Computer-Aided Design and Applications
© 2009 CAD Solutions, LLC
http://www.cadanda.com

Development of Functionality-Based Conformance Classes for ISO 10303 – Conformance Classes for Tolerancing Functionality

Mehmet I. Sarigecili[1], Utpal Roy[2], Sudarsan Rachuri[3] and Ram D. Sriram[4]

[1]Syracuse University, misarige@syr.edu
[2]Syracuse University, uroy@syr.edu
[3]National Institute of Standards and Technology, sudarsan@nist.gov
[4]National Institute of Standards and Technology, sriram@nist.gov

ABSTRACT

ISO 10303 which is STandard for the Exchange of Product model data (STEP) contains a large amount of extremely detailed information. There are no clear and easy instructions for implementing STEP, and the only way to check whether a developed software product actually satisfies STEP requirements is to test it using Conformance Classes (CCs), which are the subsets of STEP's application protocols (APs). However, existing CCs do not give users any idea about the functional capabilities of a software product conforming to STEP. These deficiencies necessitate the development of new CCs based on functionality, to help both developers and users. In this study, to achieve this goal, the functionalities expected for different product lifecycle (PLC) activities are defined at several functionality levels (FLs), using a layered approach. This approach allows the defining of a composite functionality (higher FLs) through the composition of simple, basic functionalities (lower FLs). Information requirements are defined for each functionality, and these information requirements are mapped to STEP entities. These groups of entities for functionalities constitute our functionality-based CCs. In this paper, tolerancing functionality has been studied to develop functionality-based CCs as an example.

Keywords: functionality-based CC, data exchange standards, ISO 10303, STEP.
DOI: 10.3722/cadaps.2009.167-179

1. INTRODUCTION

The basic four phases of a product lifecycle are (i) conception, (ii) design, (iii) manufacturing, and (iv) service and disposal. In these phases, activities that are performed can be grouped as follows:

- Conception: Specification and concept design
- Design: Detailed design (product design and modeling) and analysis
- Manufacturing: Manufacture and inspection
- Service: Usage, maintenance and support, and disposal.

Different tasks (functions) are required for these activities. For example, the product design and modeling activity requires representing a product by its geometry, features, structure, and semantic information. Altogether, these functions define the activity of the product design and modeling. Some of the functions might need, as prerequisites, functions of other activities which have already been completed. In this paper, the functions of activities will be called *functionalities*. They will be discussed in the following sections.

In a product lifecycle, a product moves through the hands of many players, each of whom may require different types of information. There are many different software packages that produce the necessary functions for each product handler's needs, from CAD to CAM, to accounting software, to maintenance scheduling applications. Since many different vendors provide these packages, it is imperative that some means be available to standardize the exchange of information between these software products.

ISO 10303, the Standard for the Exchange of Product model data (STEP) [1], enables such an exchange of product data among the different computer systems used throughout a product's lifecycle. The ISO 10303 is an aggregate of many parts. The Application Protocols (APs) of STEP are its implementable data specifications. Each AP consists of entities which are generic definitions of the terms (e.g., geometric entities, product attributes.) required for defining product-related data. The implementation of any AP in a software product necessitates satisfying the requirements of the Application Protocol's Conformance Classes (CCs). These CCs specify several selected groups of entities (different subsets of the total AP content) that must be completely implemented by the software. Conforming to a specific CC means that the implementation must support all entities grouped within that CC. If a vendor claims that their product conforms to a CC, it has to conform to everything in that conformance class.

The vendors implement STEP standards selectively while they develop their software products. At the end of the development, each software product must conform to all, or to a selected group of, CCs defined in the APs. However, many new data exchange standards are developed every year. As the number increases, the standards themselves become redundant, or they conflict with each other (e.g., AP203 [7] is a subset of AP214 [8]). For this reason, it has been important to make sure that the standards are compatible with each other. In order to do this, consistent conceptual models, terminology, and examples for industrial applications have to be developed. Hence, there has to be a baseline set of requirements applicable to as many standards as possible, to allow the industry to develop products that can be certified through one uniform set of requirements for acceptance in the worldwide market. This harmonization of the standards will increase the profitability, flexibility, and efficiency of a product throughout its lifecycle.

The defining of a common set of requirements can be better initiated with functionality-based CCs. Functionality-based CCs will help to solve one of the main deficiencies of existing CCs, which are not functionality-based. Even when a CAD program conforms to CCs, users can find it difficult to extract essential information from the CAD program's STEP output to carry out any of the functional analyses that may be required. For example, if a user wants to know the possibility of carrying out any tolerance or assembly analysis in a CAD program, they need to map the tolerance analysis requirements onto the entities provided by the CCs first, and then verify whether or not the software conforms to those CCs. This study suggests developing appropriate CCs based on the functionalities for the activities of a product lifecycle that will be expected from the software, so that users will be able to evaluate the software quickly and easily for its usability in their applications. The goal of this study is to evaluate the current status of STEP CCs and to recommend the development of user-friendly, functionality-based CCs, based on the information content of the current CCs, if possible.

In this study, we will report the development of functionality-based CCs for the engineering analysis of tolerancing activity. In future studies, other activities, including manufacturing, inspection, lifecycle data representation and product data management will be addressed.

In order to develop new CCs, a layered approach is proposed in this paper, where the information requirements are defined at different levels. These levels have been termed functionality levels (FLs). Five FLs are defined in this study:

- FL – 0: Generic description-related
- FL – 1: Geometry-related (2D/3D)
- FL – 2: Feature-related
- FL – 3: Product model and structure-related
- FL – 4: Product data semantics-related.

At FL-0, generic description-related, information requirements that define overall activity requirements and provide a consistent representation of functionalities about the activities, should be defined. At

FL-1, geometry-related, information requirements should be defined based on FL-0. In STEP, the geometry information (i.e., point, line, surface and solid) is defined first. Next, at FL-2, the feature-related information is defined, based on the FL-1 geometry information, by referring to the related surfaces (e.g., the definition of form features in AP214). At FL-3, product model and structure-related information are defined, based on the composition of the product in terms of its constituents. Lastly, at FL-4, the semantics of the product data are defined. The activity-specific functionalities at this level should be based on the product data semantics.

In the layered approach to functionality-based CCs, conforming to lower levels satisfies the common requirements for higher level functionalities, i. e., the lower levels are subsets of the higher levels. Hence, the functionalities are described in a hierarchical manner such that, to conform to a higher-level functionality of an activity, all lower level functionalities in that activity have to be satisfied.

In the next section, a short review of current CCs will be given. In section 3, functionality-based CCs for tolerancing will be developed, and information requirements for these tolerancing CCs will be presented. Based on these information requirements, the functionalities and CCs will be defined both for product design and modeling activity and for the engineering analysis for tolerancing activity. At the end of the paper, conclusions and recommendations for future studies are given.

2. STATUS OF CURRENT CONFORMANCE CLASSES IN STEP

The general structure of STEP is given in Fig. 1. In the **description methods** (Fig. 1), a formal language (i. e., EXPRESS [2]), was used in order to represent product data consistently and unambiguously. The language EXPRESS is both human-readable and machine interpretable for developing necessary downstream software [1].

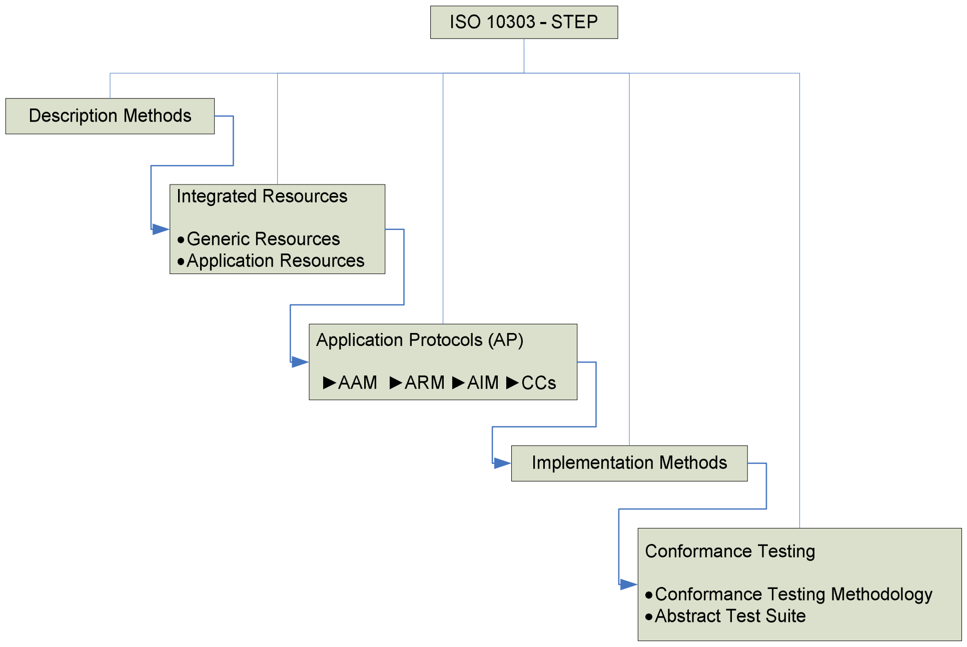

Fig. 1: Structure of ISO 10303 (STEP).

The **integrated resources** (IRs) that are represented in EXPRESS define a conceptual model for the product data (Fig. 1). They define reusable components intended for the development of the APs. They are application- and context-independent [1, 13].

There are two types of integrated resources (IRs) [1, 13]:

- application resources

- generic resources.

Application resources define the product data associated with an application, while generic resources define more general concepts that can be represented for any type of application. Generic resources are part 40 level series of STEP whereas application resources are 100 level series.

Application and generic integrated resources are used in the development of **application protocols** (APs) (Fig. 1). APs are the implementable parts of STEP. They are developed for the context of a specific application. The information requirements and constraints for data exchange in the application context are defined in the APs. The APs consist of four parts [1, 13]: (1) application activity model (AAM), (2) application reference model (ARM), (3) application interpreted model (AIM) and (4) conformance classes (CCs). A brief description of the four parts is given below:

1. The **application activity model** (AAM) is a functional model. It represents the activities of the application for the development of an AP. It also represents the information flow and control mechanisms for the application.
2. The **application reference model** (ARM) represents the information requirements and constraints for the activities given in the AAM.
3. The information requirements and constraints defined in the ARMs have to be mapped to the common STEP language for a computer-interpretable representation and data exchange. The results of this mapping are the **application interpreted models** (AIMs). During the mapping, the requirements are represented by the integrated resource constructs, by applying the necessary application-dependent constraints. New constructs may be added if they are not defined in any IRs. The AIMs are defined in EXPRESS and represented in EXPRESS-G [4] (a graphical form of EXPRESS) diagrams.
4. **Conformance classes** (CCs) are subsets of the AIMs and are defined to address different implementation interests. If a software product conforms to one of the CCs, it has to implement every entity of that CC.

Since the APs are the implementable parts of STEP, the **implementation methods** (Fig. 1) define the mapping procedure of the product data, as suggested by the EXPRESS schema of the APs. Several methods (part 20 level series) are used to implement an AP. For example, part 21 implementation gives a text file of the data. In this text file, instances of the entities defined via the EXPRESS schema are created. However, the file does not contain the schema itself [13].

The procedures for testing the software products that implement and claim conformance to STEP are defined by **conformance testing** (Fig. 1). Two different series of STEP parts (30 level series and 300 level series) constitute conformance testing. The 30 level series defines the conformance testing methodology. The 300 level series defines the abstract test suites (ATSs) for the testing of an AP implementation [1, 13].

From the developer's point of view, it is laborious and difficult to develop a software product which implements STEP. After development, the software product has to go through testing for conformance to STEP. If the testing fails, all the money and time spent on the developing process will be rendered useless. From the user's point of view, this is even more complicated, because the user wants to exchange the data that he/she needs. Unfortunately, the user cannot get any clear information about the capabilities of a piece of software which satisfies the CC requirements, except a list of entities from the conformed-to CCs. The STEP (Part 21 file) file from the software normally contains only the instances of entities from CCs which the software is conforming to. These instances are not self explanatory, and the user may have problems in understanding them. Sometimes the user is also looking for information at a much more detailed level. It is time consuming for the user to identify the necessary detailed information needed for various functionalities. So in this paper, we propose to develop functionality-based CCs that will help both developers and users. This will help developers implement STEP for some common needs. It will also help users select the software product that meets their needs.

3. FUNCTIONALITY-BASED CONFORMANCE CLASSES FOR TOLERANCING

In this section, we discuss the development of functionality-based CCs for tolerancing, which has been identified as one of the sub-activities of engineering analysis (see Tab. 2). In order to define the needs for the appropriate entities required for these particular CCs, one has to understand tolerancing methods in a broader sense. To facilitate this understanding, in sub-section 3.1 we review tolerancing and tolerance analysis. In sub-section 3.2, we then define the information requirements for tolerance analysis, in the context of a few given analysis problems. In the last section, we discuss how to obtain the STEP entities for tolerance-related information and then form a functionality-based CC for tolerancing from the extracted entities.

3.1 Methods for Tolerance Analysis

In the detailed design stage, all of the dimensions that are identified on the product's final drawings are nominal. The allowable variation in these nominal dimensions is described in the form of dimensional and geometric tolerances as well as surface finish. Dimensional tolerances define allowable variations in dimensions (in terms of size, distance, and angles). Geometric tolerances define variations pertaining to different geometric attributes, such as form, orientation and location.

Tolerancing can mean either assigning tolerances (tolerance synthesis) or analyzing the effects of assigned tolerances (tolerance analysis). In tolerance synthesis, the allocation of tolerances between the part dimensions is carried out for a given assembly response function. In tolerance analysis, the part dimensions and tolerances are given, and the aggregate behavior of those tolerances on the assembly response is determined.

There are two basic types of tolerance analysis:

- Worst-case tolerance analysis and
- Statistical tolerance analysis.

The main objective of worst-case tolerance analysis is to verify the functional requirements of an assembly (e.g., a gap between two critical mating surfaces) considering the worst case (i.e., maximum and minimum) conditions for the part/assembly dimensions. It is a deterministic analysis, and guarantees 100% part interchangeability within an assembly. Statistical tolerance analysis, on the other hand, assumes a statistical distribution for each of the dimensions manufactured and checks the distribution of the required assembly functions for different critical dimensions against the allowable tolerance specifications for a successful assembly.

3.2. Information Requirements for Tolerance Analysis

Several sample problems related to stack-up analysis, collected from different sources [10, 11, 12 and 14], have been studied. In a stack-up tolerance analysis, dimensions and tolerances are added to find the variation of an assembly (or part) requirement. These examples deal with dimensional tolerances only; examples with geometrical tolerances will be studied in the future. Due to space limitations, only one example will be discussed here: a stack-up analysis of a fairly complex assembly. Our intention is not to emphasize the solution procedure. Instead, we would like to discuss the issues involved in representing and extracting the required dimensional and tolerancing information from the product database.

The complex bolted assembly [14] for which the stack-up analysis will be carried out is given in Fig. 2. The goal of this example is to analyze the distance between part 5 and part 6, which is represented as a gap in Fig. 2.

Bolts are used to fasten the parts of this assembly (e.g., parts 1 and 2). When a bolt is tightened, the positions of the parts change. Both parts, parts 1 and 2, can shift in opposite directions or in the same direction, affecting the gap between the bolt and the hole. Because of this, assembly shifts are important and they have to be considered. The assembly shift in this case is half of the clearance between the bolt and the hole.

Our objective in this problem is to find out which assembly features, like the bolt and the hole, contribute to the gap calculation and what their critical dimensions are. This is a one-dimensional stack-up analysis problem. We need to determine one, and only one, loop of dimensions containing the gap. If there is more than one loop, the dimensioning system would have to be reviewed for over-dimensioning.

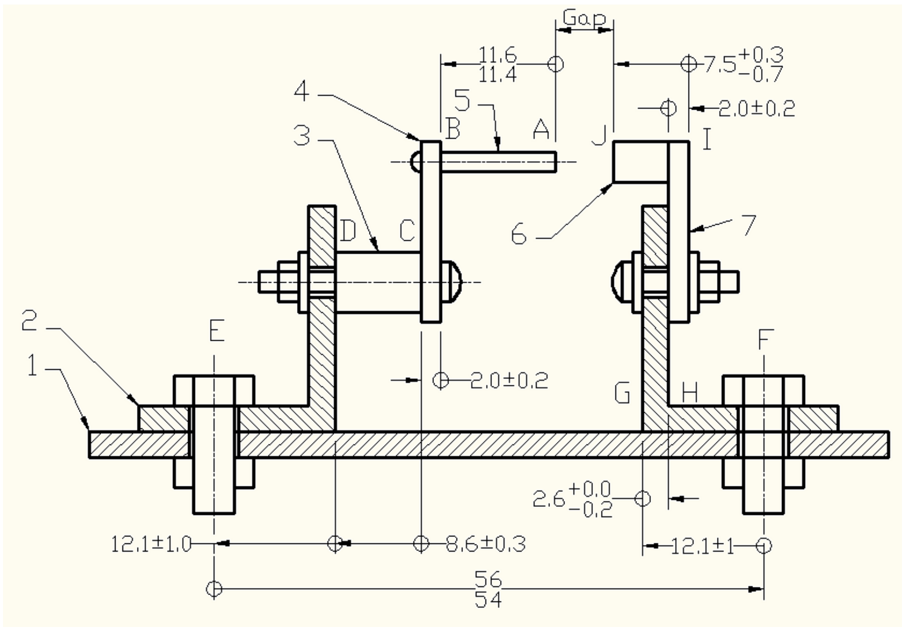

Fig. 2: Tolerance stack-up sketch for a complex assembly [14].

a) Information requirements for carrying out the tolerance analysis
For the analysis, we need the following information.

Geometry and tolerance: For a stack-up analysis, the tolerances have to be in a symmetrical bilateral format. Hence, dimensions and tolerances, converted into a suitable format, are represented by a chain of dimensions (Fig. 2): A→B→C→D→E→F→G→H→I→J→A. Also, assembly shifts of the hole and bolt pairs at E and F have to be defined.

Feature: In order to define assembly shifts at hole and bolt pairs, the hole and bolt features are needed.

Structure: Concentricity conditions and mating relations are required in order to create a chain of dimensions.

b) Analysis
The common procedure for the stack-up analysis is as follows [10, 11, 12 and 14]:

1. The stack-up direction is determined. Here, it is assumed to be positive from left to right.
2. Starting from one end of the gap, the loop is determined for the chain of dimensions.
3. In the loop, the dimensions from left to right are considered positive, while the dimensions from right to left are considered negative. The loop A→B→C→D→E→F→G→H→I→J→A is created.
4. The nominal gap is calculated by subtracting the negative sum from the positive sum, as follows:

Gap = - 11.5 – 2 – 8.6 – 12.1 + 55 – 12.1 + 2.5 + 2 -7.3 = 5.9 (3.1)

5. The worst-case variation of the nominal gap is calculated by adding the tolerances of the dimensions and any assembly shifts:

   Worst case variation of the gap = 0.1 + 0.2 + 0.3 + 1 + 1.3 + 1.3 + 1 + 1.3 + 1.3 + 1 + 0.1 + 0.2 + 0.5 = 9.6 (3.2)

   In Eq.(3.2) all four "1.3" are the assembly shifts due to the clearances between the bolts and holes. The result of worst-case variation, 9.6, is represented in bilateral format as ±9.6 which means the nominal gap can be either 5.9 - 9.6 = - 3.7 (interference) or 5.9 + 9.6 = 15.5.
6. The statistical variation of the nominal gap is calculated by squaring the tolerances, adding them, and taking the square root of the sum:

$$RSS = \sqrt{0.1^2 + 0.2^2 + 0.3^2 + 1^2 + 1.3^2 + 1.3^2 + 1^2 + 1.3^2 + 1.3^2 + 1^2 + 0.1^2 + 0.2^2 + 0.5^2}$$
$$RSS = \sqrt{10.2} = \pm 3.19 \tag{3.3}$$

The statistical variation result, ±3.19, indicates that the nominal gap can be either 5.9 – 3.19 = 2.71 or 5.9 + 3.19 = 9.09.

c) How to get the information from STEP
In STEP, dimensions and tolerances are defined by Part 47 [6]. Tolerances and dimensions are represented for shapes via the `shape_aspect` entity in Part 47. Therefore, we need to represent the shape to be able to do tolerance analysis. This means functionalities representing the shape of the product have to be conformed to before representing tolerances. The shape information of a product is defined in product design and modeling activity. In the next section, the functionalities for product design and modeling will be explained first.

3.3. Functionalities for Product Design and Modeling

The functionalities for product design and modeling are given in Tab. 1. The functionality "generic product description" (FL-0) gathers information to specify resource constructs that provide consistent representation of facts about products in different application-specific views. Information requirements for this level can be found in Part 41 [3], which defines product description and support. In this respect, the schemas in Part 41 will be employed as sub-functionalities of the generic product description functionality. The sub-functionalities are defined below:

- Application context definition
- Product definition
- Product property definition
- Product property representation.

The entities defined in these schemas in Part 41 are the conformance classes at FL-0.

The "geometry representation" functionality (FL-1) can be defined by Part 42 [4] which defines the geometric and topological representation. The schemas defined in Part 42 are sub-functionalities for the geometry representation functionality and include geometry definition, topology definition and geometric model representation. However, the geometric model representation sub-functionalities, like Boundary representation (B-rep), Constructive Solid Geometry (CSG) and/or wireframe, have to be defined by grouping the necessary entities.

The feature representation functionality (FL-2) defines form features which are determined based on the geometry of the product. The feature definitions in AP214 [8] may be used for feature representation functionality (FL-2). These definitions provide the capability to categorize an area of interest in the product's shape representation as a feature, and to associate parameters as additional information.

Functionality Levels	**Design and Manufacturing Activities**
	Product Design and Modeling
Functionality Level 0: Generic description-related	Generic product description
Functionality Level 1: Geometry-related (2D/3D)	Geometry representation
Functionality Level 2: Feature-related	Feature representation
Functionality Level 3: Product model and structure-related	Structure representation
Functionality Level 4: Product data semantics-related	Semantics representation

Tab. 1: The functionalities for product design and modeling activity.

The structure representation functionality (FL-3) defines a product in terms of its composition as a set of constituents or consumed products. The structure is defined after the geometry and feature information have been defined. The product structure schema in Part 44 [5], which defines product structure configuration, will be used for this functionality.

The semantic representation functionality at FL-4 is defined after all four levels (FL-0 through FL-3) are conformed. In addition to the first four levels of product design and modeling information, the semantics representation captures the design intent consisting of construction history, parameters, features and constraints etc.

3.4. Functionalities for the Tolerancing

The example given in section 3.2 shows us the necessary information requirements needed for the stack-up tolerance analysis. (Only the dimensional tolerances are defined in these examples.) These information requirements include both the product model information and the tolerancing information. We have to regroup the information contents, forming new groups that are organized by their relevance to defining functionalities.

The product design and modeling information requirements are as follows:

- Geometry data
- Feature data
- Product structure data.

Tolerancing requirements are as follows:

- Dimension data
- Tolerance data
- Objective
- Assumptions
- Process capability index.

For these requirements, the functionalities for tolerancing (under the engineering analysis activity) in Tab. 2 are defined at five levels, delineated in Tab. 2.

At FL-0, the shape aspect definition is used to describe the spatial characteristics of a shape. Functionality for dimension representation is defined at FL-1, based on the information requirements of FL-0. Also, we need the geometry data for which the dimensions are represented. This necessitates the retrieval of the information regarding the shape aspect definition and dimension representation from levels FL-0 and FL-1. It also requires the information regarding the generic product description and geometry representation from levels FL-0 and FL-1, regarding the product design and modeling activity. The tolerance representation functionality (for dimensional or geometric tolerances, or both) is defined at FL-2. To conform to FL-2, all requirements of levels FL-0 and FL-1 need to be satisfied. In this particular case, the requirements of FL-0 (generic product description), FL-1 (geometry representation) and FL-2 (feature representation) in the product design and modeling activity column have to be satisfied as well. These requirements are enough to carry out the tolerance stack-up analysis and synthesis at the part level. The assembly-level tolerance analysis/synthesis is defined at FL-3. The information requirements for this functionality are gathered from all four levels (FL-0 to FL-3). The product design and modeling-related data for tolerance analysis/synthesis will be retrieved from levels FL-0 to FL-3 of the CAD activity column. Finally, the final tolerance design, based on the product's real-life functions and behaviors, is defined at FL-4. The same layering procedure used in other high-level functionalities is applicable to this functionality as well.

Functionality Levels	**Design and Manufacturing Activities**	
	Product Design and Modeling (CAD)	Engineering Analysis / *Tolerancing*
Functionality Level 0: Generic description-related	Generic product description	Shape aspect definition
Functionality Level 1: Geometry-related (2D/3D)	Geometry representation	Dimension representation
Functionality Level 2: Feature-related	Feature representation	Tolerance representation (dimensional and geometric) Part level Tolerance Analysis/Synthesis
Functionality Level 3: Product model and structure related	Structure representation	Assembly level Tolerance Analysis/Synthesis
Functionality Level 4: Product data semantics related	Semantics representation	Final Tolerance Design based on product's real-life functions and behavior.

Tab. 2: Functionalities for CAD and tolerance.

3.5. Conformance Classes Based on Tolerancing Functionalities
In the previous sections, we defined the information requirements and functionalities for tolerancing. In this section, we map these information requirements onto the STEP entities. This will help us define the functionality-based CCs.

The dimension and tolerance representation-related entities can be borrowed from Part 47 [6], which is a generic integrated resource. In this resource, the `shape_aspect_definition` schema provides the definitions for the spatial characteristics of a shape that are required for dimensioning and tolerancing. Representation of the descriptions of location and size dimensions is provided by the `shape_dimension_schema` of Part 47. Also, the `shape_tolerance_schema` provides the constructs for describing tolerances. This schema includes two types of tolerance: plus-minus tolerance and geometrical tolerance.

The entities for the shape aspect definition are extracted from Part 47:

```
datum
datum_feature
datum_target
datum_reference
referenced_modified_datum
composite_shape_aspect
derived_shape_aspect
apex
centre_of_symmetry
geometric_alignment
```

```
geometric_intersection
parallel_offset
perpendicular_to
extension
tangent
shape_aspect_deriving_relationship
symmetric_shape_aspect.
```

The entities needed to represent the dimensions and limit tolerances are obtained from the `shape_dimension` schema of Part 47. For the dimension representation functionality (FL-2), we only need the dimension representation-related entities. These are given below:

```
angular_location
angular_size
dimensional_characteristic_representation
dimensional_location
dimensional_location_with_path
dimensional_size
dimensional_size_with_path
shape_dimension_representation.
```

For the tolerance representation (at FL-3) we have two functionalities: the dimensional and geometric tolerances. The plus-minus dimensional tolerance entities defined in the `Shape_tolerance_schema` from Part 47 are required for the dimension tolerances. They are:

```
limits_and_fits
plus_minus_tolerance
tolerance_value
```

The geometrical tolerance entities defined in the `Shape_tolerance_schema` from Part 47 are given. They are:

```
dimension_related_tolerance_zone_element
geometric_tolerance
geometric_tolerance_relationship
geometric_tolerance_with_datum_reference
geometric_tolerance_with_defined_unit
modified_geometric_tolerance
projected_zone_definition
runout_zone_definition
runout_zone_orientation
runout_zone_orientation_reference_direction
statistical_distribution_for_tolerance
tolerance_with_statistical_distribution
tolerance_zone
tolerance_zone_form
tolerance_zone_definition.
```

The above-mentioned entities will fulfill the requirements for functionality levels FL-0, FL-1, and FL-2. To carry out the stack-up analysis, we need to add information regarding analysis objectives (e.g., the gap to be analyzed) and assumptions, as needed. Then, the worst-case and statistical (RSS) results for the tolerance analysis will be computed, based on the information defined in the previous functionality levels.

The dimensions of the previous example (Fig. 2) are represented by the `dimensional_size` entity of STEP. The tolerances are all assumed to be converted to symmetric bilateral tolerance (± value), and they are represented by the `plus_minus_tolerance` entity of STEP. In order to represent the idea, numbers will be assigned to differentiate among instances of dimension and tolerance. The numbers are assigned such that the first dimension and the tolerance instances in the chain of dimensions loop will have 1 assigned as an identifier (e.g., dimensional_size_1), and the last dimension and tolerance instances will have 9 assigned as an identifier (e.g., dimensional_size_9). Then, the nominal gap can be calculated by following formula (`dimensional_size_#` will be shown here as D_S_#):

$$Gap = -(D\_S\_1) - (D\_S\_2) - (D\_S\_3) - (D\_S\_4) + (D\_S\_5) - (D\_S\_6) + (D\_S\_7) + (D\_S\_8) - (D\_S\_9) \quad (3.4)$$

The worst-case analysis result can be calculated by the following formula (`plus_minus_tolerance_#` will be shown here as P_M_T_i):

$$Worst - Case = \sum_{i=1}^{9} \left| P\_M\_T\_i \right| \quad (3.5)$$

In Eq.(3.5), *P_M_T_i* denotes the symmetric bilateral tolerance (± value). The statistical analysis result (RSS) can be calculated by:

$$RSS = \sqrt{\sum_{i=1}^{9} \left(P\_M\_T\_i \right)^2} \quad (3.6)$$

The definitions of the above entities are given in the related integrated resources. This means that the relationships among entities are also as they were in the integrated resources. For example, the entities and the relationships among them for dimensional tolerance representation are represented as an EXPRESS-G diagram in Fig. 3. As can be seen there, the `plus_minus_tolerance` entity has two attributes: `tolerance_dimension` and `range`. The first one is referred from the `dimensional_characteristic` entity of the `shape_dimension_schema` of Part 47. The second one is a select type. It can be either `tolerance_value` or `limits_and_fits`. `Tolerance_value` has two attributes: `upper_bound` and `lower_bound`. They are both referred from the `measure_with_unit` entity of `measure_schema` of Part 41. `Limits_and_fits` has 4 attributes: `source` is defined by the `text` entity in `support_resource_schema` of Part 41. The other attributes—`grade`, `form_variance` and `zone_variance`—are defined by the `label` entity of the `support_resource_schema` of Part 41.

To implement the proposed idea of the functionality-based conformance classes for tolerancing, we need a software program which implements the STEP schema defined in Tab. 2 (i.e., representing geometry, feature, structure, dimension, tolerance, etc.). The AP203 (2nd edition) and the AP214 are among the APs that define dimensions and tolerances. Unfortunately, no currently available software products implement the dimensioning and tolerancing sections of these APs. The software products that do implement these APs (AP203 and 214), emphasize only geometry representation. This deficiency might be solved through the development of an application programming interface (API). With an API, the dimension and tolerance information can be extracted directly from the original model file created by a commercial software modeling package.

The idea of the proposed functionality-based CCs is similar to the Data EXchange Specifications (DEXs) developed by the OASIS/PLC TC [15] in identifying a subset of the STEP standards. The DEXs have been developed to support the usage of AP239 – Product Lifecycle Support (PLCS) [9] which is an application protocol in STEP. A DEX is similar in purpose to a STEP AP Conformance Class, but supported by additional usage guidance to ensure consistent implementation of the underlying EXPRESS information model. This similarity will be investigated further in future to improve the present work.

4. CONCLUSION

As the number of redundant and conflicting standards increases, the harmonization of standards becomes more important. The first step toward harmonization is defining a common set of requirements applicable to as many standards as possible. This will allow industry to develop products that can be certified through one uniform set of requirements for acceptance in the worldwide market. This will eliminate interoperability problems and improve collaboration during product lifecycle management.

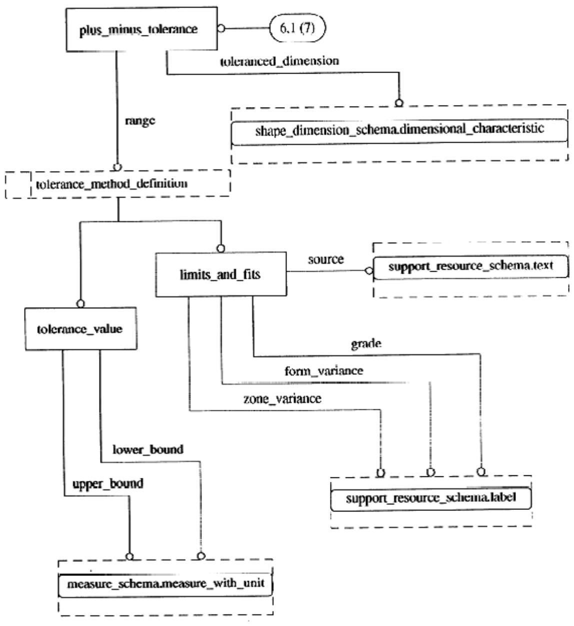
plus_minus_tolerance
6,1 (7)
toleranced_dimension
range
shape_dimension_schema.dimensional_characteristic
tolerance_method_definition
limits_and_fits
source
support_resource_schema.text
tolerance_value
grade
form_variance
zone_variance
lower_bound
upper_bound
support_resource_schema.label
measure_schema.measure_with_unit

Fig. 3: Dimensional tolerance representation in EXPRESS-G diagram (adapted from the Shape_tolerance_schema of Part 47 [6]).

The exchange of product data between different computer systems used throughout a product's lifecycle is enabled mostly by ISO 10303 – STEP. In order to implement STEP successfully, the conformance classes (CCs) defined in application protocols (APs) have to be conformed to. However, the current CCs of STEP are not developed to directly represent the capabilities of software products. The CCs only provide a list of entities that have to be implemented by a software product to claim conformance. This list is not helpful for users trying to select a software product that meets their needs. Also, it is difficult for software developers to implement the current CCs, because they are too broad and complicated. There are no instructions for implementing these CCs.

In this study, we proposed new functionality-based CCs to initiate the harmonization of STEP standards and to solve problems that developers and users have encountered with the existing CCs. In this proposed methodology, first the functionalities expected in a product development are identified. Then, they are assigned to different levels such that the higher level functionalities cover the lower level ones (the layered approach). Next, the information requirements needed to achieve the functionalities are defined. Finally, the information requirements are mapped onto the STEP entities. These entities, which are grouped for different functionalities, represent the functionality-based CCs.

The next step would be to investigate the functionality-based CCs for all activities in the product lifecycle. Also, it would be better if functionality-based CCs were improved, such that some common and some specific entities were defined at each FL for each activity. For example, an engineering analysis activity might have a finite element analysis, an assembly analysis, and a tolerance analysis.

For these three applications, the common entities of the engineering analysis and the application-specific entities should be characterized. This issue and others could be studied in the future.

However, this outline of functionality levels is a starting point, and we recognize that these exact levels might not be suitable for all product lifecycle phases. Further investigation is required for the baseline requirements applicable to different standards, so that concrete, common sets of requirements will be achieved. Also, the DEX concept of the OASIS TC will be investigated in order to improve this study.

5. ACKNOWLEDGEMENT
We would like to thank Joshua Lubell and Sharon Kemmerer of NIST for their valuable comments.

6. DISCLAIMER
No approval or endorsement of any commercial product by the National Institute of Standards and Technology or by Syracuse University is intended or implied.

7. REFERENCES

[1] ISO 10303-1: 1994, Product data representation and exchange -- Part 1: Overview and fundamental principles.
[2] ISO 10303-11: 1994, Product data representation and exchange -- Part 11: Description methods: The EXPRESS language reference.
[3] ISO 10303-41: 1994, Product data representation and exchange -- Part 41: Integrated Generic Resources: Fundamentals of product description and support.
[4] ISO 10303-42: 1994, Product data representation and exchange -- Part 44: Integrated Generic Resources: Geometric and topological representation.
[5] ISO 10303-44: 1994, Product data representation and exchange -- Part 44: Integrated Generic Resources: Product structure configuration.
[6] ISO 10303-47: 1994, Product data representation and exchange -- Part 47: Integrated Generic Resources: Shape Variation Tolerances.
[7] ISO 10303-203: 1994, Product Data Representation and exchange - AP 203: Configuration controlled 3D design of mechanical parts and assemblies.
[8] ISO 10303-214: 2003, Industrial automation systems and integration -- Product data representation and exchange -- Part 214: Application protocol: Core data for automotive mechanical design processes.
[9] ISO 10303-239: 2005, Product Data Representation and exchange - AP 239: Product life cycle support. International Organization for Standardization (ISO), Geneva, Switzerland.
[10] Drake, P. J.: Dimensioning and Tolerancing Handbook, McGraw-Hill, 1999.
[11] Fischer, B. R.: Mechanical Tolerance Stackup and Analysis, CRC Press, 2004.
[12] Geng, H.: Manufacturing Engineering Handbook, McGraw-Hill, 2004.
[13] Kemmerer, S. (Editor): STEP: The Grand Experience, NIST Special Publication 939, National Institute of Standards and Technology, Gaithersburg, MD 20899, USA, 1999, http://www.nist.gov/msidlibrary/doc/stepbook.pdf
[14] Zhang, H.-C.: Advanced Tolerancing Techniques, John Willey & Sons, Inc, 1997.
[15] OASIS TC, http://www.oasis-open.org/ 2008.