NIST Advanced Manufacturing Series
NIST AMS 300-12

Research Results and Recommendations for Universally Unique Identifiers in Product Data Standards

Thomas R. Thurman
Asa G. Trainer
Allison Barnard Feeney
Rosemary Astheimer
Martin Hardwick
Mikael Hedlind

This publication is available free of charge from:
https://doi.org/10.6028/NIST.AMS.300-12

NATIONAL INSTITUTE OF STANDARDS AND TECHNOLOGY
U.S. DEPARTMENT OF COMMERCE

NIST Advanced Manufacturing Series
NIST AMS 300-12

Research Results and Recommendations for Universally Unique Identifiers in Product Data Standards

Thomas R. Thurman
TRThurman Consulting
Marion, Iowa

Asa G. Trainer
Trainer Engineering Associates
Dighton, MA

Allison Barnard Feeney
Rosemary Astheimer
Smart Connected Systems Division
Communications Technology Laboratory

Mikael Hedlind
Sandvik Coromant
Stockholm, Sweden

Martin Hardwick
STEP Tools, Inc
Troy, New York

This publication is available free of charge from:
https://doi.org/10.6028/NIST.AMS.300-12
July 2024

U.S. Department of Commerce
Gina M. Raimondo, Secretary

National Institute of Standards and Technology
Laurie E. Locascio, NIST Director and Under Secretary of Commerce for Standards and Technology

Certain equipment, instruments, software, or materials, commercial or non-commercial, are identified in this paper in order to specify the experimental procedure adequately. Such identification does not imply recommendation or endorsement of any product or service by NIST, nor does it imply that the materials or equipment identified are necessarily the best available for the purpose.

NIST Technical Series Policies
Copyright, Use, and Licensing Statements
NIST Technical Series Publication Identifier Syntax

Publication History
Approved by the NIST Editorial Review Board on 2024-07-05

How to cite this NIST Technical Series Publication:
Thurman TR, Trainer AG, Barnard Feeney A, Hardwick M, Hedlind M, Astheimer, R (2024) Research Results and Recommendations for Universally Unique Identifiers in Product Data Standards. (National Institute of Standards and Technology, Gaithersburg, MD), NIST AMS 300-12. https://doi.org/10.6028/NIST.AMS.300-12

Author ORCID iDs
Thomas R. Thurman: 0000-0002-1550-9913
Asa G. Trainer: 0009-0009-7060-7493
Allison Barnard Feeney: 0000-0002-0866-9572
Rosemary Astheimer: 0009-0001-0534-5054

Abstract

In the model-based enterprise (MBE) paradigm, enterprises are fueled by the digital thread, an authoritative, integrated information flow that connects all phases of the product life cycle. The digital thread enables use of data-driven processes to build knowledge, make decisions, manage requirements, and control manufacturing execution. As information about a product moves through the supply chain, different systems consume or modify that information for a variety of reasons. Associating persistent and universally unique identifiers, in combination with human-readable identifiers, to key engineering requirements enables the enterprise to track all information related to that requirement over the life of the product. Neither widely-used product data standards nor commercial engineering software adequately support universally unique identifiers (UUIDs). This is a major roadblock to realizing the promises of the digital thread and model-based enterprise. This paper presents current support for UUIDs in digital thread standards, use cases and requirements for UUIDs in the product life cycle, research results, and finally, makes recommendations for use of universal identifiers in commonly used product data standards.

Keywords

computer-aided design (CAD), computer-aided inspection (CAI), computer-aided manufacturing (CAM), Digital Thread, Digital Twin, Digital Twin Manufacturing Framework, engineering design, geometrical product specification (GPS), ISO 10303, ISO 23247, product model data standard, persistent identifier, product and manufacturing information (PMI), Quality Information Framework (QIF), STandard for the Exchange of Product data (STEP), universally unique identifier (UUID).

Table of Contents

List of Tables

List of Figures

Acknowledgments

The writers would like to acknowledge Ian Parent, Jan De Nijs, Larry Maggiano, John V. Van Horn, Roland Considine, and Robert Lipman for detailed discussions of requirements, models, and approaches. The authors acknowledge Ed Paff for implementation support. The CAx-Interoperability Forum and the AP242 Product and Manufacturing Information Working Groups are also acknowledged for their contributions. Lastly, the authors would like to acknowledge Charles Manion, David Wollman, Thomas R. Kramer, William Sobel, and Benjamin Urick for their reviews of the paper.

1. Introduction

In the **model-based enterprise** (**MBE**) paradigm, enterprises are fueled by the digital thread, an authoritative, integrated information flow that connects all phases of the product life cycle. Standard data representations play a key role in facilitating the vision of **MBE** due to the role of standards in facilitating interoperability, lowering the cost of solutions, ensuring repeatability of methods, and protecting data assets from obsolescence. The digital thread enables the use of data-driven processes to build knowledge, make decisions, manage requirements, and control processes. As information about a product characteristic moves through the supply chain, different systems may change it for various reasons. Associating **universally unique identifiers** (**UUIDs**) with key engineering requirements enables the enterprise to track all information related to that requirement over the life of the product. Neither widely-used product data standards nor commercial engineering software adequately supports **UUIDs** throughout the digital thread, providing a major roadblock to realizing the model-based enterprise.

The scope of this research is the use of **UUIDs** in product data standards primarily during the design to manufacturing and inspection workflow. The included product data standards are:

- **STandard for the Exchange of Product model data** (**STEP**) [1], officially **International Organization for Standardization** (**ISO**) 10303, Product Data Representation and Exchange, is developed and maintained by the **ISO** Technical Committee 184's Subcommittee 4 on Industrial Data. **STEP** has two primary types of information models, the **STEP** product model is an integration model, comprised of generic model constructs, and domain-specific **application protocols** (**APs**) that are based on the **STEP** product model. The **APs** may be implemented in different software systems. **STEP APs** discussed in this paper include **AP 209**: Multidisciplinary analysis and design [2], **AP 238** Model based integrated manufacturing [3], and **AP 242** Managed model based 3D engineering [4];

- **MTConnect** [5] is developed by the MTConnect Institute, a subsidiary of the Association for Manufacturing Technology, that is an **American National Standards Institute** (**ANSI**)-accredited standards development organization. **MTConnect** is a model-based standard for process execution data collection and classification from manufacturing equipment; and

- **Quality Information Framework** (**QIF**) is developed and maintained by the Digital Metrology Standards Consortium (DMSC) [6], an **ANSI**-accredited standards development organization. Version 3.0 is also published as **ISO** 23952:2020 - Automation systems and integration — Quality information framework (QIF) — An integrated model for manufacturing quality information [7].

Figure 1 illustrates a design-manufacturing-inspection workflow using **Business Process Model and Notation** (**BPMN**) [8]. Each swim lane in the model may have one or more data standards for the artifacts produced and consumed in the lane. In the design engineering swim lane, the **model-based definition** (**MBD**) from the native **computer-aided design** (**CAD**) system is saved as a standard data representation, complete with **product and manufacturing information** (**PMI**) defined in **ISO geometrical product specifications** (**GPS**) series standards. That representation provides sufficient information to plan the manufacturing and inspection processes of this product. Another standard provides the manufacturing process execution instructions. A third provides a stream of data from the manufacturing tools that represents the "as-executed" machine data. The fourth provides the inspection plans and inspection results.

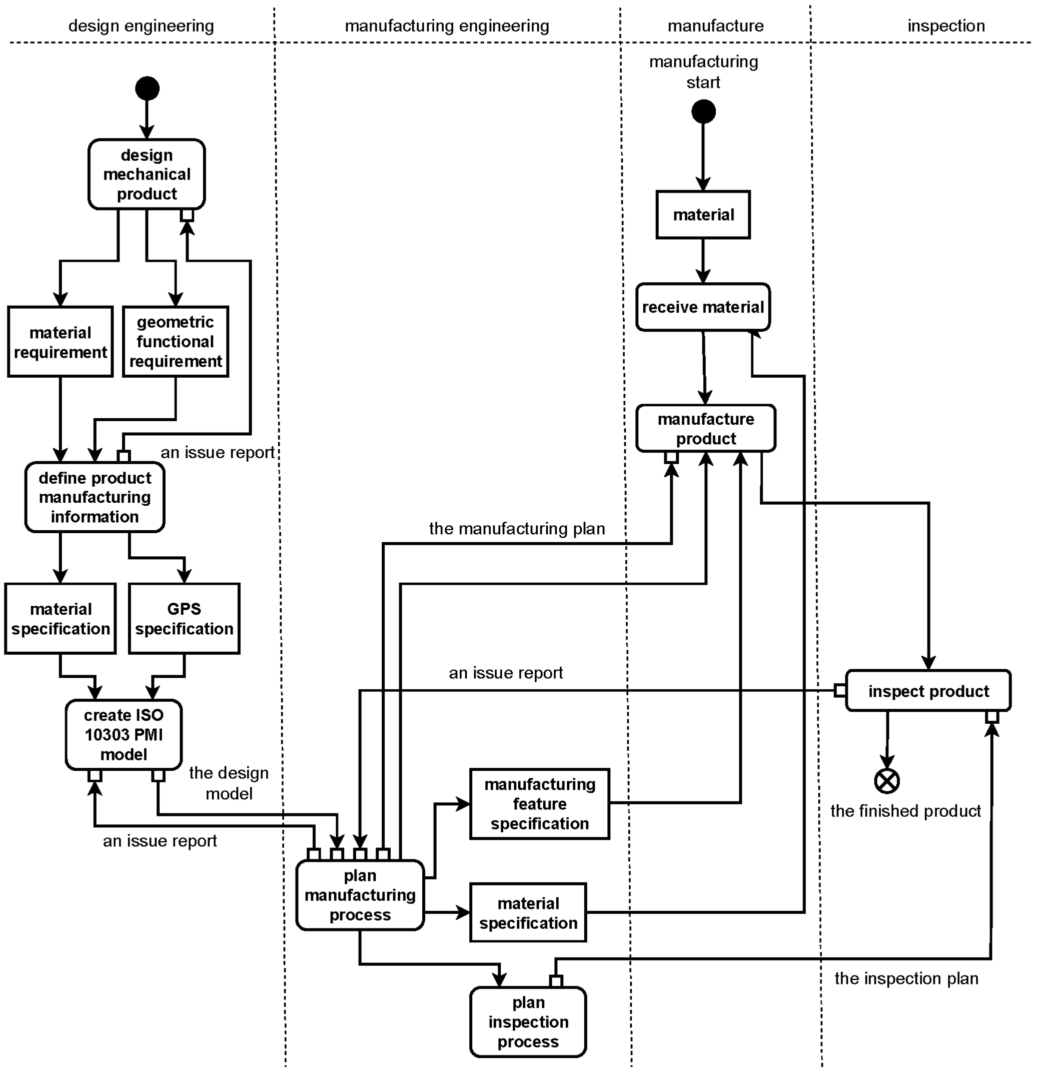


Fig. 1. The design and manufacturing workflow.

This paper proposes novel uses of **UUIDs** for a **CAD** object and for aggregated **CAD** objects that compose a single domain concept in product data standards. **UUIDs** are proposed for inclusion in the **STEP** product model to address a findability issue [9, 10] in multi-domain and multi-life-cycle engineering and manufacturing contexts. This paper provides detailed recommendations for adding **UUID**s to **AP 242** that include a proposed EXPRESS [11] information model (Sec. 11). This paper provides a recommendation for updating the **UUID** version included in future editions of **QIF**.

Section 2 of this paper describes the product data standards considered in this research.

Section 3 of this paper discusses the use cases considered in this research.

Section 4 of this paper captures common requirements for identifiers in the context of mechanical design and manufacturing model-based enterprises. Requirements are derived from industry use cases, found in the literature, patents, and specified in standards. Additionally, Sec. 4.3 describes the use of concept in this research.

Section 5 of this paper discusses **UUID** encoding requirements and characteristics. Further, it discusses types of **UUIDs**, how they are used with human-readable identifiers, and what requirements they address. The paper identifies a requirement that the **UUID** application in **STEP** applies digital signatures to deter tampering with **UUID** instances.

Section 6 of this paper describes **UUID** implementation in product data standards in the inspection domain (Sec. 6.1) and in the architecture and construction domains' Industry Foundation Classes [12] (see Sec. 6.2).

Section 8 of this paper describes the role of aggregators in **STEP** and how some aggregators are implicitly defined in a product data set.

Section 9 of this paper describes limitations of existing identifiers in **STEP**.

Section 10 of this paper discusses design options for extensions to the existing **STEP** product model. It identifies appropriate aggregates and key **STEP** ENTITY data types to use as root entities for digital signature and approval. It provides a recommended approach using [13] for classifying mechanical features for use in reference designations (Sec. 10.5).

Section 11 of this paper proposes extensions to the existing **STEP** product model. Section 11.3 of this paper discusses the application of binary hash trees in conjunction with **UUIDs** to support efficient **CAD** model comparison to detect revisions.

Section 12 of this paper discusses how the proposed model satisfies the requirements.

Section 13 of this paper provides recommendations for application behavior when implementing the proposed model.

Section 14 of this paper describes proposed extensions to existing **STEP** EXPRESS [11] declarations.

Section 15 of this paper provides description of proposed new EXPRESS declarations.

Section 16 of this paper provides an exchange example that is considered in detail.

Section 17 of this paper considers future research topics.

2. Product Data Standards

This section introduces the product data standards considered in this research.

2.1. ISO 23247 Digital Twin Manufacturing Framework

ISO 23247 [14] describes a framework for managing digital twins of manufacturing operations. It uses the **industrial internet of things** (**IIoT**) architecture described in [15] to manage the twins, and it describes how to link data representing the twins when data is distributed between multiple manufacturing systems.

2.1.1. Validation of ISO 23247 Digital Twin Manufacturing Framework

In 2020, three use case studies were used to validate ISO 23247 Part 4, Information exchange [16]. In one case study, four robots in a manufacturing cell were used to "drill and fill" holes on an aircraft wing. In this use case, the manufacturing operations were described using **AP 238**, the design constraints were described using **AP 242**, the machining results were described using **MTConnect**, and the inspection results were described using **QIF**.

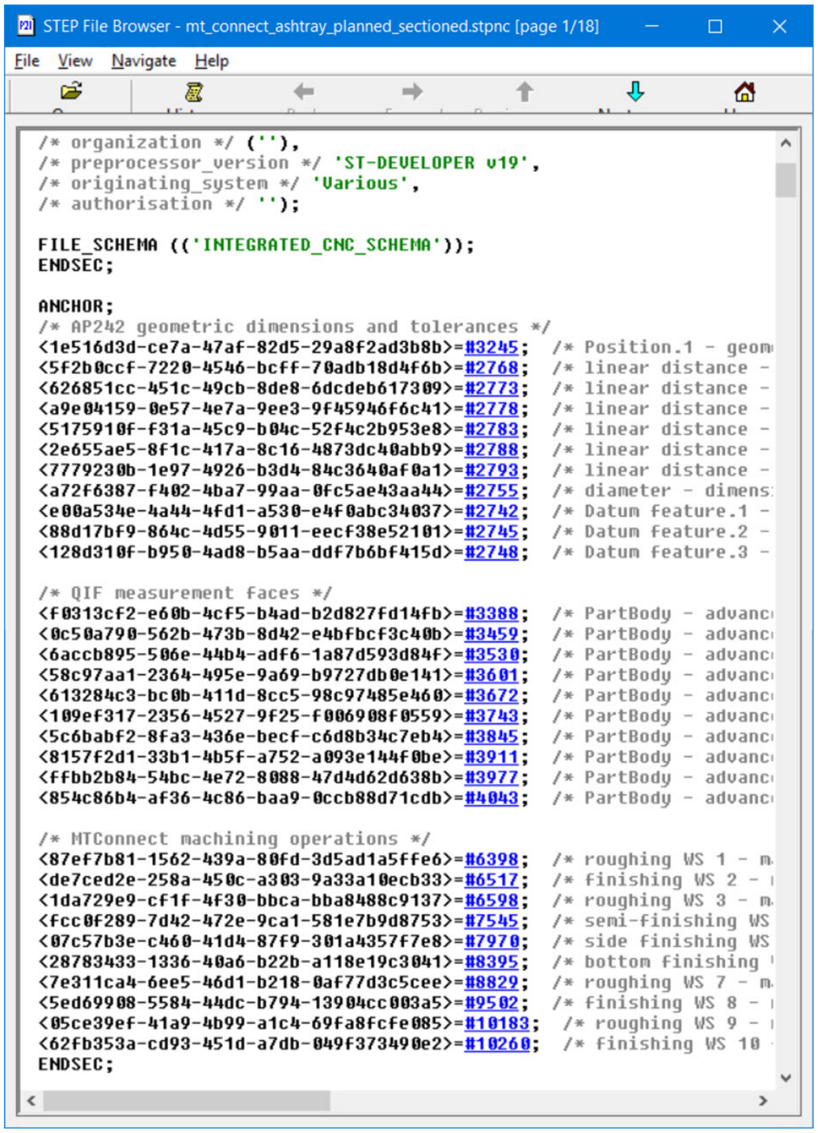
The input to the process was a digital twin of the as-inspected state of the aircraft wing. To achieve this, first an inspection planner created a QIF measurement plan. Wing components (wing skin, doubler, rib) were measured for thickness at fastener locations. Measurement results in QIF were used to adjust dimensions of the as-designed wing to reflect the as-built state. The hole stack heights were calculated when component digital twins were virtually assembled. Manufacturing process plans were updated by the process updater to consider the updated stack heights and decide which operations were to be performed by each robot. The operation list was transmitted to the robots according to ISO 6983 (**G-code**) [17] to drill and fill the necessary holes. This machining was monitored by an **MTConnect** agent to make sure that the operations were executed against the as-planned process. For example, a drilling operation that takes longer than planned would prompt an investigation to understand why this was occurring and to ensure that it was not affecting the quality of the result. The final output was a digital twin of the new state of the wing.

Digital twinning may be employed to increase quality while reducing costs. For example, in the pilot demonstration, the last-minute planning implemented by the process updater informed the robots of the as-built wing assembly and current equipment availability thus making the robots more productive. If a robot was unavailable, its work was distributed to other robots; if an operation was not necessary for a configuration of the wing, its robot was assigned other work [18].

Provided they contain accurate geometric models, the digital twins can measure in-process models of manufacturing, enabling decision-making to optimize aspects of the job. For example, if the layer stack-up for each hole is modeled accurately, each fastener can be reduced to the exact length required for each hole in the wing skin. For an airframe containing hundreds of thousands of fasteners the cumulative weight reduction can be hundreds of pounds. Measuring all the holes by hand is expensive and error-prone. In an accurate digital twin, they can be measured algorithmically over the digital model.

Digital twinning enables more efficient optimization by deploying better information models. However, many different data formats are used in manufacturing. Typically, each is aligned with a particular domain: Design, Manufacturing, or Inspection. For example, **STEP** is good at capturing design requirements, **QIF** is good at describing inspection operations, and **MTConnect** is good at monitoring machining results. To leverage the strengths of each standard while maintaining traceability of requirements through the life cycle, the ISO 23247 [14] framework describes how to manage data links between the different file formats.

When two files describe the same twin, they are required to associate the same **UUID** to that twin. Each file format is allowed to manage the associations differently. Figure 2 shows how it was managed for the **AP 238** files. An anchor section of the ISO 10303-21 [19] file lists the **UUIDs** assigned to the twins, and associates that **UUID** with an internal **STEP** entity. When an application wants to access a twin, it searches for the **UUID** and finds the internal entity.


STEP File Browser - mt_connect_ashtray_planned_sectioned.stpnc [page 1/18]
File View Navigate Help

```
/* organization */ (''),
/* preprocessor_version */ 'ST-DEVELOPER v19',
/* originating_system */ 'Various',
/* authorisation */ '');

FILE_SCHEMA (('INTEGRATED_CNC_SCHEMA'));
ENDSEC;

ANCHOR;
/* AP242 geometric dimensions and tolerances */
<1e516d3d-ce7a-47af-82d5-29a8f2ad3b8b>=#3245;  /* Position.1 - geom
<5f2b0ccf-7220-4546-bcff-70adb18d4f6b>=#2768;  /* linear distance -
<626851cc-451c-49cb-8de8-6dcdeb617309>=#2773;  /* linear distance -
<a9e04159-0e57-4e7a-9ee3-9f45946f6c41>=#2778;  /* linear distance -
<5175910f-f31a-45c9-b04c-52f4c2b953e8>=#2783;  /* linear distance -
<2e655ae5-8f1c-417a-8c16-4873dc40abb9>=#2788;  /* linear distance -
<7779230b-1e97-4926-b3d4-84c3640af0a1>=#2793;  /* linear distance -
<a72f6387-f402-4ba7-99aa-0fc5ae43aa44>=#2755;  /* diameter - dimens
<e00a534e-4a44-4fd1-a530-e4f0abc34037>=#2742;  /* Datum feature.1 -
<88d17bf9-864c-4d55-9011-eecf38e52101>=#2745;  /* Datum feature.2 -
<128d310f-b950-4ad8-b5aa-ddf7b6bf415d>=#2748;  /* Datum feature.3 -

/* QIF measurement faces */
<f0313cf2-e60b-4cf5-b4ad-b2d827fd14fb>=#3388;  /* PartBody - advanc
<0c50a790-562b-473b-8d42-e4bfbcf3c40b>=#3459;  /* PartBody - advanc
<6accb895-506e-44b4-adf6-1a87d593d84f>=#3530;  /* PartBody - advanc
<58c97aa1-2364-495e-9a69-b9727db0e141>=#3601;  /* PartBody - advanc
<613284c3-bc0b-411d-8cc5-98c97485e460>=#3672;  /* PartBody - advanc
<109ef317-2356-4527-9f25-f006908f0559>=#3743;  /* PartBody - advanc
<5c6babf2-8fa3-436e-becf-c6d8b34c7eb4>=#3845;  /* PartBody - advanc
<8157f2d1-33b1-4b5f-a752-a093e144f0be>=#3911;  /* PartBody - advanc
<ffbb2b84-54bc-4e72-8088-47d4d62d638b>=#3977;  /* PartBody - advanc
<854c86b4-af36-4c86-baa9-0ccb88d71cdb>=#4043;  /* PartBody - advanc

/* MTConnect machining operations */
<87ef7b81-1562-439a-80fd-3d5ad1a5ffe6>=#6398;  /* roughing WS 1 - m
<de7ced2e-258a-450c-a303-9a33a10ecb33>=#6517;  /* finishing WS 2 -
<1da729e9-cf1f-4f30-bbca-bba8488c9137>=#6598;  /* roughing WS 3 - m
<fcc0f289-7d42-472e-9ca1-581e7b9d8753>=#7545;  /* semi-finishing WS
<07c57b3e-c460-41d4-87f9-301a4357f7e8>=#7970;  /* side finishing WS
<28783433-1336-40a6-b22b-a118e19c3041>=#8395;  /* bottom finishing
<7e311ca4-6ee5-46d1-b218-0af77d3c5cee>=#8829;  /* roughing WS 7 - m
<5ed69908-5584-44dc-b794-13904cc003a5>=#9502;  /* finishing WS 8 -
<05ce39ef-41a9-4b99-a1c4-69fa8fcfe085>=#10183;  /* roughing WS 9 -
<62fb353a-cd93-451d-a7db-049f373490e2>=#10260;  /* finishing WS 10
ENDSEC;
```

Fig. 2. Digital twin manifest in an AP 238 file.

For this particular example, the UUIDs are created by data translation systems. As shown in Fig. 2, for the use case, UUIDs are assigned for the design tolerance data, the inspection planning geometry, and the machining operations. The design tolerance UUIDs are shared with a design system that is assumed to be managing its data using an AP 242 file. The inspection planning UUIDs are shared with an inspection system that is assumed to be managing its data using a QIF file. The machining operation UUIDs are shared with a machining system that is assumed to be managing its data using an MTConnect file.

The content of each file is created by a product lifecycle management (PLM) system. The internal identifiers will change with each iteration of the file. The ISO 23247 framework allows this, but requires the same **UUID** to be assigned to the same digital twin for external operations. This increases the burden on data translators. They need to store the **UUID** of each twin in an internal database for reuse in new versions of the data. In practice this is not difficult if the translator understands the semantics of the data. For example, when the **AP 238** translator knows that it is creating tolerance twins for the design system, geometry twins for the inspection system, and machining twins for the manufacturing system, the **UUIDs** for those twins are stored in the master data and reused for subsequent translations.

Comments are added to the **AP 238** file to help the information technology organization understand the linking and manage the systems. This includes comments to group **UUIDs** by system functionality, and comments to distinguish between twin instances. In Fig. 2, the instance comments give the machining twins unique names, but are less useful for the inspection twins because visualization is needed to aid understanding of the geometry. Software tools can be developed to assist with this data management.

2.2. ISO 10303 (STEP) Application Protocol 209

ISO 10303-209:2014 [2] titled “Multidisciplinary analysis and design”, commonly known as **AP 209**, is a product information standard for the design and analysis of composite structures. The goal of **AP 209** [20] is to support the exchange of computer-interpretable composite and metallic structural product definitions, including product shape, associated numerical analysis models (including **finite-element analysis** (**FEA**) models) and their related analysis results, and the properties of these products. The analysis model, or a set of related analysis models, idealizes a product, or aspects of a product, so that they may be utilized to validate the performance and integrity of a product. The shape definitions for design and analysis are each independently configuration-controlled. Assembly information provides the relationships necessary to identify analysis boundary conditions, and, when combined with part geometry, topology, and analysis output, provides the input necessary for downstream detail analyses. **AP 209** satisfies the need for exchange of information between the iterative design and analysis stages of the product life cycle. The current edition of **AP 209** contains an information model that is a superset of the information model for the first edition of **AP 242**, so much of the discussion of **AP 242** applies to **AP 209**. The recommendations made in this research, if accepted by industry and incorporated into **STEP**, would become available for the next edition of **AP 209**.

2.3. ISO 10303 (STEP) Application Protocol 238

ISO 10303-238:2022 [3] titled “Managed model-based integrated manufacturing”, commonly known as **AP 238**, is an information standard for describing manufacturing solutions. **AP 238** provides manufacturing execution process plans and machine code for manufacturing machining centers. In its first edition, the standard defined subtractive machining programs for **AP 203** [21] and **AP 214** [22]. A second edition, published in 2020, for model-based integrated manufacturing, defined additive and subtractive programs for processing **AP 242** Edition 2 design models into workpieces. This edition subsumed new **AP 242** capabilities for geometry, tolerances, and kinematics. A minor update was published as the third edition in 2022.

The goal of **AP 238** is to enable a form of manufacturing in which models are used to control processes [23]. In the late 1950s a manufacturing paradigm was established in which codes to control a machine tool are created by a **computer-aided manufacturing** (**CAM**) system. The **CAM** operator analyzes the geometry of a component and determines a series of operations that will convert an input (the stock) into an output (the workpiece). The operations are executed by moving a cutting tool along paths. These paths are then converted into codes for a **computer numerical control** (**CNC**) system [24].

AP 238 enables just-in-time automation of changes to products, processes, and resources during manufacturing. For example, as a tool wears its diameter changes; as a product is assembled, its dimensions are adjusted; or as a process executes, features are delayed by equipment failures or removed in response to customer requirements. An application that uses **AP 238** to model its data has the capability to detect these changes and compare an as-planned manufacturing model to an as-measured model. The **AP 238** application can then generate a new machining program to meet the new conditions [24].

AP 238 defines a language for manufacturing that divides a task into a tree of work plans containing working steps. A work plan can be decomposed into working steps. Different types of work plans can be defined for programs that have optional or parallel components. Working steps select tooling and define operations to machine features. For example, a working step may mill a pocket, or melt material to create a layer. The STEP-NC control language [25] was developed as ISO 14649 [26]. **AP 238** Edition 1 mapped this language into the integrated resources of **STEP** so that the operations, features, and tooling of a machining solution can be linked to the product geometry and topology then defined by **AP 203** and **AP 214**. **AP 238** Edition 2 extends the links to include the **PMI**, kinematics, and tessellated models defined by **AP 242** Edition 2. The **PMI** enables checking of the dimensions of a machining result against manufacturing requirements encoded in a **geometric dimensions and tolerances** (**GD&T**) language. Inclusion of kinematics enables a machine tool's capabilities to be validated against the motion requirements of the machining program before it is selected as a substitute machine for a manufacturing task. The tessellated models allow **AP 238** operations to be applied to additive manufacturing processes as well as subtractive ones. Each layer of the material is modeled as a thin mesh. Working steps are used to control the power and focus of the laser as it fuses material to create the topology.

2.4. ISO 10303 (STEP) Application Protocol 242

ISO 10303-242:2022 [4] titled "Managed model-based 3D engineering," commonly known as **AP 242**, is a broadly-used product information standard that, in its third edition, has become increasingly capable of conveying manufacturing requirements to downstream systems [27]. Figure 3 illustrates the scope of edition 3 of **AP 242**.

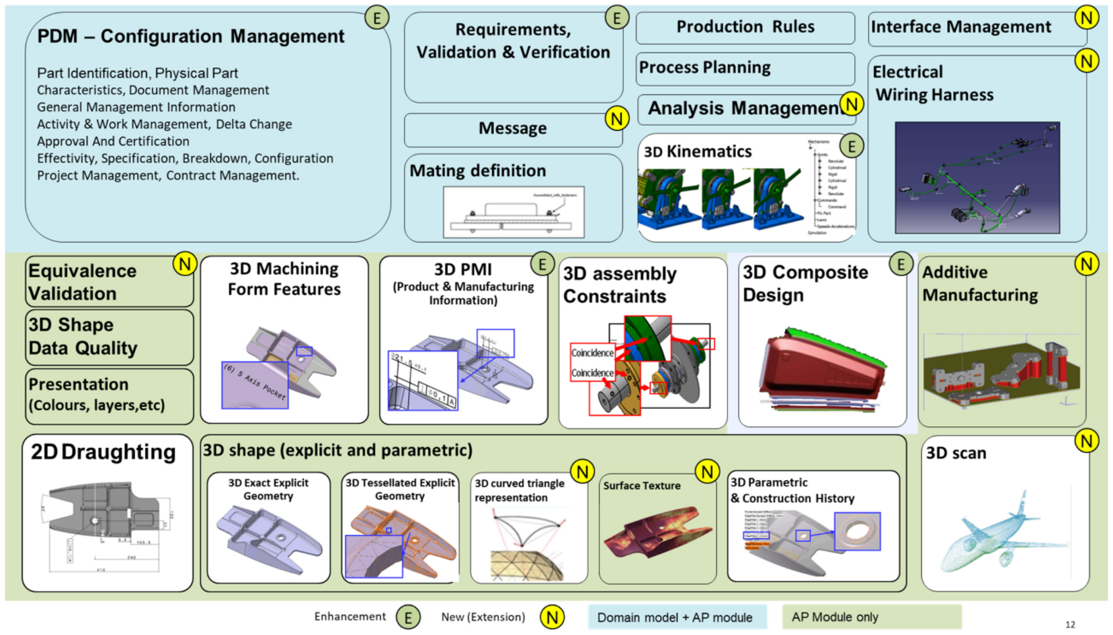

PDM – Configuration Management
Part Identification, Physical Part Characteristics, Document Management General Management Information Activity & Work Management, Delta Change Approval And Certification Effectivity, Specification, Breakdown, Configuration Project Management, Contract Management.
Requirements, Validation & Verification
Message
Mating definition
Production Rules
Process Planning
Analysis Management
3D Kinematics
Interface Management
Electrical Wiring Harness
Equivalence Validation
3D Shape Data Quality
Presentation (Colours, layers,etc)
3D Machining Form Features
3D PMI (Product & Manufacturing Information)
3D assembly Constraints
Coincidence
Coincidence
3D Composite Design
Additive Manufacturing
2D Draughting
3D shape (explicit and parametric)
3D Exact Explicit Geometry
3D Tessellated Explicit Geometry
3D curved triangle representation
Surface Texture
3D Parametric & Construction History
3D scan
Enhancement E
New (Extension) N
Domain model + AP module
AP Module only
12

Fig. 3. AP 242 edition 3 scope.

The goal of **AP 242** is to support a manufacturing enterprise with a range of standardized information models that flow through a long and wide "digital thread" that makes the manufacturing systems in the enterprise smart [28]. The information models are designed to support classification and identification schemes as well as data validation. **STEP** provides a series of standards that meet the requirements of long-term data retention for industries such as aerospace and defense [29]. Digital data conformance to **AP 242** plays a central role in achieving the goal of successful long-term data retention.

First published in December 2014, **AP 242** contains extensions and significant updates to other **STEP APs** for **PMI**, kinematics, and tessellation [30]. **PMI** is the presentation and representation of **GD&T**, requirement specifications, material specifications, surface conditions, component lists, process specifications, and inspection requirements within a **3D** product definition [31]. A second edition, published in July 2020, extended support for **American Society of Mechanical Engineers** (**ASME**) and **ISO GPS** standards, and for large-scale electrical wire harnesses for aerospace, defense, and automotive applications. Support was extended to include as-designed, as-planned, and as-measured characteristic data. Extensions were provided to better support complex fastener application at assembly, as was described in detail in Sec. 2.1. Support was extended for additive manufacturing. Piece part and assembly shape quality extensions were provided. A major enhancement was provided to support the definition of tessellated geometry using curved triangles. Extensions were provided to enhance the ability to accurately specify and control the syntactic and semantic numeric representation of real numbers in data exchange. A third edition, published in December 2022, provided extensions for **ASME** and **ISO GPS** standards, among other improvements. A study [31] comparing drawing-based processes with model-based processes concluded that **PMI** has the potential to make many life cycle processes run faster, with fewer errors, at lower cost. **AP 242** offers standards-based models that include the representation of **PMI** that is semantically rich and computer interpretable [28]. This is a major breakthrough that supports manufacturing's need for model-based **CAM** and **coordinate measurement system** (**CMS**) processes. **AP 242** increases the effectiveness of **MBE** by enabling a common path for **MBD** and model-based manufacturing (**MBM**) integration [30, 32]. **STEP** is widely adopted in all major **CAD** systems, but users must ensure that their **computer-aided technologies** (**CAx**) systems support the relatively new **AP 242**. Implementation guidance in the form of published recommended practices is provided by consortia composed of stakeholders and **CAD** software suppliers as described in [33]. **AP 242**, **AP 209**, and **AP 238** share a core **STEP** product information model.

The above capabilities make AP 242 the de facto engineering design and product information data standard. However, the ability to trace data correlated to product characteristics through the life cycle (downstream traceability) and back to the originating system (bi-directional traceability) is lost when downstream applications do not support the as-planned and as-measured capabilities in **AP 242**.

2.5. MTConnect

The **MTConnect** standard [5] is a model-based **industrial internet of things** (**IIoT**) standard using **Systems Modeling Language** (**SysML**) as its normative language for defining the information model and its behavior. **MTConnect** addresses the domain model concerns for describing and observing manufacturing equipment and its components. Developers and system integrators use this information model to provide structured, contextualized data with no proprietary format. Standardized data reduces translation costs and lets users focus on practical, productive manufacturing applications rather than low-level data wrangling. **MTConnect** data sources include production equipment, sensor packages, and other hardware.

MTConnect is not just about creating structured data with context for client applications. It's about fostering semantic interoperability between different manufacturing systems and client applications. The information models defined in **MTConnect** provide a common vocabulary and structure for manufacturing-equipment data. What's more, the standard's focus on accurate and near-real-time data from the equipment enriches the digital thread by providing information about the as-built model of a part. Using **MTConnect** leads to more efficient operations, improved production optimization, and increased productivity.

MTConnect is a standard that supports multiple protocols and representations. As a model-based standard, it doesn't dictate how the data should be communicated or represented.

This flexibility is possible using model-based normative SysML, which allows **MTConnect** to support **REpresentational State Transfer** (**REST**), **Open Platform Communications United Architecture** (**OPCUA**), **MQTT**, and **Sparkplug B** communication protocols and **XML**, **JavaScript Object Notation JSON**, and **NodeSet** data representations. This adaptability enables applications to consume the standard in a way that best suits their purpose while also allowing **MTConnect** to focus on the semantics and structure of the data.

MTConnect communicates observations about the manufacturing process and provides information about assets used in the process that cannot produce information about themselves. Examples of assets are cutting tools, fixtures, machine pallets, and data files. **MTConnect** can collect information about the assets and communicate it to applications or other machines.

Although **MTConnect** is a read-only standard, it does support inter-device deterministic communication using a choreography or observation-based architecture. In this model, the machines observe each other and react based on their observations. The machines are responsible for managing their state and will not perform an action if it sacrifices the safety or function of the machine. The interfaces model in **MTConnect** is being enhanced to support multi-device, task-based activities that can scale from cells to shop-level integration. This architecture is more secure and safe than writing directly to the controller, since the machine controls its state and activities.

MTConnect Institute is looking to the future and how to address **IIoT** concerns. **MTConnect** Institute has become involved in developing an ontological layer in association with the **Industrial Ontology Foundry** (**IOF**) [34] to enable the extension of the standard to support knowledge graphs and address concerns like device capabilities and cross-domain terminology. The new effort will bring a new dimension to the standard and allow greater interoperability with domains such as maintenance and supply chains. Ontologies also provide the ability to reason on data and infer classifications that are not stated but inferred by the characteristics of the data.

2.6. Quality Information Framework

ISO 23952:2020 [7] titled “Quality Information Framework (QIF) - An integrated model for manufacturing quality information,” commonly referred to as **QIF**, defines an integrated set of **XML** information models that enable the effective exchange of metrology data throughout the entire metrology process. **QIF** handles feature-based dimensional metrology, quality measurement planning, first article inspection, and discrete quality measurement. **QIF** supports defining or importing the product definition and reusing data for inspection planning, execution, analysis, and reporting.

QIF uses terminology and semantics from the inspection world to represent the various elements in the **QIF** specification. **QIF** uses **XSD** to specify its information models. The standard organizes **QIF XSDs** into six application areas for metrology: (1) **MBD**, (2) Rules, (3) Resources, (4) Plans, (5) Results, (6) Statistics. The standard combines the **MBD** (containing the product definition) with measurement rules and resources definitions to generate a measurement plan. The plan is then executed and the results are captured. Multiple results are combined to generate statistics. **QIF** does not perform the task of statistics and the other metrology methods. Instead, **QIF** is able to put raw inspection data into a quality context that is computer-interpretable.

From the first through third editions, **QIF** used the term **QPId** to define the text representations of the universally unique identifier described in the standard ISO/IEC 9834-8. The fourth edition, expected to be published in 2024, will refer to them using new terminology to harmonize with other industry standards. Specifically, the element of QIFDocumentType that was QPId is changed to QIFUUID; the type name QPIdType is changed to UUIDAssignmentType; and the type name QPIdReferenceType is changed to UUIDReferenceType.

Typical uses of **QIF** in industry are capturing and transmitting inspection results between organizations and describing available measurement assets. **QIF** features are different data objects than design features and are different than manufacturing features. **QIF** features are classifications of product geometry (e.g., cylinder feature, parallel planes feature, plane feature) that establish a reference object for **QIF** characteristics (e.g., diameter characteristic, flatness characteristic, position characteristic). **QIF** characteristics are different from design characteristics but may be mapped to design characteristics. Persistent ID of features and characteristics are optional in the **QIF** standard, as there is an alternative mechanism to disambiguate a feature and characteristic object in the standard. Applications that desire to ease quality traceability throughout the product life cycle should consider using **UUID** for all features and characteristics.

3. Use Cases

The following use cases identified by industrial stakeholders are used as a source of requirements for **UUIDs**:

- Design to manufacturing release [35, 36];
- Shared design collaboration;
- Design and measurement collaboration [37, 38];
- Product version comparison;
- Test and design collaboration; and
- Manufacturing Task Supervisor.

3.1. Design to Manufacturing Release

In this use case, following the process sequence illustrated in Fig. 1, a model-based product definition is released for manufacturing by design engineering. Each requirement on the form, fit, and function of the product has been associated with the modeled **CAD** objects. All manufacturing requirements have been properly identified for tracking. The exact details of what forms a design release are enterprise, industry, and practice standard specific. Some enterprises include manufacturing process planning as part of the release activity. A **STEP AP** defines prescribed information content but does not dictate business practices such as when in the life cycle sub-stage that information is created.

An initial release of a product definition has, at a minimum, the following properties:

- each **CAD** object of interest to the enterprise has a **UUID** associated to the **CAD** object;
- each engineering requirement is associated with the overall product or with one or more **CAD** objects; and
- each engineering requirement is identified with a **UUID**.

The manufacturing engineering department creates the model-based manufacturing plan. The model-based manufacturing plan has, as a minimum, these properties:

- key **CAD** object **UUIDs** have been mapped to a step in a sequential manufacturing execution requirement (program) where that step is identified with a **UUID**; and
- each engineering requirement has been mapped to a step in that sequential manufacturing execution requirement (program) that is identified with a **UUID**.

The manufacturing engineering department creates the model-based inspection plan. The model-based inspection plan has, as a minimum, these properties:

- key **CAD** object **UUIDs** have been mapped to a step in a sequential inspection execution requirement (program) where that step is identified with a **UUID**; and
- each engineering requirement has been mapped to a step in that sequential inspection execution requirement (program) that is identified with a **UUID**.

A revised product definition has these properties:

- each **CAD** object of interest to the enterprise that existed in the prior revision retains the **UUID** from that prior revision;
- each engineering requirement that existed in the prior revision retains the **UUID** from that prior revision; and

- each item in the design that is revised is identified and controlled by the enterprise quality change management process.

The model-based manufacturing and model-based inspection plans would be revised to accommodate the updated engineering requirements whilst maintaining required traceability.

Example 1. A small change occurs in a design tolerance engineering requirement. The tolerance was modified from +/-0.010 mm to +/-0.005 mm. The manufacturing engineering application that creates the model-based manufacturing plan determines that a different tool is now required to manufacture the part and assigns the new tool to the step in the new revision of the model-based manufacturing plan. The **UUID** of the step is unchanged from the previous revision of the model-based manufacturing plan.

3.2. Shared Design Collaboration

In this use case, two design partners iterate on a shared design. A shared design is interpreted such that at each stage of the workflow, the design data is synchronized so that from an external perspective, it is irrelevant which partner generated the data for external use. Several of the **CAD** objects are owned by the first design partner and other **CAD** objects are owned by the second design partner. A detailed use case is provided by [39].

Example 2. **OEM** One creates an annotation that references one surface that represents a hole in the native model. That surface is controlled by a single topological **advanced_face**. In post-processing by **OEM** Two, that surface is split into two surfaces and two **advanced_face** instances exist in the derivative model. In order for the collaboration to be successful, **OEM** Two retains knowledge that the two **advanced_face** instances in **OEM** Two's model are controlled by the single **advanced_face** instance assigned a **UUID** by **OEM** One's **CAD** system, and that the annotation applies equally to the combination of the two **advanced_face** instances in **OEM** Two's model and to the single **advanced_face** instance in **OEM** One's model.

Example 3. An **OEM** collaborates with a Supplier on a mechanical design of an equipment container, illustrated in Figure 4, that is size-constrained by its design environment (not shown). The product illustrated includes:

- a cover;
- a container; and
- a component chassis.

In this example, the design is unreleased, indicated by a ”0” in the major digit of the revision code. Both the **OEM** and the supplier use a minor digit on the revision identifier to signify an iteration during the design phase.

The Supplier is responsible for designing and delivering the component chassis that is mounted within the container as shown. The initial condition of the shared model elements considered are illustrated in table 1.

Object Owner	Iteration id	Item	UUID	Data
Supplier	0.0	Chassis keyway	1	Original orientation
OEM	0.0	Interface hole	2	Original interface hole data

Table 1. Initial condition of model.

During an iteration of the design of the chassis in the **OEM**'s container, the Supplier analyzes component placements and determines that one component must be reoriented to improve cooling efficiency. To support the component reorientation, the highlighted mounting keyway on the chassis is rotated 90° from the reference as illustrated in Figure 5a to the new position illustrated in Figure 5b. The above change, in turn, requires adjustment to the component's wiring (not shown) to accommodate the orientation change. As a result of the change, and to maintain minimum required cable bend radius of the component's interface cable, the Supplier returns an iteration of the definition of the component chassis, and returns an iteration of the container with an annotated request that the cable interface hole (identified by its **OEM**-owned **UUID**) in the container wall be moved to the right of the keyway.

The design change of the keyway by the Supplier is also identified by its **UUID**, owned by the Supplier, to make clear to the **OEM** what change has been made in the chassis. The details of the change report are not considered in this research, but the proposed change could include a report identifying that:

- Supplier-owned (reoriented chassis keyway) data orientation values identified by **UUID**(1) supersede Supplier-owned (original chassis keyway) data values identified by **UUID**(1) as illustrated in table 2; and
- **OEM**-owned data values identified by **UUID**(2) (moved interface hole in the container) supersede **OEM**-owned values identified by **UUID**(2) (original interface hole) as illustrated in table 3.

Object Owner	Iteration id	Item	UUID	Data
Supplier	0.1	Chassis keyway	1	Changed orientation

Table 2. First iteration.

Originator	Object owner	Iteration id	item	UUID value	Data value
Supplier	OEM	0.1	Interface hole	2	Technical Details

Table 3. Change request report from Supplier.

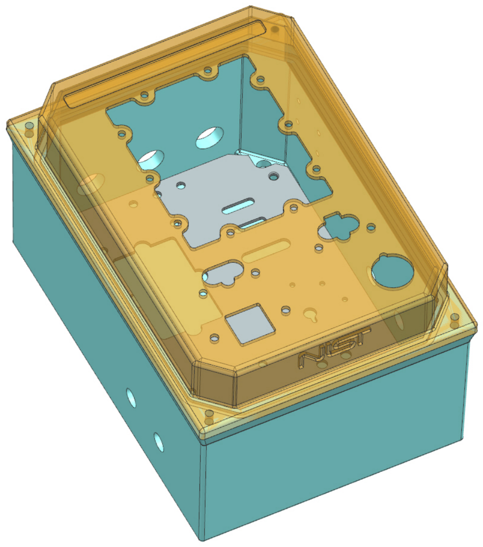
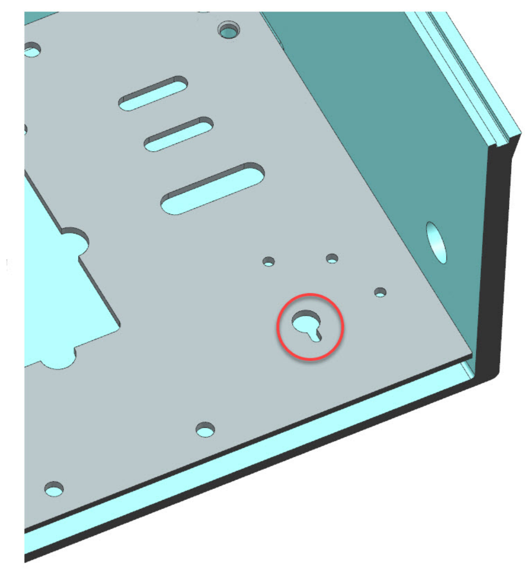


Fig. 4. An assembly of components.

Upon receiving the change report, the **OEM** would update the chassis design to the latest version supplied and would evaluate the request to change the location of the interface hole in the container. Acceptance after evaluation would result in an update to the location of the interface hole in the **OEM** design model and transmittal of the updated design to the Supplier. An evaluation resulting in rejection would initiate another round of collaboration.

Example 4. An **OEM** collaborates with a contractor to manufacture a part. The initial position tolerance specified by the contractor of .030 mm from design iteration ”0.0” is illustrated in Figure 6a.

The **OEM** requested a change in tolerance after the required assembly testing of the field-replaceable component showed that reassembly by field personnel was difficult due to the tight tolerance on the location of the mounting holes. Figure 6b illustrates the value of the position tolerance changed to .040 mm in design iteration ”0.1”. The **OEM** identifies the correct tolerance to change by transmitting the **UUID** of the tolerance annotation for the mounting hole pattern to the Supplier in a change order. The Supplier makes the


(a) A close-up of the original keyhole orientation.

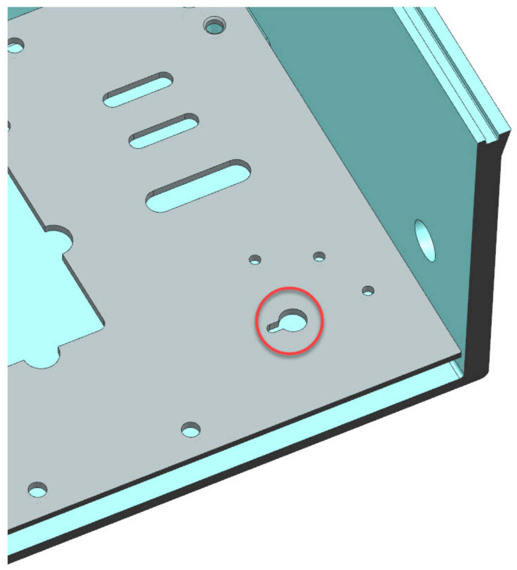
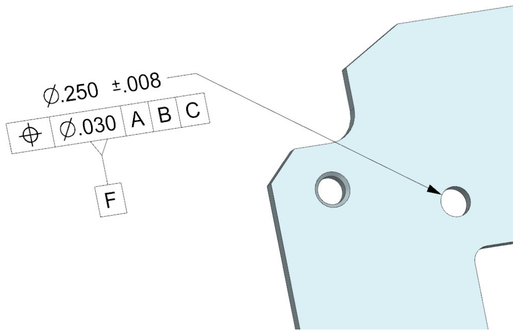

(b) A close-up of the modified keyhole orientation.

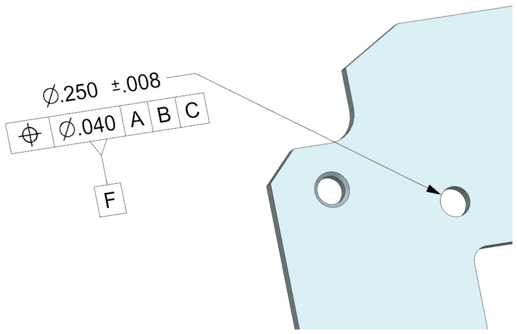
Fig. 5. Changes made to the keyhole in an assembly component.

change and returns the chassis model with updated values for the hole pattern tolerance annotation. The update package could include a report identifying that:

- Supplier-owned location tolerance data values (identified by **UUID**(1)) supersede original location tolerance identified by **UUID**(1); and
- **OEM**-owned test report action/closure record has been resolved.



(a) The original position tolerance.



(b) The position tolerance is changed.

Fig. 6. Changes made to the geometric tolerance value.

3.3. Design and Measurement Collaboration

Downstream measurement applications generally utilize geometry and topology as reference data only and are more focused on processing PMI. The downstream application provides measurement results. The availability of **CAD** system-generated **UUID** data values improves the quality and efficiency of the overall design and manufacturing process because it eliminates some associativity gaps [40] in the manufacturing life cycle workflow between the design organization and the measurement organization.

Example 5.

A design model contains a default position tolerance instance that is assigned a **UUID** (33) by the source **CAD** system. The data value assigned to that instance is (+/- 0.01 mm). Table 4 specifies management data for a workpiece based on that design model. A measurement on that workpiece at the point (1.002 mm, 2.003 mm, 2.001 mm) on the surface indicates an out-of-tolerance condition (+0.02 mm displacement). The measurement system assigns a **UUID** (99) to that measurement. The measurement system creates a measurement report containing a reference to **UUID** (99), a reference to the content of Table 4, along with a reference to the **UUID** (33) of the controlling tolerance specification (+/- 0.01 mm). Optionally, the measurement system creates a magnified graphic view of the neighborhood of interest and includes that in the measurement report. The manufacturing system at the enterprise level would encapsulate the data content with an issue record formatted in accordance with enterprise exchange agreements. The design system would perform a geometric query to identify the neighborhood of interest in the design model as part of the issue resolution process. The design system would then process the issue in accordance with the enterprise quality process. Ideally, if a design change is needed due to issues found in manufacturing, changes would be propagated through the digital system without the need for a human in the loop to validate what has changed in the models. Issue exchange is planned for edition 4 of **AP 242**. Currently, **AP 239** provides the ability to exchange issues through the ENTITY data type **observation**.

design owner	abcdef.com
part number	xyz-77-4
version	36
serial number	35
manufacturer	imanufactureparts.com

Table 4. Workpiece management data.

3.4. Derive a New Version or Product

In the product derivation use case, a new design contains a number of characteristics that are different from a reference design. For this use case, there may be new derived characteristics. In the case that components are deleted to create the new design, the reference designator values are re-sequenced because quality standards do not (usually) permit gaps in the sequence of designator identifiers. In the case that enterprise quality standards require that provenance of the data set be retained, object instance translation in the **STEP** pre-processor will create the same **UUID** as that for the object instance in the reference design.

3.5. Transfer Product Design Ownership

ASME Y14.100-2017 [41] provides guidance for transitioning product design ownership between enterprises. Following is an excerpt from that standard.

> D-9.9 Transferring Design Responsibility to Another Activity
>
> When the design responsibility for engineering drawings is transferred from one design activity to another, the drawing number(s) and part identifying number (PIN) shall be transferred to the new design activity for administration. The new assignee shall add their **CAGE** code, name, and address on the drawing by revision action to identify the change in design responsibility. In no case will the original drawing identity be changed or relocated to indicate a new **CAGE** code.

The stated requirement implies:

- a record must be provided that relates the current design activity with the previous design activity and the part identifying number, including the relevant version and product definitions;
- the change process will ensure that the quality of the **CAD** product data is not compromised in the exchange; and
- because a **CAD** model is not a drawing, but may evolve, the current design activity will need to maintain the source **CAD** model and related data in an archive for reference purposes.

Figure 7 illustrates the product data relationships (in the form of a domain requirements model) supporting requirements specified by [41], in EXPRESS-G [11] notation.

The illustration shows only one product version being transitioned for clarity. Three product definition instances are included to illustrate the case where electrical, mechanical, and requirements definitions are included in the design data set.

The **STEP** implementation model for these requirements uses **STEP** management resource entities [42] that result from interpreting the domain terms into the **STEP** product model in **AP 242** to record the ownership transition.

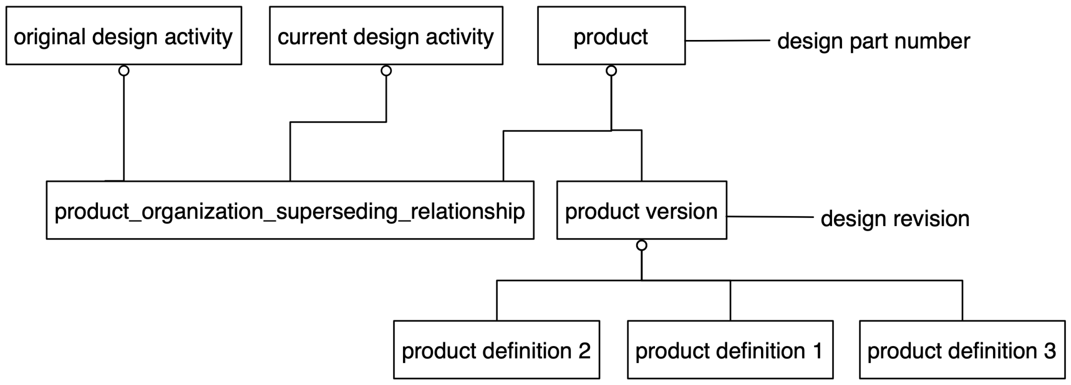


Fig. 7. Relationships between previous and current design activities and related product data.

3.6. Test and Design Collaboration

A system for managing workflow interactions between design and testing organizations is presented in [43]. That system is based on **AP 209**. The system captured test results in a few of the entities that are shared between **AP 209** and **AP 242**:

- **action**;
- **action_method**;
- **applied_action_assignment**;
- **object_role**;
- **product_definition**;
- **property_definition**; and
- **role_association**.

This is a small subset of the shared product information model between the two **APs**.

3.7. Manufacturing Task Supervisor

In a manufacturing task supervisor test case, **AP 238** as-planned data is the input to the task supervisor software. The supervisor is linked to a monitoring system that reports the current state of a machine tool, and a measurement system that reports the as-machined dimensions of a product. The measurement system may be on the machine tool or in an external **coordinate measurement system** (**CMS**). When the monitoring system reports results that differ from the as-planned process, the supervisor uses the measurement system to check the tolerances of features that may have been impacted by the discrepancy.

Machine monitoring systems report their results to the task supervisor using **MTConnect** [5]. Coordinate measurement systems report their results using **QIF**. Each language meets different domain requirements. **MTConnect** needs to report the current values of many different control registers very quickly. **QIF** needs to evaluate measurements and determine results while maintaining links to requirements. **STEP** product modeling technology was designed to unambiguously represent a product definition and is not suited for streaming real-time data from manufacturing systems. Similarly, using **STEP** to write measurement reports would be unnecessarily complicated [44].

The manufacturing supervisor needs to know when **MTConnect** is reporting data in the context of one of its manufacturing operations, and when **QIF** is measuring one of its dimensions. In **AP 238** Edition 2 this is managed using **UUIDs**. The **UUIDs** are stored in the Anchor section of the **STEP** Part 21 file [45]. An anchor links the **UUID** with its internal **STEP** representation. When an operation is started, **MTConnect** reports the **UUID** of the operation, and when a feature is measured, **QIF** reports the **UUID** of the measured dimension. The deployment of **UUIDs** in the supervisor allows each language to continue using its internal data structures. For example, if the feed override is activated by an operator, then the monitoring system reports the **UUID** of the current workingstep using **MTConnect**. The supervisor can then use **AP 238** to determine which feature on the product model was being machined and to find any tolerances that should be validated for conformance to the **AP 242** requirements. The measurement system is then activated. It inspects the product model and reports the as-measured values. The tolerances are also identified using a **UUID**. The supervisor relates each measured value to its **PMI** requirements, and requests assistance if there is potential for an issue on the final part. When many copies of the same product are machined, the features, instances, and products are each given **UUIDs**. An individual feature, on a specific instance, for a specific product can be identified using its combination of **UUIDs**. In the reverse direction, if a **QIF** application wants to show a model of a feature, then it can request a visualization of the product model containing the right combination of **UUIDs**.

4. Common Requirements

This section captures common requirements for identifiers in the context of mechanical design and manufacturing model-based enterprises. Requirements are derived from use cases identified in Sec. 3, those found in literature, patents, and specified in standards.

4.1. Product Identification Requirements

Identifiers for the enterprise (that is the product design owner), for the product model class, for the product part number, and for the product serial number are typical data elements managed at the manufacturer and customer level.

Consideration of the established widely-used standards for product marking was outside the scope of this research.

4.2. Feature Classification and Identification Requirements

4.2.1. Terminals

All technical disciplines use **terminals**, to the extent that identification of **terminals** is a common requirement across industry segments. IEC 61666 [13] provides general principles for **terminal** identification specifically within a system. From the scope statement of IEC 61666:

> IEC 61666 establishes general principles for the identification of terminals of objects within a system, applicable to all technical areas (for example mechanical engineering, electrical engineering, construction engineering, process engineering).

terminal identifiers are human-readable, and where feasible marking is provided on the object that visually identifies the **terminal**. In some cases, marking is infeasible and reference material must be reviewed to identify the **terminal**. Terminal identifiers provide increased resolution for associating corresponding aspects of different documentation or models of a system over the use of reference designators alone. Each **terminal** shall be unambiguously identified with respect to the object that owns the **terminal**. **terminals** belong to the object for which they are **terminals**. In an existence-dependent information model, when the record representing the owning object is deleted from the data set, the **terminal** is deleted from the data set. In **STEP**, **terminals** are represented by the **shape_aspect** entity for both electrical and mechanical product data [4, 46]. **terminal** identification has been harmonized to a great extent between the two domains. Determining the appropriate rules for foreign keys that are bound to **terminal** identifiers when the object the **terminal** is associated with changes over the life cycle is one objective of this document. One of the subjects of this research is to determine recommendations in industrial practice for classification of mechanical features in the design phase other than primitive geometric classifications.

4.2.2. ASME Y14.37 Product Definition for Composite Parts

ASME Y14.37 [47] specifies requirements for classification and identification of product features described as **limited length or area indicators** (**LLAIs**). The part of the definition for an **LLAI** critical for this research is:

> An **LLAI** may be used to define a specific region of the part with unique characteristics or requirements. The common requirements for a composite **LLAI** are as follows: (a) Each composite **LLAI** shall have an identifier. (b) Each **LLAI** shall have a specified type (see para. 6.4.1). The type may be specified via

attribute, naming, notes, annotation, native **CAD** feature specific to the **LLAI** type, or other similar method.

It is clear that the requirements for **LLAIs** can be related to [13], but this is a case where the mechanical and electrical disciplines are not yet in alignment.

Example 6. A class of **LLAI** included in [47] is surface preparation area (e.g., secondary bonding, electrical grounding, sacrificial machining). The requirement included is to:

> Specify surface preparation processes and attributes required to meet secondary bonding, electrical conductivity, or mating condition requirements.

The information requirements of the surface preparation area can be supported by the specialization of **characterized_object** that is **feature_definition_with_connection_area** [48] as a definition and applied to a design by either an **instanced_feature** [49] or **placed_feature** [50] found in **STEP**.

The formal interpretation of the **LLAI** information requirements, including providing formal classification structures, into the **STEP** product model is a topic for edition four of **AP 242**.

4.2.3. ASME Y14.41 Digital Product Definition Data Practices

ASME Y14.41-2019 [51] specifies practices for digital definitions of product shape data, including datums and their associated designation.

4.2.4. ISO 5459 Geometrical Tolerancing: Datums and Datum Systems

ISO 5459 [52] specifies the classification of datums and their associated designations.

4.2.5. IEC 61666 Identification of terminals within a system

IEC 61666 [13] specifies a set of general principles for identifying **terminals** of objects within a system. These general principles can be applied to various engineering domains, such as electrical, mechanical and process engineering. The principles can be applied irrespective of technology. Discussions with relevant ASME and ISO committees (ASME Y14, ISO TC 10, and ISO TC 213) indicate there is no standard equivalent to IEC 61666 in ASME or ISO. It is the recommendation of the authors that ASME and ISO adopt IEC 61666 as a reference standard with any needed extensions.

4.2.6. ISO 23952 Quality Information Framework (QIF)

ISO 23952 [7] identifies specific classes of geometric product model, **PMI**, and drawing data objects for inclusion in its **feature** type tree. The **QIF** standard includes a hierarchy of 37 parameterized feature types. Robust provisions for identification of **QIF feature** type instance data are included.

4.2.7. ISO 10303 Product Data Representation and Exchange (STEP)

There is a rich body of feature classification available in **STEP** for electrical design-oriented features, for mechanical manufacturing process-related features and for pre-defined complex hole features such as counterbored holes. For mechanical manufacturing process-related features, **AP 242** provides parametric data and English textual descriptions that enable manual interpretation for application interface development. For pre-defined complex hole features, **AP 242** provides parametric and tolerance data in a cohesive data structure that facilitates interface implementations.

Example 7. The **CAM** geometric application interprets the parametric data at run-time to construct a **CAM** model.

There exist **PMI** classifications in **STEP** for datum-related features.

Example 8. A round hole completely through a part model is specified by a user. The **CAD** system recognizes that the user has populated correct user interface data so that the **CAD** internal object classification of the associated **CAD** topological object is one or more topologically connected faces. The **CAD** system extends the geometric internal model with one or more primitive cylinders. **CAD** internal object classification of the associated **CAD** topological object is one or more topologically connected faces. The **CAD** pre-processor interface for **STEP APs** that support mechanical design would populate:

- an instance of **geometric_item_specific_usage**;
- an instance of **shape_aspect** with an identifier of 'hole 1' auto-generated by the system;
- an instance of **advanced_brep_shape_representation**;
- an instance of **manifold_solid_brep** with an identifier of 'extrude' followed by a numerical value in parenthesis that is auto-generated by the system;
- an instance of **closed_shell** that references with an identifier 'CS' followed by a numerical value in parenthesis that is auto-generated by the system;
- an instance or instances of **advanced_face** with (an) identifier(s) of "AF#" auto-generated by the system;
- an instance or instances of **face_bound** (omitted from Fig. 8 for clarity); and
- an instance or instances of **cylinder** with (an) axis placement(s) and radi(us/i) (omitted from Fig. 8 for clarity).

The instance of **geometric_item_specific_usage** specifies both the instance of **advanced_-brep_shape_representation** and the instance or instances of **advanced_face**, binding them to the instance of **shape_aspect**. NOTE - The auto-generated values listed in the example may vary per system or be user-specific data. Figure 8 provides a graphic illustration of the data set discussed in this example that was extracted from an **ISO 10303-242: Application protocol: Managed model-based 3D engineering (AP 242)** data file.

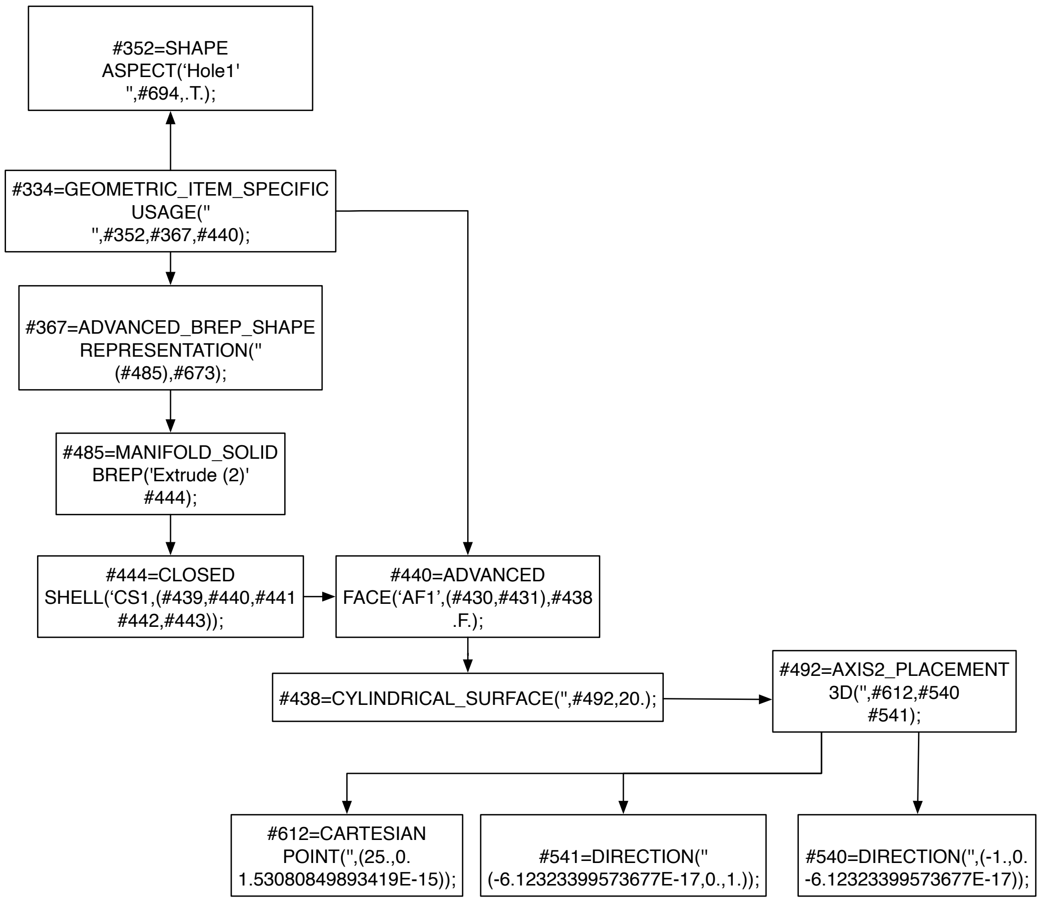

#352=SHAPE ASPECT('Hole1' '',#694,.T.);
#334=GEOMETRIC_ITEM_SPECIFIC USAGE('' '',#352,#367,#440);
#367=ADVANCED_BREP_SHAPE REPRESENTATION('' (#485),#673);
#485=MANIFOLD_SOLID BREP('Extrude (2)' #444);
#444=CLOSED SHELL('CS1,(#439,#440,#441 #442,#443));
#440=ADVANCED FACE('AF1',(#430,#431),#438 .F.);
#438=CYLINDRICAL_SURFACE('',#492,20.);
#492=AXIS2_PLACEMENT 3D('',#612,#540 #541);
#612=CARTESIAN POINT('',(25.,0. 1.53080849893419E-15));
#541=DIRECTION('' (-6.12323399573677E-17,0.,1.));
#540=DIRECTION('',(-1.,0. -6.12323399573677E-17));

Fig. 8. Key entities for Example 8.

Example 9. There is a datum in a design with a designator of ”C”. There is a datum target feature balloon [53] in that design with a designation of ”C1.” ISO 5459 [52] specifies datum targets and their string value classification. That designation indicates the first datum target in support of datum C. The string ”C1” uniquely identifies that target to a human operator. The datum target is associated in the CAD system to a UUID. Because the UUID is bound to a CAD object and not the string ”C1”, an extra processing application is required to address the human-computer associativity gap by using the tuple (type name = datum_target, value = ”C1”) as the name value for the version 5 algorithm.

4.3. Concept Classification and Identification Requirements

A released design **product_definition** provides an approved collection of property data that specifies manufacturing requirements, amongst other data deemed pertinent by the design organization.

Each class of product definition defined by the design organization provides a distinct collection of property data, consistent with the nature and requirements of the class of product the product definition is associated with. For exchanging characteristics, **AP 242** provides several predefined collections of property data and supports user-defined collections of property data. As provided in [54], the following definition for a concept is used in this document to represent an enterprise-defined collection of product data included in a product model.

> A concept is a unit of thought differentiated by a set of characteristics. Consider the concept "person." The characteristics of a person include being designed to stand upright on two legs, the ability to talk, age, marital status, skin tone, and many others. Some characteristics are indispensable for understanding a concept. These are the essential characteristics. A delimiting characteristic is a characteristic used to distinguish it from a generic concept. For example, an essential characteristic of people is they are designed to stand and walk upright. This is also a delimiting characteristic since it distinguishes people from other primates. The intention of a concept is the set of characteristics associated with the concept. The extension of a concept is the totality of objects to which a concept corresponds. A defining characteristic is a characteristic that is representative of objects in the extension of a concept. A defining characteristic of people is that they stand and walk upright. Not every person is capable of walking and standing upright, even though they are designed that way. Paralyzed or injured people may not be able to stand.

Several areas in the **STEP** product model as delineated by relevant **Computer-Aided-"x" (Design, Manufacturing, Inspection) Interoperability Forum** (**CAx-IF**) recommended practice documents [55] informed the decision to include concept as a theme in this research.

Example 10. An electrical functional network design product definition [46] based on the Modified Nodal Analysis (MNA) method [56] is used as a computer-interpretable product model for both design and simulation. No shape properties are included in the product definition.

Example 11. A mechanical design definition includes characteristics:

- directly associated to the product definition;
- directly associated to the product shape;
- directly associated to product physical features; and

- auxiliary items necessary for consistent interpretation by receiving systems.

Example 12. **ASME** Y14.45-2021 [57] provides an optional classification of tolerances for reporting measurement results. A form tolerance of circularity is designated with the string ”CIR.” In **AP 242** a form tolerance of circularity is represented by a **roundness_tolerance**. An implementation generating results would reference a **UUID** for an instance of **roundness_tolerance** as the design source specification for the circularity entry in the **ASME** record.

4.3.1. Formation of a Concept

Characteristics may be grouped or may be arranged in a tree of characteristics to form a discrete concept. Because **AP 242** specifies only the classes of characteristics and properties that may be included in a design definition, an enterprise wishing to have an explicit manifest of engineering requirements would execute a planning application that traverses the design definition to specify the explicit list of properties that will be implemented by manufacturing execution and verified by measurement systems. When the enterprise is using **UUIDs** to support data associativity, the planning application is then required to assign **UUIDs** to relevant properties in the design data to meet organizational quality requirements.

4.4. First Article Inspection

The SAE-AS9102C [58] standard does not separate reporting of feature characteristics from functional and geometric characteristics, combining all forms of characteristics into design characteristics.

The standard provides for user-defined classification of each identified characteristic.

4.4.1. Software to Report PMI in QIF and MTConnect Files

The NIST Smart Manufacturing Systems (SMS) Testbed hosts:

- prior research technical data packages with standard and native format data from design, manufacturing and inspection Technical Data Packages (TDPs) from the SMS Test Bed [59];
- NIST published reference software implementation that produces a first article inspection report from QIF 2.0 data that is available on NIST’s git hub site [60];
- NIST published software that examines QIF PMI data [61]; and
- an accompanying technical publication [62] that describes the process.

4.5. Reference Designation Requirements

Reference designators are identifiers that are visible to a user in a document or model. ISO/IEC 81346 [63] specifies structuring principles and rules for maintaining relationships among objects. In the context of ISO/IEC 81346, objects are conceptual. They are neither **CAD** objects nor **STEP** records. However, they can be mapped to **STEP** ENTITY data types. In functional, product, and location domains, [63] defines and classifies reference designators.

Reference designators may be full, in which case they start at a user-defined root and provide a traversal path to a component in a view, or they may be partial, in which case they only identify a component in the immediate context of the view the component is related to. Reference designators allow a reader to cross-correlate two views of a product definition when each product definition is in a different context. They are represented in **STEP** standards where applicable, in particular in the functional and physical product structure models. Reference designators are required to be human-readable.

A point of caution: reference designations may be changed during the life cycle as product versions are added. This may not happen often, but in a computer system, the reference designations should not be used as keys. It is better to use internal identifiers, entirely hidden from the user of the system.

Partial reference designators include two components: a classification assignment and a numeric value. Full reference designators consist of a list of partial reference designators. The classification assignment is accomplished according to the engineering domain of the component. The numeric values in a specific class are ordered for that assembly. Providing foreign keys [64] in a computer-interpretable and standard form that are bound to design objects and not reference designators is one objective of this document. During the design life cycle the numeric portion of the reference designator varies as the product design is iterated. Once the design definition for that product version is released to an archive, all data, including the reference designator, becomes permanent.

Example 13. A component in a design may have a partial reference designation [63] of A1, where the letter A signifies a class (amplifier) of which the component is a member and the numeral 1 signifies that the component is in the first designated location (upper left in a two-dimensional drawing) in the physical product. The **STEP** ENTITY data type that provides the partial reference designation is an **assembly_component_usage** [4]. In a **STEP** model, the instance of **assembly_component_usage** is assigned a transformation matrix to locate the model of the component in the model of the assembly. The **UUID** assigned to an instance of **assembly_component_usage** during release to archive is:

```
fd43400a-29bf-4ec6-b96c-e2f846eb6ff6.
```

Downstream maintenance and logistics computer-based processes then have the option to use the **UUID** to query information useful for that component without having to provide the full reference designation [63]. The user presented information for that component remains A1 in human-readable documents and models.

Full reference designators would be used to identify location information for a human-readable observation in an installation or complex product.

4.5.1. Designators vs. UUID

In **AP 242**, **UUIDs** are not assigned to designators by **CAD** preprocessor applications; they are assigned to **CAD** objects. The application of **UUIDs** does not eliminate requirements for providing designators because the life cycle for a **UUID** is different from the life cycle for a designator. A **UUID** may be associated with several different designators, each in the context of a different assembly version, over the life cycle of a product.

5. UUID Encoding Requirements and Characteristics

This section discusses **UUID** encoding requirements and characteristics. Further, it discusses types of **UUIDs**, how they are used with human-readable identifiers, and what requirements they address. This section also introduces a requirement that the **UUID** application in **STEP** applies digital signatures to deter tampering with **UUID** instances.

5.1. ISO/IEC 9834-8 Generation of UUIDs and Their Use in Object Identifiers

ISO/IEC 9834-8:2014 [65], Generation of universally unique identifiers (UUIDs) and their use in object identifiers, specifies procedures for **UUID** generation and use in the international object identifier tree under the joint UUID arc. Annex B provides functional requirements for name-based versions of a **UUID**.

> B.1 The name-based **UUID** is meant for generating a **UUID** from a name that is drawn from, and unique within, some namespace. The concept of name and namespace should be broadly construed, and not limited to textual names. The mechanisms or conventions for allocating names from, and ensuring their uniqueness within, their namespaces are beyond the scope of this Specification.
>
> NOTE – In order to avoid recursion problems, name-based **UUIDs** should not be generated from an **object identifier** (**OID**) that ends with a **UUID** which is name-based.
>
> B.2 The properties of name-based **UUIDs** generated in accordance with clause 14 and with a suitably chosen namespace will be as follows:

- The **UUIDs** generated at different times from the same name in the same namespace will be equal;

- The **UUIDs** generated from two different names in the same namespace will be different with very high probability;
- The **UUIDs** generated from the same name in two different namespaces will be different with very high probability; and
- If two name-based **UUIDs** are equal, then they were generated from the same name in the same namespace with very high probability.

5.2. IETF RFC 9562 Universally Unique IDentifiers (UUIDs)

RFC 9562 [66] clause 6.5 provides an algorithm to create a name-based version five **UUID**. ISO/IEC 9834-8:2014 [65] specifies the requirements for the encoding and interpretation of a **UUID**. There are two versions of **UUIDs** considered: random-number-based and name-based. Both versions meet the functional requirement of uniqueness.

5.2.1. UUID Versions and Critical Characteristics

In this section, a brief description for each **UUID** version is provided.

1. Version 1 produces a time-based value represented by Coordinated Universal Time (UTC) as a count of 100-nanosecond intervals since 00:00:00.00, 15 October 1582 (the date of Gregorian reform to the Christian calendar).
2. Version 2 is specific to **DCE** [67] security and is not considered further.
3. Version 3 uses a namespace and name. A compliant implementation produces the same result each time the same name in the same namespace is transformed. Version 3 uses the **MD5** hash function [68].
4. Version 4 produces a random value each time the algorithm is executed. Version 4 is used in commercially available product data exchange interface implementations.
5. Version 5 uses a namespace and name. A compliant implementation produces the same result each time the same name in the same namespace is transformed. The namespace identifier is itself a **UUID**, and any desired **UUID** may be used as a namespace designator. Either **MD5** or **SHA-1** [69] may be used but if backward compatibility is not an issue, **SHA-1** is preferred.

5.3. Data Tampering

Robust implementations of data exchange and sharing require capabilities in the data set and exchange agreements to detect data tampering [70]. The **UUID** standards do not address data tampering.

5.4. UUID registration authorities

Resolving **UUIDs** may involve establishing one or more registration authorities [71] for the parties participating in an exchange agreement. Establishment of a registration authority was not considered in this research.

6. Current UUID Implementations

This section describes **UUID** implementation in product data standards in the inspection domain and in the architecture and construction domain's Industry Foundation Classes [12].

6.1. UUID Implementation in QIF

The **QIF** file contains a QPId **XML** record that is required to include a **UUID** to identify the **QIF** file.

NOTE - The term QPId will be replaced with **UUID** in **QIF** 4.0.

QIF requires that each item record must include a local record identifier represented as an INTEGER that is unique in the context of the file. The combination of the pair (document QPId, local record identifier) is sufficient to serve as a foreign key for the item record. Alternatively, an item record may include a **UUID**, obviating the need for evaluating the combination of (document QPId, local record identifier). The **QIF** document [6] has the following definition:

> The QPIdReferenceType defines the text representation of the universally unique identifier described in the standard ISO/IEC 9834-8. As a number, it has 128 bits. As a text string, it is represented by 32 hexadecimal digits displayed in five groups separated by hyphens in the form 8-4-4-4-12, for a total of 36 characters, including hyphens, as shown below.

```
fd43400a-29bf-4ec6-b96c-e2f846eb6ff6
```

6.2. UUID Implementation in IFC

Industry Foundation Classes [12] classifies the information model into:

- core data schemas;
- shared element schemas;
- domain specific schemas; and
- resource definition schemas.

Each entity defined in core data, shared element, and domain specific schema may provide a **UUID** for identification, in addition to any human-readable identification information. The entities defined in the resource definition data schema do not include the capability to provide **UUIDs**. The IfcPropertyDefinition inherits GlobalId: IfcGloballyUniqueId from IfcRoot. The IfcVertex does not inherit GlobalId because it is not an indirect subtype of IfcRoot. The following quote is from clause 8 of Industry Foundation Classes 4.0 [12]:

> Entities and types defined in this layer can be referenced by all entities in the layers below. Unlike entities in other layers, resource definition data structures cannot exist independently, but can only exist if referenced (directly or indirectly) by one or more entities deriving from IfcRoot. As resource definitions do not have a concept of identity (such as a **GUID**), multiple objects referencing the same instance of a resource entity does not imply a relationship. For example, two polylines (IfcPolyline) sharing the same instance for a point (IfcCartesianPoint), and two polylines using different instances for identical points (such as both having coordinates 0,0,0) are semantically equivalent. It is recommended (but not required) for applications to minimize file size by sharing identical resource definition instances where possible.

7. Recommendations for QIF

It is the recommendation of the authors that the **QIF** standard be updated so **UUID**s will support version 5 namespace **UUIDs** for any use case where applications share a model and jointly update it. The rationale for the recommendation is that data that is unchanged in an iterative model should not be assigned a new **UUID**.

8. Aggregators in STEP

Aggregators are of interest in this research because they potentially define the scope of a change in a product data set. The **product_definition** [4]ENTITY data type is a domain-specific view of a version of a product that provides an implicitly defined aggregation of property-related data. Reference [72] provides an aggregation methodology that relies only on the explicit tree structure of the **STEP** data set.

The implicit nature of product_definition requires additional contextual information to be supplied for post-processors to perform coverage and design intent quality checks as the EXPRESS language rules do not prohibit incorrect interpretation of domain process standards by users. The **representation** ENTITY data type provides an explicitly defined aggregation of property-related data. The explicit nature of representation allows post-processors to perform some level of semantic checks on the content without external specifications, but because some geometric modeling requirements are not formally specified for full verification of geometry, special purpose processors are employed that have been implemented with support from geometric modeling experts. For example, several of the Euler-Poincare rules are specified in lexical form in informal propositions in **STEP**.

The **product_definition_formation** [4] aggregates several **product_definition** instances to represent a version of a product. The **product** [4] aggregates several **product_definition_formation** instances to represent a product. Neither **product_definition_formation** nor **product** are designed to directly aggregate property-related data. One exception is catalogue data which may be directly related to a product.

9. Limitations of Existing Identifiers in STEP

The authors examined existing identifiers in **STEP**, looking for available data slots to apply **UUIDs**. The following **STEP** identification-related entities were reviewed:

- **id_attribute**;
- **name_attribute**;
- **aggregate_id_attribute**;
- **applied_identification_assignment**; and
- **applied_name_assignment**.

The product_definition.id attribute (created by an instance of **id_attribute** that references the product_definition) has several **AP**-specific mandatory string values for correct interpretation of the received data for **AP** conformance obviating its use as a generic solution. The representation.id and representation.name attributes are used as data slots for predefined classification strings as part of the **STEP** interpretation process [73] of interpreting domain concepts into the **STEP** product information model and are not available as a generic solution. This eliminates **id_attribute** and **name_attribute** as general solutions. It is possible to classify an identification-related entity as a **UUID**-related identifier using **applied_identification_assignment** [4], but that does not address the aggregate issue and adds extra data to the data set, so that option was not pursued in this research.

10. Design Options

This section discusses design options for extensions to the existing **STEP** product model.

10.1. Namespace Attribute

As noted in Sec. 5.2.1, the namespace attribute is, itself, required to be a **UUID**. This document does not recommend including the namespace in the exchange data set; the type classification of the **uuid_attribute** identifies whether the uuid is a version 4 or version 5

UUID. Declaring that the **UUID** is version 5 does permit an importing application to determine that the imported **UUID** is not owned by the importing organization.

10.2. Name Attribute

The name attribute required for version 5 can be any string determined by the authoring **CAD** system that is unique in the context of the source **CAD** model. This document does not support including the name in the exchange data set.

10.3. Tampering with Data

Because **UUIDs** are explicitly present in the data set, applications could import the data set, modify the applied instances for a **UUID**, and then export the file as a valid file. To avoid that, this document recommends adoption of digital signature technology as required in ASME Y14.17-2019 and described in [74].

10.4. Detecting Changes in Data Sets

UUIDs provide opportunities for improvement in detecting atomic changes in a **CAD** model. A simple search can detect added or deleted **UUIDs**. However, a comprehensive comparison is still needed to identify changes in the **CAD** objects identified by the retained **UUIDs**. A way to detect changes in data sets is to apply the Merkle tree binary hash tree methodology [75, 76]. A brief illustration of the use of the binary hash tree methodology to compare two files:

- the root hash values of the trees are compared.
- If the root hash values are the same, then the files are identical.
- If the root hash values are different, then the trees can be traversed to identify the specific data values that differ between the two files.

Efficient implementations of Merkle trees are commonly available [77, 78]. An alternative method for detecting changes in a **STEP** data file is defined in [72].

10.4.1. Leveraging the Scope of an Object to Improve Efficiency

The scope of an object may be used to improve the efficiency of comparison by reducing the number of queries necessary to detect a change. **STEP** uses an object existence dependent model (i.e., one instance is explicitly dependent on another as specified by an attribute or constraints in the ENTITY declaration) [79]. In such cases a query can be performed to identify each object that is directly or indirectly dependent on the object selected as root for the query. **STEP** also supports establishing existence dependency using what is known as directed relationship entities. The directed relationship entities relate two instances where the existence of the instance on the related end of the relationship is dependent on the instance on the relating end of the relationship. There are a few cases in **STEP** where the relationship is not directed. In those cases, a recommended practice should be created to provide implementation guidance.

10.5. STEP Component-Related Objects that Include Reference Designators

Because a partial reference designator identifies a real component in the context of an assembly, multiple data elements are usually associated to that one designator when assembly shape data is provided. The **STEP** ENTITY data type is **assembly_component_usage** [4] and its sub-types. Uniqueness of a partial reference designator is only provided by the next_assembly_usage_occurrence [4] sub-type when the attribute reference_designator is provided and populated in conformance to **AP 242**. However, the **assembly_component_usage** ENTITY data type is an implicit aggregation data type, requiring post-processors to execute queries to ascertain the extent of data provided in the exchange data set.

10.6. STEP Feature-Related Objects that Support Designation

Individual piece part definitions include functional and shape data for their features. Features of components in an assembly have complex data associated with them specific to the particular assembly, including:

- location dependencies on the location of the component they are associated with and the network to which they may be connected;
- invariant data inherited from the definition object for that component; and
- data values specific to the component feature where the data variable is specified in the definition object.

NOTE - Feature designator representation rules for data exchange are not harmonized across **STEP** standards, IEC TS 62771:2012, and IEC 61666. Harmonization or mapping to IEC TS 62771 is out of scope of this document.

All **STEP APs** that support part features use the **shape_aspect** [4] ENTITY data type to denote a feature of a part and the application of that feature in an assembly context. The **shape_aspect** id and name [4] attributes support the feature designation functionality.

10.7. STEP Generic Characteristic Objects and Designation

There are four ENTITY data types that are generic characteristic objects:

- **characterized_object**;

- **general_property**;
- **property_definition**; and
- **representation**.

The **characterized_object** [4] ENTITY data type is used as a definitional concept where the definition is not the subject of commerce because it is independent of the **product** entity. The **product_definition** is the other major definitional concept in **STEP** and is the subject of commerce because of its dependency on the **product** entity data type. A major use case for the **characterized_object** is as a definition for a **shape_aspect** in an occurrence context.

The ENTITY data type **characterized_object** has no predefined designator in an **AP**, but an **AP** may specify predefined sub-types that are predefined classes of designators. The **characterized_object** is an implicit aggregation data type, but an **AP** may provide domain-specific subtypes of **characterized_object**, along with a specific collection of name-value property pairs that are included in a related representation to satisfy domain requirements. A query must still be performed to identify the contents of the aggregate so specified.

The **general_property** ENTITY data type has no predefined designator in an **AP**.

The **property_definition** ENTITY data type is widely applied in **AP** specific interpretations with the attendant abundance of predefined domain-specific designators.

The **representation** ENTITY data type is widely applied in **AP 242** for geometric modeling and for manufacturing specific applications that require name-value pairs for specific manufacturing plans. There are an abundance of predefined domain specific designators.

A review of **AP 242** recommended practices did not identify additional generic characteristic objects that include a predefined designator.

10.8. STEP Objects that Qualify as a Concept

The following **STEP** ENTITY data types qualify as a concept as defined in 4.3:

- **product**;
- **product_definition_formation**;
- **product_definition**;
- **property_definition**;
- **shape_aspect**; and
- **characterized_object**.

Implementations may find it useful to apply Merkle tree structures to the aggregations related to those ENTITY data types. Many application objects defined in an **AP**'s **application reference model (ARM)** (a requirements model specified in domain terminology) may qualify as concepts. Classification of application objects as being concepts is beyond the scope of this document because **ARMs** are unique to a given **AP**.

11. Proposed Model Summary

This section provides an overview of the proposed information model to address the previously described requirements for **UUIDs** in **STEP**. Entities and types proposed for inclusion in **STEP** are depicted in **EXPRESS-G**, the graphical notation for the **EXPRESS** language. Figure 9 is an **EXPRESS-G** information model depicting key entities, types, and relationships for the proposed **UUID** attribute model.

The proposed model includes the following high-level benefits:

- The name-based **UUID** version meets the functional requirement of the engineering domain requirement for repeatability;
- The random-based **UUID** version provides flexibility in implementations that require it;
- Including relationship ENTITY data types specific to **UUID** support traceability and associativity;
- Including the proposed models directly in the **STEP** information model provides that the **STEP** data set is not limited to a specific data representation format;
- The type hierarchy of the proposed model provides for future type to be added in an upwardly compatible manner; and
- Actors that assert ownership of elements in the data set can be distinguished as each actor has a unique namespace.

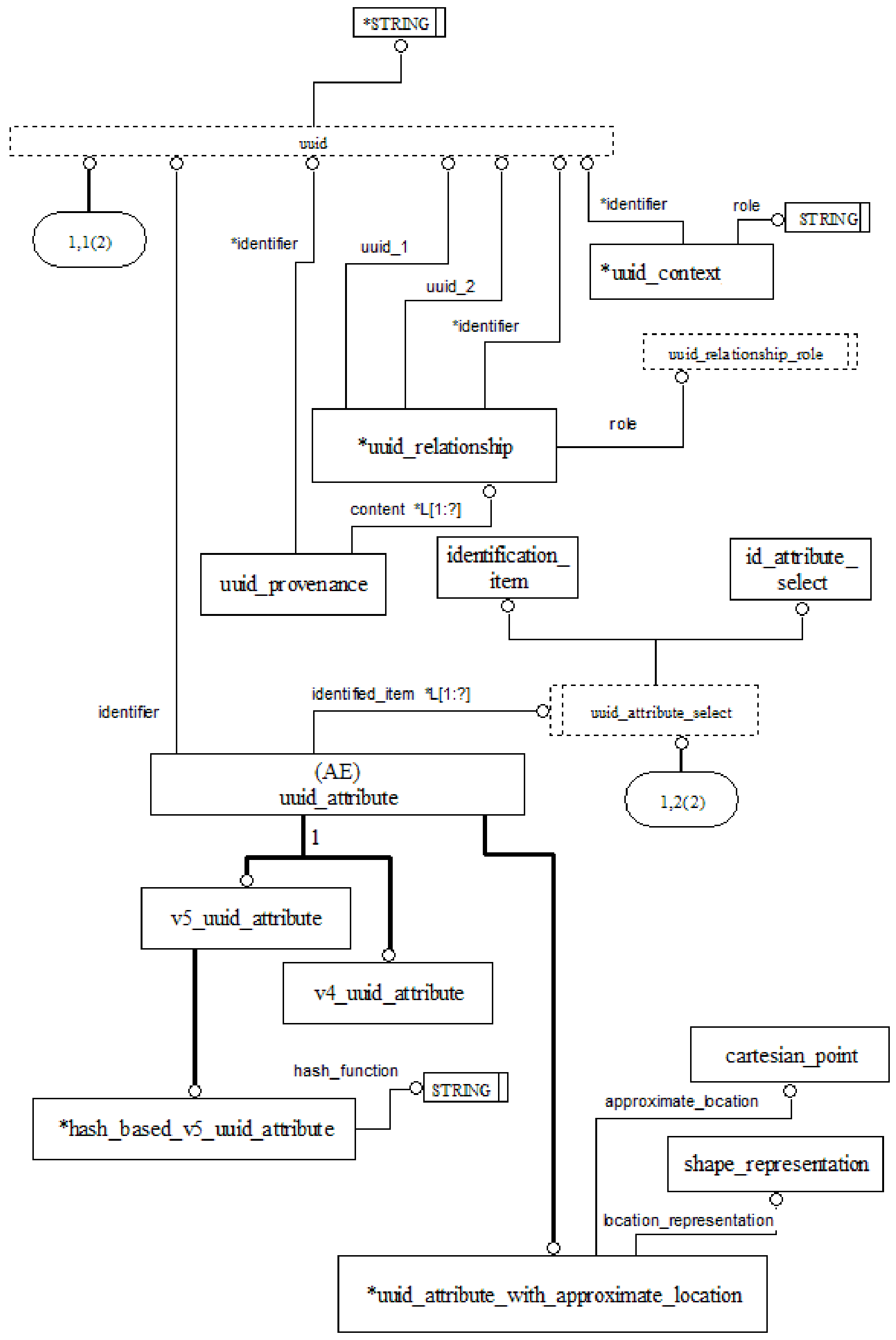


Fig. 9. EXPRESS-G model of uuid_attribute.

The proposed **EXPRESS-G** information model to support hash tree operations is provided in Fig. 10.

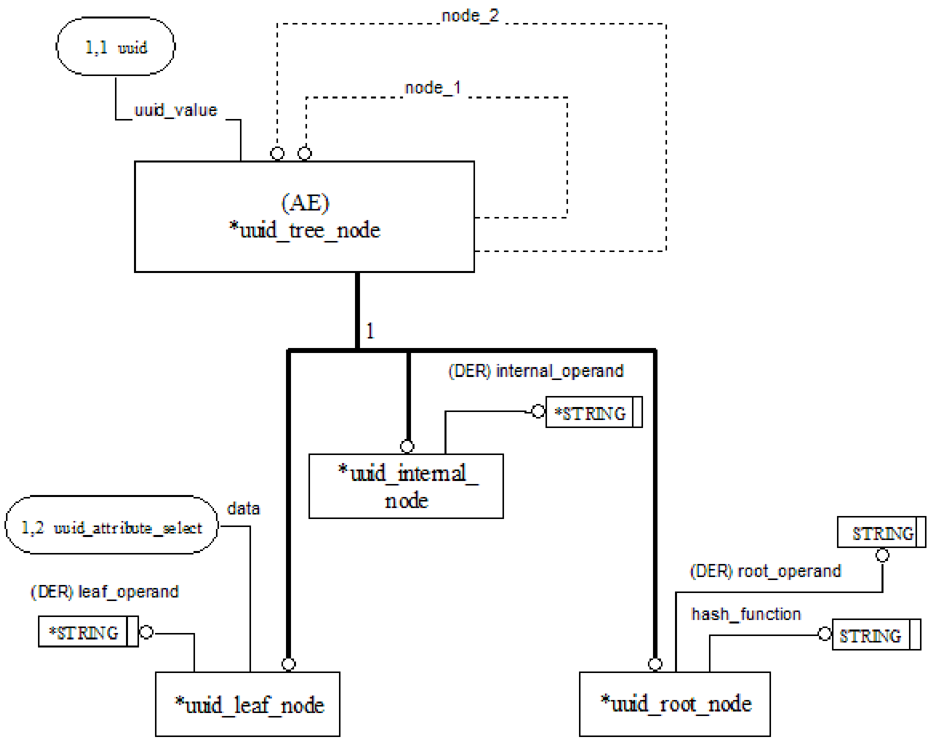


Fig. 10. EXPRESS-G model of the hash tree.

The remainder of this section describes:

- recommendations proposed in this research for implementing **UUIDs** in different contexts;
- recommendations for **UUID** encoding structure;
- recommendations for Merkle tree structure; and
- the purpose of key entities and types in the proposed model.

11.1. Recommendations for Specific Contexts

This subsection provides recommendations for UUID use in three different contexts: mechanical feature designations, **STEP** design **APs**, and digital twins.

11.1.1. Recommendations for Mechanical Feature Designators

The use of the **ARM** application object name as the class of the designator and an integer starting with 1 as the id of the designator is recommended.

This is in contrast with **QIF**, where the record identifiers increment over the entire exchange data set irrespective of the record type.

11.1.2. Recommendations for UUIDs in STEP Design APs

The recommendation for **STEP APs** that include engineering design in their scope (so-called design APs) is to use the name-based **UUID** version. **AP 238** includes **UUIDs** in data set anchor sections conforming to ISO 10303-21:2016 anchor section approach described in Sec. 2.1. This is identified as version 5 in [65] and in [66].

11.1.3. Recommendations for Digital Twins

In the scenario described in Sec. 2.1 of this paper, a software application would iterate over the design model and identify which of the **UUIDs** to provide for shop-floor data management in the form of a manifest file, which may be either an external XML file or an integral part of the **STEP** Part 21 file. Manifest data included in the data section of the **STEP** Part 21 file would be delimited by comments to maintain compatibility with existing **STEP** post-processors.

Example 14.

This manifest file contains an anchor section with references to specific **STEP** records in the data section. The manifest file is generated by an application external to the design **CAD** application.

The application object Linear_distance_dimension [26] represents a concept because it aggregates multiple **AP 238** records to form a cohesive content model.

```
ANCHOR;
<1e516d3d-ce7a-47af-82d5-29a8f2ad3b8b>=#3245;
/* Position.1 - geometric_tolerance_with_datum_reference_-
and_geometric_tolerance_with_modifiers_and_-
position_tolerance */
<5f2b0ccf-7220-4546-bcff-70adb18d4f6b>=#2768;
/* linear distance - dimensional_location */
<626851cc-451c-49cb-8de8-6dcdeb617309>=#2773;
/* linear distance - dimensional_location *

...
/**************************************************
 * Application object: LINEAR_DISTANCE_DIMENSION (#2773)
 * DIMENSION_VALUE_TOLERANCE: #2773, #2774, #2775, #3259
 * ASSOCIATED_DRAUGHTING [*]: #2773, #2776, #5701
 * TARGET: #2773, #5994
 * ORIGIN: #2773, #6191
 * PLUS_MINUS_LIMITATION: #2773, #2777, #3335
 * ID: #2773, ['linear distance']
 * DESCRIPTION: #2773, ['Dimension.2']
 */
#2773=DIMENSIONAL_LOCATION(
    'linear distance',
     'Dimension.2',
     #6191,
     #5994);
#2774=DIMENSIONAL_CHARACTERISTIC_REPRESENTATION
    (#2773,
     #2775);
#2775=SHAPE_DIMENSION_REPRESENTATION(
     '',
     (#3259),
     #754);
#2776=DRAUGHTING_MODEL_ITEM_ASSOCIATION(
    'PMI representation to presentation link',
    'Linear Size.2',#2773,#5657,#5701);
#2777=PLUS_MINUS_TOLERANCE(
    #3335,
    #2773);
```

11.2. Recommended UUID Encoding Structure

The Version 5 **UUID** uses two arguments: domain key and name string. A **UUID** value is dependent on the combination of domain key and name string. Neither of those arguments is included in the exchange data set. It is the responsibility of the originating enterprise to maintain a registry of domain keys and name strings. The domain key is typically maintained by the enterprise for its sub-domains [80, 81].

The name string is proposed to be the internal **CAD OID** or an unambiguous path to the internal **CAD** object from the internal root object in the software application model. The **UUID** proposed herein is a 36-character string which supports the requirements of version 5 encoding specified in [19, 65, 66]. It also supports the version 4 encoding specified in those recommendations and standards.

11.3. Recommended Merkle Tree Structure

The recommended structure is adopted from the industrial proposed **THEX** standard that has been interpreted [73] as part of this research and is represented as an **EXPRESS** model.

11.4. Proposed New Entities for the STEP UUID Attribute Information Model

Figure 9 is an illustration of the entities and types in the proposed **UUID** attribute model. Key entities are discussed in the following subsections.

11.4.1. Proposed uuid_attribute Entity

The **uuid_attribute** ENTITY assumes an assignment of the tuple (**UUID**, namespace, name) to one or more target instances and provides the **UUID** value calculated based on that tuple as input. The **uuid_attribute** shall be instantiated as one of its subtypes **v4_uuid_attribute**, **v5_uuid_attribute**, optionally in combination with **uuid_attribute_with_approximate_location**. The **uuid_attribute** identifier attribute complies with [65] and [66]. An informal proposition in the entity documentation will require implementation conformance. A unique rule will be included to ensure that there is no more than one member of **uuid_attribute** with a specific identifier included in a data set, preventing multiple assignments of data items to the same **UUID**.

11.4.2. Proposed uuid_relationship Entity

The **uuid_relationship** ENTITY provides the capability to build a graph of **UUID**s where the **uuid_relationship** is conceptually an edge and **UUID**s are conceptually vertices. A role attribute is provided as an enumeration for centralized management in the ISO TC 184/SC 4 WG 12 development process. The proposed values are derive_from, merge, similar_to, same_as, split, and supersedes. The values are managed by the ISO TC 184/SC 4 WG 12 ballot process. An application protocol would define the detailed interpretation for same_as. A **UUID** is provided as an identifier so that the **uuid_relationship** records may be referenced in a graph of **UUIDs**.

Example 15. Two instances of **uuid_relationship**, each with the role of ”merge” and each specifying the same target **UUID**, are provided to indicate that two instances may be merged into one instance. There is nothing in the information model that prohibits more than one **uuid_relationship** from specifying the same target **UUID**.

Example 16. A member of **uuid_relationship** with a role of ”same_as” is provided to indicate that an instance is the same as another instance.

Example 17. Two instances of **uuid_relationship**, each with the role of ”split” and each specifying the same source **UUID**, are provided to indicate that one instance has been split into two instances. Any product data set that is the result of a merge or split is required to be conformant to the relevant **AP**.

Example 18. A member of **uuid_relationship** with a role of ”supersedes” is provided to indicate that an instance replaces another instance.

11.4.3. Proposed uuid_provenance Entity

This ENTITY is proposed to record the sequence of **UUIDs** that may pertain to a process chain.

11.4.4. Proposed v4_uuid_attribute Entity

This ENTITY is proposed to support exchange of a version 4 **UUID**.

11.4.5. Proposed v5_uuid_attribute Entity

This ENTITY is proposed to support exchange of a version 5 **UUID**.

11.4.6. Proposed uuid_context Entity

This ENTITY is proposed to provide additional contextual information for a **UUID**.

11.4.7. Proposed uuid_attribute_with_approximate_location ENTITY

This ENTITY is proposed to support exchange of additional geometric information generated by a downstream application.

11.5. Proposed New Entities for the STEP Hash Tree Information Model

Figure 10 is an illustration of the entities and types in the proposed hash tree model. A hash tree provides a standard approach for collecting random data into a uniform structure. This provides maximum flexibility for enterprises to manage storage and retrieval of **STEP** data subsets.

11.5.1. Proposed uuid_tree_node Entity

This ENTITY is proposed to support exchange of hash tree nodes that are providing a **UUID** as the hashed value.

11.5.2. Proposed uuid_leaf_node Entity

This ENTITY is proposed to support exchange of a hash of product data independently of any population of the **uuid_attribute** and provides a **UUID** as the hashed value.

11.5.3. Proposed uuid_internal_node Entity

This ENTITY is proposed to support exchange of an internal node in a hash tree and provides a **UUID** as the hashed value.

11.5.4. Proposed uuid_root_node Entity

This ENTITY is proposed to support exchange of a root node in a hash tree and provides a **UUID** as the hashed value for the tree.

12. How the Proposed Model Addresses Requirements

This section describes how the proposed model in Sec. 11 addresses the requirements described in the use cases from Sec. 3.

12.1. Model Applied to Design to Manufacturing Release Use Case

The **uuid_attribute** and **hash_based_v5_uuid_attribute** can be applied to all relevant **STEP** product data entities in a data set. Refer to Section 14 for a proposed list of entities. For the use case of a design revision, regeneration of **UUID** values are facilitated because the **UUID**s' values are required to be independent of the (product, product version, product definition) tuple that is part of the fundamental **STEP** product data. It is critical for implementations to omit explicit design version data from the **UUID** calculation input data.

In this example a **UUID** is applied to a tolerance where the tolerance class is changed.

Example 19. The initial iteration of a design (Revision '-') of a part has a hole number 33 with a cylindricity_tolerance instance value of 0.01 mm. In the design pre-processor the cylindricity_tolerance instance is given a **UUID** of '6ba7b810-9dad-11d1-80b4-00c000000000'.

```
#1=PRODUCT_DEFINITION_FORMATION('-',..,..);
#2=SHAPE_ASPECT('hole_33',,);
#10=CYLINDRICITY_TOLERANCE(#2,..,..,0.01MM);
#11=UUID_ATTRIBUTE(#10, '6ba7b810-9dad-11d1-80b4-00c000000000');
```

In Revision A, the hole cylindricity_tolerance instance value is changed to a position_tolerance instance with a value of 0.005 mm. In the design preprocessor the position_tolerance instance is given a **UUID** of '7ba7b810-9dad-11d1-80b4-00c000000000'. In the **CAD** source model records the **CAD OID** entry for cylindricity_tolerance for hole 33 is marked as 'deleted' (omitted from example).

```
#16=PRODUCT_DEFINITION_FORMATION('A',..,..);
#22=SHAPE_ASPECT('hole_33',,);
#100=POSITION_TOLERANCE(#22,..,..,0.005MM);
#110=UUID_ATTRIBUTE(#100, '7ba7b810-9dad-11d1-80b4-00c000000000');
```

Furthermore, in the same Revision A data set, the **uuid_relationship** instance would be populated with

```
#1=UUID_RELATIONSHIP(
'00000000-9dad-11d1-80b4-00c000000000',
'6ba7b810-9dad-11d1-80b4-00c000000000',
'7ba7b810-9dad-11d1-80b4-00c000000000',
'supersedes',$);
```

The user organization's quality process establishes traceability requirements for characteristics. User and implementation forums are encouraged to define exchange agreements for appropriate application of the capabilities of the **UUID** ENTITY data types.

12.2. Model Applied to Shared Design Collaboration Use Case

This section is an extension of 12.1. This use case employs **STEP** change management entities to support an iteration process with the **uuid_relationship** entity specifying the product definition state transition based on changes/additions/deletions of product data instances with a **UUID** attached. An exchange agreement would be needed to identify the class of coordination involved in the collaboration.

Example 20. In this case each **OEM** uses the same application and configuration of that application. But the namespace **UUID** reflects the context of the **OEM**. The key constraint is that each application instance protects the integrity of the internal **CAD OID**s as the data is evolving. Recall that the benefit of the version 5 **UUID** is that the retained **OID**s generate the same **UUID** upon export. New **UUIDs** are only created when new **OIDs** are created. Recall that the **OID** generation process cannot depend on the iteration or revision identifier of the **CAD** file being generated. A highly interactive collaboration would bypass the **OEM PLM** system and enable direct **CAD** to **CAD** file transmission. In this example, both **OEMs** are modifying topology. A hole is added by **OEM** Two, but in a different area of the design so a new **uuid_attribute** would be provided for the additional hole.

12.3. Model Applied to Design and Measurement Collaboration Use Case

The scope of **PMI** must be adequately covered with **UUID** attributes that are generated by the authoring **CAD** application. In the design and measurement collaboration use case:

- engineering delivers a design model;
- manufacturing prepares a manufacturing plan (omitted from the scenario);
- manufacturing executes a manufacturing process according to that plan (omitted from the scenario);
- manufacturing prepares a measurement plan; and
- manufacturing provides the measurement results back to engineering noting any discrepancies of interest.

Because **AP 242** supports as-planned and as-measured data exchange, an intermediary processor could generate **AP 242**-compliant data to feed back into the design system. The receiving design system then interprets that data and generates a change request which is fed to the design department. Eventually a command is issued to the **CAD** system to change a property. The **CAD** system processes the change request and updates the model. **AP 242** edition 3 does not include issue exchange; issue exchange is being added to **AP 242** edition 4.

12.4. Model Applied to Product Version Comparison Use Case

The key entities in a comparison of two different versions of a product or of two different products are **product_definition_formation_relationship** and **product_definition_relationship**. Those entities provide a context for the comparison. In the case that the characteristics in the new design are copied from the prior design, existing **UUID** values would be populated in the new design, as **UUID** values are not a function of the design version. For each characteristic that is derived from an existing characteristic and traceability back to the existing characteristic is required, a population of **uuid_relationship** with a role of ”derived” could be provided, where the **uuid_relationship** specifies both the existing and new **UUIDs**. If a new characteristic is derived from multiple existing characteristics, there would be multiple **uuid_relationship** instances provided with the role of ”derived”.

12.5. Model Applied to Transfer Product Design Ownership Use Case

This use case identifies the need for enterprises to define domain key (see 11.2) management requirements to account for design ownership migration. Enhancements to **AP 242** or to recommended practice documents or to exchange agreements may be considered.

12.6. Model Applied to Test and Design Collaboration Use Case

The proposed model includes the ability to add **UUID** data to the ENTITY data types identified in Sec. 3.6 so that interoperability with applications conformant to **QIF** and **MTConnect** may be attained if desired.

13. Proposed Application Behavior

This section provides recommendations for application behavior when implementing the proposed model.

13.1. Creating Application

This document recommends that all **CAD** applications that claim conformance to **STEP** shall generate **UUIDs** and that the same **UUID** shall be regenerated for unchanged **CAD** objects. The internal identifier for the **CAD** object is mapped to the **UUID**. Property changes to the **CAD** object other than its internal identifier are ignored in calculating the **UUID**.

13.2. Importing Application

This document recommends that all **CAD** applications that claim conformance to **STEP** shall import **UUID**s and maintain their relationship to the product data established in the data set received. A mapping table is recommended to persistently store the external **UUID** relation to the internal **CAD OID**. For downstream applications, this document recommends that all applications shall import **UUID**s and maintain their relationship to the product data established in the initial data set received. Downstream applications that generate data derived from the imported data must specify the relationship in the exported data to the imported **UUID**. In a digital twin application, the term downstream merely implies the receiving system.

13.3. UUID Namespace Management

Each application that creates **STEP** data sets must create a namespace **UUID** for their application in the context of the using organization. The details of that transaction are outside the scope of this document. The only requirement proposed for namespace in this document is that the namespace be registered internally in an enterprise. There is no mechanism proposed in this document for exchanging namespaces.

13.4. UUID Name Management

Each application that creates **STEP** product data sets must map internal data structures to a name used to create a **UUID**. For simplicity of reference, this document refers to a **CAD OID** as being the internal representation of the name used by the **UUID** version 5 algorithm. Cases exist where **OIDs** do not exist; it is expected that the would use the full path from root to object as being the internal representation of the name used by the **UUID** version 5 algorithm. The only requirement recommended by this document is that the assignment method be repeatable.

Example 21. This example follows the recommendations of [66] by providing a **UUID** as input to the version 5 algorithm as a complete illustration. The enterprise provides an enterprise level UUID as a seed 'namespace' similarly to the default 'namespace' noted in [66]. In this example, the data set filename and **CAD** system identifier are concatenated to provide an input to the version 5 algorithm to create a **UUID**. That **UUID** is then considered to be the 'namespace' value for input to the version 5 algorithm that calculates the **UUID** for the **CAD** object. That 'namespace' **UUID** for the data set filename and **CAD** system identifier for the example is e00108f0-b388-5999-bdcc-033c5e0b2203. This example assigns the **CAD** object id value to the 'name' input of the version 5 algorithm to calculate the **UUID** value for the single **CAD** object use case.

To generate the **UUID**, newlines were removed and each space was converted to a dash. In the data illustrated below, white space was added for clarity.

```
Internal enterprise data:
Filename       : "the  CAD  FILENAME"
CAD System     : "FreeCad version 2.3.4"

Internal CAD model data:
Type           : "datum"
CAD  object ID : "12345"
value          : "A"

Resulting STEP data:
    #33=DATUM('','',$,.F.,'A');
    #44=V5_UUID_ATTRIBUTE(0d8c2a8f-0bcd-59c6-8b39-20aa4e958adf,
        (#33));
```

Example 22. This example extends the previous example by including a **datum_target** and **shape_aspect_relationship** to form an ordered collection that includes the datum in the previous example.

```
Internal CAD model data:
Type           : "datum_target"
CAD  object ID : "4567"
value          : "A1"
target         : "12345"

Resulting STEP data:
    #33=DATUM('','',#1,.F.,'A');
    #35=DATUM_TARGET('','',#1,.F.,'A1');
    #36=SHAPE_ASPECT_RELATIONSHIP('','',#33, 35);
    #44=V5_UUID_ATTRIBUTE(
        d6295c16-e110-5a99-9941-16baabf28480(#35,#36,#33));
```

13.5. STEP Object Management

This proposed model provides the ability to assign a **UUID** to a single **STEP** object or to a collection of **STEP** objects. In some cases, a single **CAD** object may map to a collection of **STEP** objects. Exchange agreements or application protocols will be required to establish object ownership. Examples include:

- data set ownership where the data set can be identified and controlled as a single thing, e.g., a zip file;
- ownership of the product for the data set;
- ownership of the product version for the data set;
- ownership of the product definition for the data set;
- ownership of a feature for the data set;
- ownership of a tolerance for the data set; and
- ownership of a feature and all the properties associated with that feature for the data set.

Each organization should assume that if they don’t own the object, they shall not modify the object. They should instead propose changes to the object.

13.6. Existence Dependence

STEP product data set instances are existence dependent. That is, if an instance is dependent on a second instance and that second instance is deleted, the first instance is also deleted.

13.7. UUID Reuse Prohibited

A **UUID** may represent the complete path to the **CAD** object from the design root with a unique key assigned for that path stored in the internal **CAD OID**. When a **CAD** object is deleted, that path is deleted in the authoring software and the internal mapping table entry for the **UUID** is noted as deleted so as to prevent reuse. When the **UUID** represents a specific **CAD OID** independent of the path, similar behavior is expected.

13.8. UUID Management in Context of Data Set Revisions

We propose that a pre-processor that processes a revision to a design shall regenerate **UUID**s that are identical to those in the previous design where the **CAD** object existence is not impacted by the revision.

Example 23. An enterprise decides to track each occurrence in an assembly with a **UUID**, and decides to track each placement transform associated with an occurrence with a **UUID**. A partial reference designation of ”A1” for an amplifier occurrence in a design is represented by the attribute reference_designator on an instance of the **AP 242** entity **assembly_component_usage**. There is also an instance of the **AP 242** entity **component_dependent_shape_representation** that provides the placement transform in its items attribute to bring the shape of the amplifier into the shape of the assembly. For convenience, the authoring application may choose to assign the amplifier occurrence and the transform to a Merkle tree or to use the aggregate option for the **UUID**.

Example 24. A **datum_system** indirectly references a situation feature [52] that is a plane. A design change causes the plane to change to an axis. The **datum_system CAD** object is deleted and replaced with a new instance in the authoring system because of the change to an axis. Therefore the **UUID** associated with that instance of **datum_system** would not be reused when the revised design was saved and an updated **STEP** data set created. If the provenance traceability for **datum_system** is a critical characteristic in the enterprise, a population of the **uuid_relationship** with ’supersedes’ could be provided to indicate the specific **UUID** that was the replacement. Alternatively, the design change management functionality in **AP 242** could be employed to indicate more complete revision information.

Example 25. A **datum_target** is included in a design. The enterprise identifies that the position characteristic of the **datum_target** is a critical characteristic for downstream applications, and a **UUID** is assigned to the position characteristic of the **datum_target** during initial file save. The **datum_target** is moved in an iteration of the design. Because the position characteristic is not removed, the **UUID** is not updated, forcing the receiving system to audit the updated design for changes. Application of a Merkle tree may reduce the complexity of the audit.

13.9. Internal Pre-Processor Export Rules

These rules are in addition to the rules inherited from the **UUID** management in the context of data set revisions above. An individual annotation cannot be merged or split during translation in order to preserve the consistency of the associated **UUID**. Any individual geometry instance that is tagged with a **UUID** must be preserved during translation. The application must export any preserved **UUID** in the model. The application must export any created **UUID** in the model. The decision of what **CAD** objects to assign **UUIDs** to is implementation dependent. We recommend that exchange agreements are executed to formalize the selection of **CAD** objects.

13.10. External Pre-Processor Export Rules

These rules are in addition to the rules inherited from the **UUID** management in the context of data set revisions above. The application must export any preserved **UUID** in the native **CAD** model. The application must export any **UUID** assignment of **CAD** objects specified for the translator configuration. Applications may choose to use a configuration file to specify the types for which **UUIDs** shall be created.

13.11. Post-Processor Import Rules

The application may create a **UUID** if it does not exist on import. The application must preserve incoming **UUID**s.

13.12. STEP Data Set State Change by an Application

The application that changes the state of a **STEP** data set respects the **UUID** population from the transmitting organization, decides what new objects need **UUIDs**, and puts **UUIDs** only on those objects when it generates an updated **STEP** data set. To identify the specific values that changed, the change management schema in ISO/TS 10303-1824 Application module: Change management [82] should be applied to communicate the detailed changes.

13.13. Downstream Applications that Create Data that is a Permanent Record

Downstream applications that create permanent records in conformance with **QIF** and **MTConnect** shall conform to the requirements in those standards.

13.14. STEP Data Set State Change Requests

Applications that request changes to a **STEP** data set shall support design change management as specified in **AP 242**. It may be necessary to preprocess the measurement results from manufacturing and from manufacturing process planning to generate the correct change request records.

Example 26. The manufacturing execution system detects that a hole is breaking through a plate because of a tolerance issue. The manufacturing execution system populates a change request and propagates the message to the data management system that issues a change request to the design organization. After review, the design organization confirms the issue and issues an update to the model. The state change of the topological model in the source **CAD** system is such that the combination of the hole and the tolerance is such that the same **UUID** is provided for the hole in the updated **STEP** data set even though detailed properties have changed. To identify specific change objects, the change management schema in ISO/TS 10303-1824 Application module: Change management should be applied to communicate the detailed changes.

14. Proposed Model Details

Several areas in the **STEP** product model as delineated by relevant **CAx-IF** recommended practice documents are useful in determining the list of entities to include in the SELECT TYPE uuid_attribute_select. A survey of recommended practice documents was performed to group the **AP 242** ENTITY data types by the research area of interest. This document proposes to extend the existing SELECT TYPE identification_item rather than creating a new SELECT TYPE. This document proposes to use the existing SELECT TYPE id_attribute_select as is with the exception that each additional ENTITY data type introduced into the integrated resource product model be specified by an extension to the id_attribute_select.

14.1. User Defined Attributes Recommended Practices

The **CAx-IF** defines recommended practices for implementation of **STEP** application protocols to enable application interoperability. The following items from the User Defined Attribute Recommended Practice [83] are included:

- an instance of the part model **product** entity or **product_definition** entity;
- an instance of the part model in an assembly;
- an instance of the **product_definition_relationship** entity;
- an instance of the **next_assembly_usage_occurrence** entity (covered by **product_-definition_relationship**);

- a portion of the shape defining the part model represented by an instance of the **shape_aspect** entity;
- a **general_property**;
- groups of user-defined attributes;
- a **dimensional_location**;
- a **dimensional_size**;
- a **geometric_tolerance**;
- a **datum_feature**; and
- a **placed_datum_target_feature**.

  NOTE - Instances of property_definition_relationship compose a group but the identification of the group is through **property_definition**.

The **IIRU** and **geometric_item_specific_usage** (**GISU**) relate the geometry to the **shape_-aspect** so are not themselves included in this part of the review. They will be included as a result of review of other requirements.

14.2. PMI Recommended Practices

The items from the PMI Recommended Practices [84] are discussed in the following sections.

14.2.1. shape_aspect as an Identifier for Topological and Geometric Elements

The **STEP** ENTITY data type **shape_aspect** may be used as an identifier for an aggregate of topological and geometric elements.

Example 27. The topological **CAD** object for a hole in system A includes one **shape_aspect**, one **item_identified_representation_usage** (**IIRU**), and two **advanced_face** ENTITY data type instances in the **STEP** data set. A **UUID**(#44) that includes the list (**shape_aspect**(#1), **IIRU**(#2)) would suffice to identify the two **advanced_face** objects. The topological **CAD** object for that hole in system B includes one **shape_aspect**, one **IIRU**, and one **advanced_face** ENTITY data type instance in the **STEP** data set. A **UUID** that includes the list (**shape_aspect**(#1), **IIRU**(#2)) would suffice to identify the single **advanced_face** object. The systems would provide internal mapping tables to retain internal model consistency.

Example 28. When converting from a high order surface, the surface will need to be split into several third order surfaces. A **shape_aspect** will identify the higher order surface and geometric relationships will be applied to record the 3rd order b spline vs. n-th order surface splits.

Some properties of the components involved in the conversion are:

- a cylinder that may be fully periodic or semi-periodic;
- one or more higher order surfaces;
- one or more surfaces (not canonical) each of which is defined by a parametric representation with two parameters and that are 3rd order surfaces; and
- a hybrid representation that includes both a tessellated and an advanced brep shape representation.

14.2.2. PMI constructive_geometric_representation Entity

A constructive_geometric_representation (**CGR**) is a representation included in the **CAD** model to provide geometry that is not part of the b-rep solid product model. The following items are included:

- constructive_geometric_representation, covered by its supertype representation; and
- constructive_geometry_representation_relationship, covered by its supertype representation_relationship.

Example 29. A line is provided to represent the axis that is the center of symmetry of a cylinder. That line is in the **CGRs** and not in the b-rep solid model for the part. The line is related to the centre_of_symmetry **shape_aspect** through a **GISU**.

14.2.3. Saved View

A review of the PMI recommended practice 4.0.8 document (superseded by [84]) and associated data identified the following additional items to include in the **uuid_attribute** related to a saved view:

- draughting_model ;
- model_geometric_view; and
- default_model_geometric_view.

14.2.4. Additional ENTITY Data Types from PMI Recommended Practices

A review of the PMI recommended practice 4.0.8 document (superseded by [84] and associated data identified the following additional items to include in the **uuid_attribute**:

- context_dependent_shape_representation;
- derived_unit;
- dimensional_characteristic_representation;
- dimension_related_tolerance_zone_element;
- founded_item;
- geometric_tolerance_auxiliary_classification;
- geometric_tolerance_relationship;
- gps_filtration_specification;
- gps_filter;
- invisibility;
- item_identified_representation_usage;
- limits_and_fits;
- maths_value_with_unit;
- measure_qualification;
- measure_with_unit;
- named_unit;
- plus_minus_tolerance;
- property_definition_representation;
- **representation_context**;
- representation_relationship;
- runout_zone_orientation;
- **shape_aspect**;

- tolerance_zone_definition;
- tolerance_value;
- tolerance_with_statistical_distribution; and
- tolerance_zone_form.

14.3. Shape Model Design Supplemental Geometry

Shape model design supplemental geometry [85] is geometry the designer creates to assist in the creation of the product shape model. There currently is no **shape_aspect** devoted to this use case of **CGR**, so we propose to use the list of items in the **uuid_attribute** to collect the **CGR** and a new associated **shape_aspect** instance.

- The constructive_geometric_representation entity is covered by representation; and
- The constructive_geometry_representation_relationship entity is covered by representation_relationship.

Example 30. There is a hole in a block. An instance of **CGR** describes the as-located base on the axis of the hole because there is no instantiated feature of the axis of the hole.

14.4. Topology

The proposal in this research is to include **topological_representation_item** by referencing id_attribute_select from the uuid or hash tree models.

14.5. Alternate Shape Representation

Alternate shape representations may be attached to the design specified **STEP** product data by downstream applications.

Example 31. Metrology includes a tessellated model and scale vectors to highlight a cylinder that is off-kilter by half-degree in the manufactured part that is not visible to the user at the model scale. The user is provided a blown-up tessellated model that had user-defined scaling for a portion of the model that is to be used for illustrating as-measured variance from as-designed brep shape.

14.6. Tessellated Representation

tessellated_representation is a subtype of the representation ENTITY data type and need not be explicitly included in the list.

14.6.1. Relationship Entities

We include relationship entities in the uuid_attribute_select to be consistent with **STEP** implementations that use relationship entities to support traceability.

15. Updated Declarations for the STEP Resource Model

The EXPRESS schema of the proposed model and data files created for model validation purposes are available on GitHub [86].

15.1. Items Identified by a UUID

The **STEP** product model provides existing structures for applying identification to items:

- identification_item; and
- id_attribute_select.

The use of existing structures provides built-in extensibility for future **APs**. The approach taken is to reference both identification_item and id_attribute_select and add needed items to identification_item only if they are not already referenced by one of (identification_item, id_attribute_select) and are also not an ENTITY data type declared in a **STEP** resource model. New ENTITY data types in **STEP** resource model should be referenced by an extension to id_attribute_select to satisfy **STEP** product model integration requirements.

15.1.1. Additional uuid_attribute Requirements

This is the list of items proposed to be added to the **STEP** product model.

```
characterized_object
characterized_object_relationship
context_dependent_shape_representation
derived_unit
dimension_related_tolerance_zone_element
dimensional_characteristic_representation
dimensional_location
founded_item
geometric_tolerance_auxiliary_classification
geometric_tolerance_relationship
gps_filter
gps_filtration_specification
invisibility
item_identified_representation_usage
limits_and_fits
measure_qualification
measure_with_unit
named_unit
plus_minus_tolerance
representation_item
representation_item_relationship
runout_zone_orientation
tolerance_value
tolerance_zone_definition
tolerance_zone_form
```

15.1.2. List of Items in id_attribute_select SELECT in ISO 10303-442:2022

The **STEP** product model provides id_attribute_select to apply identification to these items:

```
action
address
application_context
ascribable_state_relationship
dimensional_size
geometric_tolerance
group
organizational_project
product_category
property_definition
representation
shape_aspect
shape_aspect_relationship
topological_representation_item
```

15.1.3. List of Items in the identification_item SELECT in ISO 10303-442:2022

The **AM 442** list of items in identification_item is included in the trial mim_lf.exp file available here: [86].

15.1.4. Items Proposed to be Added to identification_item

The **AM 442** list of items in identification_item will be extended with these items:

```
characterized_object
characterized_object_relationship
context_dependent_shape_representation
derived_unit
dimension_related_tolerance_zone_element
dimensional_characteristic_representation
dimensional_location
founded_item
geometric_tolerance_auxiliary_classification
geometric_tolerance_relationship
gps_filter
gps_filtration_specification
invisibility
item_identified_representation_usage
limits_and_fits
maths_value_with_unit
measure_qualification
measure_with_unit
named_unit
plus_minus_tolerance
representation_item
representation_item_relationship
runout_zone_orientation
tolerance_value
tolerance_with_statistical_distribution
tolerance_zone_definition
tolerance_zone_form
```

15.2. New EXPRESS Declarations

The following sections provide the EXPRESS language declarations of the entities and types proposed for inclusion in the **STEP** product model.

15.2.1. EXPRESS Declaration for UUID Type

The uuid is a 36-character long string that complies with the requirements specified in ISO/IEC 9834-8:2014 [65] for the encoding and interpretation of a **UUID**.

```
TYPE uuid = STRING(36) FIXED;
END_TYPE;
```

15.2.2. EXPRESS Declaration for uuid_attribute_select Type

The uuid_attribute_select provides the ability to specify:

- an id_attribute_select; and
- an identification_item.

```
TYPE uuid_attribute_select = EXTENSIBLE GENERIC_ENTITY SELECT
    (id_attribute_select);
END_TYPE;
```

The EXPRESS declaration for the uuid_attribute_select does not include the **identification_item** in this schema as the uuid_attribute_select SELECT type content is extended to include the **identification_item** in an applicaton module.

15.2.3. EXPRESS Declaration for uuid_relationship_role Type

A **uuid_relationship_role** enumerates the permitted roles associated with a **uuid_relationship**. The **relating_uuid** and **related_uuid** each refer to the data accessed by a query of the relevant **UUID**. The **uuid_relationship_role** specifies the following:

- derive_from indicates that the **relating_uuid** is derived from the **related_uuid**;
- merge indicates that the **relating_uuid** is a merger of two or more other **UUIDs** where the related **UUID** is a member being merged;
- same_as indicates that the **relating_uuid** and the **related_uuid** are the same object in the exchange data set collection;
- similar_to indicates that the two **relating_uuid** and the **related_uuid** objects differ in a minor way in the exchange data set collection;
- split indicates that the **relating_uuid** is one of two or more that are the result of splitting a **UUID** into items; the related **UUID** is a member being split; and
- supersedes indicates that the **relating_uuid** supersedes the **related_uuid**.

```
TYPE uuid_relationship_role = ENUMERATION OF
  (derive_from,
   merge,
   same_as,
   similar_to,
   split,
   supersedes);
END_TYPE;
```

15.2.4. EXPRESS Declaration for uuid_attribute Entity

A **uuid_attribute** associates a **UUID** with an ordered collection of product data items. A **uuid_attribute** shall be either a **v5_uuid_attribute** or a **v4_uuid_attribute**. Additionally, a **uuid_attribute** may be a **uuid_attribute_with_approximate_location.**

```
ENTITY uuid_attribute;
  ABSTRACT SUPERTYPE OF(ONEOF(
    v5_uuid_attribute,
    v4_uuid_attribute)
    ANDOR uuid_attribute_with_approximate_location);
  identifier      : uuid;
  identified_item : LIST [1:?] OF UNIQUE LIST [1:?] OF
  UNIQUE uuid_attribute_select;
 UNIQUE
   UR1 : identifier;
END_ENTITY;
```

Attribute and domain rule descriptions:

- identifier: the **UUID** that is in the role of identifier.
- identified_item: the list of list of uuid_attribute_select that are aggregated by the **uuid_attribute**.
- UR1: specifies that no more than one **uuid_attribute** shall be associated with a specific value of **UUID**.

15.2.5. EXPRESS Declaration for v5_uuid_attribute Entity

A **v5_uuid_attribute** is a **uuid_attribute** that complies with the requirements of [66] for namespace-based **UUIDs**. Neither the namespace nor the name shall be provided in the data exchange set. The population of a **v5_uuid_attribute** conveys to the reader that the sender can re-create the associated uuid from the internal source application model.

```
ENTITY v5_uuid_attribute
  SUBTYPE OF (uuid_attribute);
END_ENTITY;
```

15.2.6. EXPRESS Declaration for v4_uuid_attribute entity

A **v4_uuid_attribute** is a **uuid_attribute** that complies with the requirements of [66] for **UUIDs** that are re-generated independently of data content at each system state save.

```
ENTITY v4_uuid_attribute
  SUBTYPE OF (uuid_attribute);
END_ENTITY;
```

15.2.7. EXPRESS Declaration for hash_based_v5_uuid_attribute Entity

A **hash_based_v5_uuid_attribute** is a **v5_uuid_attribute** where the **UUID** value is the result of applying an externally defined hash function to the list of **CAD** object instances mapped to the specified **STEP** data records specified by the identified_item inherited from the **uuid_attribute**, followed by the **UUID** version 5 hash algorithm.

```
ENTITY hash_based_v5_uuid_attribute
  SUBTYPE OF (v5_uuid_attribute);
  hash_function : STRING;
 WHERE
   WR1 : hash_function <> '';
END_ENTITY;
```

Attribute and domain rule descriptions:

- hash_function: an externally defined hash function.
- WR1: The hash function shall not be an empty string.

15.2.8. EXPRESS Declaration for uuid_attribute_with_approximate_location Entity

A uuid_attribute_with_approximate_location is a **uuid_attribute** that specifies some additional contextual geometric information. The minimal formal data provided shall be a point for the location of the associated item and the shape_representation that specifies the point in its items. The representation is expected to contain a geometric shape related to the business purpose for including the uuid_attribute_with_approximate_location in the product data set.

```
ENTITY uuid_attribute_with_approximate_location
  SUBTYPE OF(uuid_attribute);
    approximate_location    : cartesian_point;
    location_representation : shape_representation;
  WHERE
    WR1: location_representation in
    using_representations(approximate_location);
END_ENTITY;
```

Attribute and domain rule descriptions:

- approximate_location : the cartesian_point that provides the location information.
- location_representation : a shape_representation that specifies the cartesian_point in its items, either directly or indirectly.
- WR1: the location_representation shall reference the approximate_location.

15.2.9. EXPRESS Declaration for uuid_context Entity

A uuid_context associates a role to a **UUID** and is included to support the use case where the enterprise desires more granular context specifiers than that provided by the product_-definition_context ENTITY data type. The allowed values for the role are determined by exchange agreements between parties participating in the exchange. The **UUID** for which a role is assigned is determined by exchange agreements between parties participating in the exchange.

```
ENTITY uuid_context;
  identifier : uuid;
  role : STRING;
 UNIQUE
  UR1 : identifier;
 WHERE
  WR1 : role <> '';
END_ENTITY;
```

Attribute and domain rule descriptions:

- identifier: a **UUID** in the role of identifier.
- role: a STRING that specifies the role for the **UUID**.
- UR1: No more than one uuid_context can specify a particular **UUID**. NOTE - This rule is meant to prohibit multiple applications claiming to originate the same **UUID** in different roles.

- WR1 : The role shall not be an empty string.

15.2.10. EXPRESS Declaration for uuid_provenance Entity

A uuid_provenance provides a list of **uuid_relationship**s that form a record of the successive **UUIDs** in a path for a specific context. The context and application is described in an **AP** that uses this entity or by an exchange agreement.

```
ENTITY uuid_provenance;
  identifier : uuid;
  content : LIST [1:?] OF UNIQUE uuid_relationship;
 UNIQUE
   UR1 : identifier;
END_ENTITY;
```

Attribute and domain rule descriptions:

- identifier: the **UUID** that identifies the **uuid_provenance**.
- content: the list of **uuid_relationship** that form the **uuid_provenance**.
- UR1: There shall be no more than one **uuid_provenance** for a **UUID**.

15.2.11. EXPRESS Declaration for uuid_relationship Entity

A **uuid_relationship** is a relationship between two **UUIDs**. The interpretation of the relationship is that it is a relationship between the objects represented by the relevant **UUIDs**. The role of the relationship in the exchange data set is specified by the role attribute.

```
ENTITY uuid_relationship;
  identifier : uuid;
  uuid_1 : uuid;
  uuid_2 : uuid;
  role : uuid_relationship_role;
 WHERE
  WR1 : uuid_1 <> uuid_2;
  WR2 : uuid_1 <> identifier;
  WR3 : identifier <> uuid_2;
  WR4 : acyclic_uuid_relationship(SELF, uuid_1, uuid_2);
    -- A uuid_relationship shall be acyclic
END_ENTITY;
```

Attribute and domain rule descriptions:

- identifier: the **UUID** that identifies the **uuid_relationship**.

- uuid_1: the first **UUID** in the relationship.
- uuid_2: the second **UUID** in the relationship.
- role: the role of the relationship.
- WR1: uuid_1 shall not be uuid_2.
- WR2: uuid_1 shall not be the identifier.
- WR3: uuid_2 shall not be the identifier.
- WR4: **uuid_relationship**s shall be acyclic.

15.2.12. EXPRESS Declaration for uuid_tree_node Entity

A **uuid_tree_node** is a node in a hash tree [87]. It is an abstract entity and only its subtypes **uuid_leaf_node**, **uuid_internal_node**, and **uuid_root_node** may be populated. The population constraints on the references to lower-level nodes are specified in those subtypes. In order to protect against collisions between leaf hashes and other hashes, different hash constructs are used to hash the leaf nodes than are used for the internal and root nodes. This is achieved by assigning values to:

- the uuid_root_node.root_operand value of ’1’;
- the uuid_leaf_node.leaf_operand value of ’1’; and
- the uuid_internal_node.internal_operand value of ’0’.

The same hash algorithm is used as the basis of each construct, but the specified operand value is prepended to the input of each node hash, based on the type of node that is provided.

Let H() be the secure hash algorithm, for example SHA-1 [69].

- uuid_internal_node hash function = IH(X) = H(0x01, X);
- uuid_root_node hash function = RH(X) = H(0x01, X); and
- uuid_leaf_node hash function = LH(X) = H(0x00, X).

```
ENTITY uuid_tree_node
  ABSTRACT SUPERTYPE OF (ONEOF(
                         uuid_leaf_node,
                         uuid_internal_node,
                         uuid_root_node));
    node_1 : OPTIONAL uuid_tree_node;
    node_2 : OPTIONAL uuid_tree_node;
    uuid_value : uuid;
  WHERE
    WR1 : node_1 <> node_2;
END_ENTITY;
```

Attribute and domain rule descriptions:

- node_1: first node referenced by the **uuid_tree_node**.
- node_2: second node referenced by the **uuid_tree_node**.
- uuid_value: value returned by the hash function applied to lower-level nodes or product data.
- WR1: node_1 and node_2 shall be different nodes.
- IP1: The population of **uuid_tree_node** shall be acyclic.

15.2.13. EXPRESS Declaration for uuid_root_node Entity

A **uuid_root_node** is a node in a hash tree that is the root of the tree. It provides a **UUID** value for the tree and specifies the hash function that shall be used for tree members.

```
ENTITY uuid_root_node
  SUBTYPE OF(uuid_tree_node);
    hash_function : STRING;
  DERIVE
    root_operand  : STRING := '1';
 WHERE
   WR1 : SIZEOF(USED_IN('UUID_SCHEMA.UUID_TREE_NODE.NODE_1)) = 0;
   WR2 : SIZEOF(USED_IN('UUID_SCHEMA.UUID_TREE_NODE.NODE_2)) = 0;
   WR3 : EXISTS(node_1) AND EXISTS(node_2);
   WR4 : hash_function <> '';
END_ENTITY;
```

Attribute and domain rule descriptions:

- hash_function: function used to hash each node of the tree.

- root_operand: a constant prepended to the input data to the hash_function.
- WR1: No node shall reference a **uuid_root_node** in the role of node_1.
- WR2: No node shall reference a **uuid_root_node** in the role of node_2.
- WR3: node_1 and node_2 shall be provided.
- WR4: The hash function shall not be an empty string.

15.2.14. EXPRESS Declaration for uuid_internal_node Entity

A **uuid_internal_node** is a node in a hash tree that is not the root of the tree and is not a leaf of the tree.

```
ENTITY uuid_internal_node
  SUBTYPE OF(uuid_tree_node);
  DERIVE
    internal_operand : STRING (1) FIXED := '1';
 WHERE
   WR1 : EXISTS(node_1) AND EXISTS(node_2);
   WR2 : (SIZEOF(USED_IN(UUID_SCHEMA.UUID.TREE_NODE.NODE_1))=1)
   XOR (SIZEOF(USED_IN(UUID_SCHEMA.UUID.TREE_NODE.NODE_2))=1);
END_ENTITY;
```

Attribute and domain rule descriptions:

- internal_operand: a constant prepended to the input data to the hash_function.
- WR1: node_1 and node_2 shall be provided.
- WR2: An internal node shall be referenced once by a higherlevel node.

15.2.15. EXPRESS Declaration for uuid_leaf_node Entity

A **uuid_leaf_node** is a node in a hash tree that is a leaf of the tree.

```
ENTITY uuid_leaf_node
  SUBTYPE OF(uuid_tree_node);
    data          : uuid_attribute_select;
    DERIVE
    leaf_operand : STRING (1) FIXED := '0';
 WHERE
   WR1 : NOT (EXISTS (node_1) OR EXISTS(node_2));
   WR2 : (SIZEOF(USED_IN(UUID_SCHEMA.UUID.TREE_NODE.NODE_1))=1)
         XOR (SIZEOF(USED_IN(UUID_SCHEMA.UUID.TREE_NODE.NODE_2))=1);
END_ENTITY;
```

Attribute and domain rule descriptions:

- data: the input data to the leaf_node.
- leaf_operand: a fixed string with value '0'.
- WR1: Each attribute node_1, node_2 (inherited from tree_node) shall not be provided.
- WR2: A leaf_node shall be referenced by a tree_node attribute node_1 or by a tree_-node node_2 but not both simultaneously; only one tree_node shall reference a leaf_node.

16. Exchange Examples

Exchange examples considered include shared design collaboration and first article inspection.

16.1. Shared Design Collaboration

A cylinder is a combined topological and geometric representation of a hole in a three-dimensional **CAD** model that is defined by four attributes: an x,y,z coordinate tuple for location; a line that represents the axis that is the center of symmetry of that cylinder; the radius of that cylinder; and a direction of the line. The internal **CAD OID** is assigned to the resulting topology, and a **UUID** is assigned to the **shape_aspect** that represents that hole feature in the **STEP** data set. To modify the hole geometry, one of four methods is used.

Method 1: One or more of the hole attributes are modified. Because the internal **CAD OID** is maintained, the **UUID** remains as was initially defined, relying on the receiving system to audit the design for updates.

Method 2: One or more of the hole attributes are deleted and recreated while the hole feature definition remains. For example, the point that defines the location of the hole feature is deleted, and a new point is created. Because the internal **CAD** feature object remains, and its **OID** is maintained, the **UUID** does not require an update, relying on the receiving system to audit the design for updates.

Method 3: The hole is deleted, and a new hole feature is defined. Because the initial **CAD** feature is removed, a new **CAD OID** will be assigned to the new feature. The **UUID** associated to the initial **CAD OID** is flagged to become unavailable for future use, and a new **UUID** is associated to the new hole. The **uuid_relationship** will support the capability to specify that the new hole feature supersedes the previous hole feature.

Method 4: A new feature is added to the model, resulting in a new **CAD OID** in the model. A **UUID** is assigned to that new feature. Each new **CAD** object would be assigned an **OID** and when exported to **AP 242**, a **UUID**.

Example 32. This example illustrates the application of method 1 specified in 16.1 on a simple test model. The model was implemented with guidance from the first release of a recommended practice for **UUID** [88].

The model is a plate of 50.8 mm x 76.2 mm x 10 mm centered about the coordinate system. There is a 19 mm diameter thru hole located at 6.3 mm, 6.4 mm, 0 mm relative to the coordinate system. An external application or **CAD** plug-in identifies the hole feature using its **OID** and, following enterprise policy, issues a **UUID** for each **CAD** object of topology and annotation that defines the hole feature and its related **PMI** respectively. An image of the initial state of the **CAD** model is shown in Fig. 11.

An extract of the topology and annotations related to the hole in version 0.0 of the **AP 242** test file is shown in table 5.

The complete data file is available as ”NX_Plate_w_Hole_UUIDS_Rev_0.0.stp” at [86].

The parameters that locate the hole center point are modified in the CAD model during design iteration version 0.1. The thru hole center parameters are changed to move the hole to -8.3 mm, 8.9 mm, 10 mm relative to the coordinate system. An external application or **CAD** plug-in identifies the hole feature as having the same **OID**, and therefore, the **UUID** for that feature remains unchanged to maintain traceability of the modifications. An extract of the relevant instance data from the **AP 242** file of iteration 0.1 is shown in table 6. The complete data file is available as ”NX_Plate_w_Hole_UUIDS_Rev_0.1.stp” at [86].

In version 0.2, the hole feature is deleted from the CAD model. A new hole feature of the same size, with a new location -6.3 mm, 6.9 mm, 10 mm, is defined. Two scenarios result from differences in how **CAD** applications track changes.

All **PMI** remain in the model, and those whose references no longer exist become disconnected. The disconnected **PMI** are connected to the new hole geometry. Because the **PMI** remain, their **UUIDs** are maintained.

An extract of the relevant instance data from the **AP 242** file of iteration 0.2, is shown in table 7. The complete data file is available as ”NX_Plate_w_Hole_UUIDS_Rev_0.2.stp” at [86].

Version 0.3 adds a chamfer to one edge of the model. The chamfer is a new feature and the CAD application identifies it with an **OID**. Following enterprise policy, a new **UUID** is assigned to the new feature. An extract of the relevant instance data from the **AP 242** file of iteration 0.3 is shown in table 8. The complete data file is available as ”NX_Plate_w_-Hole_UUIDS_Rev_0.3.stp” at [86].

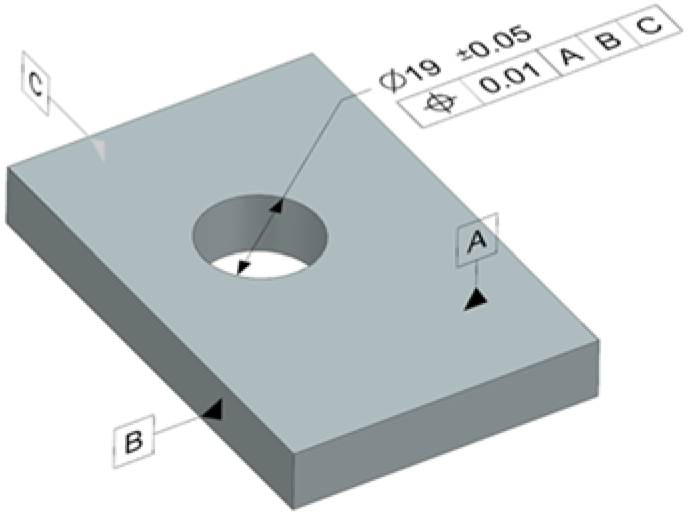


Fig. 11. CAD model with PMI.

Element Class	PMI Values	Assigned UUID
Advanced face	-	2dfe9b4e-fd8b-5006-b660-a0da88b5e2ad
Cylindrical surface	-	not provided
Hole Diameter	∅ 19	814e27a0-acee-5455-a393-2e493d372f4e
Diameter tolerance	+/- 0.05	not provided
Hole Vertex	6.3,6.4,10	03338caf-ec0c-518a-a8eb-4b7bb170724a
Hole Vertex	6.3,6.4,0	1a70557f-cc2b-57fe-8f80-fbeaaa60edce
Feature Control Frame (2)	-	df69d739-ee5e-560e-8670-8822cca41d0b
Datum Feature A	A	not provided
Datum Feature B	B	not provided
Datum Feature C	C	not provided

Table 5. Extract of data from Revision 0.0.

Element Class	PMI Values	Assigned UUID
Hole Vertex	8.3,8.9,10	03338caf-ec0c-518a-a8eb-4b7bb170724a
Hole Vertex	8.3,8.9,0	1a70557f-cc2b-57fe-8f80-fbeaaa60edce

Table 6. Extract of data from Revision 0.1 illustrating change in hole location.

16.2. First Article Inspection

With the inclusion of a **UUID** to entities, collaborating systems in the product life cycle have the ability to exchange and track model data during design iterations. This is accomplished through the ability to retain **UUIDs** from an external data source that can be referenced by the receiver. When a change to that model data occurs on the sender's side, the receiver should recognize that external change and link it to dependent data in their own models.

The ability to track model entities via permanent IDs will also allow downstream systems to update their representations of the design model and update their manufacturing and metrology planning to reflect changes in the design.

An additional benefit of the establishment of permanent IDs in **STEP** is the ability to retain a permanent audit trail of custody and connection between design and downstream systems for potential forensic analysis of critical product systems after in-service failure. The introduction of permanent IDs provides the ability of any contributor to the information stream associated with a product's life cycle to add information to the model that can be

Element Class	PMI Values	Assigned UUID
Advanced face	-	3ef5490e-c4c5-5208-8bdc-a75293cd144f
Cylindrical surface	-	not provided
Hole Diameter	∅ 19±0.05	814e27a0-acee-5455-a393-2e493d372f4e
Hole Vertex	8.3,8.9,10	ec147a04-aaad-59f0-9417-e9c7cdde9501
Hole Vertex	8.3,8.9,0	e3987c2b-bdd9-56ae-8d56-22e2443f972b
Feature Control Frame (7)	-	21dba1d4-f8c9-5c39-9163-510b94836bfb
Datum Feature A	A	not provided
Datum Feature B	B	not provided
Datum Feature C	C	not provided

Table 7. Extract of data from Revision 0.2 including hole deletion and creation of new hole.

Element Class	Related PMI	Assigned UUID
Advanced Face	-	3ef5490e-c4c5-5208-8bdc-a75293cd144f

Table 8. Extract of data from Revision 0.3 illustrating addition of chamfer face.

connected to existing model content and be retrieved by subsequent users and used as feedback from the contributor.

Example 33. Table 9 provides a first article test plan for design revision 0.1 extracted from the **STEP**file using the method in [62].

Example 34. Table 10 provides the inspection results for a workpiece created from revision 0.3 of the design extracted from the **STEP** file using the method in [60].

17. Conclusion and Future Research

In this paper, we have described research and recommendations for the use of **UUIDs** in product data standards in the design to manufacturing and inspection workflow (Sec. 2). We examined industrial use cases (Sec. 3) to discover requirements for the use of **UUIDs** in these product data standards. We discovered common requirements in the context of mechanical design and manufacturing model-based enterprises through literature, patents,

Table 9. FAIR test plan revision for version 0.1.

Element Id	PMI	FeatureNominal	QPId
Diameter (26, 21)	dia 40±0.001	-	a0dce501
Diameter (29, 24)	dia 100±0.005	-	70d69775
DistanceBetween (30, 25)	25±0.001	-	54a096a3
Datum(36)	A	Cylinder 6	-
Datum(27)	B	Plane 7	-
Datum(38)	C	Plane 8	-
DatumReference-Frame(39)	A	-	-
DatumReference-Frame(40)	B	-	-
Perpendicularity(22)	perp|0.001|A	-	4fdd2728
Perpendicularity(23)	perp|0.001|A	-	5ee8178c

Table 10. QPR FAIR data table results for version 0.3.

Element Id	PMI	Name	Measurement	QPId
Diameter (4,3)	dia 39.99-40.001	Diameter2	40. PASS	a0dce501
Diameter (14,13)	dia 99.995-100.005	Diameter1	100. PASS	70d69775
DistanceBetween (17, 16)	24.999-25.001	Distance 1	25. PASS	54a096a3
DatumReference-Frame(2)	-	-	-	
Perpendicularity(7)	perp|dia.0.001	-	0. PASS	4fdd2728
Perpendicularity(10)	perp|dia.0.001	-	0. PASS	5ee8178c

and industrial practice standards (Sec. 4). Additionally, we described our use of the term concept.

We discussed **UUID** encoding requirements and characteristics, types of **UUIDs**, how they are used with human-readable identifiers, and what requirements they address. We identified the requirement that the **UUID** application in **STEP** apply digital signatures to deter tampering with **UUID** instances (Sec. 5).

We surveyed the current state of **UUID** implementation in product data standards in the manufacturing domain, in the inspection domain, and in the architecture and construction domain's Industry Foundation Classes (Sec. 6).

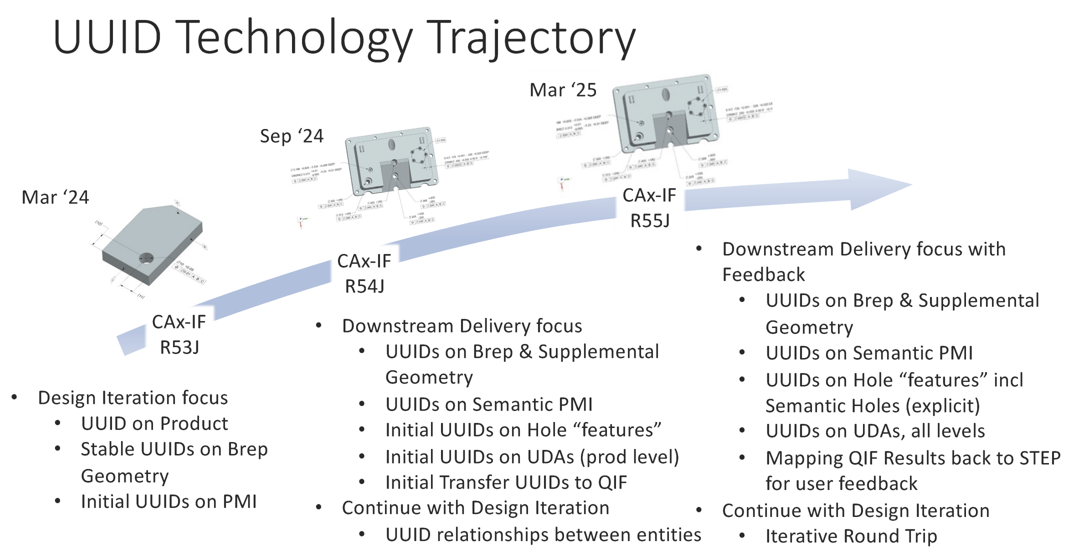
We then focused on **STEP** in particular. We described the role of aggregators in **STEP** and how some aggregators are implicitly defined in a product data set. We described limitations of existing identifiers in **STEP** (Sec. 9).

We then presented several design options for extensions to the existing **STEP** product model, enumerating appropriate aggregates and key **STEP** ENTITY data types to use as root entities for digital signature and approval. We recommended an approach using [13] to classify mechanical features for use in reference designations.

In Section 11, we proposed extensions to the existing **STEP** product model. We then described how the proposed model satisfies the requirements (Sec. 12). We provided recommendations for application behavior when implementing the proposed model (Sec. 13). We specified proposed extensions to existing **STEP** EXPRESS declarations (Sec. 14) and their descriptions (Sec. 15), and finally provide detailed exchange examples (Sec. 16).

The identification of detailed recommendations for adding **UUIDs** to the **STEP** product model is a novel contribution. However, there is much left to do. In our research, we recommended the behavior of supporting software applications that do not currently exist and will be challenging to implement. In the exchange example for the ”Shared design collaboration” use case (Sec. 16.1), methods 3 and 4 require the ability to maintain traceability that is not available in software. Exporting and importing multiple model types (such as native CAD, neutral formats, and others) from many diverse sources and their variations in geometry definition is not trivial and requires significant effort to track.

In the case of sending the data to a **CMS**, each annotation and its associated geometry in the model must be matched to one of the hundreds or thousands of configured machines, **PMI** requirements applied, and a collision-free measurement solution generated. The software systems utilizing this information must then manage gigabytes (possibly terabytes) of critical as-measured data, converted to actionable information, from multiple measurement sources in real-time, all while maintaining rock-solid persistence and interoperability. The amount of detailed data collected by an enterprise or supply chain will vary as the product design and the supply chain communication network matures.



UUID Technology Trajectory

- Design Iteration focus
  - UUID on Product
  - Stable UUIDs on Brep Geometry
  - Initial UUIDs on PMI

- Downstream Delivery focus
  - UUIDs on Brep & Supplemental Geometry
  - UUIDs on Semantic PMI
  - Initial UUIDs on Hole "features"
  - Initial UUIDs on UDAs (prod level)
  - Initial Transfer UUIDs to QIF
- Continue with Design Iteration
  - UUID relationships between entities

- Downstream Delivery focus with Feedback
  - UUIDs on Brep & Supplemental Geometry
  - UUIDs on Semantic PMI
  - UUIDs on Hole "features" incl Semantic Holes (explicit)
  - UUIDs on UDAs, all levels
  - Mapping QIF Results back to STEP for user feedback
- Continue with Design Iteration
  - Iterative Round Trip

Fig. 12. CAx-IF UUID test round objectives.

In the near term, validation of the new **AP 242 UUID** constructs will continue in **CAx-IF** [55] interoperability test rounds as shown in Fig. 12. The most recent **CAx-IF** test round focused on design iteration. In round R54J the **CAx-IF** will proceed to multi-domain interoperability by testing the initial delivery of **UUIDs** to metrology applications through **QIF**. The subsequent round will test feedback from metrology applications and round trip design iteration.

Additionally, we plan to investigate approaches for enhancing identification and communication of object state changes, using Merkle trees, to further move along the path to concept based change management.

References

[1] International Organization for Standardization (2024) ISO 10303-1 - Industrial automation systems and integration – Product data presentation and exchange – Part 1: Overview and fundamental principles. Available at https://www.iso.org/standard/83105.html.

[2] International Organization for Standardization (2014) ISO 10303-209 - Industrial automation systems and integration – Product data representation and exchange – Part 209: Application protocol: Multidisciplinary analysis and design. Available at https://www.iso.org/standard/59780.html.

[3] International Organization for Standardization (2022) ISO 10303-238 - Industrial automation systems and integration – Product data representation and exchange – Part 238: Application protocol: Managed model-based integrated manufacturing. Available at https://www.iso.org/standard/84898.html.

[4] International Organization for Standardization (2022) ISO 10303-242 - Industrial automation systems and integration – Product data representation and exchange – Part 242: Application protocol: Managed model-based 3D engineering. Available at https://www.iso.org/standard/84667.html.

[5] MTConnect Institute (2023) MTConnect Standard Version 2.2. Available at https://docs.mtconnect.org/MBSD_MTConnect_Part_1_2-2-0.pdf.

[6] Dimensional Metrology Standards Consortium (2020) Quality Information Framework Standard (QIF) – An integrated model for manufacturing quality information. Available at https://qifstandards.org/qif-download.

[7] International Organization for Standardization (2020) ISO 23952 - Automation systems and integration – Quality information framework (QIF) – An integrated model for manufacturing quality information. Available at https://www.iso.org/standard/77461.html.

[8] Object Management Group (2013) Business Process Model and Notation v2.0.2. Available at https://www.omg.org/spec/BPMN/2.0.2/PDF.

[9] Juty N, Wimalaratne SM, Soiland-Reyes S, Kunze J, Goble CA, Clark T (2020) Unique, persistent, resolvable: Identifiers as the foundation of FAIR. *Data Intelligence* 2(1-2):30–39. https://doi.org/10.1162/dint_a_00025

[10] Hedberg Jr T, Maw TMM, Rahman MM, Jadhav S, Whicker JJ, Barnard Feeney A, Helu M (2021) Defining requirements for integrating information between design, manufacturing, and inspection. *International Journal of Production Research* https://doi.org/10.1080/00207543.2021.1920057

[11] International Organization for Standardization (2004) ISO 10303-11 - Industrial automation systems and integration – Product data representation and exchange – Part 11: Implementation methods: The EXPRESS language reference manual. Available at https://www.iso.org/standard/38047.html.

[12] buildingSMART International Limited (2018) Industry Foundation Classes Version 4.0 - Addendum 2 - Technical Corrigendum 1. Available at https://standards.buildingsmart.org/IFC/RELEASE/IFC4/ADD2_TC1/HTML/.

[13] International Electrotechnical Commission (2010) IEC 61666 - Industrial systems, installations and equipment, and industrial products – Identification of terminals within a system. Available at https://webstore.iec.ch/en/publication/5705.

[14] International Organization for Standardization (2021) ISO 23247-1 - Automation systems and integration – Digital twin framework for manufacturing – Part 1: Overview and general principles. Available at https://www.iso.org/standard/75066.html.

[15] International Organization for Standardization and International Electrotechnical Commission (2018) ISO/IEC 30141 - Internet of Things (IoT) – Reference Architecture. Available at https://www.iso.org/standard/65695.html.

[16] International Organization for Standardization (2021) ISO 23247-4 - Automation systems and integration – Digital twin framework for manufacturing – Part 4: Information exchange. Available at https://www.iso.org/standard/78745.html.

[17] International Organization for Standardization (2009) ISO 6983-1 - Automation systems and integration – Numerical control of machines – Program format and definitions of address words – Part 1: Data format for positioning, line motion and contouring control systems. Available at https://www.iso.org/standard/34608.html.

[18] Fischer OJP, Matlik JF, Schindel WD, French MO, Kabir MH, Ganguli JS, Hardwick M, Arnold SM, Byar AD, Lewe JH, Duncan S, Dong JJ, Kinard DA, Maiaru M (2022) Digital twin: Reference model, realizations, and recommendations. *INSIGHT* 25.

[19] International Organization for Standardization (2024) ISO 10303-21 - Industrial automation systems and integration – Product data representation and exchange – Part 21: Implementation methods: Clear text encoding of the exchange structure. Available at https://www.iso.org/standard/63141.html.

[20] Lanza R, Haenisch J, Bengtsson K, RÃ¸lvÃ¥g T (2021) ISO 10303 AP209 – Why and how to embed nonlinear FEA. *Advances in Engineering Software* 154:102976. https://doi.org/https://doi.org/10.1016/j.advengsoft.2021.102976

[21] International Organization for Standardization (2011) ISO 10303-203 - Industrial automation systems and integration – Product data representation and exchange – Part 203: Application protocol: Configuration controlled 3D design of mechanical parts and assemblies. Available at https://www.iso.org/standard/44305.html.

[22] International Organization for Standardization (2010) ISO 10303-214 - Industrial automation systems and integration – Product data representation and exchange – Part 214: Application protocol: Core data for automotive mechanical design processes. Available at https://www.iso.org/standard/43669.html.

[23] Suh SH, Cho JH, Hong HD (2002) On the architecture of intelligent STEP-compliant CNC. *International Journal of Computer Integrated Manufacturing* 15(2):168–177. https://doi.org/10.1080/09511920110056541

[24] Brown SA, E DC, Mittman B (1963) A Description of the APT Language. *Commun ACM* 6(11):649–658. https://doi.org/10.1145/368310.368322

[25] Wikipedia (2021) STEP-NC. Available at https://en.wikipedia.org/wiki/STEP-NC.

[26] International Organization for Standardization ISO 14649-10 - Industrial automation systems and integration – Physical device control – Data model for computerized numerical controllers – Part 10: General process data.

[27] Trainer A (2016) Validation for downstream computer aided manufacturing and coordinate metrology processes (National Institute of Standards and Technology, Gaithersburg, MD), NISTGCR 15-1009. https://doi.org//10.6028/NIST.GCR.16-003

[28] Barnard Feeney A, Frechette SP, Srinivasan V (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):21001. https://doi.org/10.1115/1.4029050

[29] Williams WD (2012) A business case for long-term archiving & retrieval of the model based enterprise's data. (Sandia National Lab, Albuquerque, NM), Available at https://www.osti.gov/servlets/purl/1106409.

[30] Trainer A, Hedberg Jr T, Barnard Feeney A, Fischer K, Rosche P (2016) Gaps analysis of integrating product design, manufacturing, and quality data in the supply chain using model-based definition. *ASME 2016 11th International Manufacturing Science and Engineering Conference – Volume 2: Materials; Biomanufacturing; Properties, Applications and Systems; Sustainable Manufacturing*, , . https://doi.org/10.1115/MSEC2016-8792

[31] Hedberg Jr T, Lubell J, Fischer L, Maggiano L, Barnard Feeney A (2016) Testing the Digital Thread in Support of Model-Based Manufacturing and Inspection. *Journal of Computing and Information Science in Engineering* 16(2). https://doi.org/10.1115/1.4032697

[32] Ficher K, Rosche P, Trainer A, Barnard Feeney A, Hedberg Jr T (2015) Investigating the impact of standards-based interoperability for design to manufacturing and quality in the supply chain (Gaithersburg MD), NISTGCR 15-1009. https://doi.org/10.6028/NIST.GCR.15-1009

[33] Barnard Feeney A, Thurman T (2023) On Migrating ISO 10303 PMI Models to a Common Core. https://doi.org/10.6028/NIST.AMS.100-51.

[34] Kulvatunyou B, Drobnjakovic M, Ameri F, Will C, Smith B (2022) The Industrial Ontologies Foundry (IOF) Core Ontology, , . Available at https://tsapps.nist.gov/publication/get_pdf.cfm?pub_id=935068.

[35] Boone, T (2022) Technical data package fall workshop summary. AFRL-2022-5873.

[36] De Nijs, J (2024) Enabling Advanced Analytics Use Cases that Require Manufacturing Data. Copyright Lockheed Martin 2024, PIRA CET2024020117.

[37] De Nijs J (2021) It is all about the digital thread!. Copyright Lockheed Martin 2024, PIRA CET202109008.

[38] De Nijs J (2022) Keynote: It's all about the Digital Thread!. Copyright Lockheed Martin 2022, PIRA CET202206001.

[39] Kirkwood R, Sherwood J (2018) Sustained CAD/CAE integration: integrating with successive versions of STEP or IGES files. *Engineering with Computers* 34:1–13. https://doi.org/10.1007/s00366-017-0516-z

[40] Denno P, Thurman T (2005) Requirements on information technology for product life-cycle management. *International journal of product development* 2(1):109–122.
[41] The American Society of Mechanical Engineers (2017) ASME Y14.100 - Engineering Drawing Practices. Available at https://www.asme.org/codes-standards/find-codes-standards/y14-100-engineering-drawing-practices.
[42] International Organization for Standardization (2022) ISO 10303-41 - Industrial automation systems and integration – Product data representation and exchange – Part 41: Integrated generic resource: Fundamentals of product description and support. Available at https://www.iso.org/standard/84668.html.
[43] Lanza R, Haenisch J, Bengtsson K, RÃ¸lvÃ¥g T (2019) Relating structural test and FEA data with STEP AP209. *Advances in Engineering Software* 127:96–105.
[44] Hardwick M, Zhao YF, Proctor FM, Nassehi A, Xu X, Venkatesh S, Odendahl D, Xu L, Hedlind M, Lundgren M, Maggiano L, Loffredo D, Fritz J, Olsson B, Garrido J, Brail A (2013) A roadmap for STEP-NC-enabled interoperable manufacturing. *The International Journal of Advanced Manufacturing Technology* 68(5):1023–1037. https://doi.org/10.1007/s00170-013-4894-0
[45] Hardwick M, Spooner DL, Rando T, Morris KC (1996) Sharing manufacturing information in virtual enterprises. *Commun ACM* 39(2):46–54. https://doi.org/10.1145/230798.230803
[46] International Organization for Standardization (2021) ISO 10303-210 - Industrial automation systems and integration – Product data representation and exchange – Part 210: Application protocol for electronic interconnect, assembly and packaging design. Available at https://www.iso.org/standard/72406.html.
[47] The American Society of Mechanical Engineers (2019) ASME Y14.37 - Composite Part Drawings. Available at https://www.asme.org/codes-standards/find-codes-standards/y14-37-composite-part-drawings/2019/pdf.
[48] International Organization for Standardization (2018) ISO/TS 10303-1671 - Industrial automation systems and integration – Product data representation and exchange – Part 1671: Application module: Feature and connection zone. Available at https://www.iso.org/standard/76229.html.
[49] International Organization for Standardization (2022) ISO 10303-47 - Industrial automation systems and integration – Product data representation and exchange – Part 47: Integrated generic resource: Shape variation tolerances. Available at https://www.iso.org/standard/84677.html.
[50] International Organization for Standardization (2019) ISO/TS 10303-1764 - Industrial automation systems and integration – Product data representation and exchange – Part 1764: Application module: Shape feature. Available at https://www.iso.org/standard/78659.html.
[51] The American Society of Mechanical Engineers (2019) ASME Y14.41 - Digital product definition data practices: Engineering product definition and related documentation practices. Available at https://www.asme.org/codes-standards/find-codes-standards/y14-41-digital-product-definition-data-practices.

[52] International Organization for Standardization (2011) ISO 5459 - Geometrical product specifications (GPS) – Geometrical tolerancing – Datums and datum systems. Available at https://www.iso.org/standard/40358.html.
[53] International Organization for Standardization (2021) ISO 16792 - Technical product documentation – Digital product definition data practices. Available at https://www.iso.org/standard/73871.html.
[54] Gillman DW, Farance F (2009) Metadata and data harmonization. Available at https://nces.ed.gov/FCSM/pdf/2009FCSM_Gillman_X-A.pdf.
[55] MBx Interoperability Forum (2024) CAx Interoperability Forum. Accessed on 2024-07-05 Available at https://www.mbx-if.org/home/cax/.
[56] Nagel L (1975) Spice2: A computer program to simulate semiconductor circuits (Electronics Research Laboratory, College of Engineering, University of California, Berkeley), ERL-M520. Available at https://www2.eecs.berkeley.edu/Pubs/TechRpts/1975/ERL-m-520.pdf.
[57] American Society of Mechanical Engineers (2021) ASME Y14.45-2021 - Measurement Data Reporting.
[58] SAE International (2023) First Article Inspection Requirement RevC.
[59] Hedberg T, Helu M, Feeney AB (2016) Technical Data Packages (TDPs) from the Smart Manufacturing Systems (SMS) Test Bed.
[60] Thurman T, Trainer A, Barnard Feeney A, Astheimer R (2016) First Article Inspection Requirement Report Generation from QIF Using C++, CodeSynthesis, and Mozilla Xerces. Available at https://github.com/usnistgov/QIF.
[61] Lipman R (2019) Software to report product and manufacturing information in qif files. https://doi.org/10.6028/jres.124.036.
[62] Michaloski J (2016) First Article Inspection Requirement Report Generation from QIF Using C++, CodeSynthesis, and Mozilla Xerces. Available at https://tsapps.nist.gov/publication/get_pdf.cfm?pub_id=919140.
[63] International Organization for Standardization (2022) ISO 81346-1 - Industrial systems, installations and equipment and industrial products – Structuring principles and reference designations: Part 1: Basic rules. Available at https://www.iso.org/standard/82229.html.
[64] Codd EF (1970) A relational model of data for large shared data banks. *Communications of the ACM* 13(6):377–387.
[65] International Organization for Standardization and International Electrotechnical Commission (2014) ISO/IEC 9834-8 - Information technology – Procedures for the operation of object identifier registration authorities: Part 8: Generation of universally unique identifiers and their use in object identifiers. Available at https://www.iso.org/standard/62795.html.
[66] Davis K, Peabody B, Leach P (2024) Request for Comment 9562: Universally Unique IDentifiers (UUIDs) (Internet Engineering Task Force (IETF)),
[67] DCE: Remote Procedure Call, Open Group CAE Specification C309, ISBN 1-85912-041-5, August 1994.

[68] Rivest RL (1992) The MD5 Message-Digest Algorithm, RFC 1321. https://doi.org/10.17487/RFC1321. Available at https://www.rfc-editor.org/info/rfc1321

[69] Dang Q (2015) Secure Hash Standard, Federal Inf. Process. Stds. (NIST FIPS) (National Institute of Standards and Technology), https://doi.org/10.6028/NIST.FIPS.180-4

[70] Hedberg Jr T, Krima S, Camelio JA (2019) Method for enabling a root of trust in support of product data certification and traceability. *Journal of Computing and Information Science in Engineering* 19(4):041003. https://doi.org/10.1115/1.4042839

[71] International Organization for Standardization and International Electrotechnical Commission (2012) ISO/IEC 9834-1 - Information technology – Procedures for the operation of object identifier registration authorities: Part 1: General procedures and top arcs of the international object identifier tree. Available at https://www.iso.org/standard/58055.html.

[72] Ojal N, Giera B, Devlugt Kea (2022) A universal method to compare parts from STEP files. *J Intell Manufacturing* 33:2167–2178. https://doi.org/https://doi.org/10.1007/s10845-022-01984-3

[73] ISO TC 184/SC 4 N532:1997 Guidelines for application interpreted model development.

[74] Hedberg Jr T, Krima S, Camelio JA (2016) Embedding X.509 digital certificates in three-dimensional models for authentication, authorization, and traceability of product data. *Journal of Computing and Information Science in Engineering* 17(1):11008–11011. https://doi.org/10.1115/1.4034131

[75] Merkle RC (1978) Secure Communications over Insecure Channels. *Communications of the ACM* 2(4):294–299. https://doi.org/10.1145/359460.359473

[76] Merkle RC (1990) A certified digital signature. *Advances in Cryptology — CRYPTO' 89 Proceedings*, ed Brassard G (Springer, New York, New York, USA), , pp 218–238.

[77] Collet Y (2023) xxHash - Extremely fast hash algorithm, https://github.com/Cyan4973/xxHash.

[78] O'Connor J, Aumasson JP, Neves S, Wilcox-O'Hearn Z (2013) BLAKE3 – one function, fast everywhere, https://github.com/BLAKE3-team/BLAKE3-specs/blob/master/blake3.pdf.

[79] Danner WF, Sanford DT, Yang Y (1991) STEP (STandard for the Exchange of Product Model Data) Resource Integration: Semantic & Syntactic Rules. https://doi.org/10.6028/NIST.IR.4528.

[80] Trainer AG, Krishnan G, Varvak Y, Berkeley S (2002) Method and System for Management of Heterogeneous Assemblies. U. S. Patent 6 473 673.

[81] Trainer AG, Krishnan G, Varvak Y, Berkeley S (2014) Methods and Systems for Managing Synchronization of a Plurality of Information Items of a Computer-Aided Design Data Model. U. S. Patent 8 818 769.

[82] International Organization for Standardization (2019) ISO/TS 10303-1824 - Industrial automation systems and integration – Product data representation and exchange – Part 1824: Application module: Change management. Available at https://www.iso.org/standard/78671.html.

[83] Boy J, Lipman R, Rosche P (2021) CAx-IF Recommended Practices for User Defined Attributes, Release 1.8 (CAx Interoperability Forum), Available at https://www.mbx-if.org/home/wp-content/uploads/2024/05/rec_prac_user_def_attributes_v18.pdf.

[84] Boy J, Astheimer R, Fischer B, Paff E, Rosche P (2024) CAx-IF Recommended Practices for the Representation and Presentation of Product Manufacturing Information (PMI) (AP242), Version 4.1 (CAx Interoperability Forum), Available at https://www.mbx-if.org/home/wp-content/uploads/2024/06/rec_pracs_pmi_v41.pdf.

[85] Boy J, Lipman R, Rosche P (2021) CAx-IF Recommended Practices for Supplemental Geometry (AP242) (CAx Interoperability Forum), Available at https://www.mbx-if.org/home/wp-content/uploads/2024/05/rec_prac_suppl_geo_v12a.pdf.

[86] Thurman T, Trainer A, Barnard Feeney A, Astheimer R (2024) UUID. Available at https://github.com/usnistgov/UUID.

[87] Chapweske J, Mohr G Tree Hash EXchange format (THEX). Available at https://adc.sourceforge.io/draft-jchapweske-thex-02.html.

[88] Trainer A, Thurman T, Lipman R, Rosche P, Boy J (2024) CAx-IF Recommended Practices for Persistent IDs for Design Iteration and Downstream Exchange, Release 1.0 (CAx Interoperability Forum), Available at https://www.mbx-if.org/home/wp-content/uploads/2024/05/rec_pracs_PID_v1.pdf.

Appendix A. Abbreviations

3D	three-dimensional 9
AM 442	ISO/TS 10303-442 59
ANSI	American National Standards Institute 1
AP	application protocol 1, 9, 13, 20, 24, 33, 35–37, 39, 40, 43, 57, 65
AP 203	ISO 10303-203: Application protocol: Configuration controlled 3D design of mechanical parts and assemblies 7, 8
AP 209	ISO 10303-209: Application protocol: Multidisciplinary analysis and design 1, 7, 10, 20
AP 214	ISO 10303-214: Application protocol: Core data for auto- motive mechanical design processes 7, 8
AP 238	ISO 10303-238: Application protocol: Managed model-based integrated manufacturing 1, 4, 5, 7, 8, 10, 20, 21, 40
AP 239	ISO 10303-239: Application protocol: Product life cycle support 18
AP 242	ISO 10303-242: Application protocol: Managed model-based 3D engineering 1, 3, 4, 6–10, 18–21, 23–27, 29, 35, 36, 46, 47, 50, 52, 70, 71, 76
ARM	application reference model 37, 39
ASME	American Society of Mechanical Engineers 9, 10, 27
BPMN	Business Process Model and Notation 2
CAD	computer-aided design 2, 3, 10, 13, 14, 18, 19, 23–25, 28, 29, 34, 40, 42, 45–54, 63, 69–71
CAGE	commercial and government entity 19
CAM	computer-aided manufacturing 8, 10, 24
CAx	computer-aided technologies 10
CAx-IF	Computer-Aided-”x” (Design, Manufacturing, Inspection) Interoperability Forum 26, 52, 76
CGR	constructive geometric representation 54, 56
CMS	coordinate measurement system 10, 20, 75
CNC	computer numerical control 8
DCE	distributed computing environment 30

FEA	finite-element analysis 7
G-code	G-code 4
GD&T	geometric dimensions and tolerances 8, 9
GISU	geometric-item-specific-usage 53, 54
GPS	geometrical product specifications 2, 9, 10
GUID	globally unique identifier 32
IFC	Industry Foundation Classes 31
IIoT	industrial internet of things 4, 10, 11
IIRU	item-identified-representation-usage 53
IOF	Industrial Ontology Foundry 11
ISO	International Organization for Standardization 1, 2, 9, 10
JSON	JavaScript Object Notation 11
LLAI	limited length or area indicator 22, 23
MBD	model-based definition 2, 10–12
MBE	model-based enterprise 1, 10
MBM	model-based manufacturing 10
MD5	Message Digest algorithm 5 30
MQTT	Message Queuing Telemetry Transport 11
MTConnect	Manufacturing Technology Connect 1, 4–6, 10, 11, 21, 47, 51
OEM	original equipment manufacturer 14–17, 45, 46
OID	object identifier 29, 42, 45–48, 50, 69–71
OPCUA	Open Platform Communications United Architecture 11
PLM	product lifecycle management 6, 46
PMI	product and manufacturing information 2, 8–10, 21, 23, 24, 46, 70, 71, 75
QIF	Quality Information Framework 1, 3–6, 11, 12, 21, 23, 31, 32, 40, 47, 51, 76

QPId	QIF Persistent Identifier 12
REST	REpresentational State Transfer 11
SHA-1	Secure Hash Algorithm 1 30
STEP	STandard for the Exchange of Product model data 1, 3, 5, 7–10, 13, 19, 21–24, 26, 28, 29, 32–37, 39, 40, 44, 45, 47–53, 56–58, 60, 63, 69, 72–75
SysML	Systems Modeling Language 10
THEX	Tree Hash EXchange format 42
UUID	universally unique identifier 1, 3, 5–7, 12–19, 21, 25, 27–34, 37, 39–53, 60–67, 69–76
XML	Extensible Markup Language 11, 31
XSD	XML Schema Definition 11

Appendix B. Terms and Definitions

action

An action is the identification of the occurrence of an activity and a description of its result. An action identifies an activity that has taken place, is taking place, or is expected to take place in the future. An action has a definition that is specified by an action_method. [ISO 10303-41:2022] 20, 87

action_method

An action_method is the definition of an activity. This definition includes the activity's objectives and effects. [ISO 10303-41:2022] 20, 87

advanced_brep_shape_representation

An advanced_brep_shape_representation is a shape representation made up of one or more manifold solid B-reps. Each constituent B-rep is required to have its faces and edges explicitly defined by elementary or free-form geometry. [ISO 10303-514:1999] 24, 87

advanced_face

An advanced_face is a face defined on a surface. This face is a finite portion of the surface that has its boundaries fully defined using topological entities with associated geometric curves. The surface geometry is required to be an elementary surface, a swept surface, or a B-spline surface. [ISO 10303-511:1999] 14, 24, 53, 87

aggregate_id_attribute

An aggregate_id_attribute is an identifier that provides a single identifier for more than one product data element. [ISO 10303-41:2022, modified] 33, 87

applied_action_assignment

An applied_action_assignment assigns an action to one product data element. [ISO/TS 10303-1021:2010, modified] 20, 87

applied_identification_assignment

An applied_identification_assignment assigns an identifier to one product data element. [ISO/TS 10303-1021:2010, modified] 33, 87

applied_name_assignment

An applied_name_assignment assigns a name to one product data element. [ISO 10303-41:2022, modified] 33, 87

assembly_component_usage

An assembly_component_usage relates a constituent to its assembly. [ISO 10303-44:2022, modified] 28, 35, 50, 87

characterized_object

A characterized_object is the identification of an item that has associated property information. [ISO 10303-41:2022] 23, 35, 36, 87

closed_shell

A closed_shell is a type of connected_face_set that is a shell of dimensionality 2 which typically serves as a bound for a region in R^3. A closed_shell has no boundary, and has non zero finite extent. [ISO 10303-42:2022] 24, 87

connected_face_set

A connected_face_set is a type of topological_representation_item, which is a set of faces such that the domain of the faces together with their bounding edges and vertices is connected. [ISO 10303-42:2022] 87

cylinder

A cylinder is a surface generated by rotation of a straight line parallel to a line defined as the axis. 24, 87

datum_feature

A datum_feature is a type of shape_aspect on the boundary of the product. A datum_feature may be used to establish a datum. Each datum_feature may be either a dimensional_location_with_datum_feature or a dimensional_size_with_datum_feature. [ISO 10303-47:2014] 53, 87

datum_system

A datum_system is a type of shape_aspect that represents the ordered collection of one to three datum reference compartments within a datum reference frame. Unless otherwise specified, a datum_system shall be used in accordance with the ISO 5459 definitions or ASME Y14.5-2009 or any later edition of that standard for single datum and for a datum system. [ISO 10303-47:2014] 50, 87

datum_target

A datum_target is a type of shape_aspect that indicates a datum target on the boundary of a product shape [ISO 10303-41:2022, modified] 50, 51, 87

dimensional_location

A dimensional_location is a type of shape_aspect_relationship. A dimensional_location specifies that a spatial constraint exists between two shape_aspect elements that are represented as a non-directed measure applied along a measurement path. A dimensional_location may be either an angular_location or a dimensional_location_with_path. [ISO 10303-47:2014] 53, 87

dimensional_size

A dimensional_size is a spatial characteristic of a shape_aspect that is represented by a measure. This magnitude is independent of the location of the shape_aspect on or within the product. A dimensional_size may be either an angular_size or a dimensional_size_with_path. [ISO 10303-47:2014] 53, 87

EXPRESS

data specification language that consists of language elements that allow an unambiguous data definition and specification of constraints on the data defined [ISO 10303-11] 37, 42, 87

EXPRESS-G

A formal graphical notation for the display of data specifications defined in the EXPRESS language. 37, 38, 87

face_bound

A face_bound is a loop which is intended to be used for bounding a face. [ISO 10303-42:2022] 24, 87

feature

an aspect of a part which can be (1) a tangible portion of a physical part, (2) an element of a technical drawing, 3D model, or other abstract representation of the part, or (3) a derived, constructed, intangible portion of a part, such as a feature of size, minimum circumscribed sphere, maximum inscribed cylinder, axis, center plane, theoretically extended edge on a technical drawing, and projected point/line from the part. [ISO 23952:2020] 23, 87

feature_definition_with_connection_area

A feature_definition_with_connection_area specifies one or more areas on the feature that may be attachment areas when used in a realization [ISO/TS 10303-1671:2018-11, modified] 23, 87

general_property

A general_property is an identification of a type of property. 36, 53, 87

geometric_item_specific_usage

A geometric_item_specific_usage is a type of item_identified_representation_usage that implements the ARM concept of geometric_item_specific_usage. In this specialization of item_identified_representation_usage, the identified_item is directly included in the set of items of the used_representation. [ISO 10303-1032:2014-02] 24, 53, 87

geometric_tolerance

A geometric_tolerance is the specification of the allowable range within which a geometrical property of a product may deviate. [ISO 10303-47:2014, modified to remove list of subtypes] 53, 87

globally unique identifier

See universally unique identifier. 87

id_attribute

An id_attribute is the assignment of an identifier to product data. [ISO 10303-41:2014] 33, 87

identification_item

An identification_item is an extensible list of alternate entity data types. [ISO/TS 10303-1021:2011-10] 61, 87

instanced_feature

An instanced_feature is the identification of a preconceived form pattern on a product-definition. [ISO 10303-47:2024] 23, 87

item_identified_representation_usage

An item_identified_representation_usage is an identification of a single or a set or a list of representation_item(s) within a representation as being the element(s) that describes a particular component or part of the property that is described by the representation. Conversely, the product data item specified by the item_identified_representation_usage serves as the externally visible identification of the description elements.

Example 35. In an application protocol, an instance of representation describes the shape of a product. One element of the representation - a curve - represents the boundary of a shape_aspect, a hole, in the product. Item_identified_representation_usage is used to state that the curve referenced by ’identified_item’ is a representation item for the shape_aspect referenced by ’definition’ and that the whole representation is referenced by ’used_representation’ to indicate the structural context for the curve. Geometric_representation_context plays no role in this example. [ISO 10303-41:2014, modified] 53, 87

manifold_solid_brep

A manifold_solid_brep is a type of solid_model which is a finite, arcwise connected volume bounded by one or more surfaces, each of which is a connected, oriented, finite, closed 2-manifold. There is no restriction on the number of through holes, nor on the number of voids within the volume. [ISO 10303-42:2022] 24, 87

name_attribute

An assignment of a label by which the product data is known. [ISO 10303-41:2022] 33, 87

next_assembly_usage_occurrence

A next_assembly_usage_occurrence is a type of assembly_component_usage that specifies the relationship between a child constituent and its immediate parent assembly in a product structure.

Example 36. The position and orientation of a constituent with respect to its assembly would be computed using a transformation defined in the representation_-schema in ISO 10303-43. [ISO 10303-44:2022] 52, 87

NodeSet

NodeSet is a class in the ClusterShell Python framework that supports a unified node groups syntax and external group access. A NodeSet is a named collection of nodes for use by a job. 11, 87

object_role

An object_role is a specification of a role for the association of management type data with other aspects of product data and a description of that role. [ISO 41:2022] 20, 87

observation

An observation is a historical record of something that has occurred during the life of a product or its support environment. [ISO 10303-41: 2022, modified] 18, 87

opc-ua

The OPC Unified Architecture is a cross-platform, open-source, IEC 62541 standard for data exchange developed by the OPC Foundation. 87

placed_datum_target_feature

A placed_datum_target_feature is a type of datum_target that represents the implicit definition of a datum target for tolerancing purposes. [ISO/TS 10303-1051:2019] 53, 87

placed_feature

A placed_feature is a type of shape_aspect that is the identification of a preconceived form pattern on a part resulting from the placement of a shape_feature_definition which is defined in its own context. [ISO/TS 10303-1764:2019] 23, 87

preprocessor

software unit that translates product information from the internal format of a particular computer system to an independent public domain product data format [ISO 10303-2:2024] 48, 87

product

A product is a representation of a product or a type of product [ISO 10303-41:2014] 33, 36, 52, 87

product_definition

A product_definition is a representation of an aspect of a product, or of a class of products, for an identified life cycle stage. The life cycle stage for which a product_definition exists may be further characterized by discipline, by usage, or by both.

Example 37. The design of the SS Titanic and the as-built description of the SS Titanic can be represented as two instances of product_definition for the product that represents the ship itself.

The product_definition entity data type may represent particular products that are the members of an identified class of products.

Example 38. Each individual lifeboat on the SS Titanic can be represented by an instance of product_definition, in which the associated product represents the class of products whose members are the lifeboats.

The product_definition entity data type acts as an aggregator for information about the properties of products. The usage of a product_definition in another context is specified through its participation in a product_definition_relationship as the related_product_definition in which the using context is specified by the frame_of_-reference of the relating_product_definition. If a product_definition is considered in multiple contexts, the product_definition_context_association shall be used to specify a collection of product_definition_contexts. [ISO 10303-41:2014] 20, 26, 32, 33, 36, 52, 87

product_definition_formation

a collector of definitions of a product [ISO 10303-41:2022] 33, 36, 87

product_definition_relationship

A product_definition_relationship is a relationship between two instances of the entity data type product_definition or generic_product_definition_reference and provides an identification and description of this relationship.

Example 39. The relationships within a bill-of-materials structure are examples of product_definition_relationship entity data types that associate different products. The relationship between a sketch and a detailed design is an example of a product_-definition_relationship that associates different definitions of a single product.

Example 40. The same component could be used more than once in the same assembly. Each usage of the component would be specified as an instance of the product_definition_relationship entity. [ISO 10303-41:2022] 52, 87

product_definition_shape

A product_definition_shape is a type of property_definition. It identifies the shape of a characterized_object or of one of the types reachable as characterized_product_definition. [ISO 10303-41:2014] 87

projected_zone_definition

A projected_zone_definition represents a projected zone definition as defined in GPS standards. [ISO 10303-47:2024] 87

property_definition

A property_definition is a property that characterizes a single object. [ISO 10303-41:2014] 20, 36, 53, 87

representation

A representation is a collection of one or more representation_item instances that are related in a specified representation_context.

Example 41. Two cartesian points P and Q (described by instances of representation_item) are related in a context A (they are elements in the same representation in context A, or are elements in different representations that share context A). It is therefore possible to calculate the distance between these points. A third cartesian point R (also described by an instance of representation_item) is not related to context A. It is not possible to determine the distance between R and P, or between R and Q.

A representation_item can be related to a representation_context directly, when it occurs as an element in a representation, or indirectly, when it is referenced through any number of intervening entities, each of type representation_item or founded_item.

A representation relates a representation_context to trees of representation_item instances each tree being rooted in one member of the set of items. A representation_item or founded_item is one node in the tree; a relationship between one representation_item or founded_item and another is an edge.

Example 42. Consider a collection of two-dimensional representation_item instances used to represent the shape of a machined part. It is not a complete description of the shape, but is suitable for certain applications such as computer-aided draughting.

Two instances of representation are not related solely because the same instance of representation_item is referenced directly or indirectly from their sets of items.

Example 43. Consider a surface that is used in the respective representations of the shape of a casting die and of the shape of the part cast in that die. The same surface is related to two distinct instances of representation_context (i.e., coordinate spaces): one for the die and one for the part by the two instances of representation. However, the two instances of representation are not related; they simply share a common representation_item.

Two instances of representation are not related solely because instances of representation_item in their sets of items are related by an instance of representation_item_relationship. [ISO 10303-43:2011] 32, 36, 87

representation_context

A representation_context is a context in which instances of representation_item are related. [ISO 10303-43:2011] 55, 87

representation_item

A representation_item is an element of representation. A representation_item participates in one or more instances of representation, or contributes to the definition of another representation_item.

Example 44. Consider two instances of representation, each having the same value for context_of_items. One is a representation of the shape of a cube and indirectly references a line as one of its edges. The second simply references the line as one of its items. There are not two occurrences of the line and its sub-tree of referenced instances of representation_item in the representation_context. Rather, the use of the line in that geometric_representation_context has been asserted twice, once in each representation. [ISO 10303-43:2011, simplified] 87

REST

Representational State Transfer is a software architecture that imposes conditions on how an API should work. 87

role_association

A role_association is the assignment of an object_role to an association of management type data with other aspects of product data. [ISO 10303-41: 2022] 20, 87

roundness_tolerance

A roundness_tolerance is a type of geometric_tolerance. Unless otherwise specified, the rules governing roundness tolerance defined in ISO 1101 or ASME Y14.5-2009 shall apply. The toleranced_shape_aspect shall be of invariance class surface revolute feature but not a helical shape. The actual surface shall lie in a tolerance zone limited by two co-planar and concentric circles with a difference in radii of the tolerance value. [ISO 10303-47:2014] 27, 87

shape_aspect

A shape_aspect is an identified element of the shape of an object.

Example 45. Consider the product_definition_shape of a bolt. One might distinguish, as an element of this shape, the concept of the threaded portion of its shank. This portion of the shape could be specified using a shape_aspect entity so that other properties, such as surface finish, may be associated with it. [ISO 10303-41:2014]

22, 24, 25, 35, 36, 53–56, 69, 87

shape_feature_definition

A shape_feature_definition is a preconceived form pattern. [ISO/TS 10303-1764:2018] 87

shape_representation

A shape_representation is a type of representation in which the aggregated data elements represent a shape. [ISO 10303-42, modified] 87

Sparkplug B

MQTT Sparkplug B is an open-source software specification that provides MQTT clients the framework to seamlessly integrate data from their applications, sensors, devices, and gateways within the MQTT Infrastructure. 11, 87

terminal

A point of access to an object intended for connection to an external network [IEC 61666:2010+AMD1:2021] 22, 23, 87

tessellated_representation

A tessellated_representation is a type of shape_representation in which the geometry is approximately represented by a tessellated model with planar facets. [ISO 10303-42:2022] 56, 87

topological_representation_item

A topological_representation_item represents the topology, or connectivity, of entities that make up the representation of an object. The topological_representation_item is the supertype for all the representation_items in the topology schema. [ISO 10303-42:2022] 56, 87