename;

    // for CTF file i/o operations
    std::ifstream input_file;

    // Hash table for reading Constraints from CTF file
    std::unordered_map<CST_Face,double,hash_X,newEqual> CSTtable;

    // Hash table for line no and feature pair
    // Will be useful to quickly attach tolerance node to feature
    std::unordered_map<int,MachiningFeature*> FeatureTable;

public:
    CTFParser(){ };
    ~CTFParser(){ };

    friend class DataStructure;

    void OpenCTFfile();

    //--------------------------------------Features---------------------------------------------------
    //eg. #3=GENERAL_MIDPLANE('(FACE1&FACE21)_part1',...

    //Extract line number | returns 3
    int GetLineNo(std::string szEntireLine);

    //Extract geometric entity | returns GENERAl_MIDPLANE
    std::string GetEntity(std::string szEntireLine);

    //Extracts Face Ids of feature, this is the Unique ID (UUID)
    //of the entity present in the input AP 203 file | returns [1,21]
    std::vector<int> ExtractFeatureFaces(std::string szEntireLine);

    //Extract part id of the feature | returns 1
    int GetPartID(std::string szEntireLine);

    MachiningFeature* GetFeatureFromLineNo(int LineNo) { return FeatureTable[LineNo]; }

    //--------------------------------------Constraints------------------------------------------------
    //Extracts Distance Between two entities and the entity IDs
    //e.g #152=CST_DISTANCE(54.483, #17 ,#4) | returns a struct [17,14,54.483]
    CST_DistanceInfo ExtractConstraintDistance(std::string szEntireLine);

    //Return constraint distance when passing both faces as a struct argument
    double GetConstraintDistanceFromFaces(CST_Face obj) { return CSTtable[obj] ; }

    //--------------------------------------Tolerances--------------------------------------------
    //Checks if the line contains tolerance data, i.e checks for 'T_'
    //e.g. #442=T_SIZE(#5, (nFI, 0.13, RFS)) | returns true;
    bool IsToleranceData(std::string szEntireLine);

    //Extracts Tolerance information
    bool ReadToleranceInfo(std::string szEntireLine);

    //getting the values of tolerance type from enum.h
    int tolerance_types(std::string &Tol_type, int &ttype);

    //returns a string value for an input int tolerance type
    std::string CTFParser::sztolerance_types(int ttype);

    //finding the material condition
    int convert_MC(std::string& str);

    //getting the datum information and material condition
    void datum_types_materal_modifiers(std::string szLine, std::string Tol_type, int &pd_l, int &sd_l,
    int &td_l, int &pd_md, int &sd_md, int &td_md);
    //--------------------------------------------------------------------------------------------------
    Assembly* Parse();

};
```

Figure 17: CTFParser class code

The CTF Parser stores all the information into an intermediate graph data structure created by the DataStructure class as shown below.



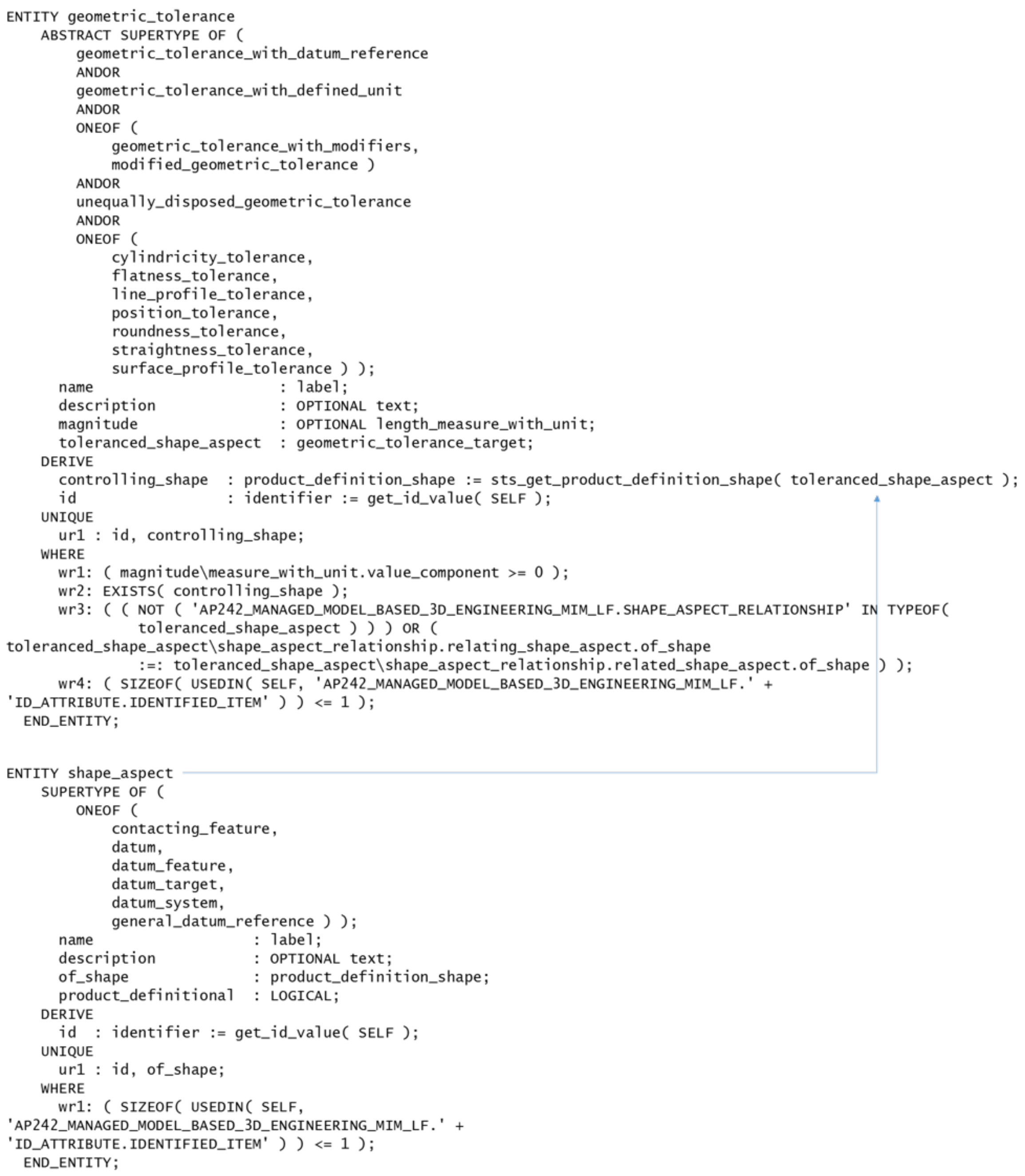
```
class DataStructure
{
    Assembly * mAssemblyObject;

    std::string mInputSTEPfilename;

    //Will be used for Parsing CTF file
    CTFParser frndParserObj;

public:
    DataStructure(){ };
    ~DataStructure() { };

    _AptStepMakerPtr oApt;//Object to hold the input file.
    _FinderPtr oFind;     //Object to query and get IDs from the input file
    _TolerancePtr oTol;   //Object to add tolerances

    friend class AP242;

    bool InitializeCOM();
   bool TerminateCOM();

    void SetInputCTFfilename( const std::string CTFfilename) { frndParserObj.mCTFfilename = CTFfilename; }
    void SetInputSTEPfilename (const std::string STEPfilename) { mInputSTEPfilename = STEPfilename; }

    //Linkedlist of Parts in the Assembly
    std::vector<Parts *> assemblyParts() const { return mAssemblyObject->GetParts(); }

    //Linkedlist of Features in a particular Part
    std::vector<MachiningFeature *> partFeatures(int PartNo) const { return assemblyParts()[PartNo]->GetFeatures(); }

    //Linkedlist of Tolerances in a particular Feature
    std::vector<ToleranceNode *> featureTolerances(int PartNo, int FeatureNo) const
    { return partFeatures(PartNo)[FeatureNo]->GetTolerances(); }

    void OpenSTEPfile(std::string mainSTEPfilename);

    void mainParse();

    void mainDisplay();

};
```

Figure 18: DataStucture class code

4.2.4 GD&T Entities in STEP AP 242

GD&T entities in STEP are modelled using the `geometric_tolerance` entity. An item of the type `geometric_tolerance` refers to a `shape_aspect` entity for identifying the target feature. Figure 19 from Step Tools Inc. explains the STEP entities that are used for assigning a Target Feature to a Feature Control Frame. An entity of type `geometric_tolerance` can have the following subtypes that will be of interest in this research,

1. `geometric_tolerance_with_datum_references` for tolerances that have datum reference frames associated with it. This includes,
    a) `angularity_tolerance`
    b) `concentricity_tolerance`
    c) `parallelism_tolerance`
    d) `perpendicularity_tolerance`
2. `geometric_tolerance_with_modifiers`
   or `modified_geometric_tolerance` for tolerances with the following material modifiers associated with it,
    a) `free_state`
    b) `least_material_requirement`
    c) `maximum_material_requirement`
3. Tolerances that do not have any datums associated with them, such as
    a) `cylindricity_tolerance`
    b) `flatness_tolerance`
    c) `position_tolerance`
    d) `roundness_tolerance`
    e) `straightness_tolerance`



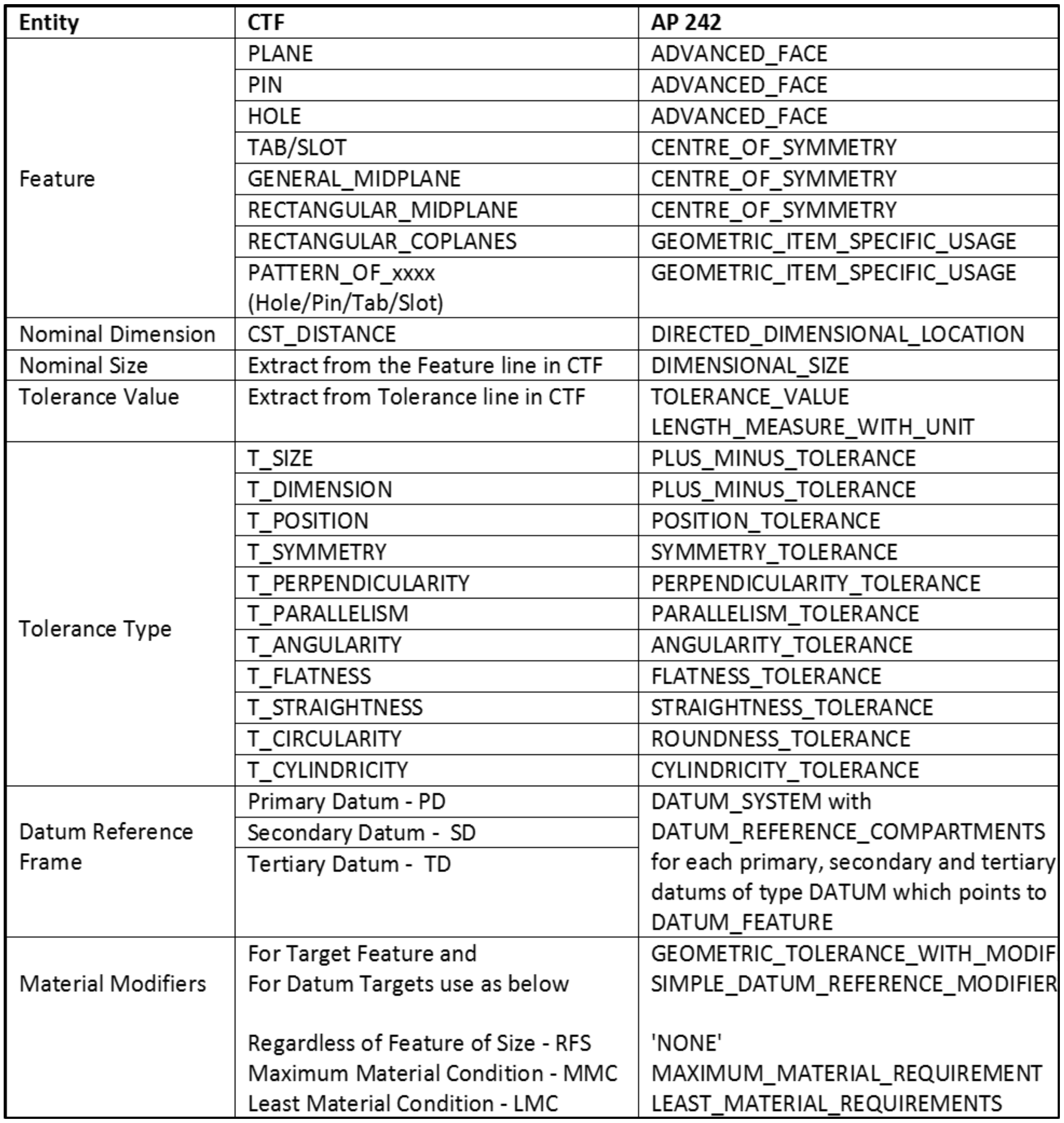
```
ENTITY geometric_tolerance
    ABSTRACT SUPERTYPE OF (
        geometric_tolerance_with_datum_reference
        ANDOR
        geometric_tolerance_with_defined_unit
        ANDOR
        ONEOF (
            geometric_tolerance_with_modifiers,
            modified_geometric_tolerance )
        ANDOR
        unequally_disposed_geometric_tolerance
        ANDOR
        ONEOF (
            cylindricity_tolerance,
            flatness_tolerance,
            line_profile_tolerance,
            position_tolerance,
            roundness_tolerance,
            straightness_tolerance,
            surface_profile_tolerance ) );
      name                     : label;
      description              : OPTIONAL text;
      magnitude                : OPTIONAL length_measure_with_unit;
      toleranced_shape_aspect  : geometric_tolerance_target;
    DERIVE
      controlling_shape  : product_definition_shape := sts_get_product_definition_shape( toleranced_shape_aspect );
      id                 : identifier := get_id_value( SELF );
    UNIQUE
      ur1 : id, controlling_shape;
    WHERE
      wr1: ( magnitude\measure_with_unit.value_component >= 0 );
      wr2: EXISTS( controlling_shape );
      wr3: ( ( NOT ( 'AP242_MANAGED_MODEL_BASED_3D_ENGINEERING_MIM_LF.SHAPE_ASPECT_RELATIONSHIP' IN TYPEOF(
              toleranced_shape_aspect ) ) ) OR (
toleranced_shape_aspect\shape_aspect_relationship.relating_shape_aspect.of_shape
              :=: toleranced_shape_aspect\shape_aspect_relationship.related_shape_aspect.of_shape ) );
      wr4: ( SIZEOF( USEDIN( SELF, 'AP242_MANAGED_MODEL_BASED_3D_ENGINEERING_MIM_LF.' +
'ID_ATTRIBUTE.IDENTIFIED_ITEM' ) ) <= 1 );
  END_ENTITY;


ENTITY shape_aspect
    SUPERTYPE OF (
        ONEOF (
            contacting_feature,
            datum,
            datum_feature,
            datum_target,
            datum_system,
            general_datum_reference ) );
      name                  : label;
      description           : OPTIONAL text;
      of_shape              : product_definition_shape;
      product_definitional  : LOGICAL;
    DERIVE
      id  : identifier := get_id_value( SELF );
    UNIQUE
      ur1 : id, of_shape;
    WHERE
      wr1: ( SIZEOF( USEDIN( SELF,
'AP242_MANAGED_MODEL_BASED_3D_ENGINEERING_MIM_LF.' +
'ID_ATTRIBUTE.IDENTIFIED_ITEM' ) ) <= 1 );
  END_ENTITY;
```

Figure 19: GD&T Entities in STEP AP 242

As suggested in the CAx-IF Recommended Practices, when available, the `advanced_face` is used for representing a feature since its topology is well-defined. However, Tab and Slot features are referenced using the mid-plane derived from the two parallel faces that constitute the feature. In GD&T derived center elements such as mid-planes are considered to be implicitly bounded where they intersect another feature of a part and hence any geometric primitives could be used to represent them. `geometric_representation_item` or `topological_representation_item` is used in this work to represent features or derived elements.

4.2.5 Entity Mapping between CTF and STEP AP 242

The next step is to map the entities in CTF to the corresponding entities in STEP. The complete list of entities that needs to be mapped from CTF to STEP AP 242 is shown in Figure 20.



Entity	CTF	AP 242
Feature	PLANE	ADVANCED_FACE
	PIN	ADVANCED_FACE
	HOLE	ADVANCED_FACE
	TAB/SLOT	CENTRE_OF_SYMMETRY
	GENERAL_MIDPLANE	CENTRE_OF_SYMMETRY
	RECTANGULAR_MIDPLANE	CENTRE_OF_SYMMETRY
	RECTANGULAR_COPLANES	GEOMETRIC_ITEM_SPECIFIC_USAGE
	PATTERN_OF_xxxx (Hole/Pin/Tab/Slot)	GEOMETRIC_ITEM_SPECIFIC_USAGE
Nominal Dimension	CST_DISTANCE	DIRECTED_DIMENSIONAL_LOCATION
Nominal Size	Extract from the Feature line in CTF	DIMENSIONAL_SIZE
Tolerance Value	Extract from Tolerance line in CTF	TOLERANCE_VALUE LENGTH_MEASURE_WITH_UNIT
Tolerance Type	T_SIZE	PLUS_MINUS_TOLERANCE
	T_DIMENSION	PLUS_MINUS_TOLERANCE
	T_POSITION	POSITION_TOLERANCE
	T_SYMMETRY	SYMMETRY_TOLERANCE
	T_PERPENDICULARITY	PERPENDICULARITY_TOLERANCE
	T_PARALLELISM	PARALLELISM_TOLERANCE
	T_ANGULARITY	ANGULARITY_TOLERANCE
	T_FLATNESS	FLATNESS_TOLERANCE
	T_STRAIGHTNESS	STRAIGHTNESS_TOLERANCE
	T_CIRCULARITY	ROUNDNESS_TOLERANCE
	T_CYLINDRICITY	CYLINDRICITY_TOLERANCE
Datum Reference Frame	Primary Datum - PD	DATUM_SYSTEM with DATUM_REFERENCE_COMPARTMENTS for each primary, secondary and tertiary datums of type DATUM which points to DATUM_FEATURE
	Secondary Datum - SD	
	Tertiary Datum - TD	
Material Modifiers	For Target Feature and For Datum Targets use as below	GEOMETRIC_TOLERANCE_WITH_MODIF SIMPLE_DATUM_REFERENCE_MODIFIER
	Regardless of Feature of Size - RFS	'NONE'
	Maximum Material Condition - MMC	MAXIMUM_MATERIAL_REQUIREMENT
	Least Material Condition - LMC	LEAST_MATERIAL_REQUIREMENTS

Figure 20: Entity Mapping from CTF to STEP AP 242

The CTF file contains the same IDs for the topological entities as that of the input STEP AP 203 file and hence attaching tolerance data to these entities can be done through the STEP NC DLL. The DLL utilizes various C++ Component Object Model (COM) objects for achieving this task.

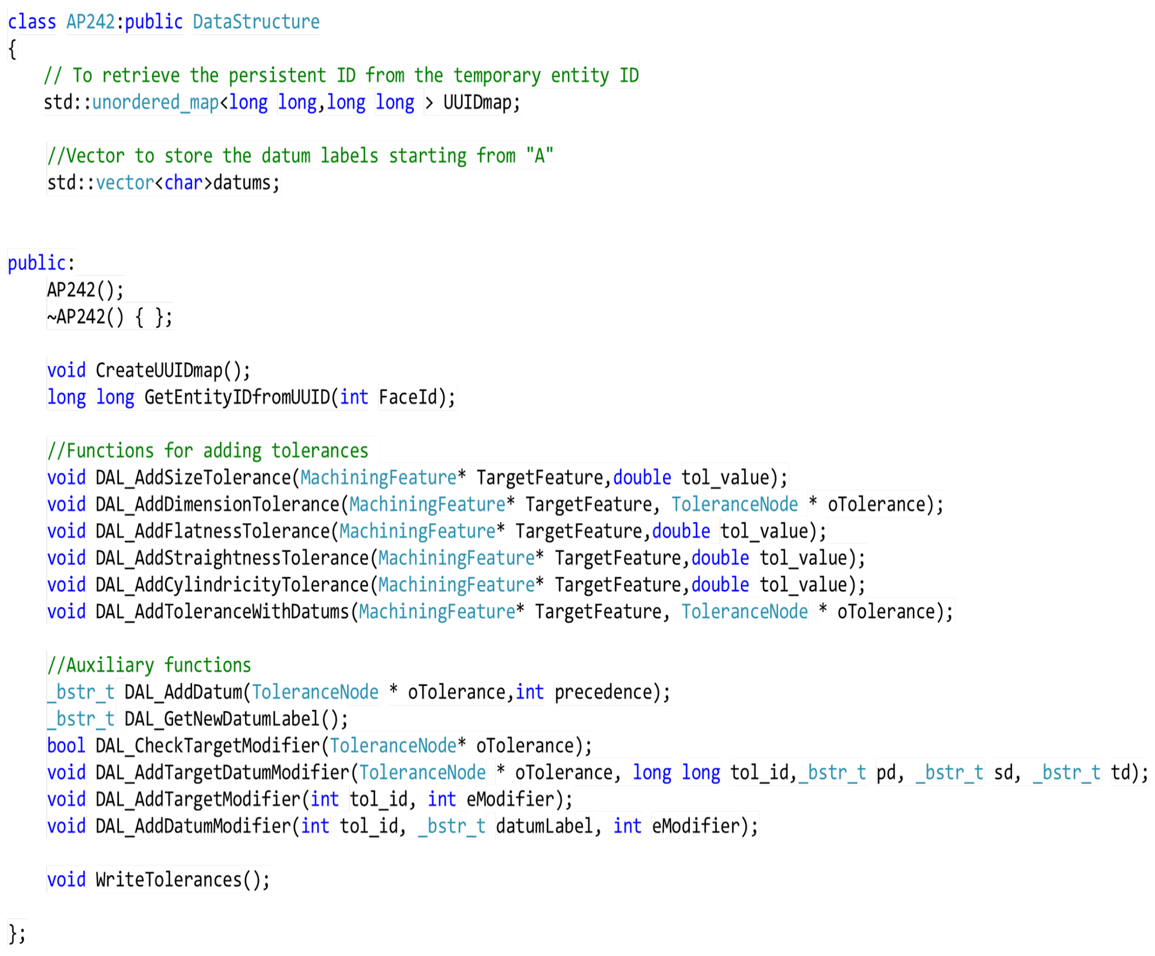
4.2.6 Writing Tolerances to STEP AP 242

After identifying the entity mapping between CTF and AP 242, the tolerance information is written to STEP AP 242 format. The pseudo code for the complete implementation is as shown in Figure 21.

```
Open and query the CTF file
Read complete information and store in the intermediate data structure
Open the input STEP file using STEP NC COM APIs
Query the 3D Assembly in the intermediate data structure
    Query the Parts
        Query the Features
            Query the Tolerances attached
                Compare the tolerance type
                    For Dimension Tolerances
                        Get Target Features
                        Extract Nominal Dimension
                        Extract plus/minus tolerance
                    For Geometric Tolerance
                        Get Target Features
                        Extract Nominal Size, if any
                        Extract  Tolerance value
                        Extract Datum Reference Frame, if any
                            If new datum
                                Create Labels for new datums
                            Else if old datum
                                Reuse the datum by obtaining existing label/ID from STEP data structure.
                Perform basic intrinsic check
                Write Tolerance using appropriate STEP NC APIs
            Go to next Tolerance node
        Go to next Feature
    Go to next Part
Write Output STEP AP 242 file.
Terminate the COM objects
```

Figure 21: Pseudo code for the translation process

The translation process starts by querying the CTF file and storing all the entities into the intermediate data structure. Subsequently, the input STEP AP 203 file is queried using the STEPNC method and all the entities are stored in another intermediate data structure developed by Step Tools Inc. The link between the entities in these two data structures is through the persistent entity IDs. The 3D Assembly available in the former data structure is queried to obtain all the Parts which constitutes the Assembly. For each Part, the target tolerance features are obtained using the persistent IDs along with any nominal parameters that are used to define the feature. After identifying the target feature, the tolerances associated with this feature is obtained. Tolerance information includes the type of tolerance, value, and datum reference frames if any. Finally, using appropriate STEP-NC methods, all the tolerance data is attached to the corresponding target feature STEP entity using persistent IDs. The AP242 class implementation used to attach tolerances using the STEPNC DLL is shown below,



```
class AP242:public DataStructure
{
   // To retrieve the persistent ID from the temporary entity ID
   std::unordered_map<long long,long long > UUIDmap;

    //Vector to store the datum labels starting from "A"
    std::vector<char>datums;


public:
    AP242();
    ~AP242() { };

    void CreateUUIDmap();
    long long GetEntityIDfromUUID(int FaceId);

    //Functions for adding tolerances
    void DAL_AddSizeTolerance(MachiningFeature* TargetFeature,double tol_value);
    void DAL_AddDimensionTolerance(MachiningFeature* TargetFeature, ToleranceNode * oTolerance);
    void DAL_AddFlatnessTolerance(MachiningFeature* TargetFeature,double tol_value);
    void DAL_AddStraightnessTolerance(MachiningFeature* TargetFeature,double tol_value);
    void DAL_AddCylindricityTolerance(MachiningFeature* TargetFeature,double tol_value);
    void DAL_AddToleranceWithDatums(MachiningFeature* TargetFeature, ToleranceNode * oTolerance);

    //Auxiliary functions
    _bstr_t DAL_AddDatum(ToleranceNode * oTolerance,int precedence);
    _bstr_t DAL_GetNewDatumLabel();
    bool DAL_CheckTargetModifier(ToleranceNode* oTolerance);
    void DAL_AddTargetDatumModifier(ToleranceNode * oTolerance, long long tol_id,_bstr_t pd, _bstr_t sd, _bstr_t td);
    void DAL_AddTargetModifier(int tol_id, int eModifier);
    void DAL_AddDatumModifier(int tol_id, _bstr_t datumLabel, int eModifier);

    void WriteTolerances();

};
```

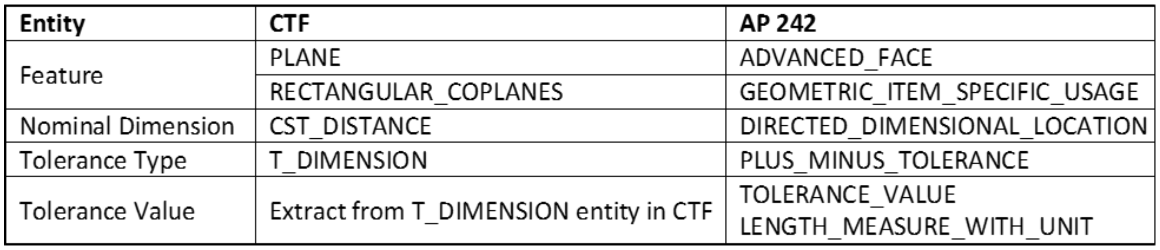
Figure 22: AP242 Class to attach tolerances

4.2.6.1 Dimensional Tolerances

Dimensional Tolerance is attached to entities that have a distance constraint associated between them. In CTF file, the nominal dimension is represented by the `CST_DISTANCE` entity which stores the nominal dimension, and also the two Features between which the constraint is applied. The first feature is the tolerance target whereas the second feature is considered as a datum from where measurements are taken. However, the second feature is not mapped to any datum feature in STEP since Dimensional tolerances do not have datums associated with them. Further, CTF stores the tolerance associated with `CST_DISTANCE` using `T_DIMENSION` entity.

- `CST_DISTANCE` (nominal dimension, #first feature, #second feature);
- `T_DIMENSION` (#first feature, (Diameter symbol, Value, Material Modifier), Primary Datum (#second feature, Material Modifier));

While writing the above data to STEP, if any of the features contain multiple faces, they are combined to a `shape_aspect` using the `GEOMETRIC_ITEM_SPECIFIC_USAGE` entity that represents multiple faces in the feature. The entity mapping from CTF to STEP for Dimensional Tolerance is shown in Figure 23.



Entity	CTF	AP 242
Feature	PLANE	ADVANCED_FACE
	RECTANGULAR_COPLANES	GEOMETRIC_ITEM_SPECIFIC_USAGE
Nominal Dimension	CST_DISTANCE	DIRECTED_DIMENSIONAL_LOCATION
Tolerance Type	T_DIMENSION	PLUS_MINUS_TOLERANCE
Tolerance Value	Extract from T_DIMENSION entity in CTF	TOLERANCE_VALUE LENGTH_MEASURE_WITH_UNIT

Figure 23: Entity mapping from CTF to STEP for Dimensional Tolerances

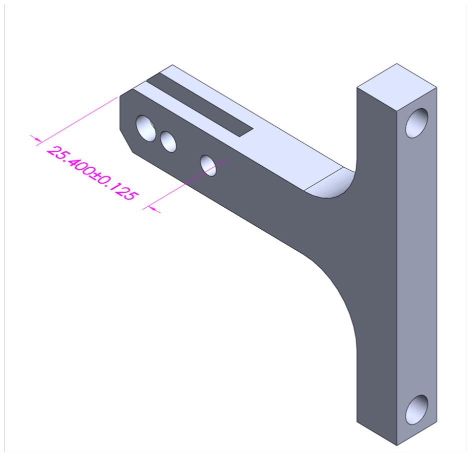
The STEPNC method used for adding multiple faces to a feature is AddFaceToCalloutAdd(). This method takes one geometric representation item, most commonly an advanced face and further attaches the other face to forms a new call out with multiple faces. The persistent IDs of both these faces are passed as arguments to the method,

```
void AddFaceToCalloutAdd (
        int id,
        int face_id
        );
```

Lastly, the distance tolerance is added using the `AddToleranceDistanceLinear()` method. The input arguments include both the feature IDs, nominal value, tolerance upper bound and tolerance lower bound as shown below. Note that the CTF file represents the total tolerance budget, whereas while converting it to STEP, it is represented using the equal bilateral tolerances.

```
int AddToleranceDistanceLinear (
        int fea1_id,
        int fea2_id,
        double value,
        double plus,
        double minus
        );
```

Consider the example shown in Figure 24. The slot on the part has a length of 25.400mm with a tolerance of 0.250 mm. This constraint is attached between the RECTANGULAR_COPLANES feature on the left end of the slot and the RECTANGULAR_PLANE on the right end of the slot. The CTF representation and the corresponding AP 203 ADVANCED_FACE entities with persistent IDs is shown below.



CTF Entities

```
#242=T_DIMENSION(#6, (nFI, 0.25, NONE), PD(#15, NONE));
#64=CST_DISTANCE(25.4, #15 ,#6);
#6=RECTANGULAR_PLANE('(FACE18)_part1', ...);
#15=RECTANGULAR_COPLANES('(FACE2&FACE25)_part1', …);
```

STEP AP203 Entities

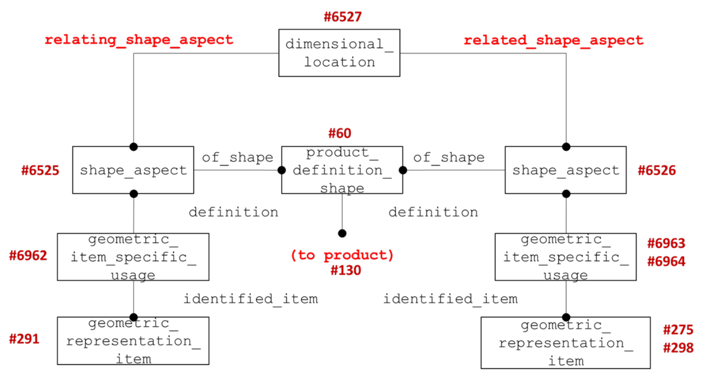
```
#275=ADVANCED_FACE('2',(#506),#507,.F.);
#291=ADVANCED_FACE('18',(#548),#549,.F.);
#298=ADVANCED_FACE('25',(#570),#571,.F.);
```

STEP NC API Instantiation

```
AddFaceToCalloutAdd (275,298);
AddToleranceDistanceLinear(291,275,25.4,0.125,0.125);
```

Figure 24: Instantiation of Dimensional Location with tolerance

Dimensional tolerance is attached to the STEP topological entities using the C++ method DAL_AddDimensionTolerance() defined in the AP242 class which contains the STEPNC API AddToleranceDistanceLinear(). The output in the AP 242 format is shown in 25. The constraint distance can be seen in #6527 under the DIRECTED_DIMENSIONAL_LOCATION entity in the AP 242 file along with the PLUS_MINUS_TOLERANCE in #6794. The RECTANGULAR_COPLANES are



STEP AP 242 Entities

```
#6730=DIMENSIONAL_CHARACTERISTIC_REPRESENTATION(#6527,#6762);
#6762=SHAPE_DIMENSION_REPRESENTATION('',(#6834),$);
#6527=DIRECTED_DIMENSIONAL_LOCATION('linear distance','',#6525,#6526);
#6525=SHAPE_ASPECT('','',#60,.F.);
#6526=SHAPE_ASPECT('','',#60,.F.);
#6962=GEOMETRIC_ITEM_SPECIFIC_USAGE('','',#6525,$,#291);
#6963=GEOMETRIC_ITEM_SPECIFIC_USAGE('','',#6525,$,#298);
#6964=GEOMETRIC_ITEM_SPECIFIC_USAGE('','',#6526,$,#275);
#275=ADVANCED_FACE('2',(#506),#507,.F.);
#291=ADVANCED_FACE('18',(#548),#549,.F.);
#298=ADVANCED_FACE('25',(#570),#571,.F.);
```

Entities between which the dimension is applied

```
#60=PRODUCT_DEFINITION_SHAPE('NONE','NONE',#131);
#131=PRODUCT_DEFINITION('NONE','NONE',#196,#1);
#196=PRODUCT_DEFINITION_FORMATION_WITH_SPECIFIED_SOURCE(' ','NONE',#130,
 .NOT_KNOWN.);
#130=PRODUCT('1','LPattern1','PART-LPattern1-DESC',(#195));

#6834=(
LENGTH_MEASURE_WITH_UNIT()
MEASURE_REPRESENTATION_ITEM()
MEASURE_WITH_UNIT(LENGTH_MEASURE(25.4),#7035)
REPRESENTATION_ITEM('linear distance')
);
```

Nominal Value

```
#6794=PLUS_MINUS_TOLERANCE(#6922,#6527);
#6922=TOLERANCE_VALUE(#6832,#6833);
#6832=(
LENGTH_MEASURE_WITH_UNIT()
MEASURE_REPRESENTATION_ITEM()
MEASURE_WITH_UNIT(LENGTH_MEASURE(0.125),#7035)
REPRESENTATION_ITEM('linear distance')
);
#6833=(
LENGTH_MEASURE_WITH_UNIT()
MEASURE_REPRESENTATION_ITEM()
MEASURE_WITH_UNIT(LENGTH_MEASURE(0.125),#7035)
REPRESENTATION_ITEM('linear distance')
);
```

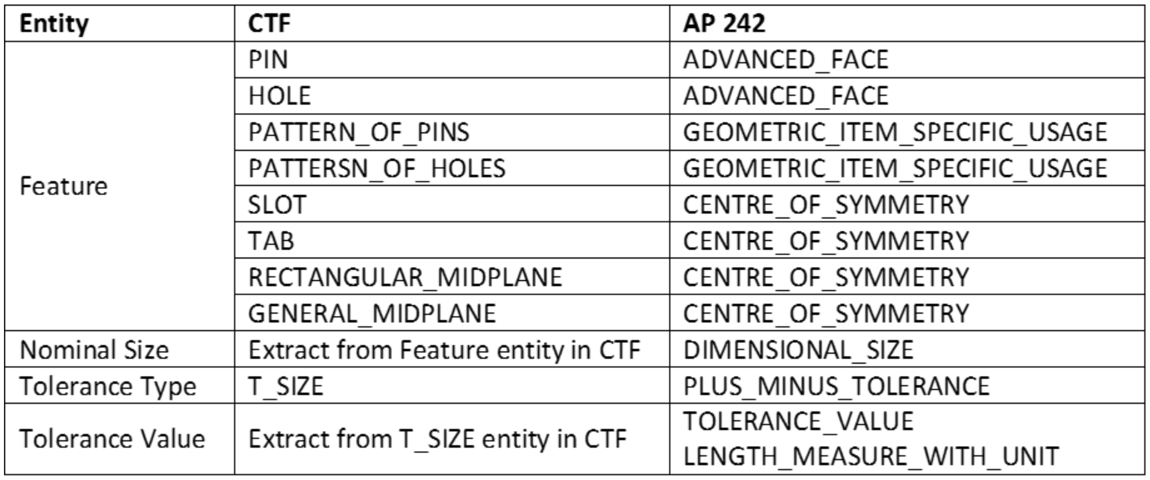
Tolerance Value

Figure 25: Dimensional Location in AP 242

mapped to the SHAPE_ASPECT in #6525 and the RECTANGULAR_PLANE on the right end of the slot is mapped to the SHAPE_ASPECT in #6526.

4.2.6.2 Size Tolerances

Size tolerances are applied to feature of size(FOS) such as Pin, Hole, Tab and Slot and do not require datums for instantiation. Like Dimensional Tolerances, Size tolerances are also specified using the +/− limits. This translator converts the tolerances budget available in the CTF file to equal bilateral tolerances. Size Tolerances are usually attached to the nominal parameters such as radius of a pin or hole, width and length of a tab or slot feature. In CTF file, the nominal size of these features are available in the corresponding `PIN, HOLE, TAB and SLOT` entities. They can also be applied to `PATTERN_OF_` any of the above features, in which case they are combined to a `shape_aspect` using the `GEOMETRIC_ITEM_SPECIFIC_USAGE` entity that represents individual features in the pattern. The entity mapping from CTF to STEP for Size Tolerance is shown in Figure 26.



Entity	CTF	AP 242
Feature	PIN	ADVANCED_FACE
	HOLE	ADVANCED_FACE
	PATTERN_OF_PINS	GEOMETRIC_ITEM_SPECIFIC_USAGE
	PATTERSN_OF_HOLES	GEOMETRIC_ITEM_SPECIFIC_USAGE
	SLOT	CENTRE_OF_SYMMETRY
	TAB	CENTRE_OF_SYMMETRY
	RECTANGULAR_MIDPLANE	CENTRE_OF_SYMMETRY
	GENERAL_MIDPLANE	CENTRE_OF_SYMMETRY
Nominal Size	Extract from Feature entity in CTF	DIMENSIONAL_SIZE
Tolerance Type	T_SIZE	PLUS_MINUS_TOLERANCE
Tolerance Value	Extract from T_SIZE entity in CTF	TOLERANCE_VALUE LENGTH_MEASURE_WITH_UNIT

Figure 26: Entity mapping from CTF to STEP for Size Tolerances

As discussed in section 4.2.6.1, the STEPNC method used for adding multiple faces to a feature is,

```
AddFaceToCalloutAdd( int id, int face_id );
```

Lastly, the Size tolerance is added using either of the following Step Tools methods

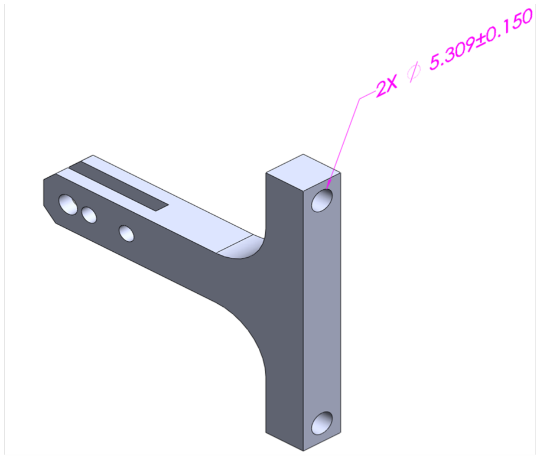
1.
```
int AddToleranceSizeRadial/Diameter (
        int fea_id,
        double value,
        double plus,
        double minus
        );
```

2.
```
int  AddToleranceSizeLength/Height/Width(
    int fea_id,
        double value,
        double plus,
        double minus
        );
```

3. Or using the earlier described,

```
int AddToleranceDistanceLinear (
        int fea1_id,
        int fea2_id,
        double value,
        double plus,
        double minus
        );
```

The input arguments for instantiation 1 is usually a `shape_aspect` entity such as the `advanced_face` that is required to define the target feature. However, instantiation 2 requires the feature to be defined by passing all the constituting faces as arguments to the `AddFaceToCalloutAdd()` method. Instantiations 3 requires both the feature IDs, nominal value, tolerance upper bound and tolerance lower bound to be passed as argument. Consider the example shown in Figure 27. The two holes form a pattern and a size tolerance of 0.150mm is applied to the 5.309 mm diameter of the two holes. This information is represented in the CTF file using the `T_SIZE` and `PATTERN_OF_HOLES` entity as shown. The corresponding AP 203 `ADVANCED_FACE` entities with persistent IDs is also shown below.



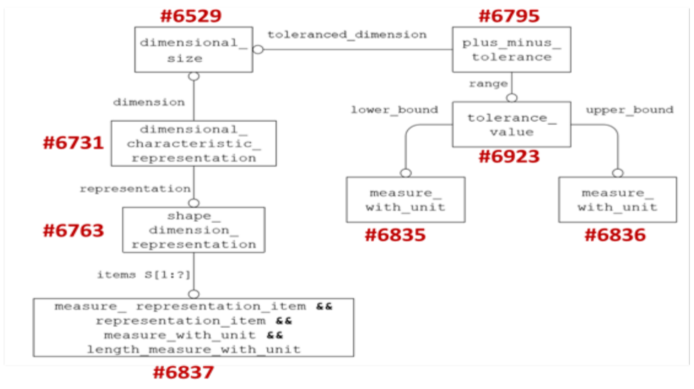
CTF Entities
```
#244=T_SIZE(#7, (nFI, 0.15, RFS));
#7=PATTERN_OF_HOLES(2'(FACE23&FACE22)_part1', [-1, 5.20417e-017, 0], 2.6543, 9.525);
#8=HOLE('(FACE23)_part1', (48.1243, -23.5114, 10.2133), [-1, 5.20417e-017, 0], 2.6543, 9.525);
#9=HOLE('(FACE22)_part1', (48.1243, 33.6386, 10.2133), [-1, 5.20417e-017, 0], 2.6543, 9.525);
```

STEP AP203 Entities
```
#295=ADVANCED_FACE('22',(#562,#563),#564,.F.);
#296=ADVANCED_FACE('23',(#565,#566),#567,.F.);
```

```
AddFaceToCalloutAdd (295,296);
AddToleranceSizeRadial(295,2.6543,0.075,0.075);
```

Figure 27: Instantiation of Radial Size with tolerance

The CTF pin and hole features represent their size using the radius and hence tolerance is attached to the STEP topological entities using the C++ method `DAL_AddSizeTolerance()` defined in the AP242 class. This method contains the STEPNC API `AddToleranceSizeRadial()`. However, for Tab and Slot features, the CTF feature represents the width and depth of the slot using the same method, but internally calls the `AddToleranceDistanceLinear()` method. The output AP 242 entities for the above example is shown in 28. The radial size of the holes can be seen in #6529 under the `DIMENSIONAL_SIZE` entity in the AP 242 file along with the `PLUS_MINUS_TOLERANCE` in #6795. The `PATTERN_OF_HOLES` is mapped to the `SHAPE_ASPECT` in #6528 using the `GEOMETRIC_ITEM_SPECIFIC_USAGE` entities in #6965 and #6966 which contains the `ADVANCED_FACE` entities that represent the two `HOLE`s.



STEP AP 242 Entities

```
#6731=DIMENSIONAL_CHARACTERISTIC_REPRESENTATION(#6529,#6763);
#6529=DIMENSIONAL_SIZE(#6528,'radius');
#6763=SHAPE_DIMENSION_REPRESENTATION('',(#6837),$);
#6528=SHAPE_ASPECT('','',#60,.F.);
#6965=GEOMETRIC_ITEM_SPECIFIC_USAGE('','',#6528,$,#296);
#6966=GEOMETRIC_ITEM_SPECIFIC_USAGE('','',#6528,$,#295);
#295=ADVANCED_FACE('22',(#562,#563),#564,.F.);
#296=ADVANCED_FACE('23',(#565,#566),#567,.F.);
#6795=PLUS_MINUS_TOLERANCE(#6923,#6529);
#6923=TOLERANCE_VALUE(#6835,#6836);
#6835=(
LENGTH_MEASURE_WITH_UNIT()
MEASURE_REPRESENTATION_ITEM()
MEASURE_WITH_UNIT(LENGTH_MEASURE(0.075),#7034)
REPRESENTATION_ITEM('radial size')
);
#6836=(
LENGTH_MEASURE_WITH_UNIT()
MEASURE_REPRESENTATION_ITEM()
MEASURE_WITH_UNIT(LENGTH_MEASURE(0.075),#7034)
REPRESENTATION_ITEM('radial size')
);
#6837=(
LENGTH_MEASURE_WITH_UNIT()
MEASURE_REPRESENTATION_ITEM()
MEASURE_WITH_UNIT(LENGTH_MEASURE(2.6543),#7034)
REPRESENTATION_ITEM('radial size')
);
```

Two holes patterned through
`GEOMETRIC_ITEM_SPECIFIC_USAGE`

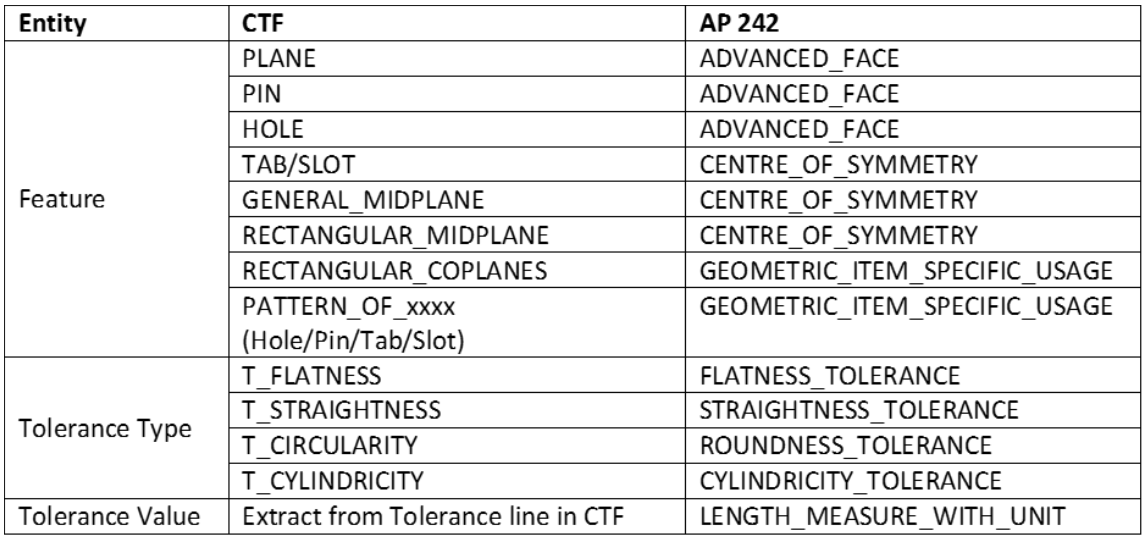
Tolerance Value

Nominal Value

Figure 28: Dimensional Size in AP 242

4.2.6.3 Geometric Tolerance (without Datums) with Material Modifiers

The following form tolerances are implemented without datums during instantiation. However, they can have material modifiers associated with the target features. The entity mapping from CTF to STEP for Form Tolerance is shown in Figure 29.



Entity	CTF	AP 242
Feature	PLANE	ADVANCED_FACE
	PIN	ADVANCED_FACE
	HOLE	ADVANCED_FACE
	TAB/SLOT	CENTRE_OF_SYMMETRY
	GENERAL_MIDPLANE	CENTRE_OF_SYMMETRY
	RECTANGULAR_MIDPLANE	CENTRE_OF_SYMMETRY
	RECTANGULAR_COPLANES	GEOMETRIC_ITEM_SPECIFIC_USAGE
	PATTERN_OF_xxxx (Hole/Pin/Tab/Slot)	GEOMETRIC_ITEM_SPECIFIC_USAGE
Tolerance Type	T_FLATNESS	FLATNESS_TOLERANCE
	T_STRAIGHTNESS	STRAIGHTNESS_TOLERANCE
	T_CIRCULARITY	ROUNDNESS_TOLERANCE
	T_CYLINDRICITY	CYLINDRICITY_TOLERANCE
Tolerance Value	Extract from Tolerance line in CTF	LENGTH_MEASURE_WITH_UNIT

Figure 29: Entity mapping from CTF to STEP for Form Tolerances

1. **Flatness** : In the CTF file, Flatness is represented using,

   - `T_FLATNESS`(#Target Feature, (Diameter Symbol(nFI), Tolerance Value, Material Modifier)).

   To write out AP 242 file, the `DAL_AddFlatnessTolerance`() method is called which internally calls the `AddToleranceFlatness`() method if there are no material modifiers associated with the target feature. However, when there is a material modifier associated with the target feature, `AddTolerance-FlatnessWithFlags`() is called internally.

```
void DAL_AddFlatnessTolerance(
    MachiningFeature* TargetFeature,
    double tol_value
    );
int AddToleranceFlatness[WithFlags] (
        int fea_id,
        double value,
        [int flags]
        );
```

2. **Cylindricity** : Cylindricity is represented as follows in the CTF file,

   - `T_CYLINDRICITY`(#Target Feature, (Diameter Symbol, Tolerance Value, Material Modifier))

The `DAL_AddCylindricityTolerance`() method is called to write out AP 242 file, which internally calls the `AddToleranceCylindricity`() method if there are no material modifiers associated with the target feature. Similar to Flatness tolerance, if there is a material modifier associated with the target feature, `AddToleranceCylindricityWithFlags`() is called internally.

```
void DAL_AddCylindricityTolerance(
    MachiningFeature* TargetFeature,
    double tol_value
    );
int AddToleranceCylindricity[WithFlags] (
        int fea_id,
        double value,
        [int flags]
        );
```

3. **Circularity (Roundness)**: Same instantiation as above using,

```
void DAL_AddCylindricityTolerance(
    MachiningFeature* TargetFeature,
    double tol_value
    );
int AddToleranceRoundness[WithFlags] (
        int fea_id,
        double value,
        [int flags]
        );
```

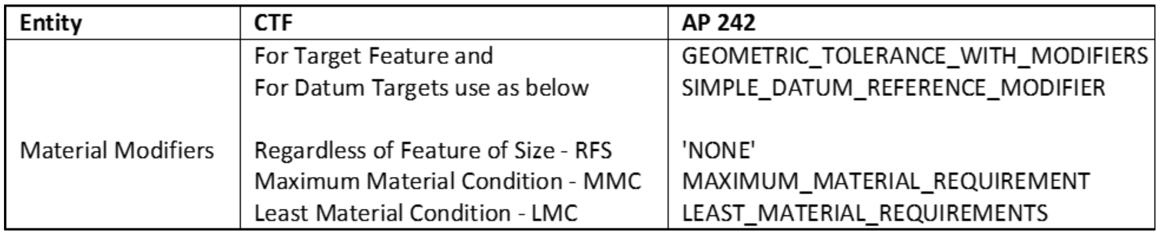
4. **Straightness**: Again, same instantiation as above using,

```
void DAL_AddStraightnessTolerance(
    MachiningFeature* TargetFeature,
    double tol_value
    );
int AddToleranceStraightness[WithFlags] (
        int fea_id,
        double value,
        [int flags]
        );
```

In this implementation, the target feature for Straightness is a shape_aspect entity defined by the advanced_face entity.

5. Also, Profile and Position tolerance is sometimes instantiated without datums.

Whenever tolerance targets have material modifiers associated with them, the entity mapping from CTF to STEP for Material Modifiers is as shown in Figure 30.



Entity	CTF	AP 242
	For Target Feature and	GEOMETRIC_TOLERANCE_WITH_MODIFIERS
	For Datum Targets use as below	SIMPLE_DATUM_REFERENCE_MODIFIER
Material Modifiers	Regardless of Feature of Size - RFS	'NONE'
	Maximum Material Condition - MMC	MAXIMUM_MATERIAL_REQUIREMENT
	Least Material Condition - LMC	LEAST_MATERIAL_REQUIREMENTS

Figure 30: Entity mapping from CTF to STEP for Material Modifiers

```
void DAL_AddTargetModifier(
    int tol_id,
    int eModifier
    );
```

The above method internally calls the [WithFlags] version of the STEPNC methods, e.g. `AddToleranceStraightnessWithFlags()`. In this case, another method is called to assign material modifier to the target feature as follows,

```
void AddModifierToTolerance (
    int tol_id,
    string^ modifier
    );
```

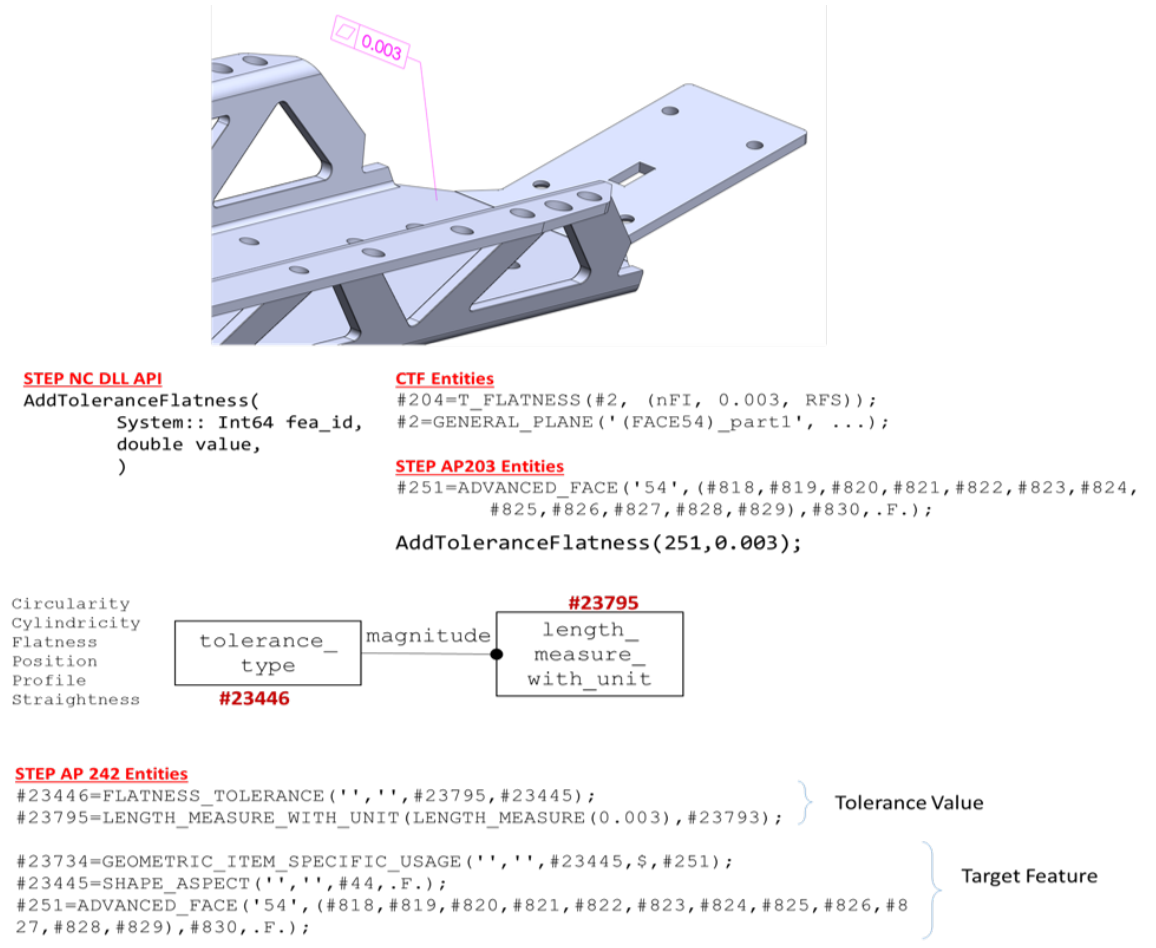
where `string modifier` can accept the following arguments in this translator,

1. "any_cross_section"
2. "free_state"

3. “least_material_requirement”

4. “maximum_material_requirement”

Consider the example shown in Figure 31. A Flatness tolerance of 0.003 mm is applied to the bigger face shown in the model. This information is represented in the CTF file using the `T_FLATNESS` and `GENERAL_PLANE` entity as shown. The corresponding AP 203 `ADVANCED_FACE` entities with persistent IDs is also shown below.



STEP NC DLL API

```
AddToleranceFlatness(
        System:: Int64 fea_id,
        double value,
        )
```

CTF Entities

```
#204=T_FLATNESS(#2, (nFI, 0.003, RFS));
#2=GENERAL_PLANE('(FACE54)_part1', ...);
```

STEP AP203 Entities

```
#251=ADVANCED_FACE('54',(#818,#819,#820,#821,#822,#823,#824,
        #825,#826,#827,#828,#829),#830,.F.);
AddToleranceFlatness(251,0.003);
```

STEP AP 242 Entities

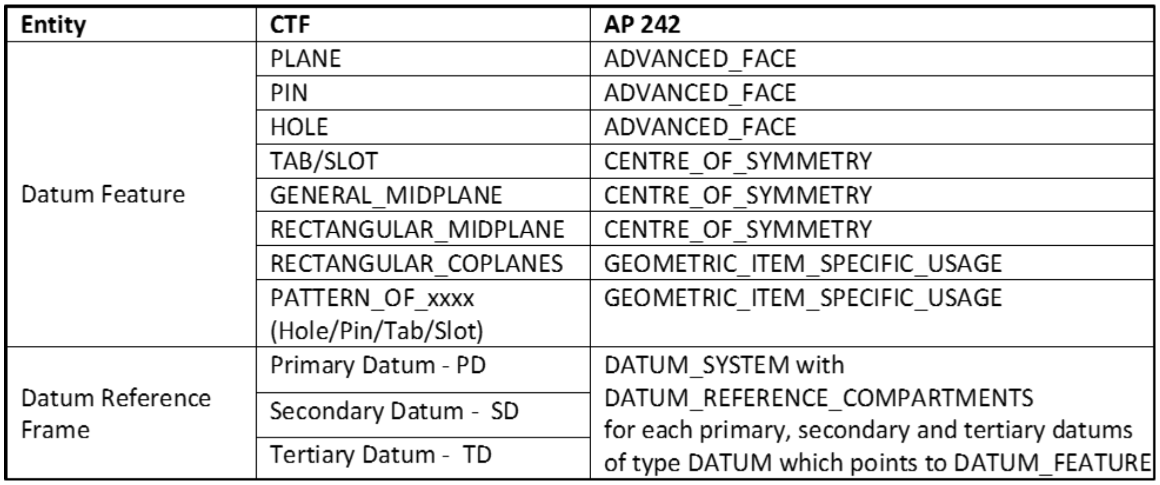
```
#23446=FLATNESS_TOLERANCE('','',#23795,#23445);
#23795=LENGTH_MEASURE_WITH_UNIT(LENGTH_MEASURE(0.003),#23793);

#23734=GEOMETRIC_ITEM_SPECIFIC_USAGE('','',#23445,$,#251);
#23445=SHAPE_ASPECT('','',#44,.F.);
#251=ADVANCED_FACE('54',(#818,#819,#820,#821,#822,#823,#824,#825,#826,#8
27,#828,#829),#830,.F.);
```

Figure 31: Tolerance without datums in AP 242

Flatness tolerance is attached to the STEP topological entities using the C++ method as described above. The output in the AP 242 format is also shown. #23446 represents the FLATNESS_TOLERANCE entity in the AP 242 file along with the value in LENGTH_MEASURE_WITH_UNIT entity in #23795. The GENERAL_PLANE target feature is mapped to the SHAPE_ASPECT in #23445 which points to the ADVANCED_FACE entity in #251 represented by the target face.

4.2.6.4 Datums and Datum Features

All tolerances other than size and form require the specification of datums during instantiation. Datums are theoretically exact points, axes, or planes used as references for measurements on the actual part and hence lie on the boundary of a part. Whenever available, advanced_face is used to represent the faces that constitute the datums. When feature of size (FOS) is used as a datum, it is actually using the resolved entity of the FOS such as axis or mid-plane for reference. A Datum Reference Frame (DRF) is a set of two or three mutually perpendicular datums used to establish the coordinate system for measurements. Datum Precedence refers to Datums listed in order: primary, secondary, tertiary to control 6 degrees of freedom. Also, there can be several DRF on the same part and each datum can be used any number of times in DRFs, as required. The entity mapping from CTF to STEP for Datums and Datum Feature is shown in Figure 32.

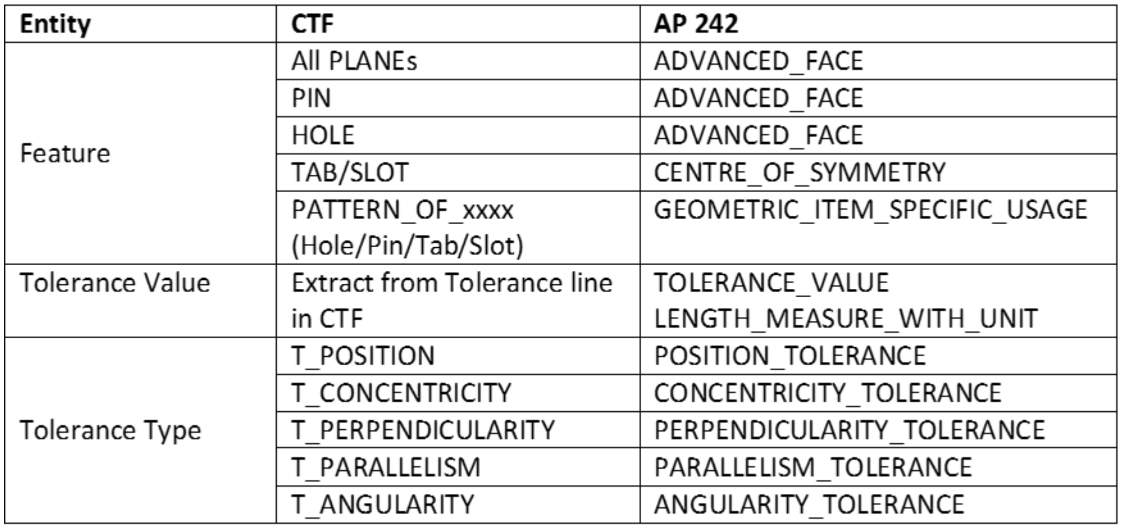


Entity	CTF	AP 242
Datum Feature	PLANE	ADVANCED_FACE
	PIN	ADVANCED_FACE
	HOLE	ADVANCED_FACE
	TAB/SLOT	CENTRE_OF_SYMMETRY
	GENERAL_MIDPLANE	CENTRE_OF_SYMMETRY
	RECTANGULAR_MIDPLANE	CENTRE_OF_SYMMETRY
	RECTANGULAR_COPLANES	GEOMETRIC_ITEM_SPECIFIC_USAGE
	PATTERN_OF_xxxx (Hole/Pin/Tab/Slot)	GEOMETRIC_ITEM_SPECIFIC_USAGE
Datum Reference Frame	Primary Datum - PD	DATUM_SYSTEM with DATUM_REFERENCE_COMPARTMENTS for each primary, secondary and tertiary datums of type DATUM which points to DATUM_FEATURE
	Secondary Datum - SD	
	Tertiary Datum - TD	

Figure 32: STEP Entities for Datum and Datum Features

4.2.6.5 Geometric Tolerance with Modifiers and Datums

Orientation and Location tolerances require datums for their instantiation. These datum features can also have material modifiers associated with it. The entity mapping from CTF to STEP for Geometric Tolerance with Datums is shown in Figure 33.



Entity	CTF	AP 242
Feature	All PLANEs	ADVANCED_FACE
	PIN	ADVANCED_FACE
	HOLE	ADVANCED_FACE
	TAB/SLOT	CENTRE_OF_SYMMETRY
	PATTERN_OF_xxxx (Hole/Pin/Tab/Slot)	GEOMETRIC_ITEM_SPECIFIC_USAGE
Tolerance Value	Extract from Tolerance line in CTF	TOLERANCE_VALUE LENGTH_MEASURE_WITH_UNIT
Tolerance Type	T_POSITION	POSITION_TOLERANCE
	T_CONCENTRICITY	CONCENTRICITY_TOLERANCE
	T_PERPENDICULARITY	PERPENDICULARITY_TOLERANCE
	T_PARALLELISM	PARALLELISM_TOLERANCE
	T_ANGULARITY	ANGULARITY_TOLERANCE

Figure 33: STEP Entities for Geometric Tolerance with Datums

1. **Location** : In the CTF file, Location Tolerance is represented using `T_POSITION/T_CONCENTRICITY(` #Target Feature, (Diameter Symbol(nFI), Tolerance Value, Target Material Modifier), PD(#Datum Feature, Datum Material Modifier), SD(#Datum Feature, Datum Material Modifier), TD(#Datum Feature, Datum Material Modifier));
2. **Orientation** : Similarly, Orientation tolerance is represented using `T_PARALLELISM/T_ANGULARITY/T_PERPENDICULARITY(` #Target Feature, (Diameter Symbol(nFI), Tolerance Value, Target Material Modifier), PD(#Datum Feature, Datum Material Modifier)); The `DAL_AddToleranceWithDatums()` method is called to write out AP 242 file, which internally calls the `AddTolerance[tolerance type]()` or `AddTolerance[tolerance type]WithFlags()` method.

   ```
   void DAL_AddToleranceWithDatums(
       MachiningFeature* TargetFeature,
       double tol_value
       );
   int AddTolerancePosition[WithFlags](
           int fea_id,
           double value,
           string^ datums,
           [int flags]
           );
   ```

   The Datums which are passed as int arguments to the above method is created using the `AddDatum()` function. It creates a new datum with the given label and returns the id of the new object. There is another function

DAL_GetNewDatumLabel() which progressively generates new datum labels starting from alphabet 'A'.

```
int AddDatum (
        string^ label,
        int face_id
        );
```

If an existing datum is to be re-used, then the GetDatumCount() function returns a count of the number of datums previously defined in the model and GetDatumNext() iterates over the datums returning the object identifier of the datum and several attributes as out parameters. The label of the returned datum object can be obtained by passing the identifier to the GetDatumLabel() method.

```
int GetDatumCount();
int GetDatumNext(
        int index,
        string^ label,
        string^ modifier
        );
string GetDatumLabel(
        int dat_id
        );
```

```
void DAL_AddStraightnessTolerance(
    MachiningFeature* TargetFeature,
    double tol_value
    );
int AddToleranceStraightness[WithFlags](
        int fea_id,
        double value,
        [int flags]
        );
```

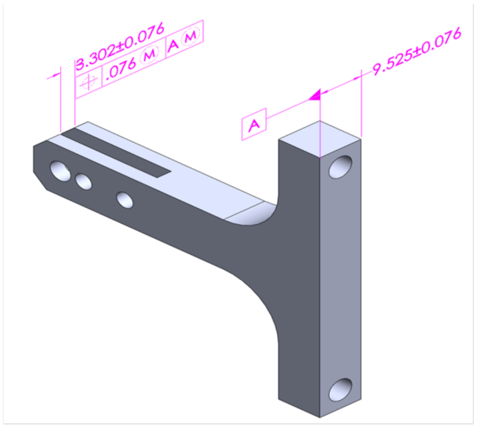
In this implementation, the target feature for Straightness is a shape_aspect entity defined by the advanced_face entity.

3. Also, Profile and Position tolerance is sometimes instantiated without datums.

Whenever datum features have material modifiers associated with them, the `DAL_AddDatumModifier()` method is called which internally calls the `AddModifierToDatumTolerance()`. Material Modifier is an enum type in the former function whereas it is a string argument in the latter.

```
void DAL_AddDatumModifier(
    int tol_id,
    _bstr_t datumLabel,
    int eModifier
    );
void AddModifierToDatumTolerance(
        int tol_id,
        string^ datum_label,
        string^ modifier
        );
```

Consider the example shown in Figure 34. A Position tolerance of 0.076 mm is applied to the slot shown in the model. This information is represented in the CTF file using the `T_POSITION` in #258 and `SLOT` in #4. The Datum Feature 'A' is the mid-plane represented by the `GENERAL_MIDPLANE` in #2. The corresponding AP 203 `ADVANCED_FACE` entities with persistent IDs is also shown below.

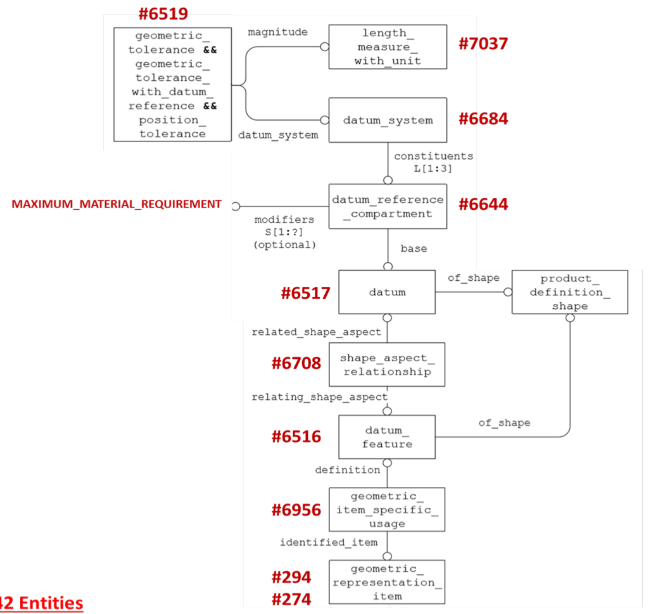


CTF Entities
```
#258=T_POSITION(#4, (FI ,0.076, MMC)), PD(#2, MMC));
#4=SLOT('CTP(FACE17&FACE19)_part1', (88.7643, 5.06365, 10.2133), [-1, 1.77575e-016, 3.737e-
016], 3.302, 12.7, 25.4, [0, -1, 3.33067e-016]);
#2=GENERAL_MIDPLANE('(FACE1&FACE21)_part1', (68.6725, 5.06365, 10.2133), [1.358e-016, 3.33067e-
016, 1], 9.525, 66.675, 66.04, [0, 1, 0]);
```
STEP AP203 Entities
```
#290=ADVANCED_FACE('17',(#543,#544,#545,#546),#547,.F.);
#292=ADVANCED_FACE('19',(#550,#551,#552,#553),#554,.F.);
#274=ADVANCED_FACE('1',(#501,#502,#503,#504),#505,.F.);
#294=ADVANCED_FACE('21',(#557,#558,#559,#560),#561,.T.);
AddDatum('A',274);              //returns #6517
AddFaceToCalloutAdd (6517,294); // Add second parallel face to Datum A
CalloutCenterOfSymmetry(290);        // Create a  derived entity; center of symmetry for the slot
AddTolerancePositionWithFlags(290,0.76,'A',MMC);
```

Figure 34: CTF File with Datums and Modifiers

Position tolerance along with Datum and Modifier is attached to the STEP topological entities using the C++ method as described above. The output in the AP 242 format is shown in Figure 35. #6519 represents the `POSITION_TOLERANCE` entity in the AP 242 file along with the value in LENGTH_MEASURE_WITH_UNIT entity in #7037. The SLOT target feature is mapped to the SHAPE_ASPECT in

#6513 which points to the ADVANCED_FACE entities in #290 and #292 represented by the target faces. Finally, the GENERAL_MIDPLANE datum feature is mapped to the DATUM_FEATURE in #6516 which points to the ADVANCED_FACE entities in #294 and #274 represented by the datum faces.



STEP AP 242 Entities

```
#6519=(
GEOMETRIC_TOLERANCE('','',#7037,#6513)
GEOMETRIC_TOLERANCE_WITH_DATUM_REFERENCE((#6684))
GEOMETRIC_TOLERANCE_WITH_MODIFIERS((.MAXIMUM_MATERIAL_REQUIREMENT.))
POSITION_TOLERANCE()
);
#7037=LENGTH_MEASURE_WITH_UNIT(LENGTH_MEASURE(0.076),#7035);
```

Tolerance Frame with Material modifiers

```
#6513=SHAPE_ASPECT('','',#60,.F.);
#6953=GEOMETRIC_ITEM_SPECIFIC_USAGE('','',#6513,$,#290);
#6958=GEOMETRIC_ITEM_SPECIFIC_USAGE('','',#6513,$,#292);
#290=ADVANCED_FACE('17',(#543,#544,#545,#546),#547,.F.);
#292=ADVANCED_FACE('19',(#550,#551,#552,#553),#554,.F.);
```

Target Feature

```
#6684=DATUM_SYSTEM('',$,#60,.F.,(#6644));
#6644=DATUM_REFERENCE_COMPARTMENT('',$,#60,.F.,#6517,(
SIMPLE_DATUM_REFERENCE_MODIFIER(.MAXIMUM_MATERIAL_REQUIREMENT.)));
#6517=DATUM('','',#60,.F.,'A');
#6708=SHAPE_ASPECT_RELATIONSHIP('','',#6516,#6517);
#6516=DATUM_FEATURE('A','',#60,.F.);
```

Datum Reference Frame With material modifiers

```
#6956=GEOMETRIC_ITEM_SPECIFIC_USAGE('','',#6516,$,#294);
#6955=GEOMETRIC_ITEM_SPECIFIC_USAGE('','',#6516,$,#274);
#294=ADVANCED_FACE('21',(#557,#558,#559,#560),#561,.T.);
#274=ADVANCED_FACE('1',(#501,#502,#503,#504),#505,.F.);
```

Datum Feature

Figure 35: Tolerances with Datum and Modifier in AP 242

Chapter 5

CASE STUDY AND CONFORMANCE CHECKING

As part of the Auto-tolerancing project, this translator has been tested using different 3D assembly models. Three assembly models are presented as test cases for conformance checking and also to ensure readability with other CAD systems. The translated STEP AP 242 files of these models will be opened using the following three packages,

1. **NIST Step File Analyzer v1.72**: The output STEP AP 242 file has been tested using the Lipman and Free 2016 released by NIST for conformance checking produced same results as the input CTF file.
2. **STEP NC Machine v12.12**: There are only few GUIs at present that support viewing STEP AP 242 semantic PMI as graphical annotations. STEP NC machine is a GUI provided by Step Tools Inc. which displays the part tolerances in a tree view and also highlights the features.
3. **MBDVidia v3.5.160525**: MBDVista developed by Capvidia is another CAD viewer that can read semantic PMI from STEP AP 242 file and generate annotations of GD&T. It supports creation and modification of tolerance frames and exports the final data to other formats such as QIF and XML.

The first test case is an assembly of a Cam Follower that was provided by RECON services as shown in Figure 36.

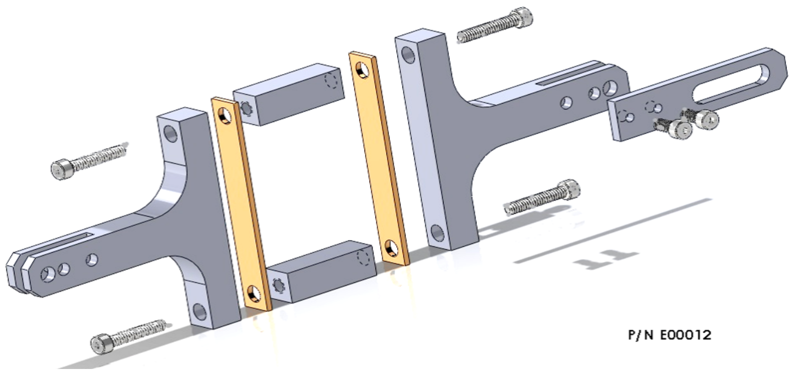
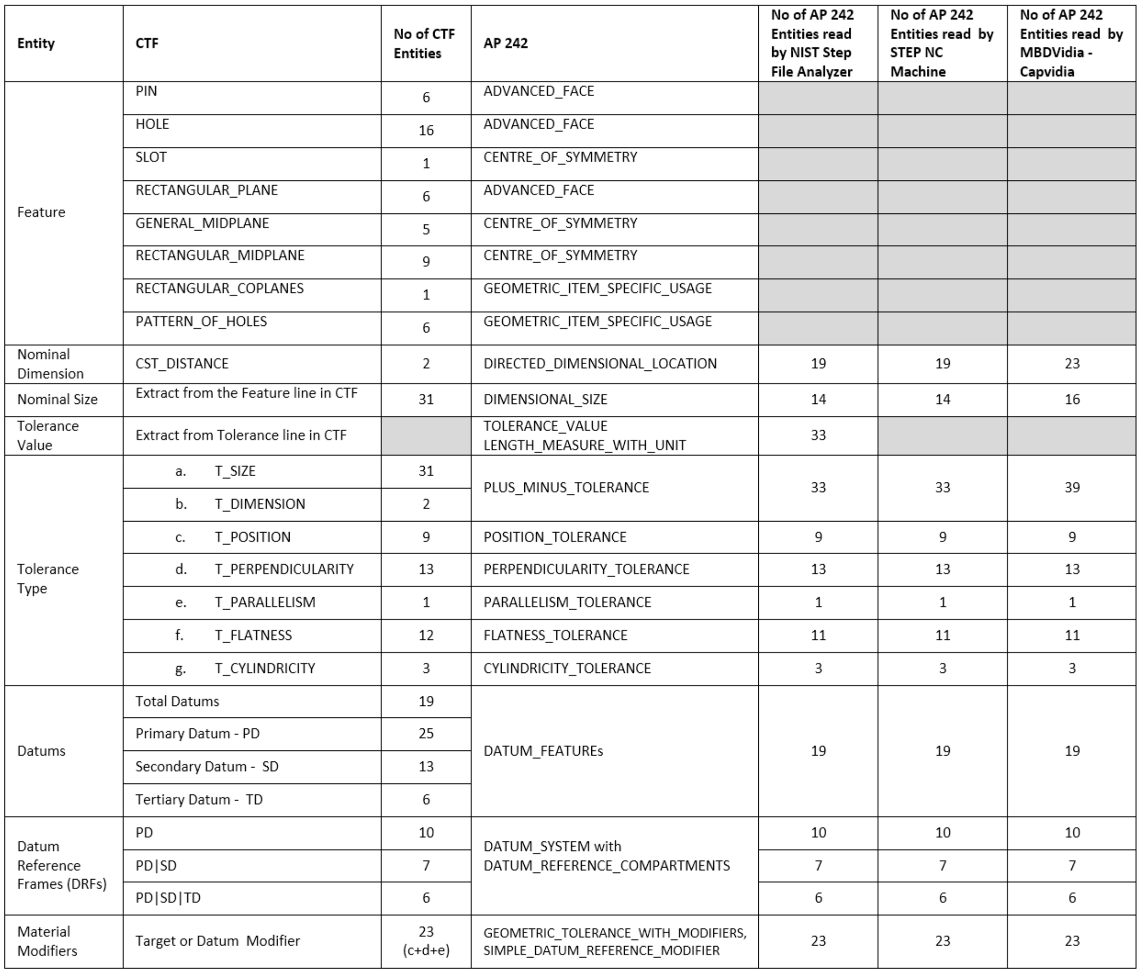
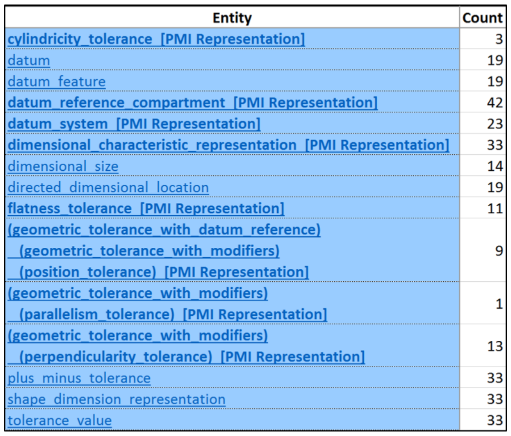


Figure 36: Cam Follower Assembly from RECON services



Entity	CTF	No of CTF Entities	AP 242	No of AP 242 Entities read by NIST Step File Analyzer	No of AP 242 Entities read by STEP NC Machine	No of AP 242 Entities read by MBDVidia - Capvidia
Feature	PIN	6	ADVANCED_FACE			
	HOLE	16	ADVANCED_FACE			
	SLOT	1	CENTRE_OF_SYMMETRY			
	RECTANGULAR_PLANE	6	ADVANCED_FACE			
	GENERAL_MIDPLANE	5	CENTRE_OF_SYMMETRY			
	RECTANGULAR_MIDPLANE	9	CENTRE_OF_SYMMETRY			
	RECTANGULAR_COPLANES	1	GEOMETRIC_ITEM_SPECIFIC_USAGE			
	PATTERN_OF_HOLES	6	GEOMETRIC_ITEM_SPECIFIC_USAGE			
Nominal Dimension	CST_DISTANCE	2	DIRECTED_DIMENSIONAL_LOCATION	19	19	23
Nominal Size	Extract from the Feature line in CTF	31	DIMENSIONAL_SIZE	14	14	16
Tolerance Value	Extract from Tolerance line in CTF		TOLERANCE_VALUE LENGTH_MEASURE_WITH_UNIT	33		
Tolerance Type	a. T_SIZE	31	PLUS_MINUS_TOLERANCE	33	33	39
	b. T_DIMENSION	2				
	c. T_POSITION	9	POSITION_TOLERANCE	9	9	9
	d. T_PERPENDICULARITY	13	PERPENDICULARITY_TOLERANCE	13	13	13
	e. T_PARALLELISM	1	PARALLELISM_TOLERANCE	1	1	1
	f. T_FLATNESS	12	FLATNESS_TOLERANCE	11	11	11
	g. T_CYLINDRICITY	3	CYLINDRICITY_TOLERANCE	3	3	3
Datums	Total Datums	19	DATUM_FEATUREs	19	19	19
	Primary Datum - PD	25				
	Secondary Datum - SD	13				
	Tertiary Datum - TD	6				
Datum Reference Frames (DRFs)	PD	10	DATUM_SYSTEM with DATUM_REFERENCE_COMPARTMENTS	10	10	10
	PD|SD	7		7	7	7
	PD|SD|TD	6		6	6	6
Material Modifiers	Target or Datum Modifier	23 (c+d+e)	GEOMETRIC_TOLERANCE_WITH_MODIFIERS, SIMPLE_DATUM_REFERENCE_MODIFIER	23	23	23

Figure 37: Conformance Checking for Cam Follower Assembly



Entity	Count
cylindricity_tolerance [PMI Representation]	3
datum	19
datum_feature	19
datum_reference_compartment [PMI Representation]	42
datum_system [PMI Representation]	23
dimensional_characteristic_representation [PMI Representation]	33
dimensional_size	14
directed_dimensional_location	19
flatness_tolerance [PMI Representation]	11
(geometric_tolerance_with_datum_reference) (geometric_tolerance_with_modifiers) (position_tolerance) [PMI Representation]	9
(geometric_tolerance_with_modifiers) (parallelism_tolerance) [PMI Representation]	1
(geometric_tolerance_with_modifiers) (perpendicularity_tolerance) [PMI Representation]	13
plus_minus_tolerance	33
shape_dimension_representation	33
tolerance_value	33

Figure 38: PMI summary NIST Step File Analyzer for Cam Follower Assembly

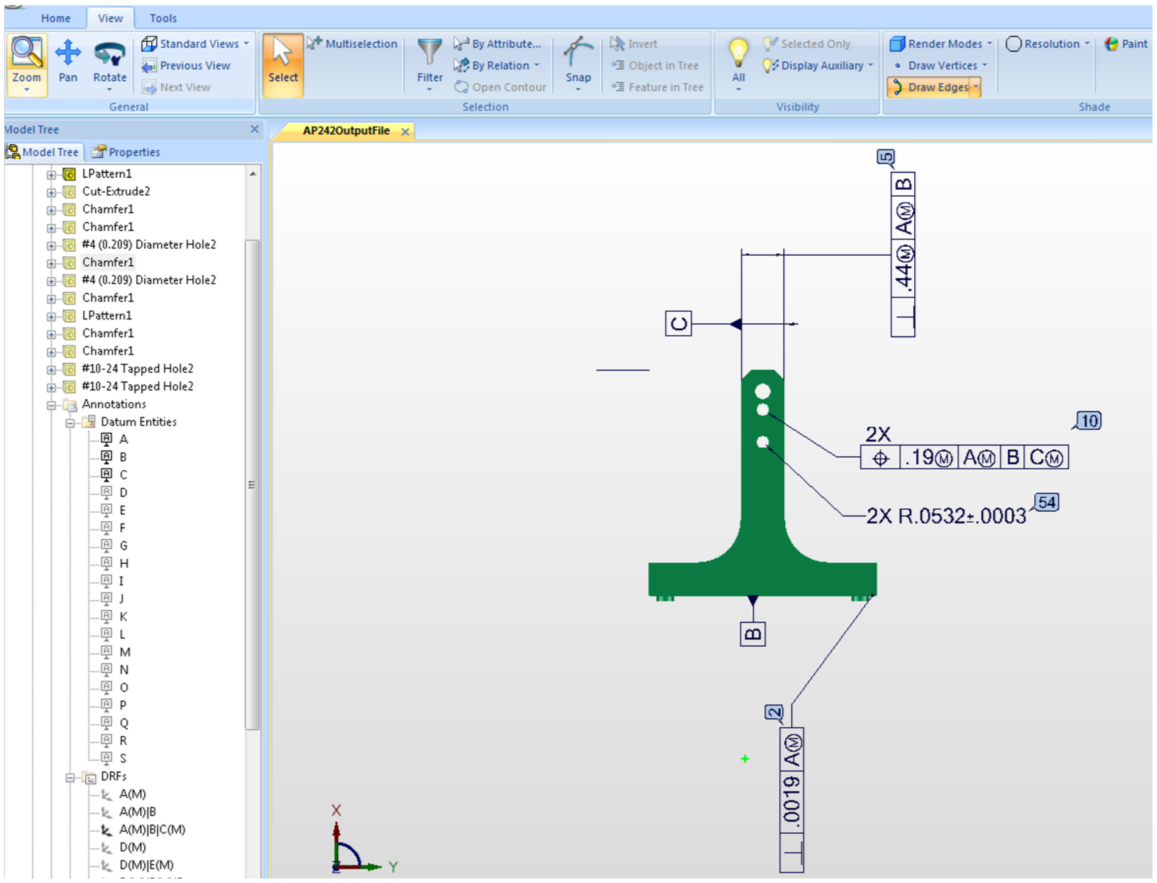
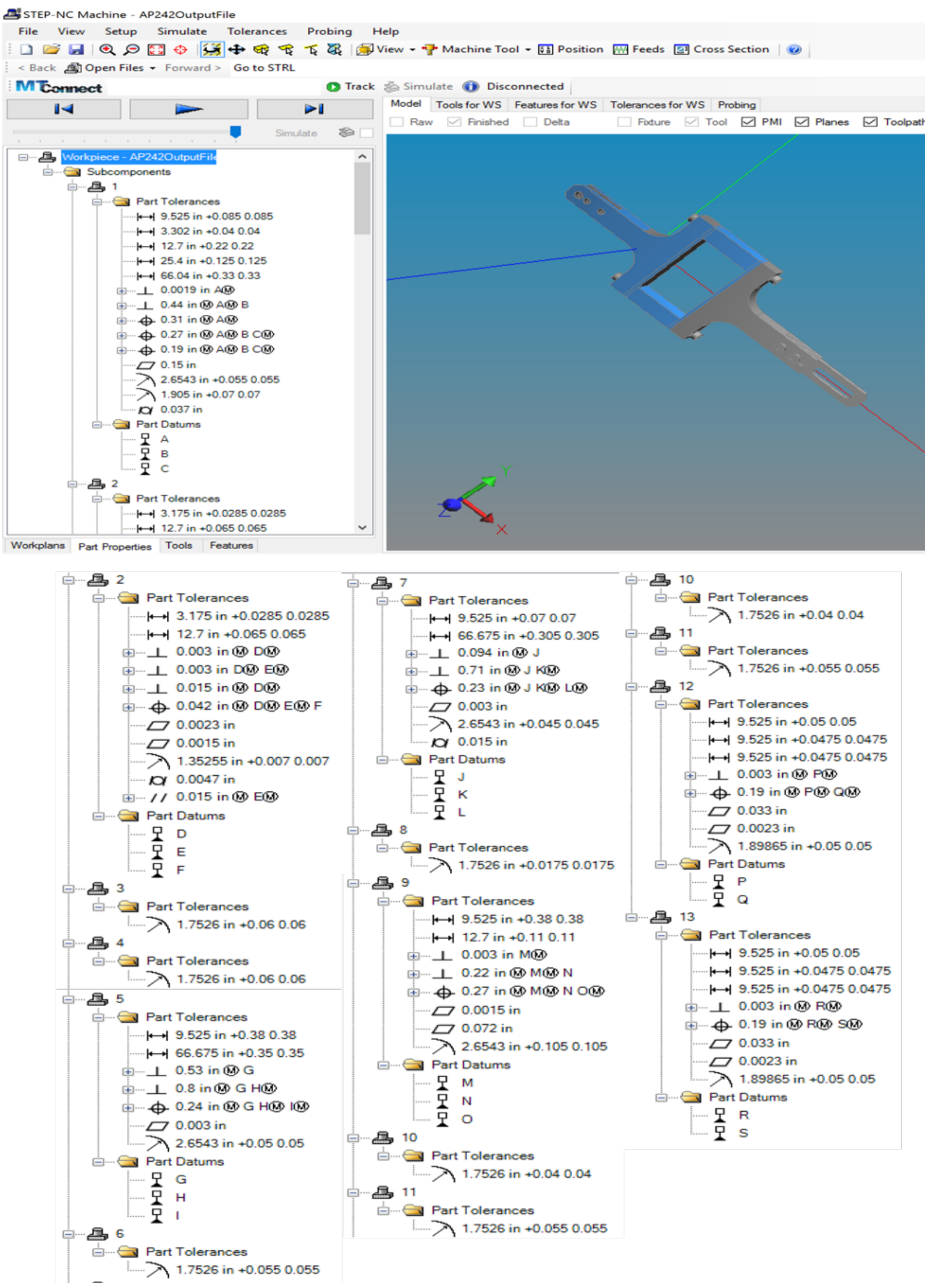


Figure 39: MBDVidia GD&T shown as annotations for Cam Follower component


STEP-NC Machine - AP242OutputFile
File View Setup Simulate Tolerances Probing Help
View Machine Tool Position Feeds Cross Section
< Back Open Files Forward > Go to STRL
MTConnect
Track Simulate Disconnected
Simulate
Model Tools for WS Features for WS Tolerances for WS Probing
Raw Finished Delta Fixture Tool PMI Planes Toolpath
Workplans Part Properties Tools Features
Y
Z
X

```
Workpiece - AP242OutputFile
  Subcomponents
    1
      Part Tolerances
        |←→| 9.525 in +0.085 0.085
        |←→| 3.302 in +0.04 0.04
        |←→| 12.7 in +0.22 0.22
        |←→| 25.4 in +0.125 0.125
        |←→| 66.04 in +0.33 0.33
        ⊥ 0.0019 in AⓂ
        ⊥ 0.44 in Ⓜ AⓂ B
        ⌖ 0.31 in Ⓜ AⓂ
        ⌖ 0.27 in Ⓜ AⓂ B CⓂ
        ⌖ 0.19 in Ⓜ AⓂ B CⓂ
        ▱ 0.15 in
        ⌒ 2.6543 in +0.055 0.055
        ⌒ 1.905 in +0.07 0.07
        ⌭ 0.037 in
      Part Datums
        A
        B
        C
    2
      Part Tolerances
        |←→| 3.175 in +0.0285 0.0285
        |←→| 12.7 in +0.065 0.065
```

```
2
  Part Tolerances
    |←→| 3.175 in +0.0285 0.0285
    |←→| 12.7 in +0.065 0.065
    ⊥ 0.003 in Ⓜ DⓂ
    ⊥ 0.003 in DⓂ EⓂ
    ⊥ 0.015 in Ⓜ DⓂ
    ⌖ 0.042 in Ⓜ DⓂ EⓂ F
    ▱ 0.0023 in
    ▱ 0.0015 in
    ⌒ 1.35255 in +0.007 0.007
    ⌭ 0.0047 in
    // 0.015 in Ⓜ EⓂ
  Part Datums
    D
    E
    F
3
  Part Tolerances
    ⌒ 1.7526 in +0.06 0.06
4
  Part Tolerances
    ⌒ 1.7526 in +0.06 0.06
5
  Part Tolerances
    |←→| 9.525 in +0.38 0.38
    |←→| 66.675 in +0.35 0.35
    ⊥ 0.53 in Ⓜ G
    ⊥ 0.8 in Ⓜ G HⓂ
    ⌖ 0.24 in Ⓜ G HⓂ IⓂ
    ▱ 0.003 in
    ⌒ 2.6543 in +0.05 0.05
  Part Datums
    G
    H
    I
6
  Part Tolerances
    ⌒ 1.7526 in +0.055 0.055
```

```
7
  Part Tolerances
    |←→| 9.525 in +0.07 0.07
    |←→| 66.675 in +0.305 0.305
    ⊥ 0.094 in Ⓜ J
    ⊥ 0.71 in Ⓜ J KⓂ
    ⌖ 0.23 in Ⓜ J KⓂ LⓂ
    ▱ 0.003 in
    ⌒ 2.6543 in +0.045 0.045
    ⌭ 0.015 in
  Part Datums
    J
    K
    L
8
  Part Tolerances
    ⌒ 1.7526 in +0.0175 0.0175
9
  Part Tolerances
    |←→| 9.525 in +0.38 0.38
    |←→| 12.7 in +0.11 0.11
    ⊥ 0.003 in MⓂ
    ⊥ 0.22 in Ⓜ MⓂ N
    ⌖ 0.27 in Ⓜ MⓂ N OⓂ
    ▱ 0.0015 in
    ▱ 0.072 in
    ⌒ 2.6543 in +0.105 0.105
  Part Datums
    M
    N
    O
10
  Part Tolerances
    ⌒ 1.7526 in +0.04 0.04
11
  Part Tolerances
    ⌒ 1.7526 in +0.055 0.055
```

```
10
  Part Tolerances
    ⌒ 1.7526 in +0.04 0.04
11
  Part Tolerances
    ⌒ 1.7526 in +0.055 0.055
12
  Part Tolerances
    |←→| 9.525 in +0.05 0.05
    |←→| 9.525 in +0.0475 0.0475
    |←→| 9.525 in +0.0475 0.0475
    ⊥ 0.003 in Ⓜ PⓂ
    ⌖ 0.19 in Ⓜ PⓂ QⓂ
    ▱ 0.033 in
    ▱ 0.0023 in
    ⌒ 1.89865 in +0.05 0.05
  Part Datums
    P
    Q
13
  Part Tolerances
    |←→| 9.525 in +0.05 0.05
    |←→| 9.525 in +0.0475 0.0475
    |←→| 9.525 in +0.0475 0.0475
    ⊥ 0.003 in Ⓜ RⓂ
    ⌖ 0.19 in Ⓜ RⓂ SⓂ
    ▱ 0.033 in
    ▱ 0.0023 in
    ⌒ 1.89865 in +0.05 0.05
  Part Datums
    R
    S
```

Figure 40: STEP-NC Machine GD&T tree view

The second test case is an assembly of a Radio Car Chassis as shown in Figure 41.

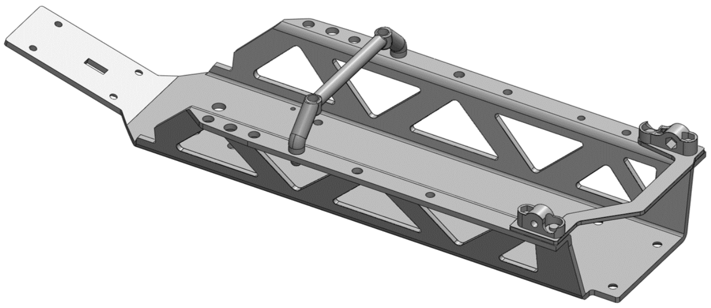
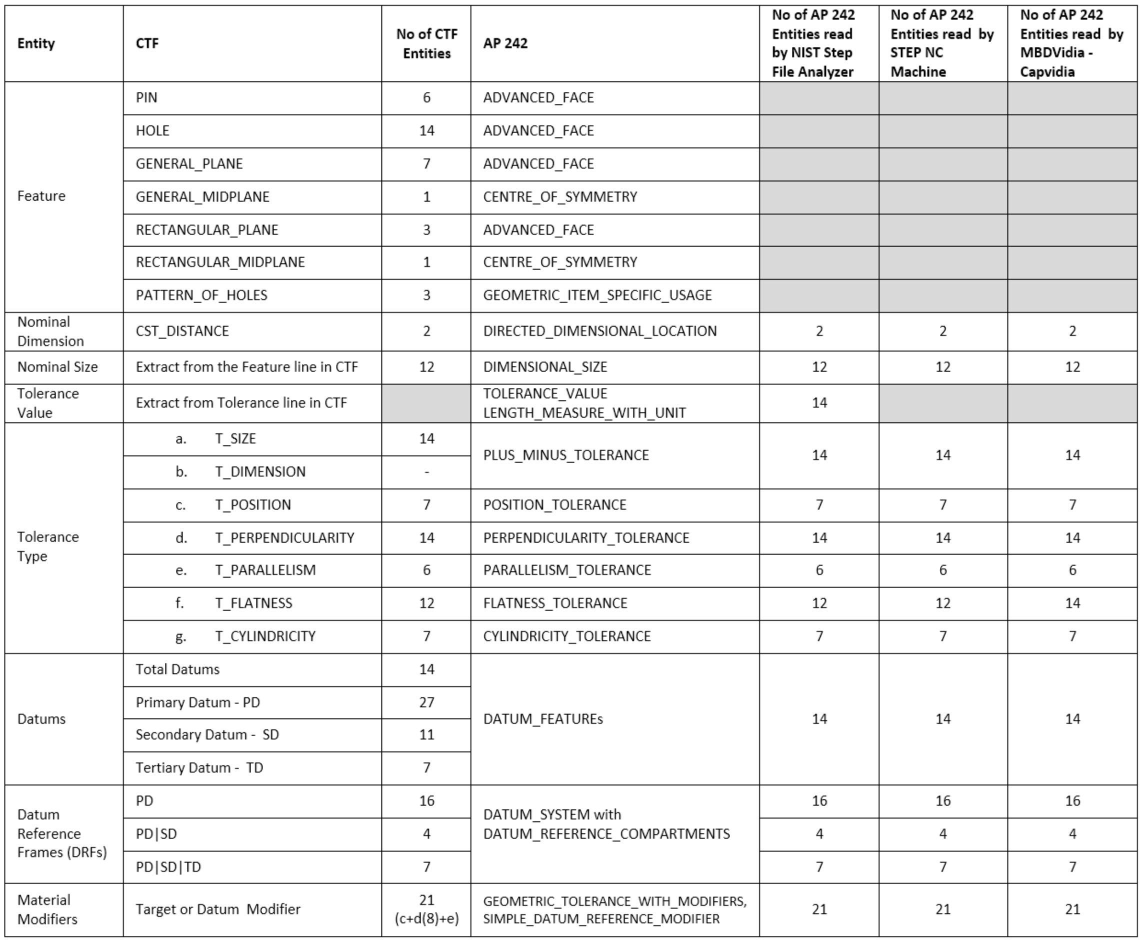
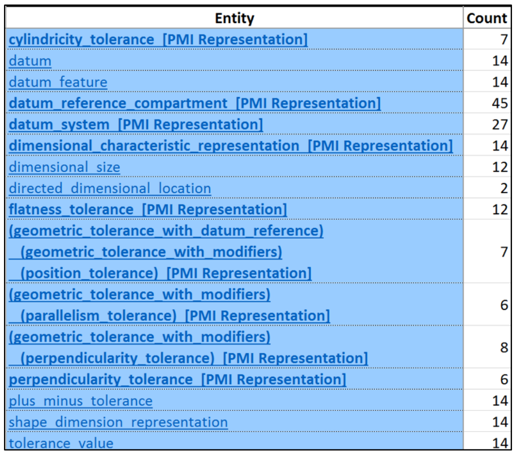


Figure 41: Radio Car Chassis model



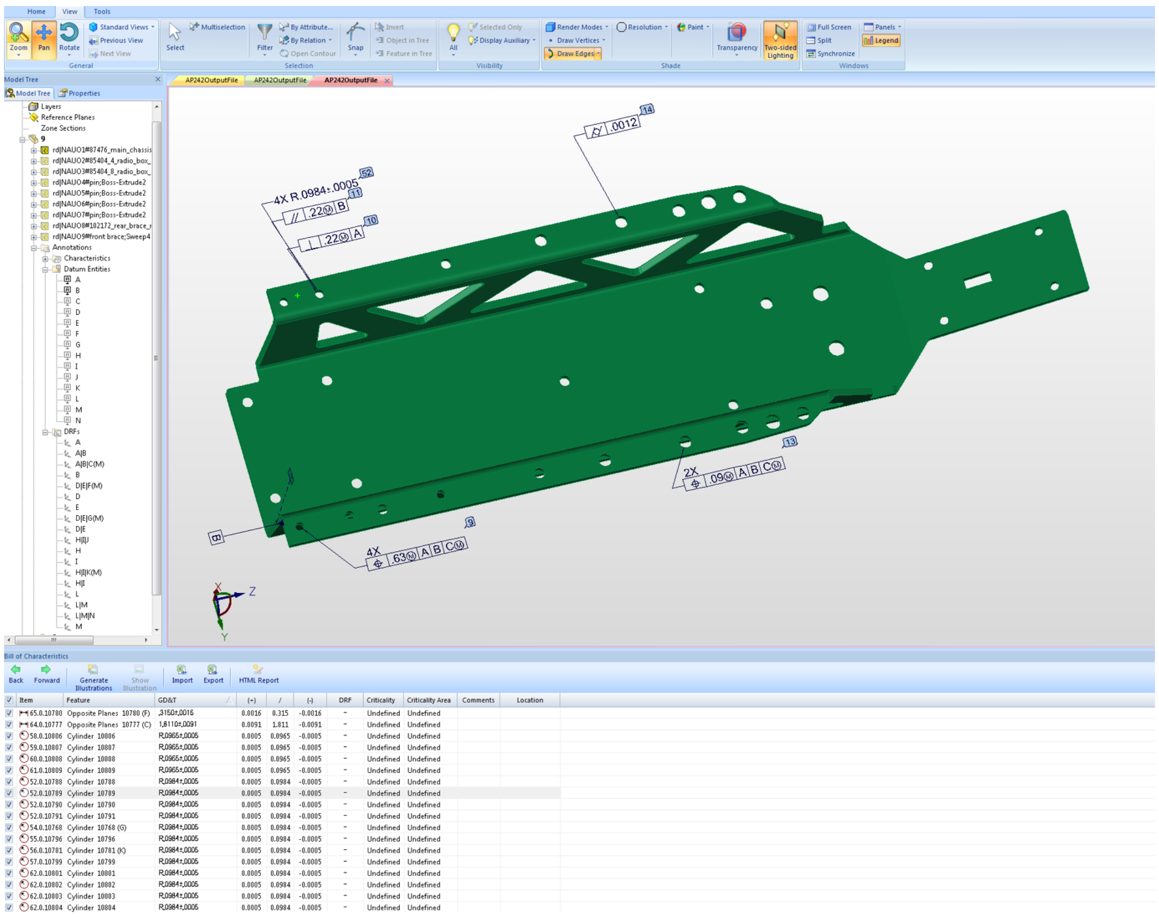
Entity	**CTF**	**No of CTF Entities**	**AP 242**	**No of AP 242 Entities read by NIST Step File Analyzer**	**No of AP 242 Entities read by STEP NC Machine**	**No of AP 242 Entities read by MBDVidia - Capvidia**
Feature	PIN	6	ADVANCED_FACE			
	HOLE	14	ADVANCED_FACE			
	GENERAL_PLANE	7	ADVANCED_FACE			
	GENERAL_MIDPLANE	1	CENTRE_OF_SYMMETRY			
	RECTANGULAR_PLANE	3	ADVANCED_FACE			
	RECTANGULAR_MIDPLANE	1	CENTRE_OF_SYMMETRY			
	PATTERN_OF_HOLES	3	GEOMETRIC_ITEM_SPECIFIC_USAGE			
Nominal Dimension	CST_DISTANCE	2	DIRECTED_DIMENSIONAL_LOCATION	2	2	2
Nominal Size	Extract from the Feature line in CTF	12	DIMENSIONAL_SIZE	12	12	12
Tolerance Value	Extract from Tolerance line in CTF		TOLERANCE_VALUE LENGTH_MEASURE_WITH_UNIT	14		
Tolerance Type	a. T_SIZE	14	PLUS_MINUS_TOLERANCE	14	14	14
	b. T_DIMENSION	-				
	c. T_POSITION	7	POSITION_TOLERANCE	7	7	7
	d. T_PERPENDICULARITY	14	PERPENDICULARITY_TOLERANCE	14	14	14
	e. T_PARALLELISM	6	PARALLELISM_TOLERANCE	6	6	6
	f. T_FLATNESS	12	FLATNESS_TOLERANCE	12	12	14
	g. T_CYLINDRICITY	7	CYLINDRICITY_TOLERANCE	7	7	7
Datums	Total Datums	14	DATUM_FEATUREs	14	14	14
	Primary Datum - PD	27				
	Secondary Datum - SD	11				
	Tertiary Datum - TD	7				
Datum Reference Frames (DRFs)	PD	16	DATUM_SYSTEM with DATUM_REFERENCE_COMPARTMENTS	16	16	16
	PD|SD	4		4	4	4
	PD|SD|TD	7		7	7	7
Material Modifiers	Target or Datum Modifier	21 (c+d(8)+e)	GEOMETRIC_TOLERANCE_WITH_MODIFIERS, SIMPLE_DATUM_REFERENCE_MODIFIER	21	21	21

Figure 42: Conformance Checking for Radio Car Chassis Assembly



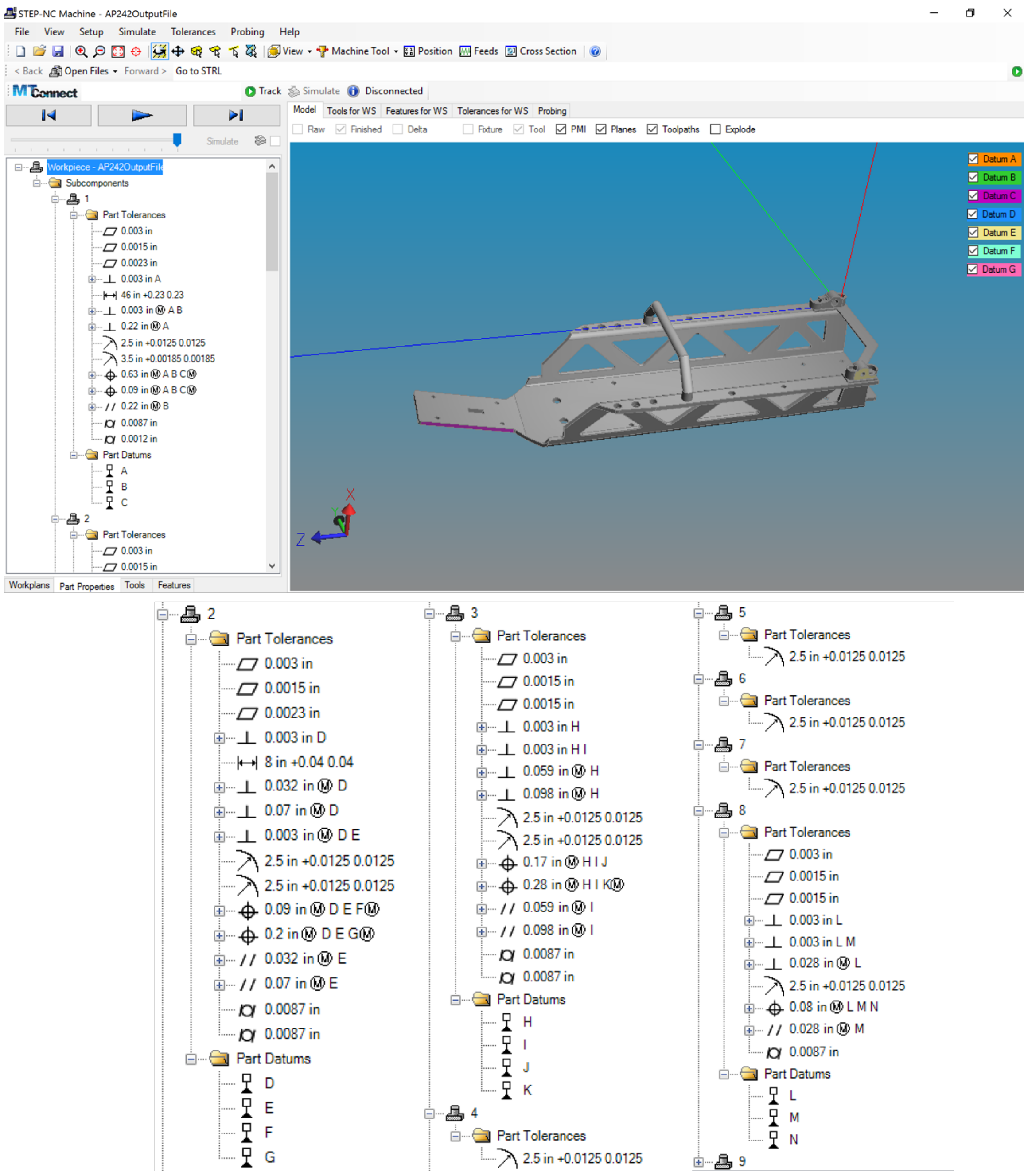
Entity	Count
cylindricity_tolerance [PMI Representation]	7
datum	14
datum_feature	14
datum_reference_compartment [PMI Representation]	45
datum_system [PMI Representation]	27
dimensional_characteristic_representation [PMI Representation]	14
dimensional_size	12
directed_dimensional_location	2
flatness_tolerance [PMI Representation]	12
(geometric_tolerance_with_datum_reference) (geometric_tolerance_with_modifiers) (position_tolerance) [PMI Representation]	7
(geometric_tolerance_with_modifiers) (parallelism_tolerance) [PMI Representation]	6
(geometric_tolerance_with_modifiers) (perpendicularity_tolerance) [PMI Representation]	8
perpendicularity_tolerance [PMI Representation]	6
plus_minus_tolerance	14
shape_dimension_representation	14
tolerance_value	14

Figure 43: PMI summary NIST Step File Analyzer for Radio Car Chassis Assembly



✓	Item	Feature	GD&T	(+)	/	(-)	DRF	Criticality	Criticality Area	Comments	Location
✓	65.0.10780	Opposite Planes 10780 (F)	.3150±.0016	0.0016	0.315	-0.0016	-	Undefined	Undefined		
✓	64.0.10777	Opposite Planes 10777 (C)	1.8110±.0091	0.0091	1.811	-0.0091	-	Undefined	Undefined		
✓	58.0.10806	Cylinder 10806	R.0965±.0005	0.0005	0.0965	-0.0005	-	Undefined	Undefined		
✓	59.0.10807	Cylinder 10807	R.0965±.0005	0.0005	0.0965	-0.0005	-	Undefined	Undefined		
✓	60.0.10808	Cylinder 10808	R.0965±.0005	0.0005	0.0965	-0.0005	-	Undefined	Undefined		
✓	61.0.10809	Cylinder 10809	R.0965±.0005	0.0005	0.0965	-0.0005	-	Undefined	Undefined		
✓	52.0.10788	Cylinder 10788	R.0984±.0005	0.0005	0.0984	-0.0005	-	Undefined	Undefined		
✓	52.0.10789	Cylinder 10789	R.0984±.0005	0.0005	0.0984	-0.0005	-	Undefined	Undefined		
✓	52.0.10790	Cylinder 10790	R.0984±.0005	0.0005	0.0984	-0.0005	-	Undefined	Undefined		
✓	52.0.10791	Cylinder 10791	R.0984±.0005	0.0005	0.0984	-0.0005	-	Undefined	Undefined		
✓	54.0.10768	Cylinder 10768 (G)	R.0984±.0005	0.0005	0.0984	-0.0005	-	Undefined	Undefined		
✓	55.0.10796	Cylinder 10796	R.0984±.0005	0.0005	0.0984	-0.0005	-	Undefined	Undefined		
✓	56.0.10781	Cylinder 10781 (K)	R.0984±.0005	0.0005	0.0984	-0.0005	-	Undefined	Undefined		
✓	57.0.10799	Cylinder 10799	R.0984±.0005	0.0005	0.0984	-0.0005	-	Undefined	Undefined		
✓	62.0.10801	Cylinder 10801	R.0984±.0005	0.0005	0.0984	-0.0005	-	Undefined	Undefined		
✓	62.0.10802	Cylinder 10802	R.0984±.0005	0.0005	0.0984	-0.0005	-	Undefined	Undefined		
✓	62.0.10803	Cylinder 10803	R.0984±.0005	0.0005	0.0984	-0.0005	-	Undefined	Undefined		
✓	62.0.10804	Cylinder 10804	R.0984±.0005	0.0005	0.0984	-0.0005	-	Undefined	Undefined		

Figure 44: MBDVidia GD&T shown as annotations for Radio Car Chassis


STEP-NC Machine - AP242OutputFile
File View Setup Simulate Tolerances Probing Help
View Machine Tool Position Feeds Cross Section
< Back Open Files Forward > Go to STRL
MTConnect
Track Simulate Disconnected
Model Tools for WS Features for WS Tolerances for WS Probing
Raw Finished Delta Fixture Tool PMI Planes Toolpaths Explode
Simulate
Datum A
Datum B
Datum C
Datum D
Datum E
Datum F
Datum G
X
Z
Workplans Part Properties Tools Features

```
Workpiece - AP242OutputFile
  Subcomponents
    1
      Part Tolerances
        ⏥ 0.003 in
        ⏥ 0.0015 in
        ⏥ 0.0023 in
        ⟂ 0.003 in A
        ↔ 46 in +0.23 0.23
        ⟂ 0.003 in Ⓜ A B
        ⟂ 0.22 in Ⓜ A
        ⌒ 2.5 in +0.0125 0.0125
        ⌒ 3.5 in +0.00185 0.00185
        ⌖ 0.63 in Ⓜ A B CⓂ
        ⌖ 0.09 in Ⓜ A B CⓂ
        ∥ 0.22 in Ⓜ B
        ⌭ 0.0087 in
        ⌭ 0.0012 in
      Part Datums
        A
        B
        C
    2
      Part Tolerances
        ⏥ 0.003 in
        ⏥ 0.0015 in
```

```
2
  Part Tolerances
    ⏥ 0.003 in
    ⏥ 0.0015 in
    ⏥ 0.0023 in
    ⟂ 0.003 in D
    ↔ 8 in +0.04 0.04
    ⟂ 0.032 in Ⓜ D
    ⟂ 0.07 in Ⓜ D
    ⟂ 0.003 in Ⓜ D E
    ⌒ 2.5 in +0.0125 0.0125
    ⌒ 2.5 in +0.0125 0.0125
    ⌖ 0.09 in Ⓜ D E FⓂ
    ⌖ 0.2 in Ⓜ D E GⓂ
    ∥ 0.032 in Ⓜ E
    ∥ 0.07 in Ⓜ E
    ⌭ 0.0087 in
    ⌭ 0.0087 in
  Part Datums
    D
    E
    F
    G
```

```
3
  Part Tolerances
    ⏥ 0.003 in
    ⏥ 0.0015 in
    ⏥ 0.0015 in
    ⟂ 0.003 in H
    ⟂ 0.003 in H I
    ⟂ 0.059 in Ⓜ H
    ⟂ 0.098 in Ⓜ H
    ⌒ 2.5 in +0.0125 0.0125
    ⌒ 2.5 in +0.0125 0.0125
    ⌖ 0.17 in Ⓜ H I J
    ⌖ 0.28 in Ⓜ H I KⓂ
    ∥ 0.059 in Ⓜ I
    ∥ 0.098 in Ⓜ I
    ⌭ 0.0087 in
    ⌭ 0.0087 in
  Part Datums
    H
    I
    J
    K
4
  Part Tolerances
    ⌒ 2.5 in +0.0125 0.0125
```

```
5
  Part Tolerances
    ⌒ 2.5 in +0.0125 0.0125
6
  Part Tolerances
    ⌒ 2.5 in +0.0125 0.0125
7
  Part Tolerances
    ⌒ 2.5 in +0.0125 0.0125
8
  Part Tolerances
    ⏥ 0.003 in
    ⏥ 0.0015 in
    ⏥ 0.0015 in
    ⟂ 0.003 in L
    ⟂ 0.003 in L M
    ⟂ 0.028 in Ⓜ L
    ⌒ 2.5 in +0.0125 0.0125
    ⌖ 0.08 in Ⓜ L M N
    ∥ 0.028 in Ⓜ M
    ⌭ 0.0087 in
  Part Datums
    L
    M
    N
9
```

Figure 45: STEP-NC Machine GD&T tree view

The last test case is a Body Cap assembly as shown in Figure 46.

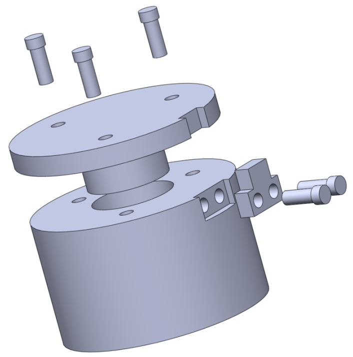
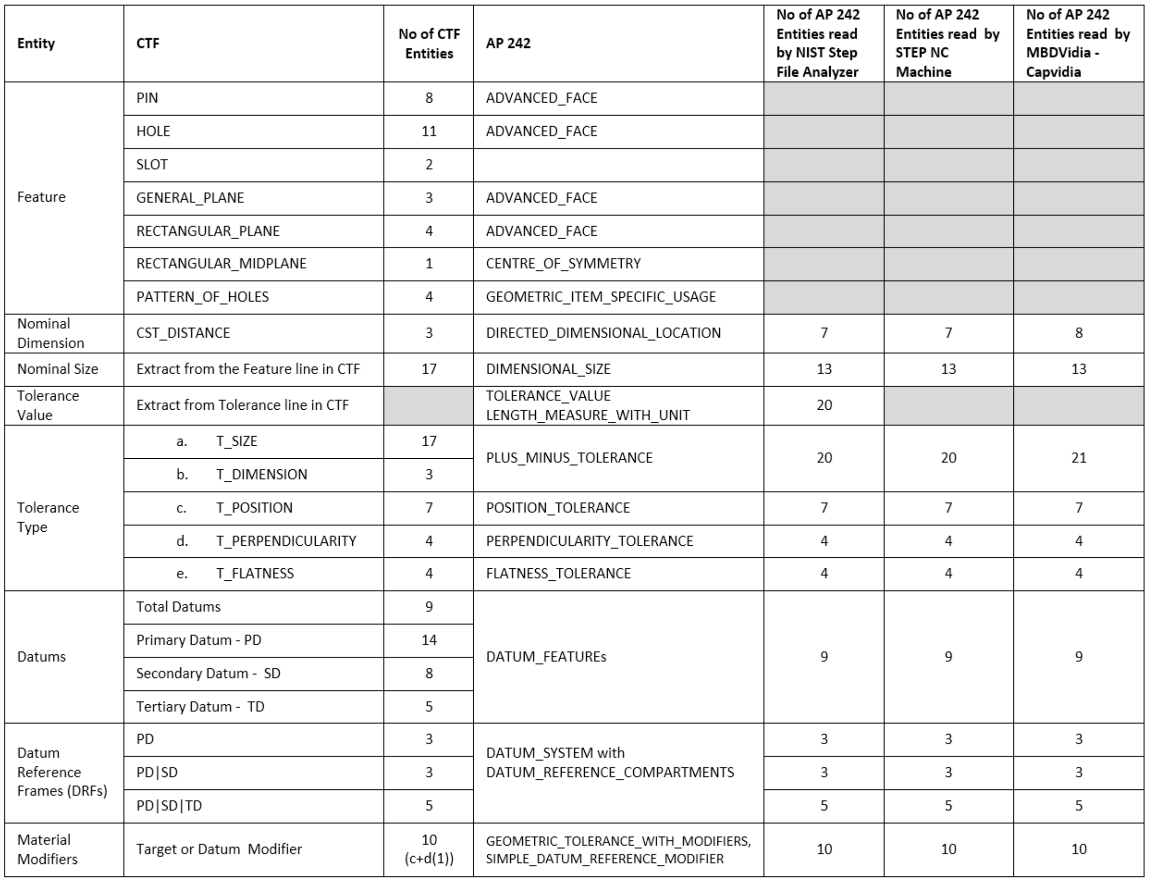
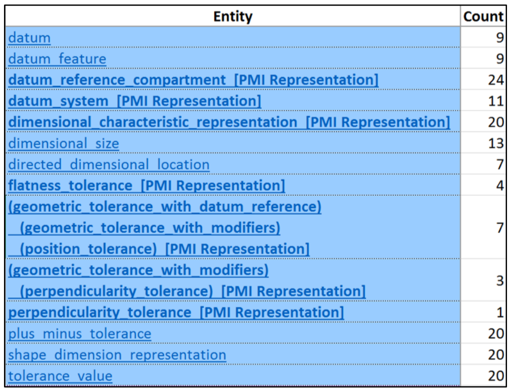


Figure 46: Body Cap Assembly



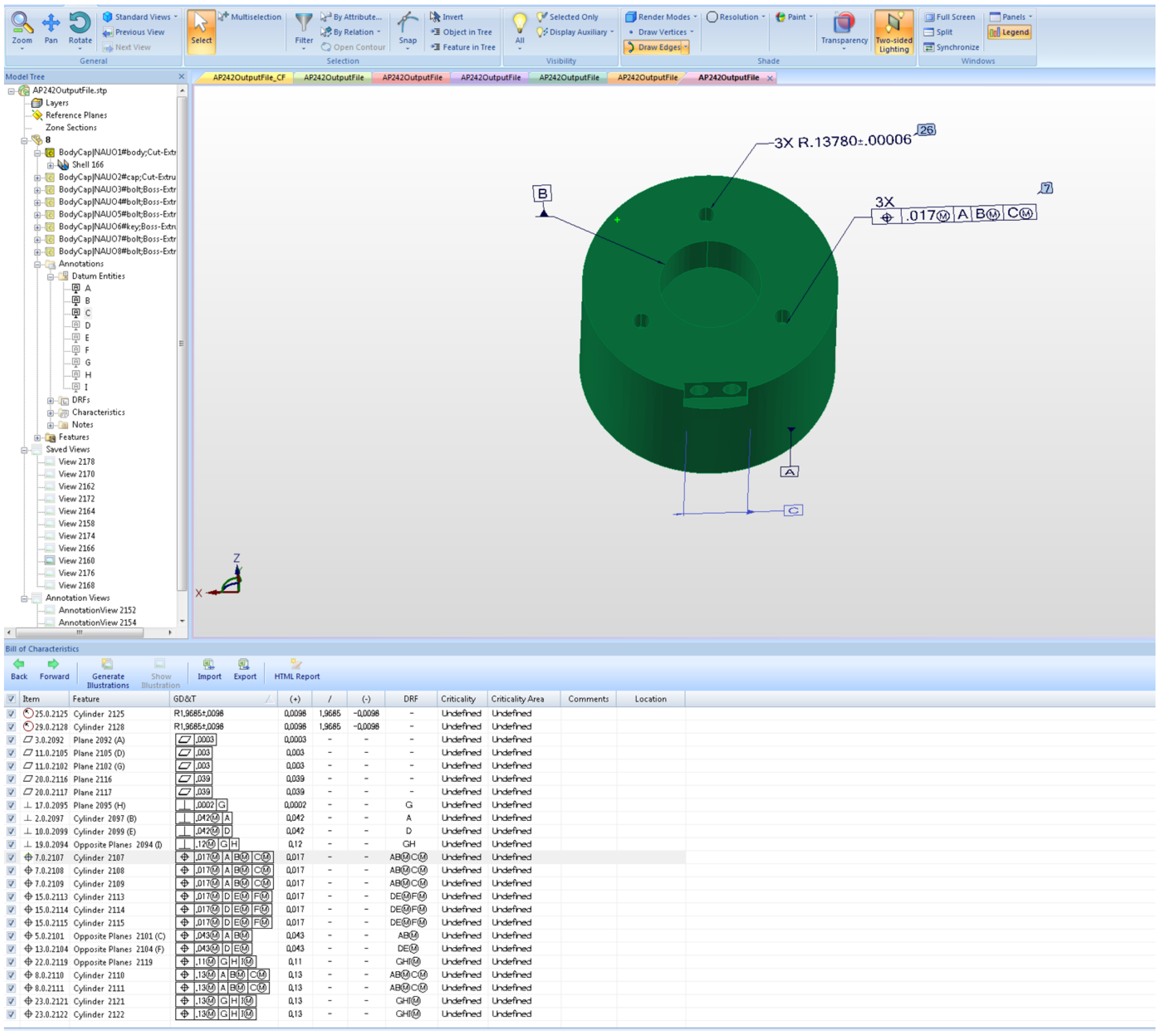
Entity	CTF	No of CTF Entities	AP 242	No of AP 242 Entities read by NIST Step File Analyzer	No of AP 242 Entities read by STEP NC Machine	No of AP 242 Entities read by MBDVidia - Capvidia
Feature	PIN	8	ADVANCED_FACE			
	HOLE	11	ADVANCED_FACE			
	SLOT	2				
	GENERAL_PLANE	3	ADVANCED_FACE			
	RECTANGULAR_PLANE	4	ADVANCED_FACE			
	RECTANGULAR_MIDPLANE	1	CENTRE_OF_SYMMETRY			
	PATTERN_OF_HOLES	4	GEOMETRIC_ITEM_SPECIFIC_USAGE			
Nominal Dimension	CST_DISTANCE	3	DIRECTED_DIMENSIONAL_LOCATION	7	7	8
Nominal Size	Extract from the Feature line in CTF	17	DIMENSIONAL_SIZE	13	13	13
Tolerance Value	Extract from Tolerance line in CTF		TOLERANCE_VALUE LENGTH_MEASURE_WITH_UNIT	20		
Tolerance Type	a. T_SIZE	17	PLUS_MINUS_TOLERANCE	20	20	21
	b. T_DIMENSION	3				
	c. T_POSITION	7	POSITION_TOLERANCE	7	7	7
	d. T_PERPENDICULARITY	4	PERPENDICULARITY_TOLERANCE	4	4	4
	e. T_FLATNESS	4	FLATNESS_TOLERANCE	4	4	4
Datums	Total Datums	9	DATUM_FEATUREs	9	9	9
	Primary Datum - PD	14				
	Secondary Datum - SD	8				
	Tertiary Datum - TD	5				
Datum Reference Frames (DRFs)	PD	3	DATUM_SYSTEM with DATUM_REFERENCE_COMPARTMENTS	3	3	3
	PD|SD	3		3	3	3
	PD|SD|TD	5		5	5	5
Material Modifiers	Target or Datum Modifier	10 (c+d(1))	GEOMETRIC_TOLERANCE_WITH_MODIFIERS, SIMPLE_DATUM_REFERENCE_MODIFIER	10	10	10

Figure 47: Conformance Checking for Body Cap Assembly



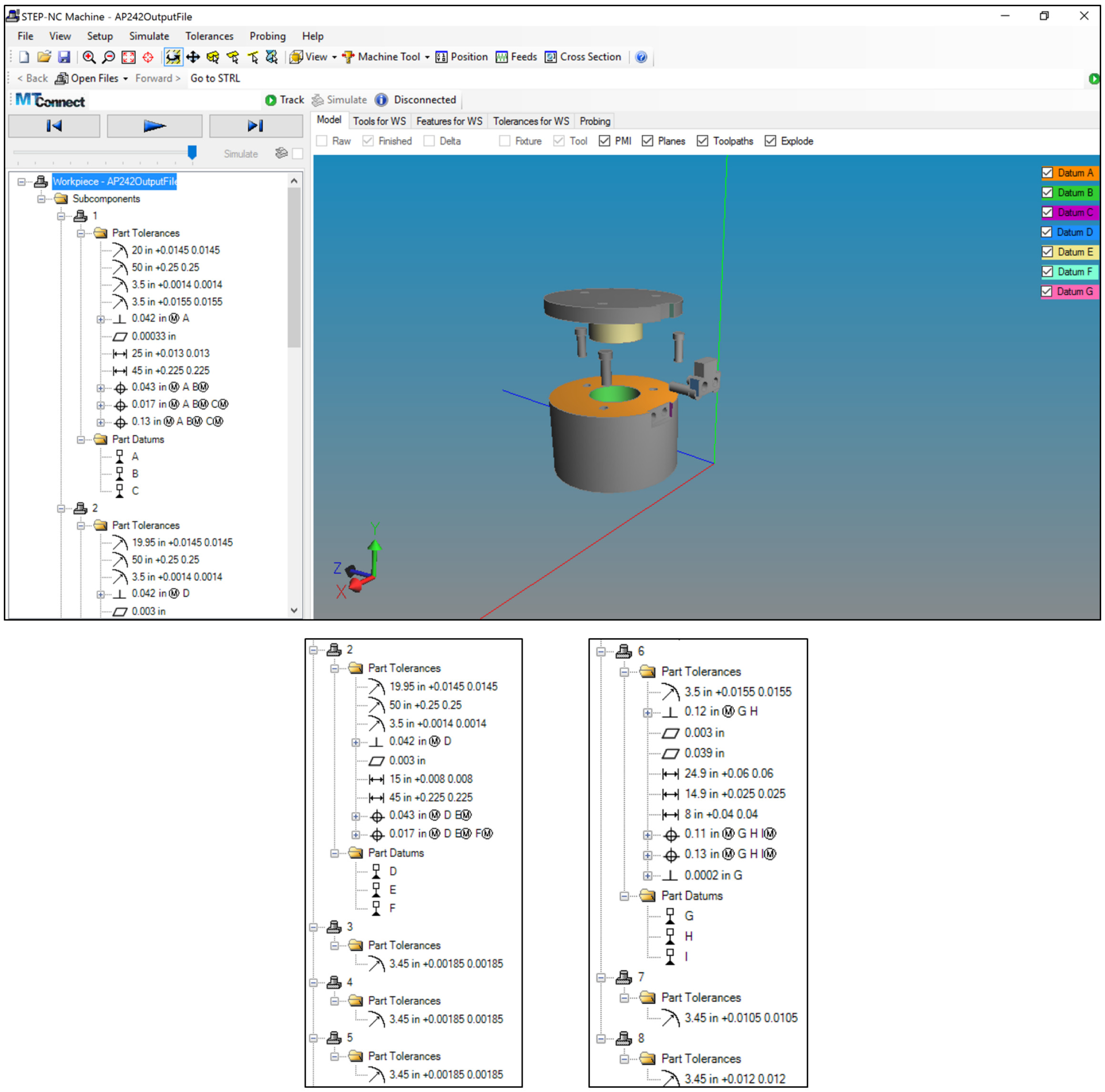
Entity	Count
datum	9
datum_feature	9
datum_reference_compartment [PMI Representation]	24
datum_system [PMI Representation]	11
dimensional_characteristic_representation [PMI Representation]	20
dimensional_size	13
directed_dimensional_location	7
flatness_tolerance [PMI Representation]	4
(geometric_tolerance_with_datum_reference) (geometric_tolerance_with_modifiers) (position_tolerance) [PMI Representation]	7
(geometric_tolerance_with_modifiers) (perpendicularity_tolerance) [PMI Representation]	3
perpendicularity_tolerance [PMI Representation]	1
plus_minus_tolerance	20
shape_dimension_representation	20
tolerance_value	20

Figure 48: PMI summary NIST Step File Analyzer for Body Cap Assembly



Item	Feature	GD&T	(+)	/	(-)	DRF	Criticality	Criticality Area	Comments	Location
25.0.2125	Cylinder 2125	R1,9685±,0098	0,0098	1,9685	-0,0098	-	Undefined	Undefined		
29.0.2128	Cylinder 2128	R1,9685±,0098	0,0098	1,9685	-0,0098	-	Undefined	Undefined		
⏥ 3.0.2092	Plane 2092 (A)	⏥ ,0003	0,0003	-	-	-	Undefined	Undefined		
⏥ 11.0.2105	Plane 2105 (D)	⏥ ,003	0,003	-	-	-	Undefined	Undefined		
⏥ 11.0.2102	Plane 2102 (G)	⏥ ,003	0,003	-	-	-	Undefined	Undefined		
⏥ 20.0.2116	Plane 2116	⏥ ,039	0,039	-	-	-	Undefined	Undefined		
⏥ 20.0.2117	Plane 2117	⏥ ,039	0,039	-	-	-	Undefined	Undefined		
⊥ 17.0.2095	Plane 2095 (H)	⊥ ,0002 G	0,0002	-	-	G	Undefined	Undefined		
⊥ 2.0.2097	Cylinder 2097 (B)	⊥ ,042Ⓜ A	0,042	-	-	A	Undefined	Undefined		
⊥ 10.0.2099	Cylinder 2099 (E)	⊥ ,042Ⓜ D	0,042	-	-	D	Undefined	Undefined		
⊥ 19.0.2094	Opposite Planes 2094 (I)	⊥ ,12Ⓜ G H	0,12	-	-	GH	Undefined	Undefined		
⌖ 7.0.2107	Cylinder 2107	⌖ ,017Ⓜ A BⓂ CⓂ	0,017	-	-	ABⓂCⓂ	Undefined	Undefined		
⌖ 7.0.2108	Cylinder 2108	⌖ ,017Ⓜ A BⓂ CⓂ	0,017	-	-	ABⓂCⓂ	Undefined	Undefined		
⌖ 7.0.2109	Cylinder 2109	⌖ ,017Ⓜ A BⓂ CⓂ	0,017	-	-	ABⓂCⓂ	Undefined	Undefined		
⌖ 15.0.2113	Cylinder 2113	⌖ ,017Ⓜ D EⓂ FⓂ	0,017	-	-	DEⓂFⓂ	Undefined	Undefined		
⌖ 15.0.2114	Cylinder 2114	⌖ ,017Ⓜ D EⓂ FⓂ	0,017	-	-	DEⓂFⓂ	Undefined	Undefined		
⌖ 15.0.2115	Cylinder 2115	⌖ ,017Ⓜ D EⓂ FⓂ	0,017	-	-	DEⓂFⓂ	Undefined	Undefined		
⌖ 5.0.2101	Opposite Planes 2101 (C)	⌖ ,043Ⓜ A BⓂ	0,043	-	-	ABⓂ	Undefined	Undefined		
⌖ 13.0.2104	Opposite Planes 2104 (F)	⌖ ,043Ⓜ D EⓂ	0,043	-	-	DEⓂ	Undefined	Undefined		
⌖ 22.0.2119	Opposite Planes 2119	⌖ ,11Ⓜ G H IⓂ	0,11	-	-	GHIⓂ	Undefined	Undefined		
⌖ 8.0.2110	Cylinder 2110	⌖ ,13Ⓜ A BⓂ CⓂ	0,13	-	-	ABⓂCⓂ	Undefined	Undefined		
⌖ 8.0.2111	Cylinder 2111	⌖ ,13Ⓜ A BⓂ CⓂ	0,13	-	-	ABⓂCⓂ	Undefined	Undefined		
⌖ 23.0.2121	Cylinder 2121	⌖ ,13Ⓜ G H IⓂ	0,13	-	-	GHIⓂ	Undefined	Undefined		
⌖ 23.0.2122	Cylinder 2122	⌖ ,13Ⓜ G H IⓂ	0,13	-	-	GHIⓂ	Undefined	Undefined		

Figure 49: MBDVidia GD&T shown as annotations for Body Cap component



```
Workpiece - AP242OutputFile
    Subcomponents
        1
            Part Tolerances
                20 in +0.0145 0.0145
                50 in +0.25 0.25
                3.5 in +0.0014 0.0014
                3.5 in +0.0155 0.0155
                0.042 in Ⓜ A
                0.00033 in
                25 in +0.013 0.013
                45 in +0.225 0.225
                0.043 in Ⓜ A BⓂ
                0.017 in Ⓜ A BⓂ CⓂ
                0.13 in Ⓜ A BⓂ CⓂ
            Part Datums
                A
                B
                C
        2
            Part Tolerances
                19.95 in +0.0145 0.0145
                50 in +0.25 0.25
                3.5 in +0.0014 0.0014
                0.042 in Ⓜ D
                0.003 in
```

```
2
    Part Tolerances
        19.95 in +0.0145 0.0145
        50 in +0.25 0.25
        3.5 in +0.0014 0.0014
        0.042 in Ⓜ D
        0.003 in
        15 in +0.008 0.008
        45 in +0.225 0.225
        0.043 in Ⓜ D EⓂ
        0.017 in Ⓜ D EⓂ FⓂ
    Part Datums
        D
        E
        F
3
    Part Tolerances
        3.45 in +0.00185 0.00185
4
    Part Tolerances
        3.45 in +0.00185 0.00185
5
    Part Tolerances
        3.45 in +0.00185 0.00185
```

```
6
    Part Tolerances
        3.5 in +0.0155 0.0155
        0.12 in Ⓜ G H
        0.003 in
        0.039 in
        24.9 in +0.06 0.06
        14.9 in +0.025 0.025
        8 in +0.04 0.04
        0.11 in Ⓜ G H IⓂ
        0.13 in Ⓜ G H IⓂ
        0.0002 in G
    Part Datums
        G
        H
        I
7
    Part Tolerances
        3.45 in +0.0105 0.0105
8
    Part Tolerances
        3.45 in +0.012 0.012
```

Figure 50: STEP-NC Machine GD&T tree view

5.1 Summary of Conformance Checking

All three softwares were able to read the test STEP AP 242 files. The following observations were made during the conformance checking process,

1. NIST Step File Analyzer produces output and shows 100% conformance of the output to STEP AP 242.
2. STEP-NC Machine displays all the tolerances in the tree view and also highlights the target and datum features. Features with multiple faces and pattern features are also captured and shows 100% conformance to the AP 242 schema.
3. MBDVidia also displays all the tolerances information in both tree view as well as graphical annotations. Also it is observed that, for features with multiple faces, separate tolerance frames are created for individual faces of the feature in annotations. This information is not redundant, but increases the number of tolerance frames in the final output file. The increase in number of Dimension Location and Dimension Size entities in Figure 35 is due to the above reason.

Chapter 6

DISCUSSION

This thesis describes the implementation of a STEP AP 242 translator and also provides an overview of the essential elements that need to be exchanged using a complete digital product data exchange standard. The current architecture of STEP cannot fully support the exchange of data between heterogeneous CAD systems because the internal representations of most CAD systems vary considerably from the ISO STEP standard. To overcome such difficulties, CAD vendors need to provide seamless support to the industry by implementing STEP Application Protocols. Industries will need to strictly adhere to the use of standard product model and make a cause to continuously improve MBD. Once translators are available, the advantages of migrating to 3D MBD in the field of digital design and manufacturing will definitely start to show up. Semantic representation of design and manufacturing data can enable automated consumption by downstream applications, saving significant time.

6.1 Scope

The scope of implementation of the current translator is as follows,

1. GD&T is limited to the semantic Representation. Presentation or graphical annotation of GD&T is not available in the Part 21 file.
2. Profile and Runout tolerances are not considered in this work.
3. Datum Targets defined in the standard such as point, line, rectangle, circle and

target area which are used for conveying a specific region is not implemented here. All datums are represented by the Datum Feature alone.

4. Tolerance class which is used to specify limits and fits is not part of the available GD&T in CTF file.

6.2 Future Work

In addition to rectifying the above limitations, the following work can be undertaken in future,

1. Test the readability of the AP 242 file with CMM, CATS and CAI software packages.
2. Translate GD&T data from AP 242 to CTF file format for doing various kinds of tolerance analysis. This would prove to be very useful for companies with restriction in sharing geometry information; because CTF file does not require the complete geometry to do tolerance analysis.

REFERENCES

ASME Y14.5: Dimensioning and Tolerancing. 2009. Technical report. New York: The American Society of Mechanical Engineers.

Bettig, Bernie, and J Shah. 2001. “Derivation of a standard set of geometric constraints for parametric modeling and data exchange.” *Computer-Aided Design* 33 (1): 17–33.

CAx-IF Recommended Practices. 2014. http://www.cax-if.de/documents/rec_pracs_pmi_v40.pdf.

Clement, A, A Riviere, and P Serre. 1996. “A declarative information model for functional requirements.” In *Computer-aided Tolerancing,* 3–16. Springer.

Collins, Rockwell. 2016. “Investigating the Impact of Standards-Based Interoperability for Design to Manufacturing and Quality in the Supply Chain.”

Computer Aided Manufacturing-International, inc, R.H. Johnson, and Inc Associates. 1985. *Dimensioning and Tolerancing: Final Report.* CAM-I. https://books.google.com/books?id=dDJqtwAACAAJ.

Desrochers, A, and R Maranzana. 1996. “Constrained dimensioning and tolerancing assistance for mechanisms.” In *Computer-aided Tolerancing,* 17–30. Springer.

Desrochers, Alain, and Alain Rivière. 1997. “A matrix approach to the representation of tolerance zones and clearances.” *The International Journal of Advanced Manufacturing Technology* 13 (9): 630–636.

Feeney, Allison Barnard, Simon P Frechette, and Vijay Srinivasan. 2015. “A portrait of an ISO STEP tolerancing standard as an enabler of smart manufacturing systems.” *Journal of Computing and Information Science in Engineering* 15 (2): 021001.

Gao, Jinsong, Kenneth W Chase, and Spencer P Magleby. 1998. “Generalized 3-D tolerance analysis of mechanical assemblies with small kinematic adjustments.” *IIE transactions* 30 (4): 367–377.

Haghighi, Payam, Prashant Mohan, Nathan Kalish, Prabath Vemulapalli, Jami J Shah, and Joseph K Davidson. 2015. “Toward automatic tolerancing of mechanical assemblies: first-order GD&T schema development and tolerance allocation.” *Journal of Computing and Information Science in Engineering* 15 (4): 041003.

Hardwick, Martin. n.d. "Manufacturing Integration using the STEP-NC DLL."

Hillyard, R, and Ie Braid. 1978. "Analysis of dimensions and tolerances in computer-aided mechanical design." *Computer Aided Design* 10 (3): 161–166.

ISO 1101, Geometrical product specifications (GPS). 2012. Technical report. International Organisation for Standardization.

ISO, SAH. 2006. "STEP Application Handbook." *SCRA, North Charleston, SC.*

Jayaraman, Rangarajan, and Vijay Srinivasan. 1989. "Geometric tolerancing: I. Virtual boundary requirements." *IBM Journal of Research and Development* 33 (2): 90–104.

Kandikjan, Tatjana, Jami J Shah, and Joseph K Davidson. 2001. "A mechanism for validating dimensioning and tolerancing schemes in CAD systems." *Computer-Aided Design* 33 (10): 721–737.

Kemmerer, Sharon J. 1999. *STEP: the grand experience.* US Department of Commerce, Technology Administration, National Institute of Standards / Technology.

Kim, Junhwan, Michael J Pratt, Raj G Iyer, and Ram D Sriram. 2008. "Standardized data exchange of CAD models with design intent." *Computer-Aided Design* 40 (7): 760–777.

Kramer, Thomas R, HM Huang, Elena Messina, Frederick M Proctor, and Harry Scott. 2001. "A feature-based inspection and machining system." *Computer-Aided Design* 33 (9): 653–669.

Krishnan, Krishna K, Osama K Eyada, and Jin B Ong. 1997. "Modeling of manufacturing processes characteristics for automated tolerance analysis." *INTERNATIONAL JOURNAL OF INDUSTRIAL ENGINEERING-APPLICATIONS AND PRACTICE* 4 (3): 187–196.

Lipman, Robert, and STEP Free. 2016. "STEP File Analyzer v1.70."

Maeda, T, and N Tokuoka. 1996. "Toleranced feature modeling by constraint of degree of freedom for assignment of tolerance." In *Computer-aided Tolerancing,* 89–103. Springer.

Medichalam, Madhu S, Jami J Shah, and Roshan D'Souza. 2004. "N-rep: a neutral feature representation to support feature mapping and data exchange across applications." In *ASME 2004 International Design Engineering Technical Conferences* *and Computers and Information in Engineering Conference,* 599–609. American Society of Mechanical Engineers.

Mohan, Prashant, Payam Haghighi, Prabath Vemulapalli, Nathan Kalish, Jami J Shah, and Joseph K Davidson. 2014. “Toward Automatic Tolerancing of Mechanical Assemblies: Assembly Analyses.” *Journal of Computing and Information Science in Engineering* 14 (4): 041009.

Murshed, SM Mahbub, Adam Dixon, and Jami J Shah. 2009. “Neutral definition and recognition of assembly features for legacy systems reverse engineering.” In *ASME 2009 International Design Engineering Technical Conferences and Computers and Information in Engineering Conference,* 615–628. American Society of Mechanical Engineers.

Murshed, SM Mahbub, Jami J Shah, Vadivel Jagasivamani, Ayman Wasfy, and David W Hislop. 2007. “OAM+: An assembly data model for legacy systems engineering.” In *ASME 2007 International Design Engineering Technical Conferences and Computers and Information in Engineering Conference,* 869–881. American Society of Mechanical Engineers.

Ranyak, Paul S, and Richard Fridshal. 1988. “Features for tolerancing a solid model.” In *ASME Computers in Engineering Conference,* 1:262–274.

Requicha, Aristides AG. 1983. “Toward a theory of geometric tolerancing.” *The International Journal of Robotics Research* 2 (4): 45–60.

Rivest, L, C Fortin, and A Desrochers. 1993. “Tolerance modeling for 3D analysis-presenting a kinematic formulation.” In *Proceedings of 3rd CIRP Seminars on Computer Aided Tolerancing,* 27–28.

Roy, Utpal, and Ying-Che Fang. 1996. “Tolerance representation scheme for a three-dimensional product in an object-oriented programming environment.” *IIE transactions* 28 (10): 809–820.

Roy, Utpal, and CR Liu. 1993. “Integrated CAD frameworks: Tolerance representation scheme in a solid model.” *Computers & Industrial Engineering* 24 (3): 495–509.

Sambhoos, Kedar, Bahattin Koc, and Rakesh Nagi. 2009. “Extracting assembly mating graphs for assembly variant design.” *Journal of computing and information science in engineering* 9 (3): 034501.

Shah, Jami J, and David W Miller. 1990. “Structure for supporting geometric tolerances in product definition systems for CIM.” *Manufacturing Review* 3 (1): 23–31.

Shah, Jami J, Yong Yan, and Bing-Chun Zhang. 1998. “Dimension and tolerance modeling and transformations in feature based design and manufacturing.” *Journal of Intelligent Manufacturing* 9 (5): 475–488.

Shen, Zhengshu, Jami J Shah, and Joseph K Davidson. 2008. “Analysis neutral data structure for GD&T.” *Journal of Intelligent Manufacturing* 19 (4): 455–472.

Tsai, Jhy-Cherng, and Mark R Cutkosky. 1997. “Representation and reasoning of geometric tolerances in design.” *Artificial Intelligence for Engineering, Design, Analysis and Manufacturing* 11 (04): 325–341.

Turner, JU. 1993. “A feasibility space approach for automated tolerancing.” *Journal of Engineering for industry* 115 (3): 341–346.

Vemulapalli, Prabath, Prashant Mohan, Jami J Shah, and Joseph K Davidson. 2014. “User Defined Assembly Features and Pattern Recognition From STEP AP203.” In *ASME 2014 International Design Engineering Technical Conferences and Computers and Information in Engineering Conference,* V01AT02A067–V01AT02A067. American Society of Mechanical Engineers.

Venkataraman, Subramanian, Jami J Shah, and Joshua D Summers. 2001. “An investigation of integrating design by features and feature recognition.” In *International Conference FEATS.*

Wu, Yanyan. 2002. *Development of mathematical tools for modeling geometric dimensioning and tolerancing.*

Wu, Yanyan, Jami J Shah, and Joseph K Davidson. 2003. “Computer modeling of geometric variations in mechanical parts and assemblies.” *Journal of Computing and Information Science in Engineering* 3 (1): 54–63.

Zhang, Bing Chun. 1992. “Geometric modeling of dimensioning and tolerancing.” PhD diss., Arizona State University.