NIST Technical Note 1777

Design and Usage Guide for Version 0.92 of the Quality Information Framework Data Model and XML (Extensible Markup Language) Schemas

Yaoyao (Fiona) Zhao
Thomas Kramer
Robert Stone
Matt Hoffman
Scott Hoffman
William Rippey

http://dx.doi.org/10.6028/NIST.TN.1777

National Institute of Standards and Technology
U.S. Department of Commerce

NIST Technical Note 1777

Design and Usage Guide for Version 0.92 of the Quality Information Framework Data Model and XML (Extensible Markup Language) Schemas

Yaoyao (Fiona) Zhao
Thomas Kramer
William Rippey
Intelligent Systems Division
Engineering Laboratory

Robert Stone
Origin International Inc.

Matt Hoffman
Scott Hoffman
Validation Technologies Inc.

http://dx.doi.org/10.6028/NIST.TN.1777

November 2012

U.S. Department of Commerce
Rebecca Blank, Acting Secretary

National Institute of Standards and Technology
Patrick D. Gallagher, Under Secretary of Commerce for Standards and Technology and Director

National Institute of Standards and Technology Technical Note 1777
Natl. Inst. Stand. Technol. Tech. Note 1777, 42 pages (November 2012)
http://dx.doi.org/10.6028/NIST.TN.1777
CODEN: NTNOEF

Design and Usage Guide for Version 0.92 of the Quality Information Framework Data Model and XML (Extensible Markup Language) Schemas

Yaoyao (Fiona) Zhao, Thomas Kramer, Robert Stone, Matt Hoffman, Scott Hoffman, William Rippey

Dimensional Metrology Standards Consortium
June 2012

Contents

Table of Figures

Design and Usage Guidelines for Version 0.92 of the Quality Information Framework Data Model and XML Schemas

Foreword

The Dimensional Metrology Standards Consortium (DMSC, Inc.) is an American National Standards Institute (ANSI) accredited standards developing organization, as well as an A-Liaison to the International Organization for Standardization (ISO). The mission of the DMSC is to identify urgently needed standards in the field of dimensional metrology, and to promote, foster, and encourage the development and interoperability of these standards, along with related and supporting standards that will benefit the industry as a whole. More information about DMSC's Quality Information Framework (QIF) effort can be found at www.qifstandards.org.

The Quality Information Framework was developed by domain experts from the manufacturing quality community representing a wide variety of industries and quality measurement needs. Specific contributors include Honeywell, Lockheed Martin, National Institute of Standards and Technology, Chrysler, Mitutoyo America, MetroSage, Metrology Integrators, Validation Technologies, and others.

Executive Summary

This document describes the design approach used for implementing version 0.92 of the Quality Information Framework (QIF) data model, including implementation practices for the Extensible Markup Language (XML) Schema Definition Language (XSD) schemas that define the model and describe the rules for reading and writing QIF data files.

The purpose is to orient potential users of QIF to the organization of the data model to make their study of the details more rewarding and efficient. It should also help solution providers and users to evaluate QIF for their uses, without needing to go to the lowest technical details of the XML schemas. The material on XML practices describes consistent design practices to be used by QIF working groups who will be designing new schemas, and it should help data processing experts who will be writing code to serialize and deserialize manufacturing quality data using the XML schemas. The data model narrative focuses on the approach to modeling the initial requirements of QIF, namely the content of ANSI/ASME Y14.5 (1994), and the plans and results data elements defined in Dimensional Measuring Interface Standard (DMIS) 5.2 [1].

1 Introduction

Usage guide goal: to describe the design of the data model comprising elements defined by the QIF XML schema files.

Usage guide status: Release V0.92 of QIF comprises the QIF Library V0.92, the QMPlans V0.92 XML schema, and the QMResults V0.92 XML schema. A subsequent document, "QIF Data Dictionary" will be published that describes all details of QIF data types, and their semantics related to dimensional metrology and quality information workflow. The QMPlans.xsd file was developed after this document was edited, so the schema file will be included in release 0.92, but there is no description of its data model in this document. New versions of this Guide will be issued when substantive additions are made to the data model, or when design principles are revised.

Version 0.9 of QIF was released in September 2011. Two moderate revisions were subsequently made to improve the data model design, add the initial QMPlans.xsd, add some traceability data, and modify the object reference approach.

Intended audience: Two groups may be interested in this document for evaluating QIF for solving interoperability problems: developers of software applications and information management systems related to quality management, and users of quality management software solutions. Additionally, this document prescribes practices to be followed by DMSC working groups that will design application area data models and XML schemas in the future.

Intended use: Potential users of QIF will gain insight into the organization of the QIF data model to make their study of the details more rewarding and efficient. This guide should also help solution providers and users to evaluate QIF for their uses, without needing to go to the lowest technical details of the XML schemas. The material on XML practices, and data file examples, describe consistent design practices to be used by QIF working groups who will be designing new schemas, and it should help data processing experts who will be using the XML schemas and writing code to serialize and deserialize manufacturing quality data.

2 Design Goals of the QIF Specification

The goal of the QIF specification is to facilitate interoperability of manufacturing quality data between system software components. The QIF data model, implemented using neutral data formats, should ensure simple, effective, and accurate import and export of data.

The quality-system wide interoperability is supported by partitioning the data model between a QIF Library of common, reusable components, and several application area models. The reusable library components are referenced throughout the comprehensive quality data model

thereby ensuring interoperability and extensibility between any data producer and consumer that supports the QIF formats.

Solving the metrology interoperability problem will benefit manufacturers by avoiding wasted resources spent on non-value-added costs of translating data between the different components of manufacturing quality systems. Users should gain flexibility in configuring quality systems and in choosing commercial components, and achieve effortless and accurate flow of data within their factory walls as well as with suppliers and customers. Solution providers should be able to eliminate their efforts previously spent in data translations, and there should be increased opportunities to sell their products and expand features.

3 Simplified Work Flow for QIF Information

QIF designers believe that the scope of QIF information flow may eventually pervade modern manufacturing systems. This section focuses on the narrower scope of the QIF V0.92 data model, the first public release, which is limited to dimensional characteristics, inspection results reporting, and inspection plan generation.

The total scope of QIF will include flow of inspection data for improving manufacturing processes and product design, describing available inspection resources, specifying measurement rules, and reporting attribute inspection results. The details of the QIF data model are evolving as working groups move forward and incorporate more requirements. References [2] and [3] provide a more detailed description of the total scope of QIF.

We model a manufacturing quality system in five components as shown in Figure 1: product definition, measurement planning, measurement programming, measurement execution, and quality results analysis and reporting. Product definition is the process in which a part is designed, using computer aided design (CAD) software, based on customer requirements. High level components of product definition include: CAD, Product Lifecycle Management (PLM), and Enterprise Resource Planning (ERP) components. The key activity in the product definition process is to create a geometric model of the product from the conceptual ideas, which can then be augmented with further engineering information pertaining to fabrication processes.

To any manufacturing quality system, the crucial information generated from a product definition process is the product manufacturing information (PMI). PMI commonly includes information such as material, surface texture, roughness, color, hardness, and geometric dimensioning and tolerancing (GD&T) information. GD&T includes associations between geometric elements of the product and dimensions, tolerances, and datums.

Quality information generated in QIF format can be used as input by many other quality and manufacturing management components, including but not limited to, statistical process control (SPC), materials resource planning (MRP), measurement systems analysis (MSA),

manufacturing execution systems (MES), and computer aided manufacturing (CAM). Specific data flows are not shown because it is a design goal of QIF to not constrain the architecture of systems that will exchange data. The QIF specification only describes the content of the QIF library, and data models of each application area.

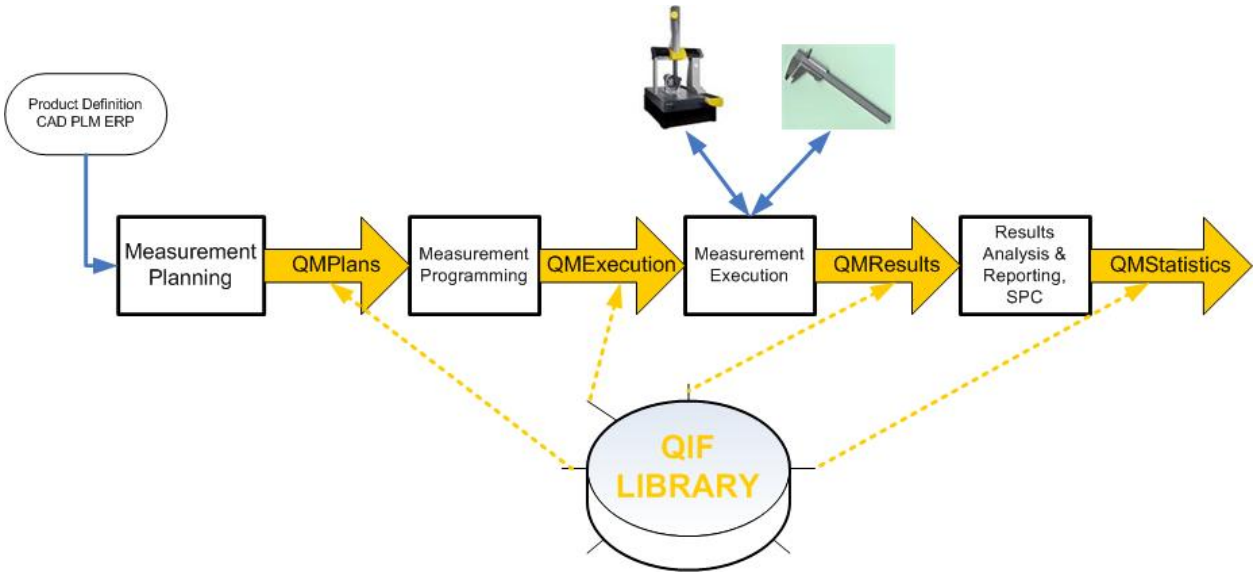


Figure 1. Simplified QIF Workflow. The QIF Library defines data elements that may be shared among the application areas QMPlans, QMExecution, QMResults, and QMStatistics.

Version 0.92 of QIF comprises the QIF library, QMPlans.xsd, and QMResults.xsd. The library is a collection of eleven XML schema files defining data elements and models that may be referenced by all quality application areas. The QMResults.xsd schema describes the formatting of data conveyed via the *QMResults* interface shown in Figure 1. The QMPlans.xsd schema describes the *QMPlans* interface. As of June 2012, the other QIF interfaces have been outlined, but work on data models has not begun.

The measurement planning activity receives all the information shown above, and takes into consideration the availability of quality measurement resources, to generate measurement plans formatted according to the *QMPlans* XML schema. These high level plans identify the measurement features and their characteristics. The plans may also specify the measuring sequence, and resources to be used, but are not required to do so. The inspection programming activity generates the machine-level measurement programs, formatted according to the *QMExecution* data model, or for example, DMIS, that provide detailed measurement operation information (i.e., probing points, scanning routes, etc.). Part inspection is carried out by the measurement execution activity, whose responsibility is typically to interpret the machine-level measurement plans, give equipment level commands to specific coordinate measuring machine (CMM) control units, collect point data, fit features to data, and output feature and characteristic

data formatted according to the *QMResults* data model. Measurement execution can also include software solutions that issue instructions to human operators using calipers, go/no-go gauges, and specialized inspection equipment, and generate results data in QIF compatible format.

The measurement results data is collected, reported, and analysed. The collection and the analysis of multiple part inspections are formatted according to the *QMStatistics* data specification. Results of single part inspection, as well as analysis of groups of parts, can provide feedback to upstream processes, such as computer aided design (CAD), statistical quality control (SQC), computer aided manufacturing (CAM), etc., to improve the production of future products.

4 Terminology

Measurement - an estimate of a dimension associated with a feature or features on a physical part generated using some physical device

Inspection - measurement of features on a physical part to determine whether the features are within allowed tolerance, commonly in order to accept or reject the part

Information element - the precise specification of format and meaning for a data element

Data element - a specific value for an information element

*Feature***:** a collection of associated points that form a continuous surface separating solid matter from free space, and are bounded by other similar constructs. Examples: 1) the surface of bore bounded by the opposed planar surfaces of a slab, 2) one surface of a propeller blade. [4]

Geometric Characteristic: a concept characterizing the size, form, orientation or location of a Feature or of a component of a feature. Examples: Diameter, Flatness, Parallelism, or Position. [4]

5 QIF Design Requirements

There are three categories of requirements on the content of the QIF data model, and on how the data are encoded for exchange:

- **Business case functional requirements**. The design of the QIF data model is driven by the functional requirements of activities that import, process, and export manufacturing quality data. Requirements are expressed via natural language rules, scenarios, use case notation, identification of specifications and/or standards, and examples. The DMSC began with a baseline requirement to model ANSI/ASME Y14.5 (1994), and the plans

and results information in DMIS 5.2. The requirements list is evolving as the members of the quality community identify more workflow requirements. Functional requirements involve engineering data, workflow details, as well as business case justifications and requirements.

- **Interoperability requirements**. This is primarily the requirement that developers from different companies, without cooperation, can develop software applications that will successfully exchange QIF data packages. This requirement is essential because solutions are developed globally by diverse developers, but also because QIF information must flow between companies, between original equipment manufacturers and contractors, and between primaries and subcontractors. This requirement is met primarily through complete and accurate semantic annotations embedded within the XML schema files, and by publishing specifications and usage documents.
- **Computational requirements**. These are good practices of computer science that lead to efficiency of coding and use of computation resources. Examples are use of class inheritance, and schema design practices that minimize size of data files.

The following functional requirements are the current goals for QIF. Not all have been implemented in version 0.92. Requirements that have been validated by inspection and examples in V0.92 include ASME Y14.5 (1994) and DMIS 5.2. Requirements of inspection planning, traceability, and first article inspection reports are current topics for working groups in 2012. Requirements for statistical process control and analysis workflow will be formalized when the QMStatistics working group is formed.

Requirements met in QIF V0.92:

- Information requirements for QIF V0.92 encompass the principles of geometric dimensioning and tolerancing as described in ANSI/ASME Y14.5 (1994), as well as workflow practices and quality management functions. DMSC plans to meet requirements of the 2009 version in the near future.
- QIF V0.92 encompasses all planning and results information defined in DMIS 5.2. DMIS data describing sensors will be considered by a working group that will develop a model for quality resources.
- The neutral data format specifications are accompanied by fully defined semantics derived where applicable from other standards like DMIS 5.2, and AS9102a First Article Inspection [5]. The semantics ensure that data cannot be misinterpreted between sender and receiver of QIF data files,
- Inspection results data, QMResults, can be used for both a reverse engineering process where actual measurement data is stored without the presence of nominal information because it is unknown, or for a conventional measurement process where inspection is planned using nominal part feature data.

- QIF V0.92 data files will carry all information defined for first article inspection reports as defined in AS9102a.
- QIF facilitates traceability of quality results to inspection and measurement processes, including identification of measurement devices and operators, software applications, CAD models. Currently this is a general requirement that may be refined by validation against formal standards and specifications.
- QIF data files maintain links so that inspection data can be used to monitor, control, and improve manufacturing processes. Scope includes support of manufacturing process and product validation.
- A general design guideline is to allow re-analysis of measured point data, after a QMResults file has been written.
- A general design guideline is to support quality systems based on model-based design, as well as systems that implement 2D drawing based processes.

Requirements to be met in future versions of QIF:

- QIF must carry data to support functionality of variation management as defined in IAGC AS9103 - Variation Management of Key Characteristics.
- Identify key and critical characteristics in inspection plans and results files.
- Support managing measuring and test equipment (MTE) performance as specified in ANSI/NCSLI Z540.3-2006, Requirements for the Calibration of Measuring and Test Equipment.
- Non-dimensional information will include: Surface Textures, Specifications (e.g., Thread Specs, Welding, General Property Attributes (e.g., Notes, Markings, Cosmetics)
- QIF will convey attribute data (e.g., Paint scratches, Dents, Burrs, etc.) and Binary data as described by AIAG QMD 1.0, D-25 [6].

Requirements to be reviewed for applicability to QIF:

- ISO 8062 Standard for Casting Tolerance
- Capability to convey a reference to point clouds of measured data to enable re-analysis of inspection results.

QIF is an evolving specification and additional requirements will be added. Users and developers of quality systems are encouraged to nominate new requirements, whether they are informal or standards-based.

6 QIF Data Model Structures

QIF V0.92 attempts to follow a decoupled normalized relationship model. As such, many relations between instanced types in the QIF schemas are made using identifiers rather than allowing a parent type to directly contain a child type (as you would find in a strictly hierarchical model). Each “object” or “instanced type” that needs to be referenced by another type is given a unique identification designator (id). The referencing object may then establish a relation by calling out the id rather than redefining the entire data element within itself. By decoupling the child from the parent we introduce the ability to reuse components of the relationship without duplicating definitions. This concept of decoupling is one of the pillars of QIF extensibility.

The QIF models represent measurement feature and characteristic objects with four aspects: instance, definition, nominal, and actual. The relationships between data objects of each aspect type in a data file are implemented using a relationship scheme described in more detail in section 6.5. Because the scope of QIF covers the entire lifecycle of quality systems, the aspects allow QIF data files to describe nominal and measured GD&T quality data, as well as express the relationships and other data generated during the workflow of planning, execution, reporting, and analysis.

We will use the terms “data object” and “object” to mean a grouping of information in a QIF XML data file, defined by elements of the QIF data model.

6.1 Four Aspects of Features Data

Feature information is defined in the QIF library using four aspects: definition, nominal, actual, and instance. In a QIF data file each feature data object has a unique identifier, and relationships between objects are expressed by references to the identifiers as shown in Figure 5. These four library data types were designed to express quality information beyond the scope of solely inspection results reporting. The instance aspect, in particular, includes information related to part design as well as information generated by planning activities.

The four aspects of feature data will be illustrated with the simple example of a plate with four holes.

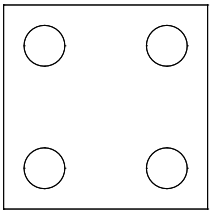


Figure 2. A plate with four holes

This plate with holes can exist in several contexts: it may be a 2D drawing, it may be a 3D CAD model, it may be a CMM inspection plan or program, it may be an actual physical part, it may be a CMM results report, or it may exist as all of the above. Regardless, in QIF the four holes would be considered as cylinder features. In the CAD or drawing contexts, nominal information for these cylinders will exist. In the physical part context actual information about the cylinders can exist if they are measured. So naturally, one would assume that QIF would only need to contain objects that define feature nominal and feature actual information. Such a simple approach can result in the redundant expression of data and may not mirror the design intent.

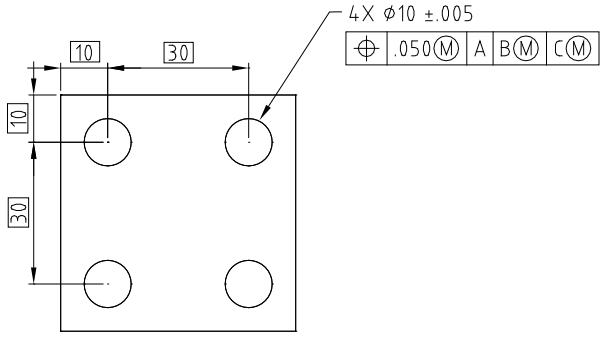


Figure 3. A plate with four holes and GD&T

The design of the plate in Figure 3 illustrates that some nominal information about the holes is shared and some is not. Each cylinder has a unique nominal location defined by basic dimensions but all have a shared nominal diameter. QIF therefore splits the nominal information over a shareable feature definition and a non-shareable feature nominal.

A **feature definition** data object is intended to be reusable, in that it includes information (e.g., cylinder: diameter), that is independent of a specific instance of the feature (e.g., a specific hole in our example). A single feature definition can be referenced by many nominal feature objects. Only nominal feature objects may reference feature definition objects.

A **feature nominal** data object adds additional feature information to the feature definition by defining information unique to a particular instance of a feature. For example, a **CylinderFeatureDefinitionType** provides the diameter for a cylinder, while a **CylinderFeatureNominalType** references the **CylinderFeatureDefinitionType**, and gives the location of the axis and optional target points on a specific cylinder on a part to be measured. Nominal objects may only reference feature definition objects.

If a part is measured then each cylinder will have unique location and size information. Nothing can be shared when it comes to feature actual data.

A **feature actual** data object provides feature information that has been measured or constructed. For example, a **CylinderFeatureActualType** will contain the actual location, orientation, and size of a cylinder. For an inspection that has been programmed from CAD data, QMResults data files may also include a related nominal feature object (which in turn, has a related feature definition object), with a reference linking the two. For feature actual data generated during a reverse engineering process, a QMResults data file may not contain nominal feature data. Thus, feature actual objects may reference a feature nominal object but do not necessarily have to do so.

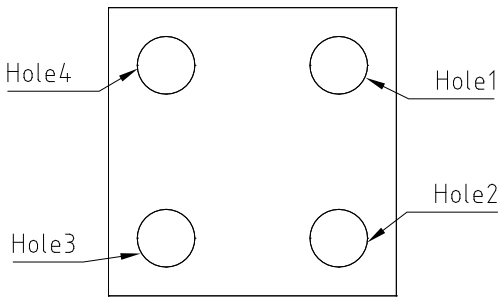


Figure 4. A plate with four holes with names.

A **feature instance** data object represents an instance of a feature at any stage of the metrology process - before or after a feature has been measured. The feature instance data object provides: the name assigned to the feature (as in Fig 4), optional links to upstream CAD data, reference to a part definition object ID, and a reference to either a nominal feature object or an actual feature object. If an actual feature is referenced, the corresponding nominal feature (if there is one) may be found through the actual feature object.

If a feature is measured several times, it is expected that a feature instance data object will be defined for each measurement, and will reference a different actual feature data object for each measurement. Some other examples of workflow that cause instantiation of a feature instance object include: any process at any time that requires a named feature, planning inspection of a part, free-form measurement of a part for reverse engineering purposes, bringing a legacy CMM report into QIF, or mining a legacy CMM program for nominal features and characteristics.

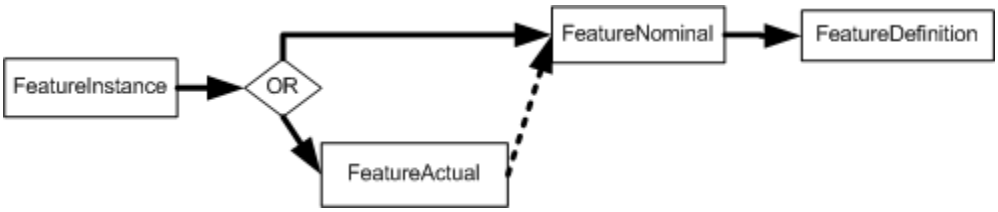


Figure 5. The reference connections among feature data objects in an XML data file. Solid lines show required references, dashed lines show optional references.

6.2 Four Aspects of Characteristic Types

As with features, characteristics have four aspects defined in the QIF library: definition, nominal, actual, and instance. Data objects of each aspect type can be linked in an XML data file to express the semantics of GD&T and quality workflow using the scheme described in section 6.5. As with features, the aspects cover a quality workflow scope wider than solely results reporting.

The **characteristic definition** is the part of a characteristic that can be shared among different characteristics. An example would be a standard diameter tolerance; one manufacturer, for instance, has a standard diameter tolerance for sheet metal parts of (+.25/-.04) mm regardless of the diameter. As another example, one often sees tolerances specified for dimensions based on the number of decimal places: dimensions to one decimal place are ±.2 mm, those to two decimal places are ±.05 mm, etc.

The **characteristic nominal** is that part of a characteristic that is not shared among different characteristics, not to be confused with sharing a characteristic among several features. An example would be a diameter tolerance for a set of holes in a pattern all with the same diameter. That shared diameter becomes the target value in the nominal characteristic. Very often, each

characteristic definition will only be referenced by a single characteristic nominal, the pair together representing one call-out on a print such as the diameter with tolerance in Fig 3.

The **characteristic actual** is the evaluation of the characteristic based on feature measurement data. There will be one characteristic actual for each measured characteristic. Just as with feature actual information, there is no shareable information among characteristic actuals.

The **characteristic instance** is the mechanism used to apply a tolerance to an individual feature. Our plate with holes would have one characteristic instance for each hole because each hole will have an individual tolerance condition: the characteristic actual in QMResults. Each instance would reference the single shared characteristic nominal, and in turn that characteristic nominal would reference a single characteristic definition which may or may not be referenced by other characteristic nominals.

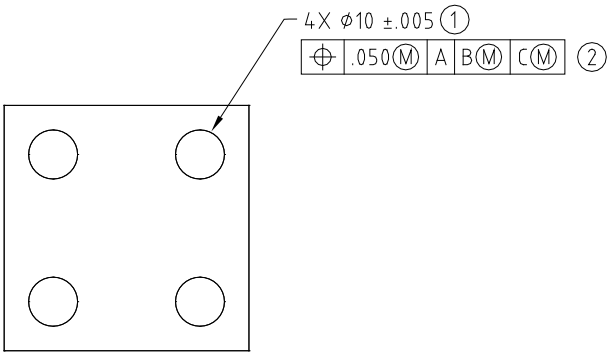


Figure 6. A plate with ballooned tolerances.

In the world of QMPlans, the idea of a characteristic instance may at first seem a bit redundant, but it allows for a unique identifying name, label, or “balloon” to be assigned on a per-feature basis. Our diameter tolerance might be “ballooned” on the print (as in Fig 6) as characteristic number 1, then each characteristic instance could be labeled 1A, 1B, 1C or 1_1, 1_2, 1_3 depending on company standards.

6.3 Two Aspects of Part Information

A part is a physical object that contains one or more features. The **Parts** element is a second level element in the QMResults data model, a sub-element of **MeasurementResults**, which is a sub-element of the top level element **Inspection**. The two sub-elements of the Parts element are **PartDefinitions** and **PartInstances**.

A **PartDefinition** element includes: material, units, and names of part drawings, CAD models, or files containing simple graphics renderings of the part.

A **PartInstance** element includes: **PartSerialNumber**, a reference to a PartDefinition element, reference to a transform element, feature instance sub-elements, feature actual sub-elements, and inspection status for the part (e.g., pass, fail, rework, etc.).

6.4 Handling Assemblies of Parts in QIF

Standard AS9102a [5] requires the tracking of inspection results for assemblies; that is, inspected objects that may comprise more than one discrete part. To describe this condition the QIF data model defines substitution group elements for **PartDefinition** and **PartInstance** that describe assemblies, or types **AssemblyDefinitionType** and **AssemblyInstanceType**. The Assembly types reuse or extend the Part definitions in a recursive way. Thus a Part can be a single physical object, possibly with unique serial number, or the QIF data file may contain a substitution of several assembly elements, each of which extends the properties of the **PartInstanceType** by adding a **SequencedPartsInstances** element. A **SequencedAssemblyPartInstance** contains a sequence ordinal paired with a **PartInstanceID**. The sequence ordinal may indicate order of inspection or may be used as an index to group related part information in blocks 15, 16, 17, and 18 on AS9102a Form 1. Individual parts in an assembly may make reference to their own individual **PartTraceability** elements.

6.5 Making Connections Between Data Objects

In QIF data files, many connections between data objects are made by using an identifier and a reference to the identifier. For example the connection from a nominal feature to a feature definition is made by putting an identifier in the feature definition and a reference to the identifier in the nominal feature. All identifiers in the QIF schemas are of type **QIFIdType**. All references to identifiers are of type **QIFReferenceType**. In a data file, the reference contains the same symbol as the identifier of the object being referenced. For example, if the value of the **id** attribute of an instance of **ArcFeatureDefinitionType** is `321` then the value of the **FeatureDefinitionId** element of an instance of **ArcFeatureNominalType** that uses that definition is also `321`.

We want to ensure that connections made using identifiers and references join the correct types of data objects. For example, a reference from a nominal arc feature to its definition must identify an arc feature definition, not any other type of object such as a transformation or a cone definition. To ensure that identifier/reference pairs make matches between objects of the correct types, the QIF schemas contain several hundred key/keyref pairs, one for each identifier/reference pair. Key and keyref are standard parts of the XML schema language. Readily available XML data file checkers will check whether or not key/keyref constraints are satisified in a data file governed by an XML schema. A detailed description of how key and keyref work may be found in books about the XML schema language such as [7]. In a nutshell (and oversimplifying), a key/keyref pair locates two places in an (upright) tree of data that must contain identical data. The two places are identified by describing the two paths that go upward

to them from a common starting point. The key/keyref pair is located at the common starting point.

The key/keyref pair mechanism is very good at catching errors in matching identifiers and references, but it is not foolproof. In all cases, when there are many objects of the same type, no automated check (such as key/keyref) will know which one is intended. For example, if a data file has a set of instances of **ArcFeatureNominalType** and a set of instances of **ArcFeatureDefinitionType**, only the builder of the data file will know which nominal is supposed to go with which definition. In other cases, there is no way to define a key that discriminates between similar types of object because they are mixed together at the same location (in a set of transformations containing both actual transformations and nominal transformations, for example).

Finally, there are a few cases in the QIF schemas where a key/keyref pair could be written, but has not been.

7 QIF Handling of Units

The QIF design seeks to handle units simply and unambiguously in data files, especially to allow quantities to appear without explicit units specified for each value. QIF uses a scheme where primary/default and alternate units are specified once in a data file. Quantities using the default units for length, angle, or temperature can occur in data files without an explicit attribute giving the name of the unit. Alternate units can be assigned to individual quantities by including an attribute giving the name of the unit. All unit names must be unique for a given unit type.

Primary and alternate units are specified in a QIF data file, such as a QMResults data file, by using the **FileUnits** element defined in Units.xsd. The **FileUnits** element specifies a primary unit for each of the unit types used in the data file, and optional alternate units. The three primary units types used in QMResults are angle, length, and temperature. Quantities expressed in the primary units can be written in a data file without any explicit mention of a unit. If any quantity of a given unit type appears in a data file, the corresponding data type must appear in the **PrimaryUnits** or the **OtherUnits** of the **FileUnits**. Common XML file checkers will signal an error if this rule is violated.

For example, a data file might give a diameter as follows:

```
<Diameter>7.5</Diameter>.
```

If the **LengthUnit** in the **PrimaryUnits** is millimeter, the data on the line above would mean that the diameter is 7.5 millimeters. This association occurs because the **Diameter** element is of **LengthValueType** in the schema.

The default unit for all unit types is the SI unit (meter, radian, kelvin, etc.). If it is desired to have a primary unit type not be a SI unit, a **UnitConversion** element must be included in the declaration of the primary unit. The **UnitConversion** element gives an **Offset** and a multiplication **Factor** that may be used to convert values of the primary unit type to values in terms of SI units.

For example, the meter is the SI unit for length. If a user wants to use the millimeter as the primary length unit in a data file, the user puts the following lines into the **FileUnits** portion of the data file:

```
<PrimaryUnits>
  <LengthUnit>
   <SIUnitName>meter</SIUnitName>
   <UnitName>millimeter</UnitName>
   <UnitConversion>
    <Factor>0.001</Factor>
    <Offset>0</Offset>
   </UnitConversion>
  </LengthUnit>
 ...
</PrimaryUnits>
```

In the **FileUnits** portion of the data file, wherever a unit is declared, the name of the SI unit may be given regardless of whether it is the primary unit or not. If the **UnitConversion** is not included in the data file, the **UnitName** just serves as an alias for the SI unit. For example if the unit type is **LengthUnit**, the **SIUnitName** must be **meter** if it used, but the **UnitName** might be **meter** or **m**, or anything else the user likes. If the **UnitConversion** is included in the data file, naming the SI unit makes it clear what units result from applying the conversion. The conversion is always accomplished using the equation:

SI = ((X plus **Offset**) times **Factor**)

where SI is the value in SI units, and X is the value in declared units.

The **FileUnits** element also includes an **OtherUnits** sub-element for specifying alternate units. For example, a **LengthUnit** named inch could be defined in the **OtherUnits** element as follows.

```
<OtherUnits>
  <LengthUnit>
   <SIUnitName>meter</SIUnitName>
   <UnitName>inch</UnitName>
   <UnitConversion>
    <Factor>0.0254</Factor>
    <Offset>0</Offset>
```

```
    </UnitConversion>
  </LengthUnit>
  ...
</OtherUnits>
```

If a quantity in a data file is represented using an alternate unit, the name of the unit type must be given. If the definition for inch just given is used in a data file, a diameter of 5 inches in a data file would be expressed as follows:

```
<Diameter lengthUnit="inch">5</Diameter>
```

8 XML Naming and Design Rules

The QIF data model is implemented using XML [8]. XML was chosen for QIF design and data file encoding because the basic XML specifications are supported as open, public domain, royalty-free standards, and because XML is very widely used: educated experts and users are easy to find. Furthermore, tools for incorporating XML into software projects are widely available.

Certain conventions have been used in the development of the QIF XML schemas:

- All type definitions are declared globally, i.e., as direct children of a root schema element. In other words, no type definition is embedded inside another type definition. This convention is commonly called using the venetian blind pattern.
- Names are descriptive and formed by concatenation without abbreviation. All concatenated words except possibly the first start with an upper case letter. The next bullet gives the rule for the case of the first letter of the first word.
- XML Type and Element names start with an upper case letter. XML Attribute names start with a lower case letter.
  Example Type name: ArcFeatureNominalType
  Example Attribute name: nominalsCalculated
- Almost all type names end in "Type".
- The universal resource identifier (URI) for the QIF namespace is:

  qif="http://www.qifstandards.org".

The XML schema language includes a "documentation" node type that may be used to put documentation into a schema. Documentation nodes must be preserved by XML tools. Comments may also be inserted in schemas but are not necessarily preserved by XML tools. A first round of inserting documentation nodes in the QIF schemas has been performed, but adding

more documentation nodes is planned. In many places, the meaning of the schemas is not obvious and requires explanation. Documentation nodes can provide that.

9 QIF Library Schema Files

A design goal for the centralized data model library is to avoid overlapping or redundant and syntactically different models for the same concept. There are eleven files in the QIF library.

9.1 FeatureTypes.xsd

Measurement features are generated by analyzing the association between GD&T, PMI, and product design geometric elements. In the QIF realm and widely accepted in the metrology and manufacturing quality fields, features are defined as the underlying targets to which GD&T is applied. These features are usually tangible and are often defined as points, curves, planes, spheres, etc. that can actually be seen on the physical part to be measured.

The FeatureTypes.xsd schema defines 28 feature types. Almost all of the feature types defined in FeatureTypes.xsd have an equivalent type definition in the DMIS 5.2 standard. A comparison of feature definitions in QIF and DMIS is shown in Figure 7. QIF features are described using Cartesian coordinates. Notes may be attached to any of the feature aspects.

QIF Feature	**Equivalent DMIS Feature**
Arc	ARC (format 1)
Attribute	GEOM, OBJECT (possibly)
Circle	CIRCLE
Composite	
Compound	COMPOUND
Cone	CONE
ConicalSegment	CONRADSEGMNT
ConstantCrossSection	*no equivalent*
Cuboid	RCTNGL
Cylinder	CYLNDR
CylindricalSegment	CYLRADSEGMNT
EdgePoint	EDGEPT
Ellipse	ELLIPS
ElongatedCylinder	ELONGCYL
Line	LINE
Pattern	PATERN
Plane	PLANE
PointCurve	GCURVE
Point	POINT
PointSurface	GSURF
Slot2D	CPARLN
Slot3D	PARPLN
Slot3DWithDraft	SYMPLN
Sphere	SPHERE
SphericalSegment	SPHRADSEGMNT

```
SurfaceOfRevolution          REVSURF
ToroidalSegment              TORRADSEGMNT
Torus                        TORUS
```

Figure 7. Comparison of feature definitions in QIF and DMIS

The hierarchy of feature type derivations is illustrated in Figure 8, using the example of the circle feature family. The types for definition, instance, nominal, and actual define the four aspects of feature data objects that can occur in an XML data file.

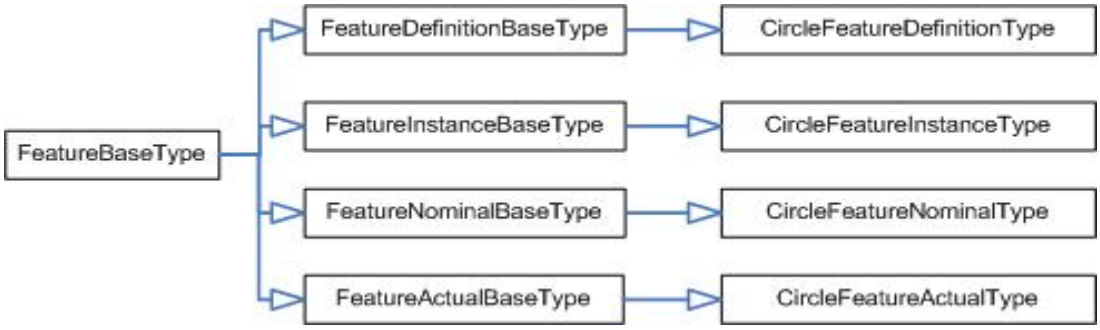


Figure 8. Hierarchy of type definitions for feature data object types

9.2 CharacteristicAspects.xsd

The top level element of the CharacteristicAspects.xsd schema file is Characteristics. The Characteristics element is used (via "ref") in both the QMPlans schema and the QMResults schema. Of the 2500 lines in the file, only about 200 are used to give the hierarchy under the Characteristics element. About 1600 lines are used to define several hundred key/keyref pairs that check that the four aspects of characteristics are connected correctly across all characteristic types. Roughly 700 lines contain substitution group declarations for characteristics.

9.3 CharacteristicTypes.xsd

GD&T and PMI information is categorized as the characteristic information in QIF data models, and is modeled in the CharacteristicTypes.xsd schema file. As with features, characteristics have four aspects: definition, nominal, actual, and instance. Also as with features, the XML Schema language definitions for characteristics form a type hierarchy, but the characteristics hierarchy is longer and deeper. There are four hierarchies, one for each aspect. Each of the four is headed by **CharacteristicBaseType**. The prototype followed by each of the four hierarchies is shown in Figure 8. The word "Aspect" is shown in the names in the figure where one of the four aspect names (instance, definition, nominal, or actual) would appear.

The **CharacteristicBaseType** and other types whose name includes “Base” are abstract types and cannot be instantiated. The hierarchy has 43 leaf nodes that are not abstract and can be instantiated.

```
CharacteristicBaseType
  CharacteristicAspectBaseType
    AttributeCharacteristicAspectType
    DimensionalCharacteristicAspectBaseType
      AngularCharacteristicAspectBaseType
        AngleBetweenCharacteristicAspectType
        AngleCharacteristicAspectType
      CoordinateCharacteristicAspectType
      LinearCharacteristicAspectBaseType
        ChordCharacteristicAspectType
        CurveLengthCharacteristicAspectType
        DiameterCharacteristicAspectType
        DistanceBetweenCharacteristicAspectType
        HeightCharacteristicAspectType
        LengthCharacteristicAspectType
        RadiusCharacteristicAspectType
        SquareCharacteristicAspectType
        WidthCharacteristicAspectType
    GeometricCharacteristicAspectBaseType
      FormCharacteristicAspectBaseType
        CircularityCharacteristicAspectType
        CylindricityCharacteristicAspectType
        FlatnessCharacteristicAspectType
        StraightnessCharacteristicAspectType
        StraightnessCylindricalZoneCharacteristicAspectType
      LocationCharacteristicAspectBaseType
        ConcentricityCharacteristicAspectBaseType
          ConcentricityCylindricalZoneCharacteristicAspectType
          ConcentricitySphericalZoneCharacteristicAspectType
        PositionCharacteristicAspectBaseType
          PositionAngularZoneCharacteristicAspectType
          PositionBoundaryZoneCharacteristicAspectType
          PositionConicalZoneCharacteristicAspectType
          PositionCylindricalZoneCharacteristicAspectType
          PositionElongationZoneCharacteristicAspectType
          PositionPlanarZoneCharacteristicAspectType
          PositionRadialZoneCharacteristicAspectType
          PositionSphericalZoneCharacteristicAspectType
        SymmetryCharacteristicAspectType
      OrientationCharacteristicAspectBaseType
        OrientationCylindricalZoneCharacteristicAspectBaseType
          AngularityCylindricalZoneCharacteristicAspectType
          ParallelismCylindricalZoneCharacteristicAspectType
          PerpendicularityCylindricalZoneCharacteristicAspectType
        OrientationPlanarZoneCharacteristicAspectBaseType
          AngularityPlanarZoneCharacteristicAspectType
          ParallelismPlanarZoneCharacteristicAspectType
          PerpendicularityPlanarZoneCharacteristicAspectType
      ProfileCharacteristicAspectBaseType
        LineProfileCharacteristicAspectType
        PointProfileCharacteristicAspectType
```

SurfaceProfileCharacteristicAspectType
SurfaceProfileNonUniformCharacteristicAspectType
RunoutCharacteristicAspectBaseType
CircularRunoutCharacteristicAspectType
TotalRunoutCharacteristicAspectType
NumericalCharacteristicAspectType
UserDefinedCharacteristicAspectType

Figure 9. Characteristic definitions in QIF

9.4 ConstructedFeatureTypes.xsd

Constructed features have the same range of shapes as other features, but a constructed feature can be a non-tangible feature type constructed from other tangible feature types. An example might be an offset plane, where a plane is defined in space based upon several target points at given offset distances.

Constructions of actual features are modeled in the ConstructedFeatureTypes.xsd schema file. Sixteen of the 28 feature types can be constructed, as shown in Table 1. There is a complexType in the schema file for each X in the table. Each of those complexTypes is a construction definition type and includes a documentation node containing a brief explanation of how the construction should be done.

In addition to the eleven construction types shown in the table, four features have additional construction types, as follows.

- Circle – FromCone, Tangent
- Line – Parallel, Perpendicular
- Plane – Offset, Parallel, Perpendicular
- Point – FromCone, CenterOfGravity, Pierce, MovePoint, MovePointVector, MovePointAxis, Extreme

Each construction definition type used in a data file is attached to a constructed feature instance type by being the value of a **ConstructionDefinition** element. The constructed feature instance type is a derived type of the corresponding feature instance type with that one added element. For example, **ConstructedPlaneFeatureInstanceType** is derived from **PlaneFeatureInstanceType**. Also, each feature type has a constructed version for the definition, nominal, and actual aspects; none of these has additional elements or attributes. For example, **ConstructedPlaneFeatureActualType** is derived from **PlaneFeatureActualType**.

9.5 PrimitiveTypes.xsd

The PrimitiveTypes.xsd schema file defines 54 complexTypes and 21 simpleTypes of various sorts that are used by other schema files. Nineteen of the simpleTypes are enumerations (for example a **SlotEndType** must be one of “ROUND”, “FLAT”, or “OPEN”).

Construction Feature	BestFit	Cast	Copy	Extract	FromScan	Intersection	Mid	Projection	Recompensated	TangentThrough	Transform
Arc	X	X	X	X	X			X	X		X
Circle	X	X	X		X	X		X	X	X	X
Cone	X	X	X		X				X		X
Cylinder	X	X	X		X				X		X
EdgePoint		X	X		X				X		X
Ellipse	X	X	X		X	X		X	X		X
Line	X	X	X	X	X	X	X	X	X	X	X
Plane	X	X	X	X	X		X		X	X	X
Point		X	X		X	X	X	X	X		X
PointCurve	X		X	X							X
PointSurface	X		X	X							X
Slot2D	X	X	X		X			X	X		X
Slot3D	X	X	X		X				X		X
Slot3DWithDraft	X	X	X		X				X		X
Sphere	X	X	X		X				X		X
Torus	X	X	X		X				X		X

Table 1: Feature constructions

The complexTypes in this schema file are almost all very different from one another. One modest exception to that is that there are seven transform types.

9.6 QIFTypes.xsd

The QIFTypes.xsd schema file is similar to PrimitiveTypes.xsd in that it defines various types used in other schema files. However, the only application area schema files that are currently technically complete are QMResults.xsd and QMPlans.xsd, so it is not possible to determine whether types are assigned appropriately to PrimitiveTypes.xsd and QIFTypes.xsd. When other application area schema files are technically complete, it will be useful to consider whether PrimitiveTypes.xsd and QIFTypes.xsd might be merged or have types moved from one to the other.

9.7 PartTypes.xsd

The top level element in the PartTypes.xsd schema file is Parts. The Parts element is used (via "ref") in the QMPlans schema and the QMResults schema. Of the 2800 lines in the file, only about 400 are used to define the hierarchy under Parts. About 1700 lines are used to define several hundred key/keyref pairs that check that the aspects of features are connected correctly. About 700 lines define substitution groups for features.

9.8 Transforms.xsd

The Transforms.xsd schema file defines five coordinate system transform types and a Transforms element.

9.9 Units.xsd

This schema file defines units for values of angle, area, force, length, mass, pressure, speed, temperature, and time. The QMResults.xsd schema currently uses only **AngleValueType**, **LengthValueType**, and **TemperatureValueType**.

All quantities except tolerances in the QIF schemas have a specified type of unit. The Units.xsd schema file defines the following value types that have units:

- AngleValueType
- AreaValueType
- ForceValueType
- LengthValueType
- MassValueType

- PressureValueType
- SpeedValueType
- TemperatureValueType
- TimeValueType

9.10 MeasurementDevices.xsd

This schema file defines the basic information describing measurement devices such as probe accuracy, stylus length, touch-triggering force, etc. QIF V0.92 contains a modest outline of data elements. The DMSC plans to establish a working group, called QMResources, that will define the data model for quality measurement resources. The QIF concept is that QIF XML data formats would be used for: the description of available measurement resources in an enterprise, resources assigned to inspection plans and programs, and maintaining traceability and archiving of resources applied to products.

9.11 Traceability.xsd

The purpose of traceability information is to make it possible to associate QIF data files with the product inspected, and the resources used to design it, manufacture it, program its inspection, and inspect it. DMSC divides process traceability information between inspection processes and manufacturing processes. Currently the QIF data model describes inspection resources in more detail than manufacturing resources, relying on activities like MES to use QIF references to manufacturing databases to extract details. Traceability information includes: part serial number, workpiece temperature, customer and order number descriptions, inspection operator id, manufacturing process id, process machine id, detailed description of measurement devices, etc. The current data were largely influenced by the requirement of QIF to convey the data specified by AS 9102a. DMSC is seeking input on other requirements for traceability data.

10 The QMResults.xsd Schema File

The current XSD module structure for QMResults.xsd is shown in Figure 10. In the figure, a schema at one level of indentation includes all the schemas below it at the next level of indentation. There are no circular references, but some schemas are included multiple times; this is allowed by XSD rules and does not confuse XML tools. Using this centralized QIF data model library, the different but complementary application area models avoid the creation of redundant and syntactically different modes for the same concept.

QMResults.xsd
  CharacteristicAspectsTypes.xsd
  PartTypes.xsd

Transforms.xsd

CharacteristicAspectsTypes.xsd
CharacteristicTypes.xsd
CharacteristicTypes.xsd
PrimitiveTypes.xsd
QIFTypes.xsd
ConstructedFeatureTypes.xsd
FeatureTypes.xsd
PrimitiveTypes.xsd
FeatureTypes.xsd
PrimitiveTypes.xsd
QIFTypes.xsd
MeasurementDevices.xsd
QIFTypes.xsd
PartTypes.xsd
ConstructedFeatureTypes.xsd
QIFTypes.xsd
Traceability.xsd
PrimitiveTypes.xsd
Units.xsd
QIFTypes.xsd
PrimitiveTypes.xsd
Traceability.xsd
MeasurementDevices.xsd
QIFTypes.xsd
Transforms.xsd
PrimitiveTypes.xsd

Figure 10. Structure of Include Directives in QMResults.xsd

The top few levels of structure of the QMResults data model are shown in Figure 11. As shown in the figure, the root element is **Inspection**, which is of type **InspectionType**. The **InspectionType** has two elements, **Version** and **MeasurementResults**; it also has one attribute, id (not explicit on the figure). The **MeasurementResults** element is of type **MeasurementResultsType**, which has seven required elements (boxes with solid line edges) and eight optional elements (boxes with dashed line edges). Descriptions of the elements follow.

- **InspectionNotes** (optional) – whatever human-readable notes are desired. Each note has an id which can be referenced by other data objects.

- **Settings** – a File Units element (see section 3.5) and a boolean indicating whether the data is compensated
- **Traceabilities** – Each Traceability data object may contain one or more elements of **QMRTraceabilityType**, which includes: **PartID**, **OrderNumber**, **SerialNumber**, **LotNumber**, **FixtureId**, inspection start and end times, **InspectionOperator**, etc. Each element has a unique id that can be referenced by other data objects in the file, typically **Parts** elements, but potentially by **Features** and **Characteristics** as well. The QIF design for traceability will evolve as more workflow and business case requirements are incorporated.
- **DatumTargetReferences** (optional) – a list of **DatumTargetReference** elements, each of which associates a **DatumDefinition** with a **DatumTargetDefinition**.
- **DatumDefinitions** (optional) – a list of **DatumDefinition** elements, each of which assigns a **DatumLabel** to a reference feature and, optionally, has **DatumTargets**
- **DatumTargetDefinitions** (optional) – a list of **DatumTargetDefinition** elements
- **Transforms** (optional) – of **TransformsType** whose elements are optional lists of **ReportingTransforms**, **CharacteristicTransforms**, **DatumReferenceFrameTransforms**, and **CompositeDatumReferenceFrameTransforms**. All Transform elements have a unique id which can be referenced by other data objects.
- **DatumReferenceFrames** (optional) – a list of **DatumReferenceFrame** elements
- **AssociatedInformation** (optional) – contains references to additional files generated during the measurement process.
- **PartModels** (optional) – one or more definitions of part geometry.
- **Parts** – lists of **Part** elements. A part is a physical object that contains one or more features. A **Part** element links pairs of **PartDefinitions** and **PartInstances**. This also lists assemblies.
- **Characteristics** – of **CharacteristicsAndConstructsType**. Characteristic Definitions describe GD&T characteristics according to ANSI/ASME Y14.5. QIF **Construct** elements associate triads of characteristic **Definitions**, **Nominals**, and **Actuals**. The optional **CharacteristicGroups** element is a way of relating characteristics based on common association, e.g., the manufacturing process, or print callout.
- **CharacteristicFeatureRelations** – of **CharacteristicFeatureRelationsType**, is a list of **FeatureRelationType** elements, each of which associates a characteristic with a part and one or more features.
- **InspectionStatus** – of **InspectionStatusAndErrorsType**, containing a **Status** (PASS, FAIL, REWORK, etc.) and an optional list of error strings.

The example results file shown in Appendix A has the structure shown in Figure 11, which describes the top-level structure of data files written in conformance to QMResults.xsd.

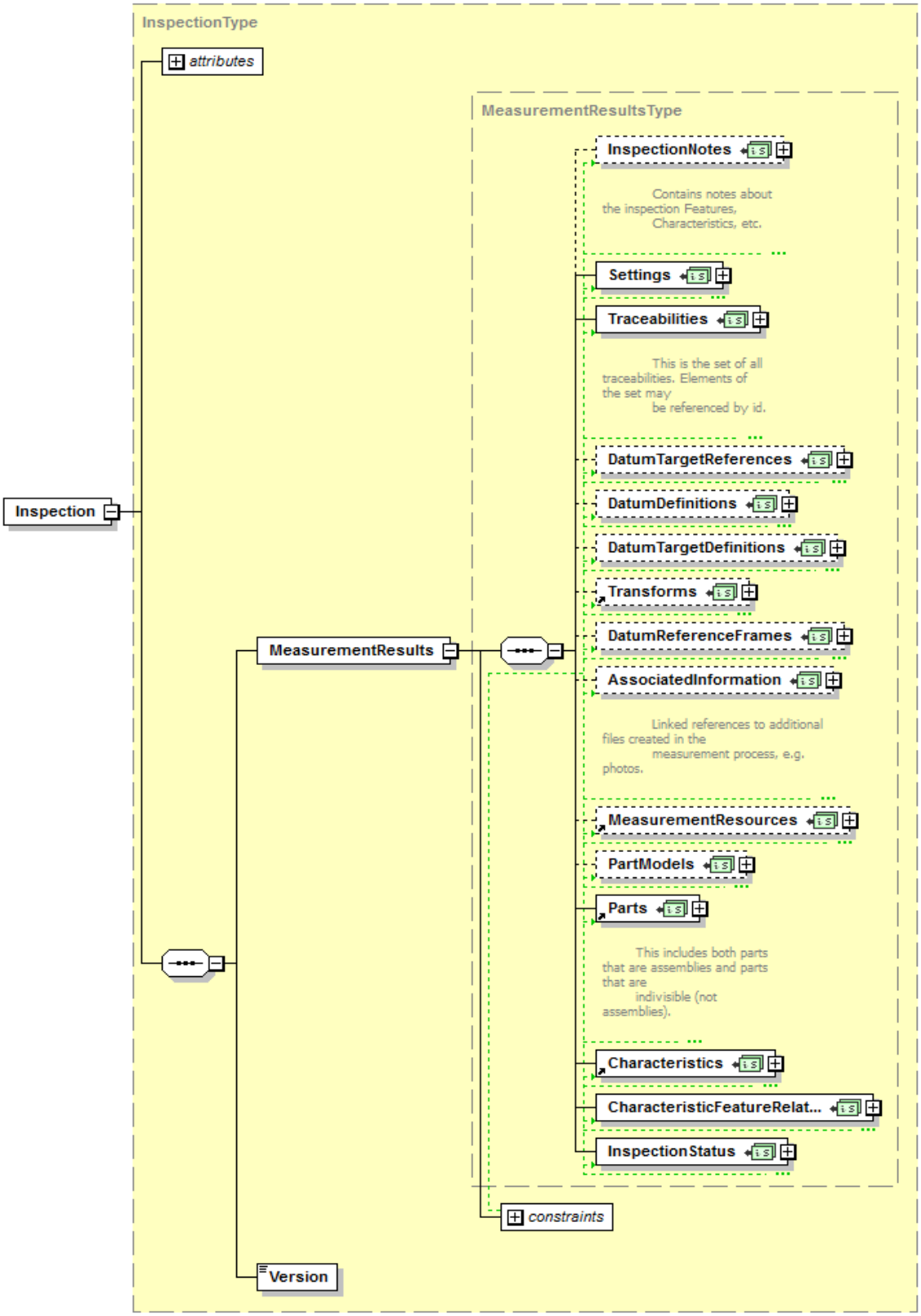


Figure 11. Top-level structure of a data file written according to QMResults.xsd

11 QIF Case Study Parts

Sample test files of simulated inspection results and plans have been generated to test the correctness and completeness of QIF V0.92. The test parts, shown in Figure 12, are:

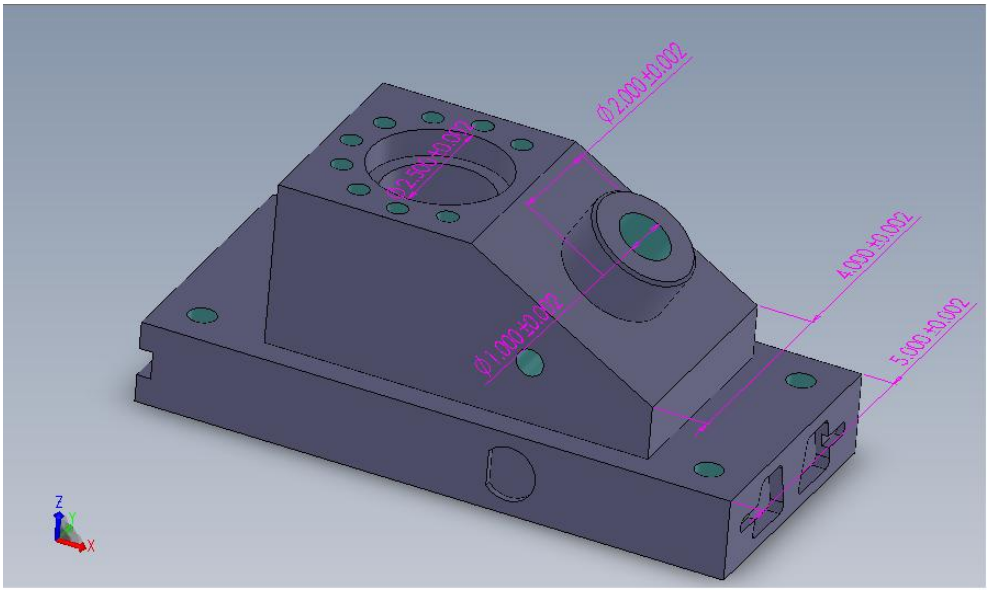
- a CMM calibration master ball
- an ANC-101 (Advanced Numerical Control program of CAM-I (Consortium for Advanced Management, International)) example part
- a sheet metal scanning measurement example part.

These case studies showed that the QMResults.xsd schema is able to represent different types of measurement features and characteristics. Since the format of results data is independent of how the data was collected, the QMResults.xsd schema is able to collect measurement results from common CMMs and optical or vision scanning devices. This is one of the key capabilities that QMResults schema must have in order to support interoperability of results data generated by different types of measurement equipment. For major end users, such as airplane manufacturers and automobile manufacturers, multiple types of measurement device are used in any shop floor. Therefore, QMResults.xsd provides a method of collecting and exchanging measurement results seamlessly without costly data translation.

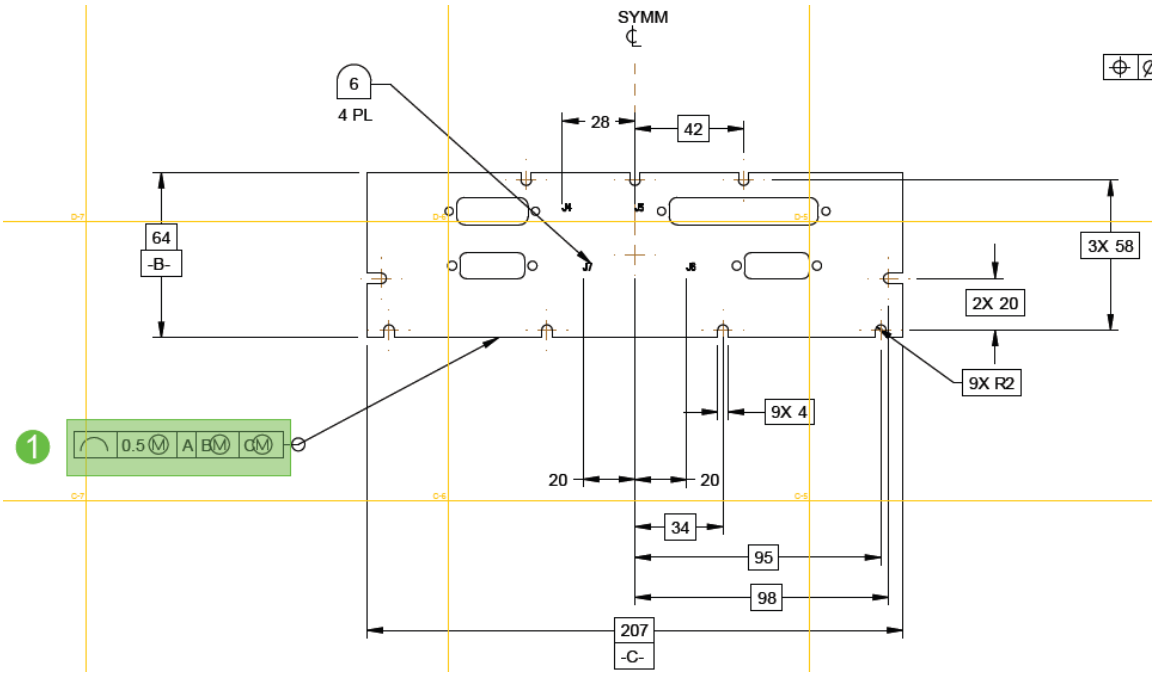
The next step in the evolution of QIF will be to create software libraries (with C++, Java, or .NET) to write data into (serialize) QMResults XML files, read data from (deserialize) QMResults XML files, and provide convenient data access functions that application programmers can use. This software development will be an open source, public domain project so members of the manufacturing quality community can suggest improvements or improve the code for the benefit of all users. Once the open source libraries are developed, it will be easy for software developers to link in the libraries and access the data, allowing them to avoid the effort of developing code that imports and exports between the QMResults schema and QIF structures. The libraries would be a toolkit for extended functionality to better use the base libraries, with continuous improvement through the open source model.

Example QMResults and QMPlans XML files for the master ball can be found in Appendix B. Examples data files for the other parts can be obtained from the QMResults Working Group.

(a) Case study 1 – CMM calibration master ball



(b) Case study 2 – ANC 101 example part



(c) Case study 3 – sheet metal example part

Figure 12. Case studies to validate QMResults.xsd

Bibliography

[1] "Dimensional Measuring Interface Standard, DMIS 5.2 Standard, Part 1," ANSI, 2009.

[2] Y. F. Zhao, T. Kramer, W. Rippey, J. Horst and F. Proctor, "An Integrated Data Model for Quality Information Exchange in Manufacturing Systems," in *37th MATADOR Conference*, UK, 2012.

[3] Y. F. Zhao, J. A. Horst, T. R. Kramer, W. Rippey and R. Brown, "Quality Information Framework – Integrating Metrology Processes," in *INCOM*, Bucharest, Romania, 2012.

[4] W. Tandler, "A miniature GD&T glossary".

[5] SAE Aerospace, *SAE 9102a Aerospace First Article Inspection Requirement, Rev A*, SAE Aerospace, 2004.

[6] AIAG, "Quality Management Data (QMD) Version 1.0," Automotive Industry Action Group,

Southfield, MI, 2007.

[7] P. Walmsley, Definitive XML Schema, Vols. ISBN-13: 978-0130655677 , 2001.

[8] W3C, "W3C XML Schema Definition Language (XSD) 1.1," World Wide Web Consortium (W3C), 2012.

APPENDICES

Appendix A – Acronyms Used

ANSI – American National Standards Institute

ASCII – American Standard Code for Information Interchange (a standard for character coding)

ASME – American Society of Mechanical Engineers

CAD – Computer-Aided Design

CAIPP – Computer-Aided Inspection Process Planning

CAM – Computer-Aided Machining

CMM – Coordinate Measuring Machine

DME – Dimensional Measuring Equipment

DMIS – Dimensional Measuring Interface Standard

DMSC – Dimensional Metrology Standards Consortium

ERP – Enterprise Resource Planning

GD&T – Geometric Dimensioning and Tolerancing

ISO – International Organization for Standardization

MES - Manufacturing Execution Systems

MRI – Measurement Resources Information

MRP – Materials Resource Planning

MSA – Measurement Systems Analysis

PMI – Product Manufacturing Information

QIF – Quality Information Framework

QMPlanning – Quality Measurement Planning

QMPlans – Quality Measurement Plans

QMResources - Quality Measurement Resources

QMResults – Quality Measurement Results

QMRules – Quality Measurement Rules

QMS – Quality Management Systems (a committee)

QMStatistics – Quality Measurement Statistics

R&R – Repeatability and Reproducibility

SI – The International System of Units

SPC – Statistical Process Control

SQC – Statistical Quality Control

STEP – STandard for the Exchange of Product model data (ISO 10303)

XML – eXtensible Markup Language (a file format)

XSD – XML Schema Definition language (an information modeling language)

Appendix B. Sample XML data files

This appendix includes one inspection results file and one inspection plan file for the master ball part. Additional sample XML data files are available from the QMResults Working Group, with the releases of the QIF schema files.

Master Ball Example file - QMResults format

```
<?xml version="1.0" encoding="UTF-8"?>
<!-- Example QMResults data file for a master ball.              -->
<!-- Created by Fiona Zhao and Tom Kramer at NIST,               -->
<Inspection
 id="OTH_MasterBall"
 xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
 xsi:noNamespaceSchemaLocation="../applications/QMResults.xsd">
 <MeasurementResults>
  <Settings>
   <Compensated>true</Compensated>
   <FileUnits>
    <PrimaryUnits>
     <AngleUnit>
      <SIUnitName>radian</SIUnitName>
      <UnitName>degree</UnitName>
      <UnitConversion>
       <Factor>0.017453293</Factor>
      </UnitConversion>
     </AngleUnit>
     <LengthUnit>
      <SIUnitName>meter</SIUnitName>
      <UnitName>mm</UnitName>
      <UnitConversion>
       <Factor>0.001</Factor>
```

```
      </UnitConversion>
     </LengthUnit>
     <TemperatureUnit>
      <SIUnitName>kelvin</SIUnitName>
      <UnitName>kelvin</UnitName>
     </TemperatureUnit>
    </PrimaryUnits>
    <OtherUnits/>
  </FileUnits>
</Settings>
<Traceabilities>
  <Traceability id="traceability1">
    <PartInstanceId>PRT_MasterBall</PartInstanceId>
    <MeasurementDeviceIds>
      <MeasurementDeviceId><Id>CMM_1</Id></MeasurementDeviceId>
    </MeasurementDeviceIds>
    <StandardAndVersion>ASME-Y14.5 1994</StandardAndVersion>
  </Traceability>
</Traceabilities>
<DatumReferenceFrames>
  <DatumReferenceFrame id="DRF_1">
    <NominalDatumFeature>
      <FeatureInstanceId>FISP_MasterBall</FeatureInstanceId>
    </NominalDatumFeature>
  </DatumReferenceFrame>
</DatumReferenceFrames>
<AssociatedInformation id="OTH_3">
  <URI/>
</AssociatedInformation>
<MeasurementResources>
  <MeasurementDevices>
    <CMM id="CMM_1">
      <Name>Our CMM</Name>
      <InspectionTemperatures>
        <MachineTemperature id="OTH_2">
          <Temperature>300</Temperature>
          <TimeStamp>2011-09-12T11:36:21Z</TimeStamp>
        </MachineTemperature>
      </InspectionTemperatures>
    </CMM>
  </MeasurementDevices>
</MeasurementResources>
<Parts>
  <PartDefinitions>
    <PartDefinition id="PRD_MasterBall">
      <FeatureAspects>
        <FeatureDefinitions>
          <SphereFeatureDefinition id="FDSP_MasterBall">
            <Diameter lengthUnit="mm">20.0</Diameter>
          </SphereFeatureDefinition>
        </FeatureDefinitions>
        <FeatureNominals>
          <SphereFeatureNominal id="FNSP_MasterBall">
            <FeatureDefinitionId>FDSP_MasterBall</FeatureDefinitionId>
            <CenterPoint>
              <X>0</X> <Y>0</Y> <Z lengthUnit="mm">10.0</Z>
```

```
            </CenterPoint>
           </SphereFeatureNominal>
          </FeatureNominals>
         </FeatureAspects>
        </PartDefinition>
       </PartDefinitions>
       <PartInstances>
        <PartInstance id="PRT_MasterBall">
         <PartDefinitionId>PRD_MasterBall</PartDefinitionId>
         <FeatureInstances>
          <SphereFeatureInstance id="FISP_MasterBall"
           geometry="ThreeDimensional"
           materialDirection="NOT_APPLICABLE"
           featureOfSize="true">
           <PartDefinitionId>PRD_MasterBall</PartDefinitionId>
<!--            <FeatureActualId>FASP_MasterBall</FeatureActualId> -->
           <FeatureNominalId>FNSP_MasterBall</FeatureNominalId>
           <FeatureName>MasterBall</FeatureName>
           <AnalysisMode>LeastSquares</AnalysisMode>
          </SphereFeatureInstance>
         </FeatureInstances>
         <FeatureActuals>
          <SphereFeatureActual id="FASP_MasterBall">
           <PointList>
            <MeasurePoint id="MPA_MasterBallZ">
             <Point> <X>0</X> <Y>0</Y> <Z>10.01</Z> </Point>
             <Normal> <I>0</I> <J>0</J> <K>1</K> </Normal>
             <MeasurementDeviceId>CMM_1</MeasurementDeviceId>
             <Sequence>1</Sequence>
             <Compensated>true</Compensated>
            </MeasurePoint>
            <MeasurePoint id="MPA_MasterBallXPositive">
             <Point> <X>10.01</X> <Y>0</Y> <Z>0</Z> </Point>
             <Normal> <I>1</I> <J>0</J> <K>0</K> </Normal>
             <MeasurementDeviceId>CMM_1</MeasurementDeviceId>
             <Sequence>2</Sequence>
             <Compensated>true</Compensated>
            </MeasurePoint>
            <MeasurePoint id="MPA_MasterBallXNegative">
             <Point> <X>-10.00</X> <Y>0</Y> <Z>0</Z> </Point>
             <Normal> <I>-1</I> <J>0</J> <K>0</K> </Normal>
             <MeasurementDeviceId>CMM_1</MeasurementDeviceId>
             <Sequence>3</Sequence>
             <Compensated>true</Compensated>
            </MeasurePoint>
            <MeasurePoint id="MPA_MasterBallYPositive">
             <Point> <X>0</X> <Y>10.02</Y> <Z>0</Z> </Point>
             <Normal> <I>0</I> <J>1</J> <K>0</K> </Normal>
             <MeasurementDeviceId>CMM_1</MeasurementDeviceId>
             <Sequence>4</Sequence>
             <Compensated>true</Compensated>
            </MeasurePoint>
            <MeasurePoint id="MPA_MasterBallYNegative">
             <Point> <X>0</X> <Y>-10.01</Y> <Z>0</Z> </Point>
             <Normal> <I>0</I> <J>-1</J> <K>0</K> </Normal>
             <MeasurementDeviceId>CMM_1</MeasurementDeviceId>
```

```
            <Sequence>5</Sequence>
            <Compensated>true</Compensated>
          </MeasurePoint>
        </PointList>
        <FeatureNominalId>FNSP_MasterBall</FeatureNominalId>
        <CenterPoint> <X>0</X> <Y>0</Y> <Z>0</Z> </CenterPoint>
        <Diameter>20</Diameter>
        <DiameterMin>19.95</DiameterMin>
        <DiameterMax>20.05</DiameterMax>
      </SphereFeatureActual>
      <CircleFeatureActual id="FACIR_MasterBall">
        <CenterPoint> <X>0</X> <Y>0</Y> <Z>0</Z> </CenterPoint>
        <Normal> <I>0</I> <J>0</J> <K>1</K> </Normal>
        <Diameter>20</Diameter>
      </CircleFeatureActual>
    </FeatureActuals>
    <Status>PASS</Status>
  </PartInstance>
 </PartInstances>
</Parts>
<Characteristics>
 <CharacteristicDefinitions>
  <DiameterCharacteristicDefinition id="CHDD_MasterBall">
   <Name>MasterBallDiameter</Name>
   <Tolerance>
    <MaxValue> <Lval>0.05</Lval> </MaxValue>
    <DefinedAsLimit>true</DefinedAsLimit>
   </Tolerance>
  </DiameterCharacteristicDefinition>
 </CharacteristicDefinitions>
 <CharacteristicNominals>
  <DiameterCharacteristicNominal id="CHND_MasterBall">
   <Name>MasterBallDiameterCharacteristicNominal</Name>
   <DefinitionId>CHDD_MasterBall</DefinitionId>
   <TargetValue>20</TargetValue>
  </DiameterCharacteristicNominal>
 </CharacteristicNominals>
 <CharacteristicActuals>
  <DiameterCharacteristicActual
   id="CHAD_MasterBall">
   <Name>MasterBallDiameterCharacteristicActual</Name>
   <Status>INTOL</Status>
   <NominalId>CHND_MasterBall</NominalId>
   <CoordinateType>CARTESIAN</CoordinateType>
   <Value>
    <ActualValue> <Lval>20</Lval> </ActualValue>
   </Value>
  </DiameterCharacteristicActual>
 </CharacteristicActuals>
 <CharacteristicInstances>
  <DiameterCharacteristicInstance id="CHID_MasterBall">
   <Name>MasterBallDiameterCharacteristic</Name>
   <Measurements>
    <Measurement>
     <Sequence>1</Sequence>
     <ActualFeatureId>FASP_MasterBall</ActualFeatureId>
```

```
        </Measurement>
        <Measurement>
          <Sequence>2</Sequence>
          <ActualFeatureId>FASP_MasterBall</ActualFeatureId>
        </Measurement>
        <Measurement>
          <Sequence>3</Sequence>
          <ActualFeatureId>FASP_MasterBall</ActualFeatureId>
        </Measurement>
        <Measurement>
          <Sequence>4</Sequence>
          <ActualFeatureId>FASP_MasterBall</ActualFeatureId>
        </Measurement>
        <Measurement>
          <Sequence>5</Sequence>
          <ActualFeatureId>FASP_MasterBall</ActualFeatureId>
        </Measurement>
      </Measurements>
      <ActualId>CHAD_MasterBall</ActualId>
<!--    <NominalId>CHND_MasterBall</NominalId> -->
    </DiameterCharacteristicInstance>
   </CharacteristicInstances>
  </Characteristics>
  <CharacteristicFeatureRelations>
   <Relation id="OTH_5">
    <PartInstanceId>PRT_MasterBall</PartInstanceId>
    <CharacteristicInstanceId>CHID_MasterBall</CharacteristicInstanceId>
    <FeatureInstanceIds>
     <FeatureInstanceId><Id>FISP_MasterBall</Id></FeatureInstanceId>
    </FeatureInstanceIds>
   </Relation>
  </CharacteristicFeatureRelations>
  <InspectionStatus>
   <Status>PASS</Status>
  </InspectionStatus>
 </MeasurementResults>
 <Version>1</Version>
</Inspection>
```

Master Ball Example file - QMPlans format

```
<?xml version="1.0" encoding="UTF-8"?>
<!-- Example QMPlans data file for a master ball.             -->
<!-- Created by Tom Kramer at NIST.                           -->
<!-- This plan file might be the plan whose execution creates -->
<!-- the qmresults2masterball.xml file.                       -->
<MeasurementPlan
  id="OTH_MasterBall"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:noNamespaceSchemaLocation="../applications/QMPlans.xsd">
  <FileUnits>
    <PrimaryUnits>
      <AngleUnit>
        <SIUnitName>radian</SIUnitName>
        <UnitName>degree</UnitName>
        <UnitConversion>
          <Factor>0.017453293</Factor>
```

```
      </UnitConversion>
    </AngleUnit>
    <LengthUnit>
      <SIUnitName>meter</SIUnitName>
      <UnitName>mm</UnitName>
      <UnitConversion>
        <Factor>0.001</Factor>
      </UnitConversion>
    </LengthUnit>
    <TemperatureUnit>
      <SIUnitName>kelvin</SIUnitName>
      <UnitName>kelvin</UnitName>
    </TemperatureUnit>
  </PrimaryUnits>
  <OtherUnits/>
</FileUnits>
<Traceabilities>
  <Traceability id="traceability1">
    <PartInstanceId>PRT_MasterBall</PartInstanceId>
    <MeasurementDeviceIds>
      <MeasurementDeviceId><Id>CMM_1</Id></MeasurementDeviceId>
    </MeasurementDeviceIds>
    <StandardAndVersion>ASME-Y14.5 1994</StandardAndVersion>
  </Traceability>
</Traceabilities>
<MeasurementResources>
   <MeasurementDevices>
    <CMM id="CMM_1">
      <Name>Our CMM</Name>
    </CMM>
  </MeasurementDevices>
</MeasurementResources>
<Setup>
  <DatumReferenceFrames>
    <DatumReferenceFrame id="DRF_1">
      <NominalDatumFeature>
        <FeatureInstanceId>FISP_MasterBall</FeatureInstanceId>
      </NominalDatumFeature>
    </DatumReferenceFrame>
  </DatumReferenceFrames>
</Setup>
<QualityControlPlan>
  <Sampling>
    <SamplingPlans>
      <SamplingPlan id="SamplingPlan_1">
        <SampleSize>1</SampleSize>
      </SamplingPlan>
    </SamplingPlans>
    <MeasurementCycleTime>
      P1D
    </MeasurementCycleTime>
  </Sampling>
</QualityControlPlan>
<Parts>
  <PartDefinitions>
    <PartDefinition id="PRD_MasterBall">
      <FeatureAspects>
```

```
        <FeatureDefinitions>
          <SphereFeatureDefinition id="FDSP_MasterBall">
            <Diameter lengthUnit="mm">20.0</Diameter>
          </SphereFeatureDefinition>
        </FeatureDefinitions>
        <FeatureNominals>
          <SphereFeatureNominal id="FNSP_MasterBall">
            <FeatureDefinitionId>FDSP_MasterBall</FeatureDefinitionId>
            <CenterPoint>
              <X>0</X> <Y>0</Y> <Z lengthUnit="mm">10.0</Z>
            </CenterPoint>
          </SphereFeatureNominal>
        </FeatureNominals>
      </FeatureAspects>
    </PartDefinition>
  </PartDefinitions>
  <PartInstances>
    <PartInstance id="PRT_MasterBall">
      <PartDefinitionId>PRD_MasterBall</PartDefinitionId>
      <FeatureInstances>
        <SphereFeatureInstance id="FISP_MasterBall"
          geometry="ThreeDimensional"
          materialDirection="NOT_APPLICABLE"
          featureOfSize="true">
          <PartDefinitionId>PRD_MasterBall</PartDefinitionId>
          <FeatureNominalId>FNSP_MasterBall</FeatureNominalId>
          <FeatureName>MasterBall</FeatureName>
          <AnalysisMode>LeastSquares</AnalysisMode>
        </SphereFeatureInstance>
      </FeatureInstances>
      <Status>NOT_MEASURED</Status>
    </PartInstance>
  </PartInstances>
</Parts>
<Characteristics>
  <CharacteristicDefinitions>
    <DiameterCharacteristicDefinition id="CHDD_MasterBall">
      <Name>MasterBallDiameter</Name>
      <Tolerance>
        <MaxValue> <Lval>0.05</Lval> </MaxValue>
        <DefinedAsLimit>true</DefinedAsLimit>
      </Tolerance>
    </DiameterCharacteristicDefinition>
  </CharacteristicDefinitions>
  <CharacteristicNominals>
    <DiameterCharacteristicNominal id="CHND_MasterBall">
      <Name>MasterBallDiameterCharacteristicNominal</Name>
      <DefinitionId>CHDD_MasterBall</DefinitionId>
      <TargetValue>20</TargetValue>
    </DiameterCharacteristicNominal>
  </CharacteristicNominals>
  <CharacteristicInstances>
    <DiameterCharacteristicInstance id="CHID_MasterBall">
      <Name>MasterBallDiameterCharacteristic</Name>
      <NominalId>CHND_MasterBall</NominalId>
    </DiameterCharacteristicInstance>
  </CharacteristicInstances>
```

```
  </Characteristics>
  <CharacteristicFeatureRelations>
    <Relation id="OTH_5">
      <PartInstanceId>PRT_MasterBall</PartInstanceId>
      <CharacteristicInstanceId>CHID_MasterBall</CharacteristicInstanceId>
      <FeatureInstanceIds>
        <FeatureInstanceId><Id>FISP_MasterBall</Id></FeatureInstanceId>
      </FeatureInstanceIds>
    </Relation>
  </CharacteristicFeatureRelations>
  <UnorderedPlanRoot>
    <Steps>
      <MeasureEvaluateAll/>
    </Steps>
  </UnorderedPlanRoot>
</MeasurementPlan>
                                    --/--
```