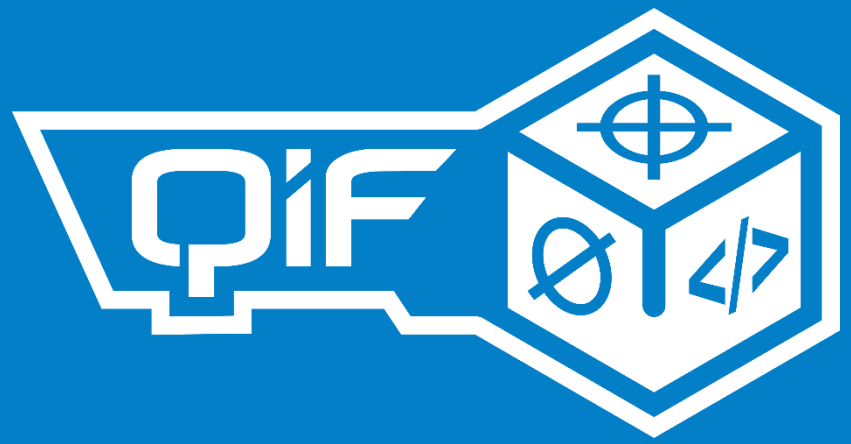


Digital Metrology Standards Consortium DMSC

DMSC's QIF Tutorial Suite

April 23, 2020

QIF Tutorials

Basics, Technology, and Applications

QIF 101: Understanding QIF Basics –
Curtis Brown, Honeywell FM&T (April 23, 2020)

QIF 201: Reviewing QIF Technology –
Daniel Campbell, Capvidia (April 30, 2020)

QIF 301: Coding QIF Applications –
Tom Kramer, NIST Guest Researcher (May 7, 2020)

QIF 101:
Understanding QIF Basics

Curtis Brown
Lead Mechanical Engineer
Honeywell FM&T*
Kansas City, MO
DMSC, President

* The Department of Energy's Kansas City National Security Campus is operated and managed by Honeywell Federal Manufacturing & Technologies, LLC under contract number DE-NA0002839

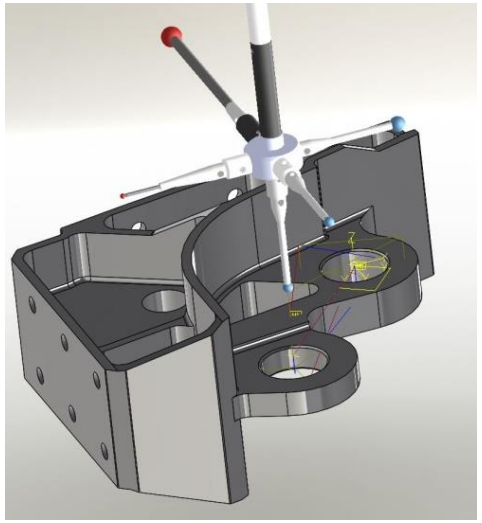
QIF 101: Syllabus

- Digital Metrology Standards Consortium
- Manufacturing Quality Activity Workflow
- Quality Information Framework
- Digital Transformation via MBE

Who is the DMSC?

Digital Metrology Standards Consortium

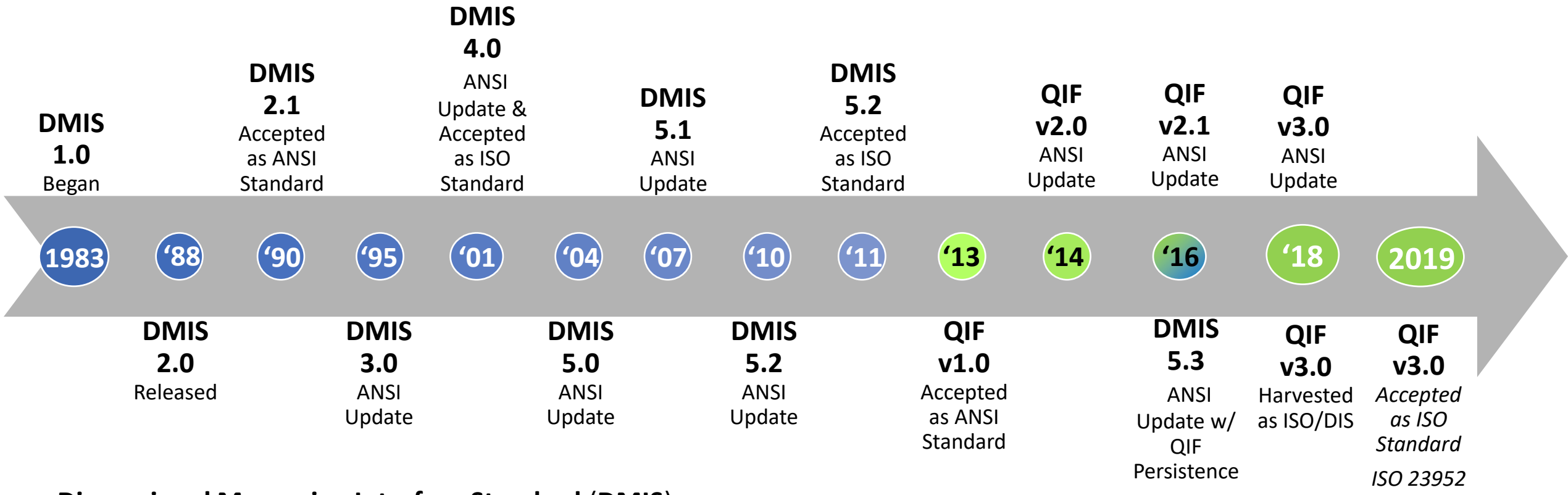
- A **non-profit**, cooperative sponsorship, **consortium** organization. Conceived & sponsored in 1983; Separate legal entity 2005, dba Digital MSC, 2018.
- Dedicated to identifying, promoting, fostering, and encouraging the **development** and **interoperability** of information standards that benefit the digital metrology community.
- Preparing standards that **impact** digital **model-based quality** enterprise.
- A professional **group** of manufacturing metrologists, software developers, and innovators worldwide. Note: 500+ years of experience contributed to the QIF.
- Maintainers of **Dimensional Measuring Interface Standard** (**DMIS**) standard.
- Developers & maintainers of **Quality Information Framework** (**QIF**) standard.
- **ANSI accredited** standards making organization
- **A-Liaison** member of **ISO / TC 184 / SC 4** (allows for harvesting ANSI standards)



DMSC Objectives . . .

- To **reduce** the **cost of quality**
- To **develop** and maintain **trusted digital** interoperability standards
- To **impact** digital **metrology** within manufacturing and specifically within Model-Based Enterprise
- To **enable** the **freedom** to choose solutions
  - **best** in **class**
  - **best** in **value**

DMSC's Quality Standards Pedigree



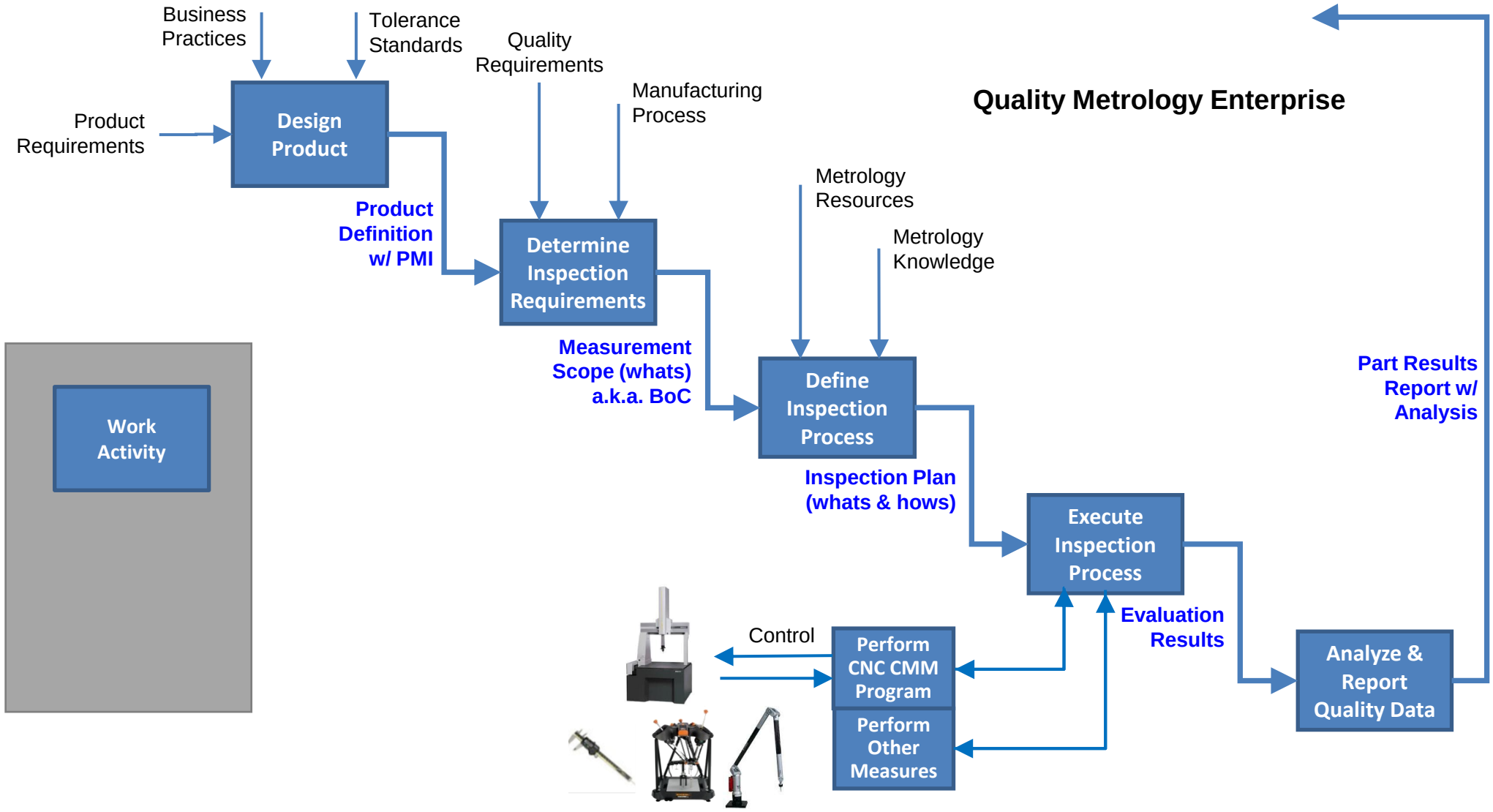
Dimensional Measuring Interface Standard (DMIS)

Quality Information Framework (QIF)

ISO 22093:2011 – Industrial automation systems and integration – Physical device control – Dimensional Measuring Interface Standard (DMIS)

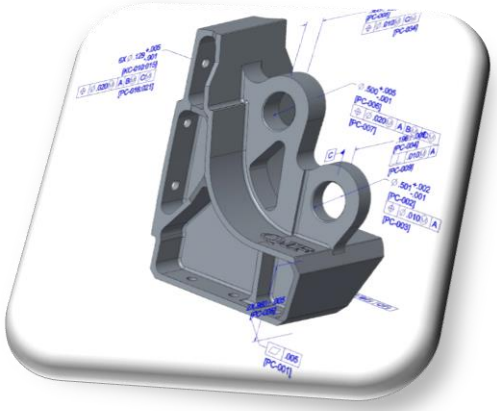
ISO 23952:2020– Quality Information Framework (QIF) – An integrated model of manufacturing quality information via ISO /TC 184/SC 4/WG 15

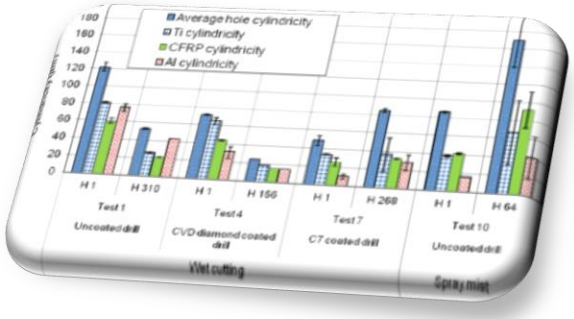
Manufacturing Verification / Product Acceptance Activity Workflow



The QIF Standard

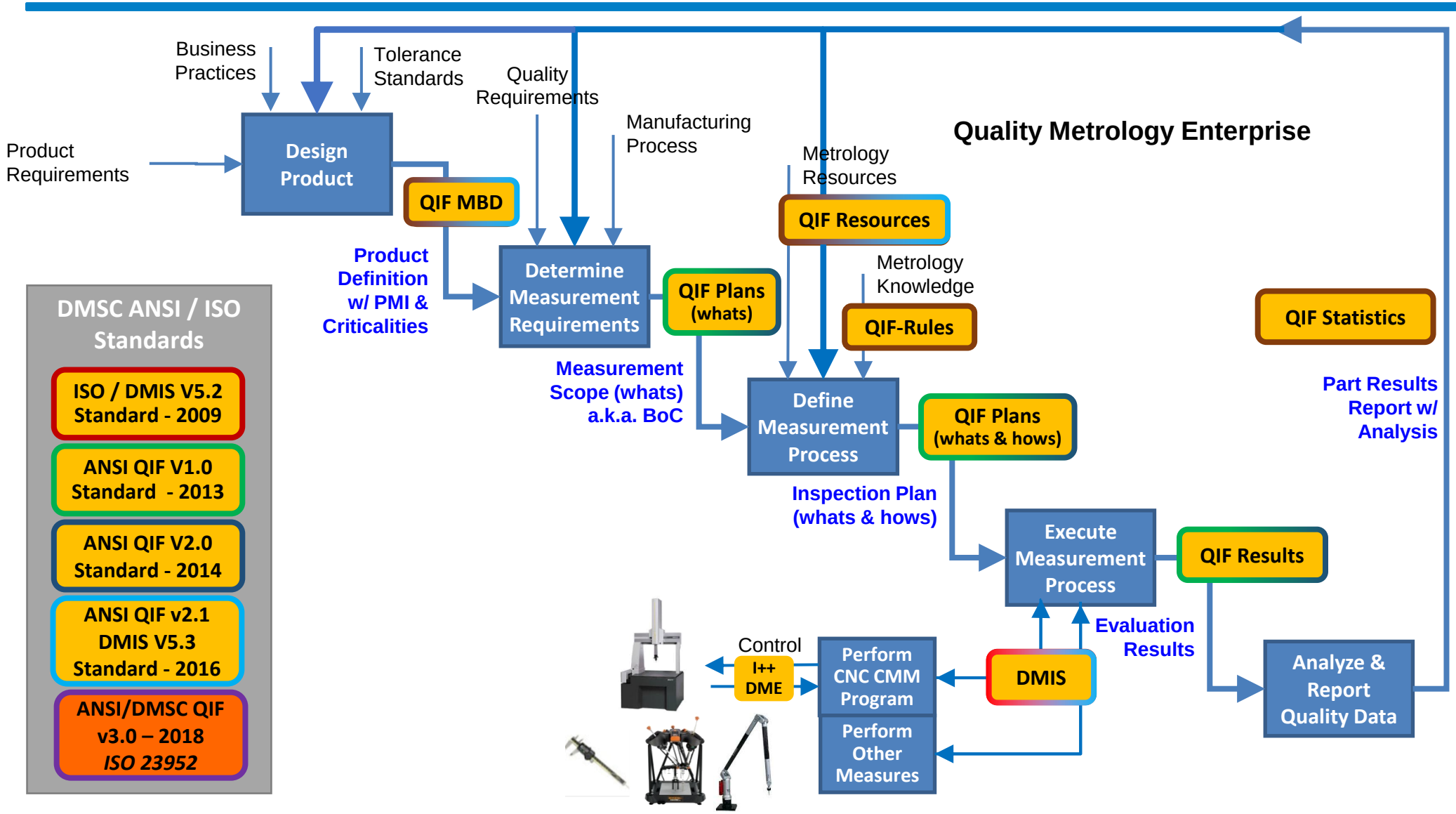
- *Quality Information Framework (QIF) – ANSI/DMSC QIF v3.0 - 2018*
- *An Integrated Model for Manufacturing Quality Information*
- Defines, Constrains, and Exchanges:
  - *Model-Based Definition*
    - *Feature-Based (Metrology/Measurement)*
    - *Semantic PMI (Characteristics)*
  - *Quality Planning*
    - *Whats: qBOM = Bill of Characteristics (BoC)*
    - *Hows: Inspection Plan (Methods)*
  - *Measurement Execution*
    - *DMIS 5.3 w/QIF QPIds*
  - *Measurement Results*
    - *Piece Part*
    - *Statistical*



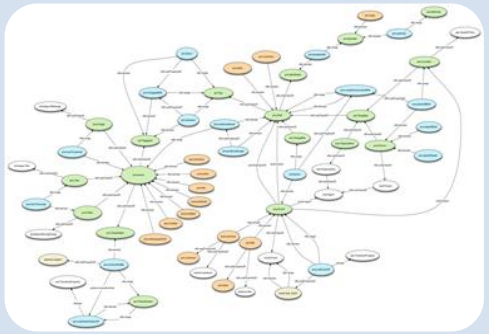


- *Enterprise Connectivity for Quality Feedback*
  - *QIF's Quality Persistent ID (QPId) (i.e., Universal Unique ID (UUID))*
  - 651aded1-ff04-498a-968e-044147a2506d

QIF Activity Workflow



What is the QIF?



Characteristic-Centric, Feature-Based Ontology of Manufacturing Quality Metadata

(Structured Data)

< />

XML Technology: Simple, Modern Implementation with Built-In Code Validation

(Modern Approach)

Information Semantically Linked to the Model-Based Definition for Full Information Traceability

(Connected Data)

ANSI
American National Standards Institute

Approved ANSI Interoperability Standard

Harvested by ISO/TC 184/SC 4 as ISO/QIF 23952

(Standard Artifacts)

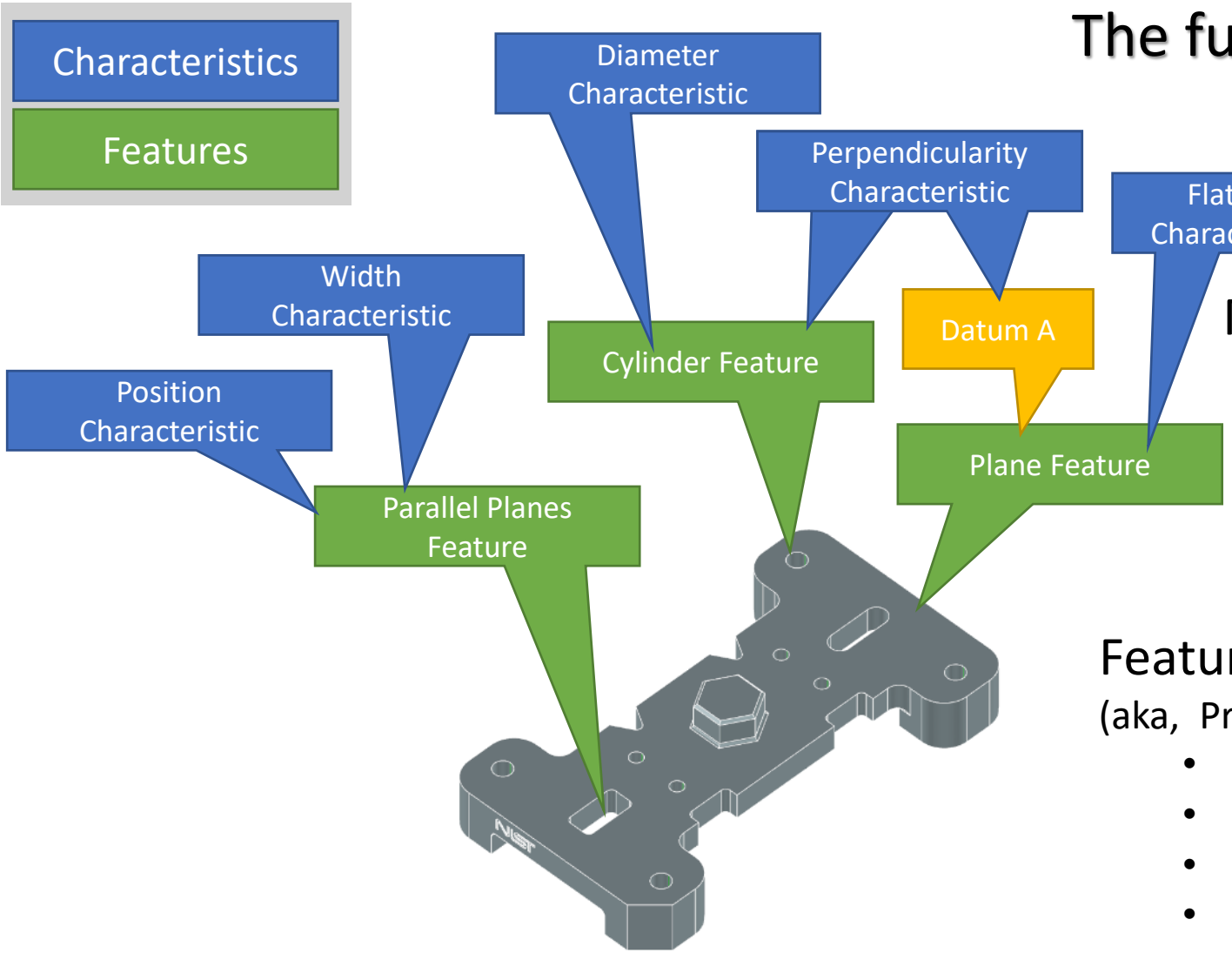
QPIds – Persistent UUID within the QIF

QIF Persistent Identifier (QPId) *noun* Cu·pid \ˈkyü-pəd\ 

- Universally Unique Identifier (UUID) (adopted by Microsoft as a GUID)
  - ISO/IEC 9834-8
  - 550e8400-e29b-41d4-a716-446655440000
- Chances of generating two that are the same within the universe are practically nil.
  - 340000000000000000000000000000000000 (3.4×10^{38}) possible UUIDs
- **Allows information to be combined later without resolving identifier conflicts**
- Many software development libraries generate UUIDs
- QPIds uniquely identify
  - QIF Document
  - QIF Plan
  - QIF Result
  - QIF Rule Set
  - Feature Item
  - Characteristic Item 
  - Product Item
  - Resource Item

An Important Mechanism that facilitates Lifecycle Connectivity w/ Traceability

QIF: Features & Characteristics



The fundamental constructs behind QIF:

- Features &
- Characteristics

Model geometry is wrapped by ***Features***

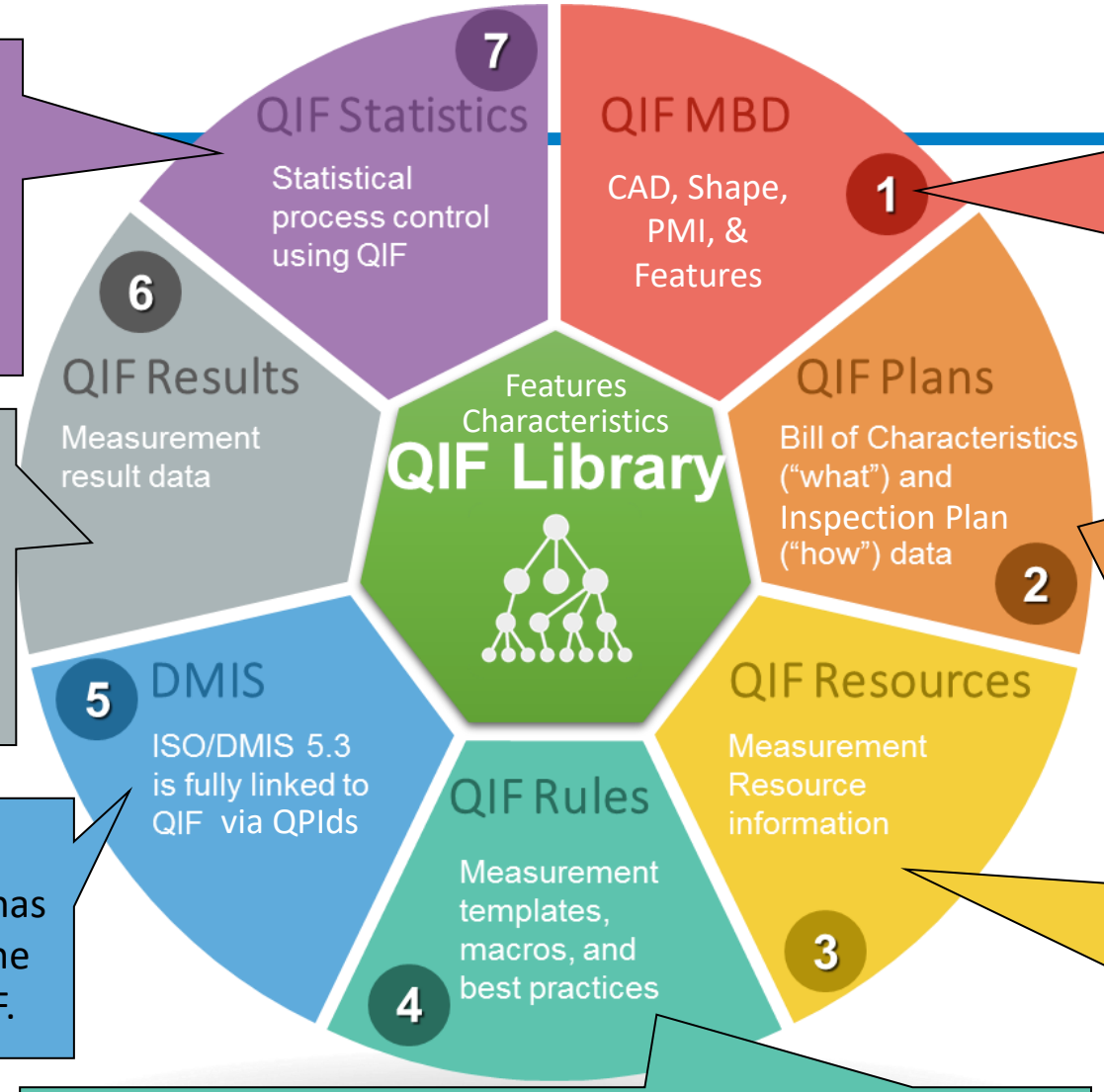
- Different concept than CAD features!
- Sometimes referred to as:
  - Tolerance Features
  - Metrology Features
  - Measurement Features

Features are referenced by ***Characteristics***

(aka, Product Characteristics, Inspection Requirements)

- Geometric Tolerances
- Dimensional Tolerances
- Specifications
- Notes

QIF Application Areas



Reference a bundle of QIF Results sets and specify a statistical analysis method to be carried out. Can optionally include the results of the statistical analysis as well

QIF MBD (Model-Based Definition) is the basis for providing traceability to authority CAD data. It is not required for basic QIF use cases. Considered to be the strongest semantic CAD+PMI standard available.

Measurement results data, associated with the MBD! This can be just tolerance evaluation results, and can even include all the point cloud data from the features.

Wide range of optional levels of detail for measurement plans:

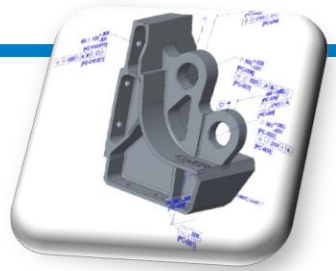
- What to Measure: Bill of Characteristics
- How to Measure: Inspection Plan
- Assign measurement resources
- Specify sampling point locations

DMIS is <u>not</u> part of QIF, ISO 22093, however the latest ANSI DMIS 5.3 has been updated to harmonize with the data traceability mechanisms in QIF.

Specify basic or highly detailed information about available measurement equipment (e.g., CMMs, probes, calipers, gages). As always, this data is contextual and semantic.

Rule templates for creating measurement rule instances. (e.g., *If a Surface Profile tolerance value is less than **x**, then use a CMM method with at least **y** number of point/sq.in.)*

QIF Enables a Quality Digital Thread



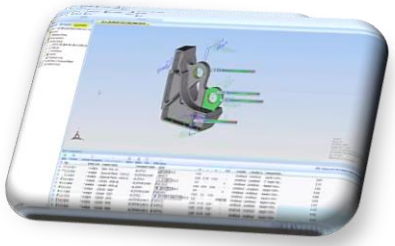
QIF-MBD

QIF-Plans

QIF-Resources

QIF-Document

QUALITY DIGITAL THREAD

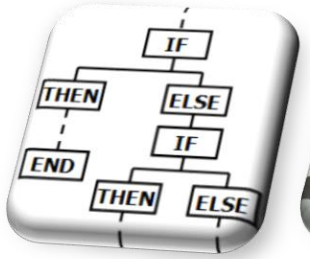


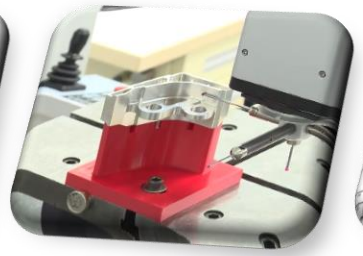
QIF-Rules

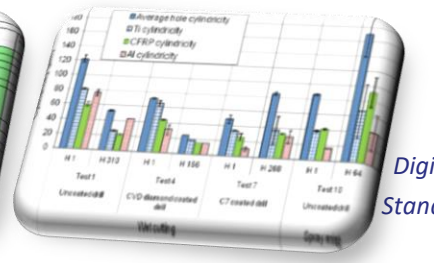
QIF-Execution w/ DMIS 5.3

QIF-Results

QIF-Statistics







Digital Metrology Standards Consortium

Workflow Example

Process Stage 1:
Search the PMI applied to the QIF MBD model, and identify the necessary measurement tasks.
This list of tasks is called a Bill of Characteristics

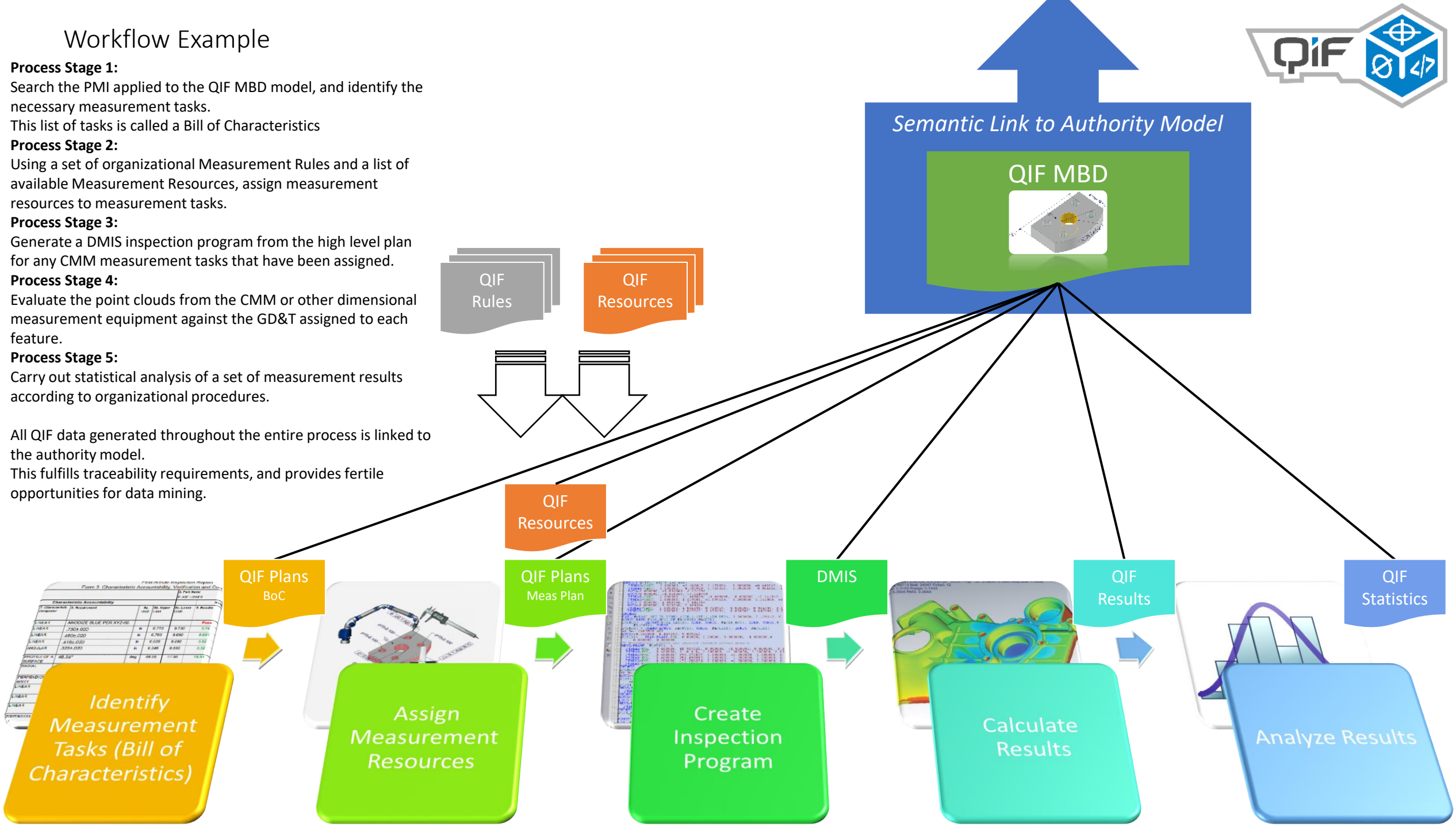
Process Stage 2:
Using a set of organizational Measurement Rules and a list of available Measurement Resources, assign measurement resources to measurement tasks.

Process Stage 3:
Generate a DMIS inspection program from the high level plan for any CMM measurement tasks that have been assigned.

Process Stage 4:
Evaluate the point clouds from the CMM or other dimensional measurement equipment against the GD&T assigned to each feature.

Process Stage 5:
Carry out statistical analysis of a set of measurement results according to organizational procedures.

All QIF data generated throughout the entire process is linked to the authority model.
This fulfills traceability requirements, and provides fertile opportunities for data mining.



QIF Business Value

Manufacturing Perspective

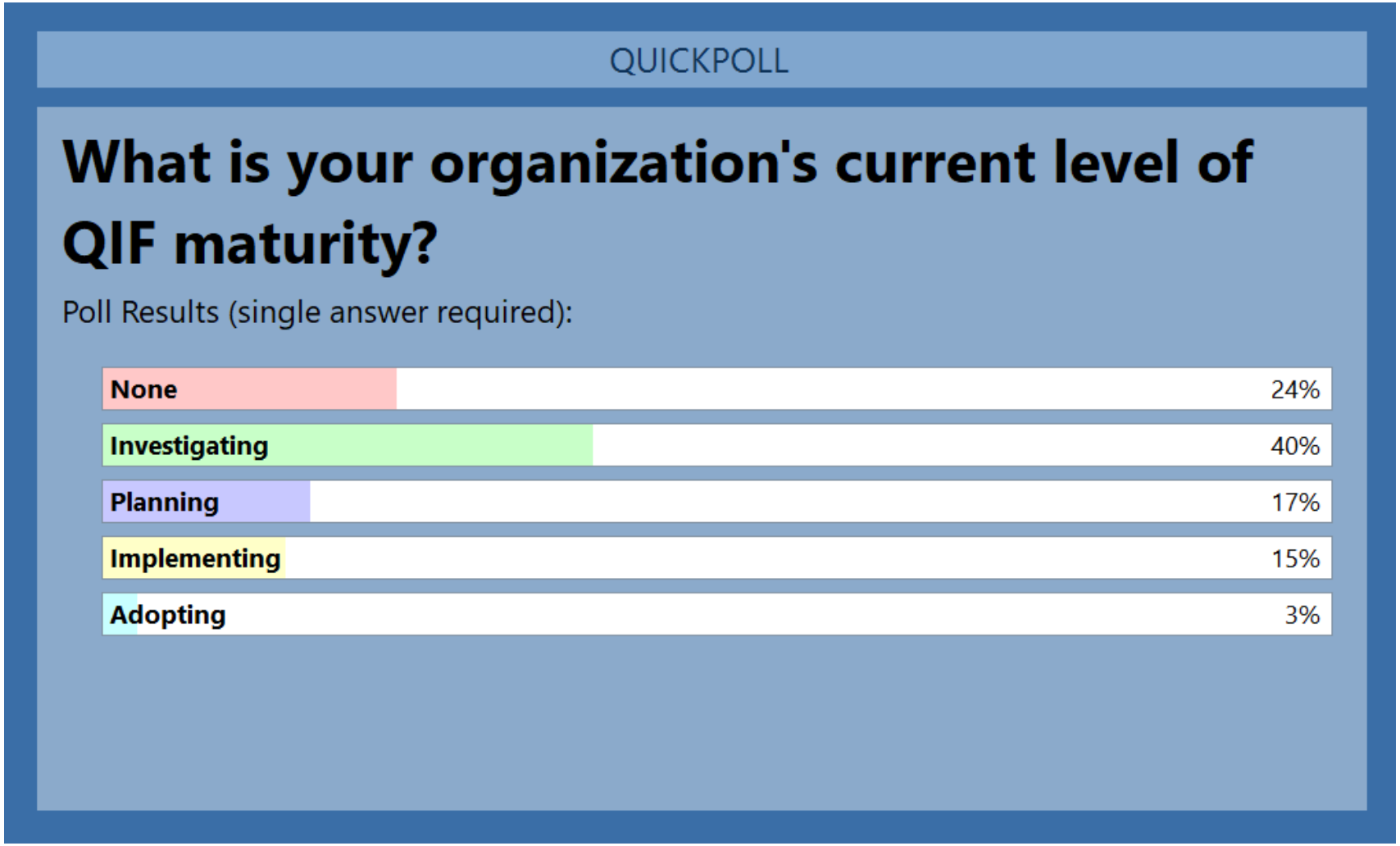
- Reduce Cost
- A **common information interface**
- **Freedom of choice** in selection of both value and performance based metrology solutions.
- **Interoperability** allows for Best-In-Class / Best-in-Value (hybrid solution)
- **One standard format** for measurement **results**.
- **Persistent traceability** between disciplines and back to the Model
- Enables Quality solutions to **communicate** inspection **results back to design** & manufacturing.
- **Impacts quality** functions within a Model-Based Enterprise
- **Common** format of PMI & feature-based MBD, **linked** w/ quality information
- ANSI **recognized standard**, an international standard even better

QIF Business Value

Vendor's Perspective

- Reduce Cost
- Concentrate on Core Competency
- **Inherent validations** for data correctness
- **Ease of (XML Schema** Based) Implementation
- **Cost efficient** to implement and use
- Minimize software development
- Avoid non-value (proprietary systems) development
- Complete and **extensible**
- Expanded **market opportunities** (penetrate proprietary systems)
- **One standard format** for measurement **results**.
- **Common** format of PMI & feature-based MBD **linked** w/ quality information
- ANSI **recognized standard**, an international standard even better

Webinar Poll Results



Critical Enabler for Digital Transformation

Steps for Digital Transformation

Digitization

the process of converting information into a digital format

Interoperable; Connected Systems

authoritative sources connected to other disciplines

Automation; Make Better Decisions

more information, allows for making smarter decisions, and to develop innovative tools to make work faster, better, cheaper

- For Product Realization, Digitalization is via Model-Based Enterprise
- The Quality Information Framework (QIF) Standard enables, contextual, semantic, digital interoperability

DIKW Pyramid

Application of Knowledge towards a specific goal

Applied Information

Data with context

Wisdom

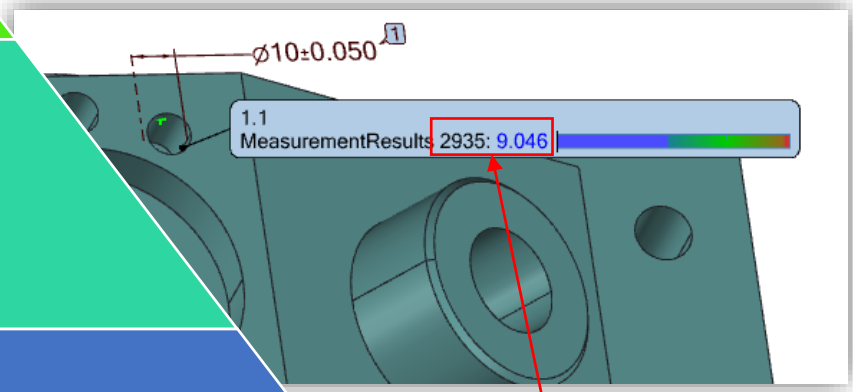
Knowledge

Information

Data

"Remove more material"

"This hole size is out of tolerance; the mating pin will not fit into this hole during assembly"



2935: 9.046

Making Decisions from …

Decision from Data

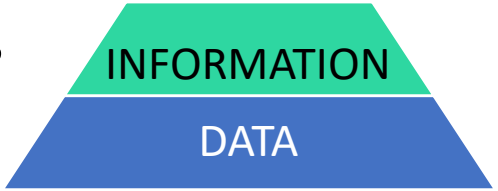
Decisions from Disconnected Data requires “human-in-the-loop”

- Human Creativity
- Tedious Process
- High Cognitive Load
- More Opportunities for Error
- Costly solutions
- Inconsistent Solutions

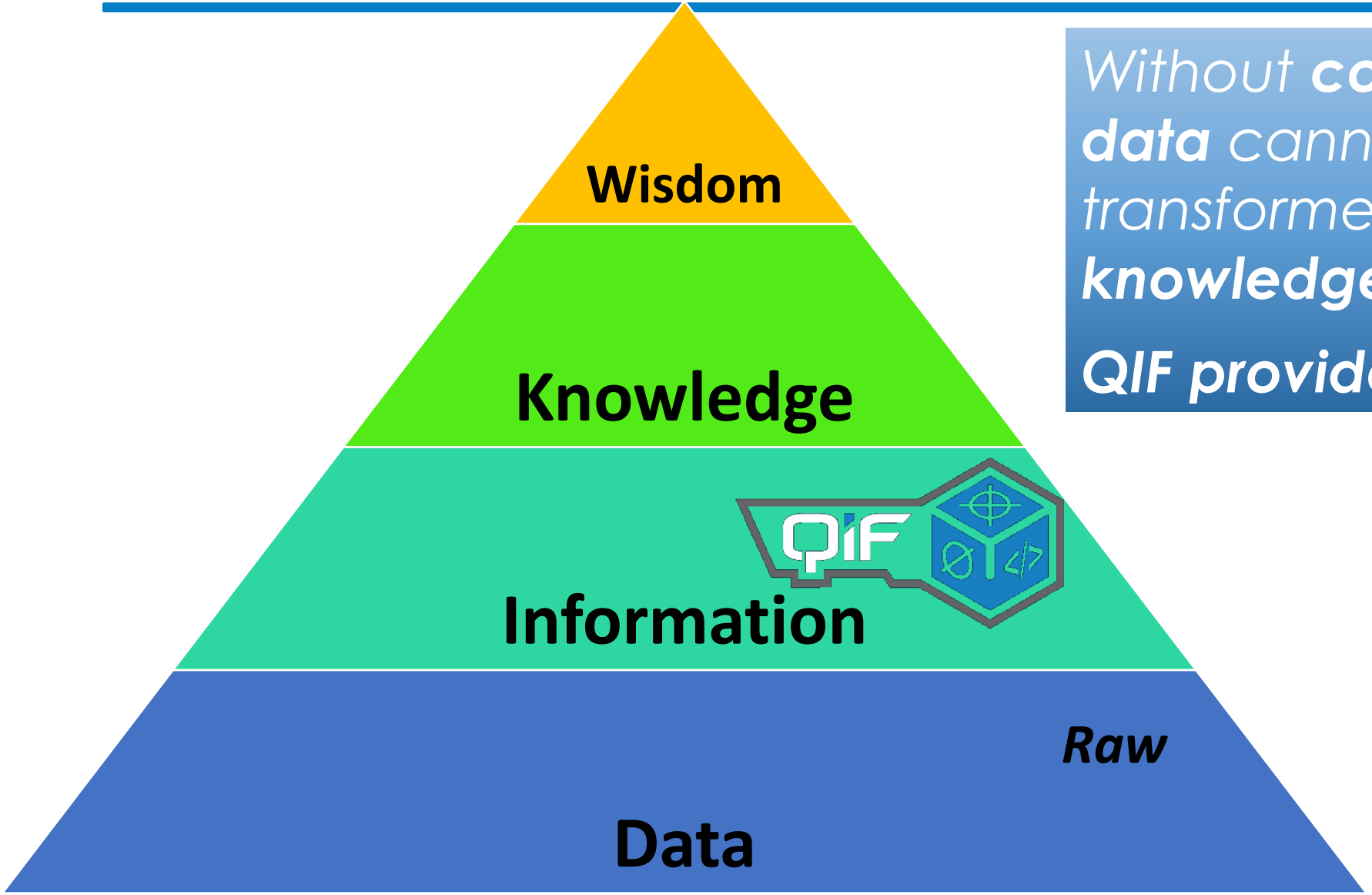
Decision from Information

Automation comes from data with context and structured.

- Increases speed of task completion
- Lowers costs due to decreased labor requirements
- Frees up valuable personnel for other tasks more suited for the human mind
- Repeatable Solutions
- Lower risks



DIKW Pyramid & QIF



*Without **context**, **data** cannot be transformed into **knowledge**.*

QIF provides this context.

DMSC's Roadmap for Success

- Active Schema Development
- Data Integrity
- Standardization Efforts
- Facilitate software development
- Free Open Source Tools
- QIF Community

DMSC Members

Digital Metrology Standards Consortium DMSC

QIF

ACTION ENGINEERING

metrologic group

TDP 360

ZEISS

METROSAGE

AAT3D
Metrology Solutions for Smart Manufacturing

Origin

SAMWELL GROUP

ITI
International TechneGroup

Honeywell

STEP Tools, Inc.
www.steptools.com

JOHN DEERE

PURDUE UNIVERSITY

UNC CHARLOTTE

ELYSIUM

NIST

CAPVIDIA

Schneider Electric

KOTEM

TechAzul

QIF SOLUTIONS

innovalia METROLOGY
INNOVATION AND ACCURACY

Quality Vision International
Precision for People

LK metrology

HIGH QA
HIGH QUALITY AUTOMATION

Mitutoyo
Mitutoyo America Corporation

DMSC BoD

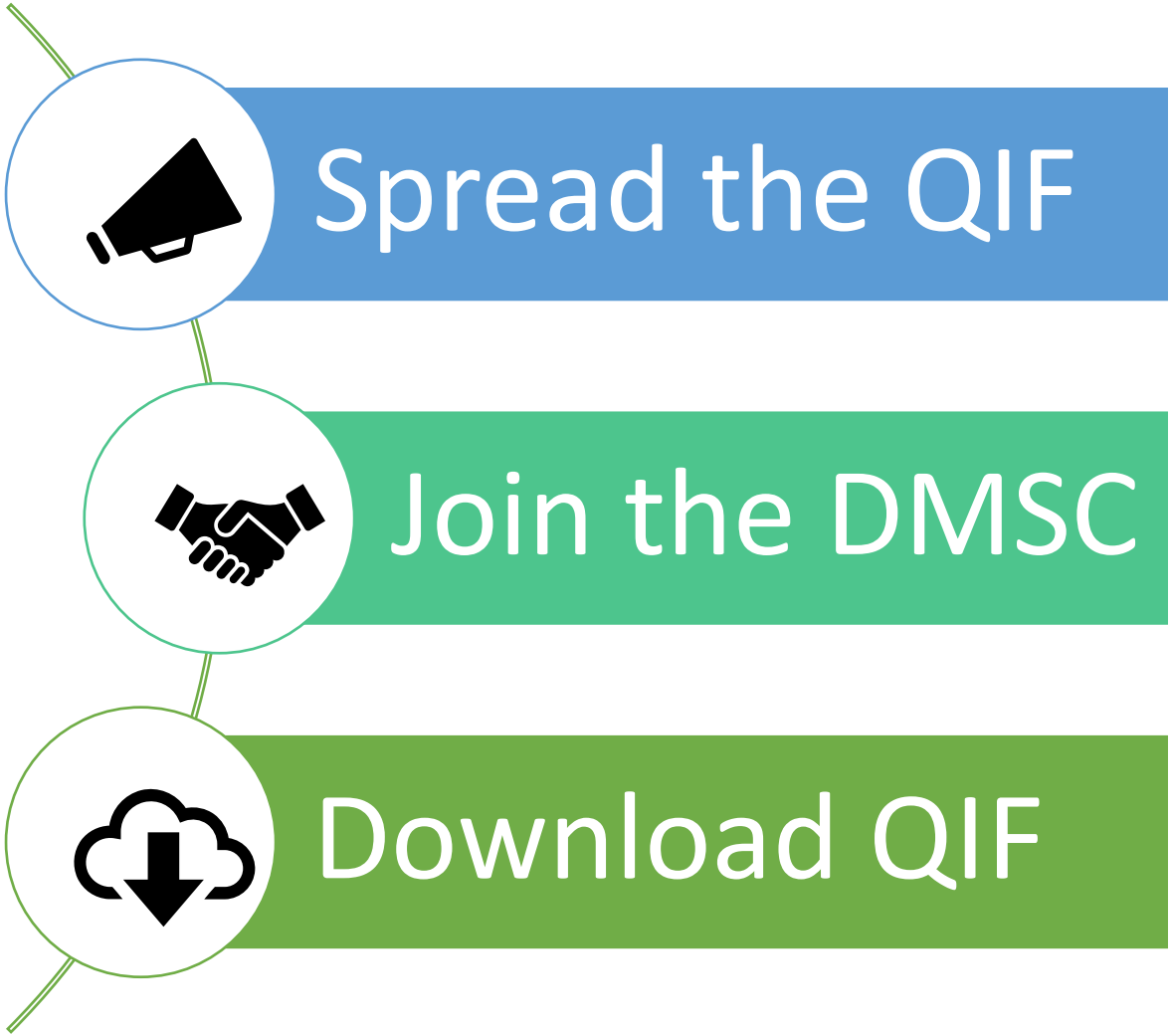
DMSC Board of Directors & Officers

Robert Brown Mitutoyo America Corporation	**Jennifer Herron** Action Engineering
Cory Leland Deere & Co.	**Ray Stahl** KOTEM / QVI
Daniel Campbell Capvidia	**Rosemary Astheimer** Purdue University
Curtis Brown - President Honeywell FM&T	**Mark Thomas – Executive Director** Blue Tuna
Ray Admire - Treasurer QIF Solutions	**Bailey Squier – Executive Director Emeritus** BHSA

Take Action!

Digital Metrology Standards Consortium DMSC

QIF

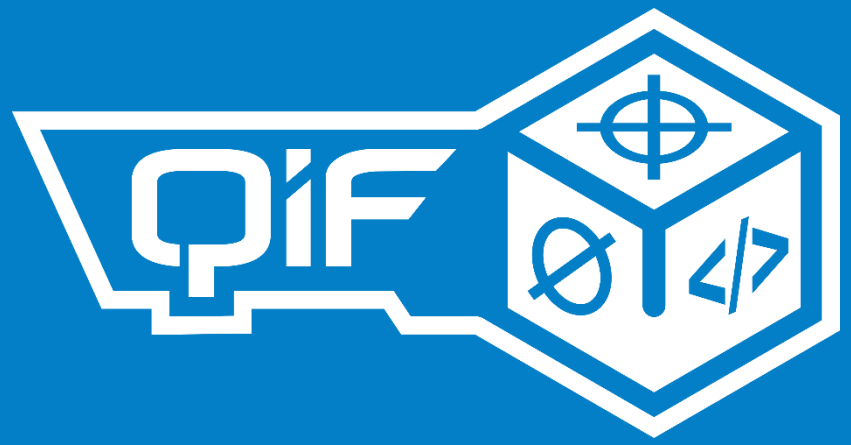


Visit us at:

www.qifstandards.org

Questions

www.qifstandards.org



Digital Metrology Standards Consortium DMSC

DMSC's QIF Tutorials

April 30, 2020

QIF Tutorials

Basics, **Technology,** and Applications

QIF 101: Understanding QIF Basics –
Curtis Brown, Honeywell FM&T (April 23, 2020)

QIF 201: Reviewing QIF Technology –
Daniel Campbell, Capvidia (April 30, 2020)

QIF 301: Coding QIF Applications –
Tom Kramer, NIST Guest Researcher (May 7, 2020)

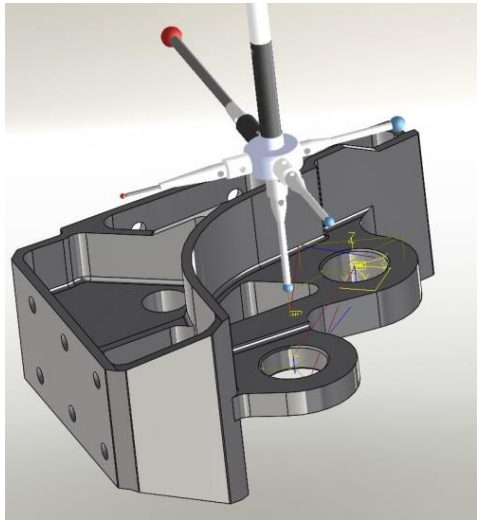
QIF 201:
Outlining the QIF Schemas

Daniel Campbell
VP Model-Based Definition
Capvidia
www.capvidia.com

Who is the DMSC?

Digital Metrology Standards Consortium

- A **non-profit**, cooperative sponsorship, **consortium** organization. Conceived & sponsored in 1983; Separate legal entity 2005, dba Digital, 2018.
- Dedicated to identifying, promoting, fostering, and encouraging the **development** and **interoperability** of information standards that benefit the digital metrology community.
- Preparing standards that **impact** digital **model-based quality** enterprise.
- A professional **group** of manufacturing metrologists, software developers, and innovators worldwide. Note: 500+ years of experience contributed to the QIF.
- Maintainers of **Dimensional Measuring Interface Standard** (**DMIS**) standard.
- Developers & maintainers of **Quality Information Framework** (**QIF**) standard.
- **ANSI accredited** standards making organization
- **A-Liaison** member of **ISO / TC 184 / SC 4** (allows for harvesting ANSI standards)



What is **XML**, XSD, XLST?

What is XML?

- Extensible Markup Language
- A set of rules for text based data formatting

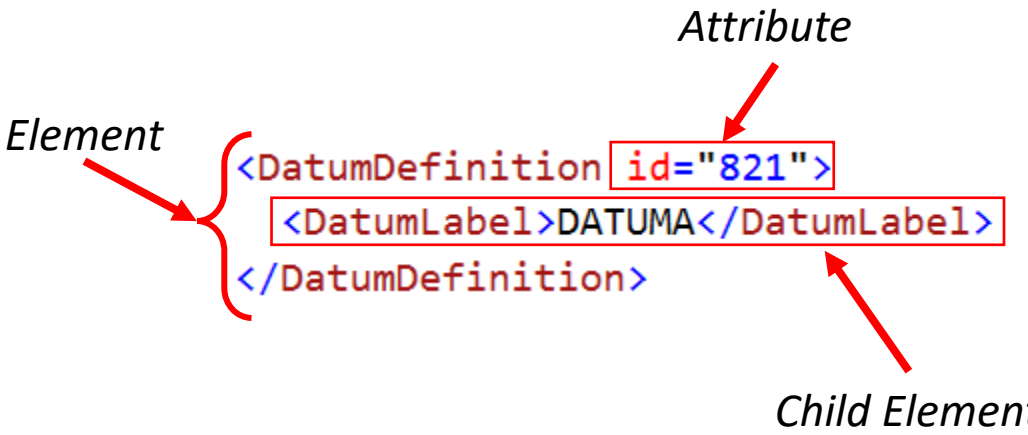
Why does it matter?

- Great for human and machine consumption
- Choice of XML makes QIF data super easy to read and write
- Lots of XML tools out there: XPath, XQuery, XSLT, XML software apps and libraries…

Vocabulary

Element: a block of XML data surrounded by start and end tags (or, an empty element tag). Elements can contain child elements.

Attributes: a piece of metadata to an Element, contained within the element start tag.



What is XML, **XSD**, XLST?

What is XSD?

- XML Schema Definition
- A tool for specifying a required structure for an XML document

Why does it matter?

- Provides a standard, formal definition for an XML based format → QIF
- Validation: Can be used to check ✓ an XML instance file to ensure that it conforms to the Schema

Source Code Generators

Did you know?

The rigorous definition of QIF by using XSD makes it very easy to automatically generate source code that can read and write QIF.

Most engineering graduates these days are pretty good with programming languages like Python – with just a couple hours of work, they could become QIF hackers!

The QIF Community website has a bunch of free tools and instructions on how to generate source code bindings for QIF for C++, C#, and Python. Check it out: https://qualityinformationframework.github.io/

What is XML, XSD, **XLST**?

What is XSLT?

- Extensible Stylesheet Language Transformations
- A Turing-complete language for manipulating XML documents

Why does it matter?

- XSLT scripts can be executed on any modern operating system 🔓
- More Validation: QIF uses XSLT to perform robust checks✔ on a QIF instance file

XSLT and QIF

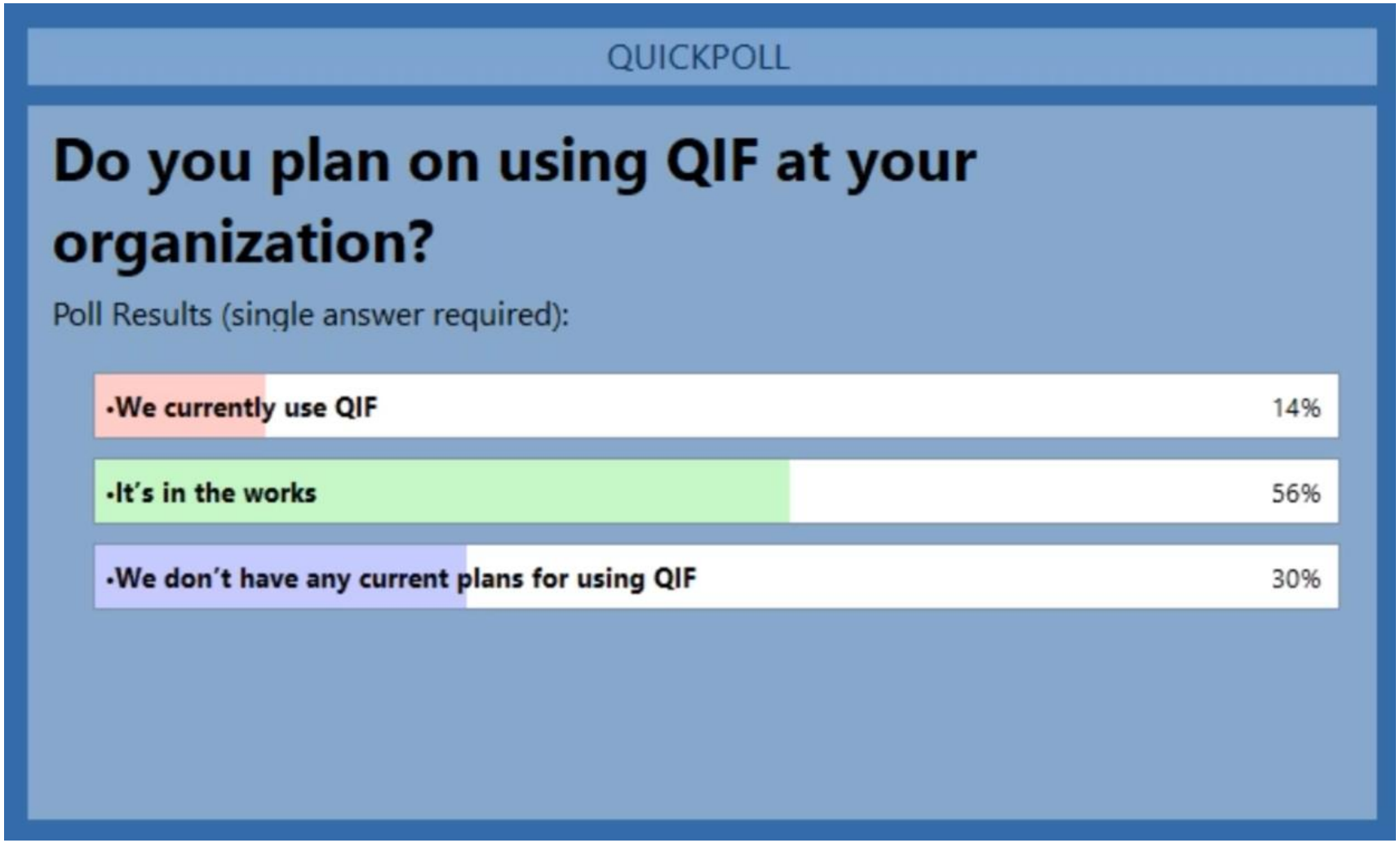
The XSLT files can be found in the download package for QIF. (Download it here!) Look for the .XSL files in the `XSD/Check/` directory.

They provide much richer validation than the XSD schemas can provide. For example, here are some things that are checked:

- Valid ID references to external documents
- Correct unit vector length
- Data integrity checks for array sizes

Want to run an XLST script yourself? Check out this page of the QIF Community GitHub site for info on how to do that.

Poll


QUICKPOLL
Do you plan on using QIF at your organization?
Poll Results (single answer required):
•We currently use QIF
14%
•It's in the works
56%
•We don't have any current plans for using QIF
30%

Major QIF Elements
QIF Document

- Top level container for an instance file
  - 1 instance file = 1 QIFDocument
- At the top level, it contains all the key elements we will discuss in the next slides
- Almost all the elements at this level are optional

```
<QIFDocument>

      ...

      <Header> ... </Header>

      <DatumDefinitions> ... /DatumDefinitions>

      <DatumReferenceFrames> ... </DatumReferenceFrames>

      <MeasurementResources> ... </MeasurementResources>

      <Product> ... </Product>

      <Features> ... </Features>

      <Characteristics> ... </Characteristics>

      <Results> ... </Results>

</QIFDocument>
```

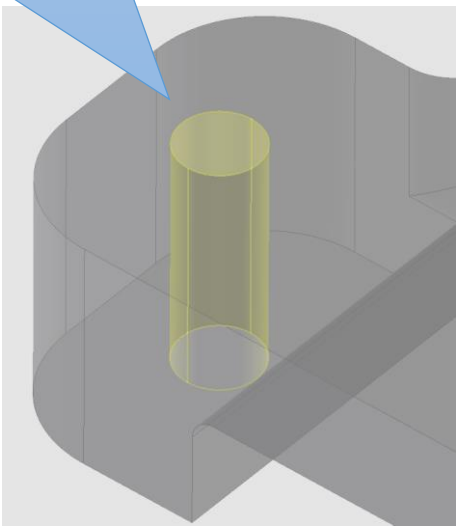
Major QIF Elements
Library: Features

An abstraction for referencing a portion of a part.
In MBD, this means:
A container for referencing one or more geometrical entities on the model

There are lots of feature types! Some examples:

- Cylinder
- Plane
- Cone
- Opposite Parallel Planes (slot)
- Freeform (generic)
- Circles
- Lines
- Ellipse
- Compound Features
- Pattern Features
- Etc.

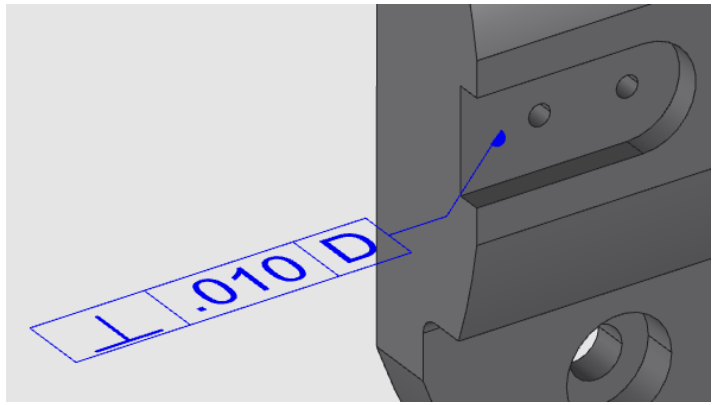
This Cylinder feature is made up of 2 CAD surfaces. (Pretty typical.)
But the CAD's mathematical representation of this geometry is irrelevant – this is a functional hole, and needs to be treated as such!



Major QIF Elements
Library: Characteristics

A control placed on a Feature.

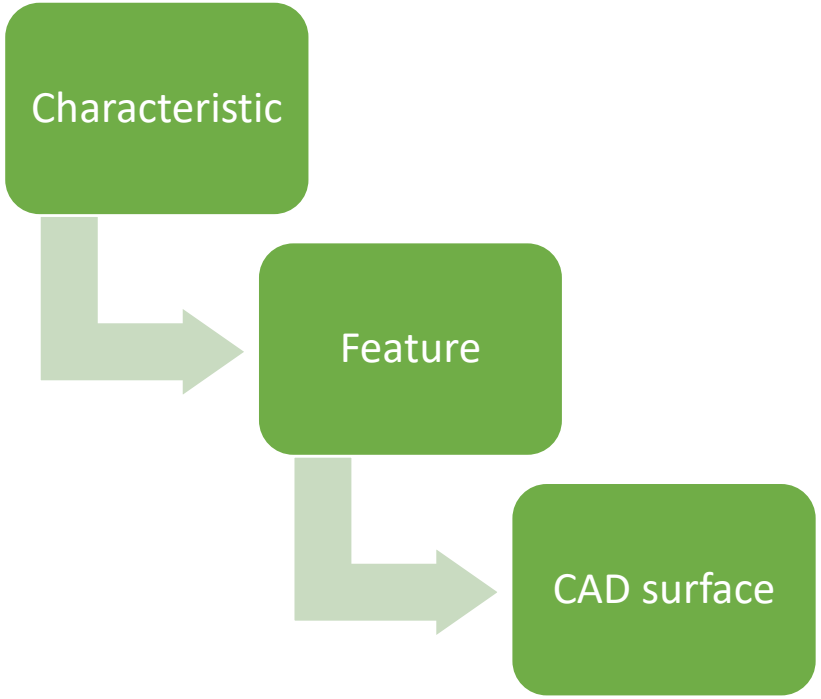
For example, a Size/Form/Orientation /Location tolerance, a Surface Finish, a Weld specification, etc.



With QIF MBD, it is also possible for a Characteristics to have a 3D presentation element (e.g., an annotation). But that's for human consumption.

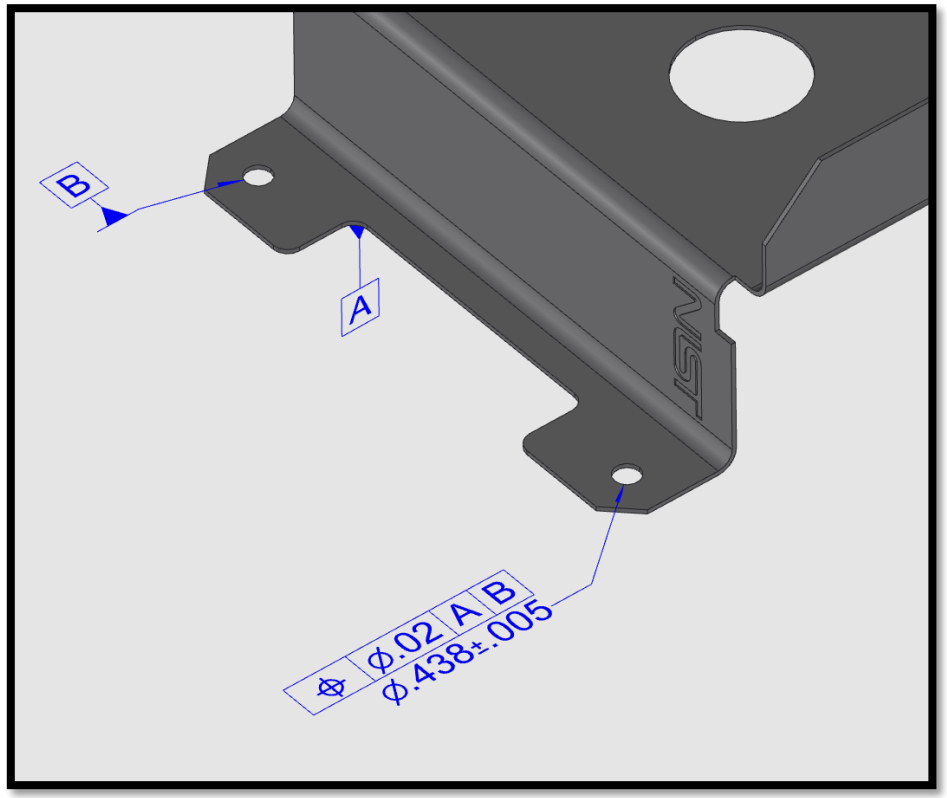
How is a Characteristic connected to the MBD?

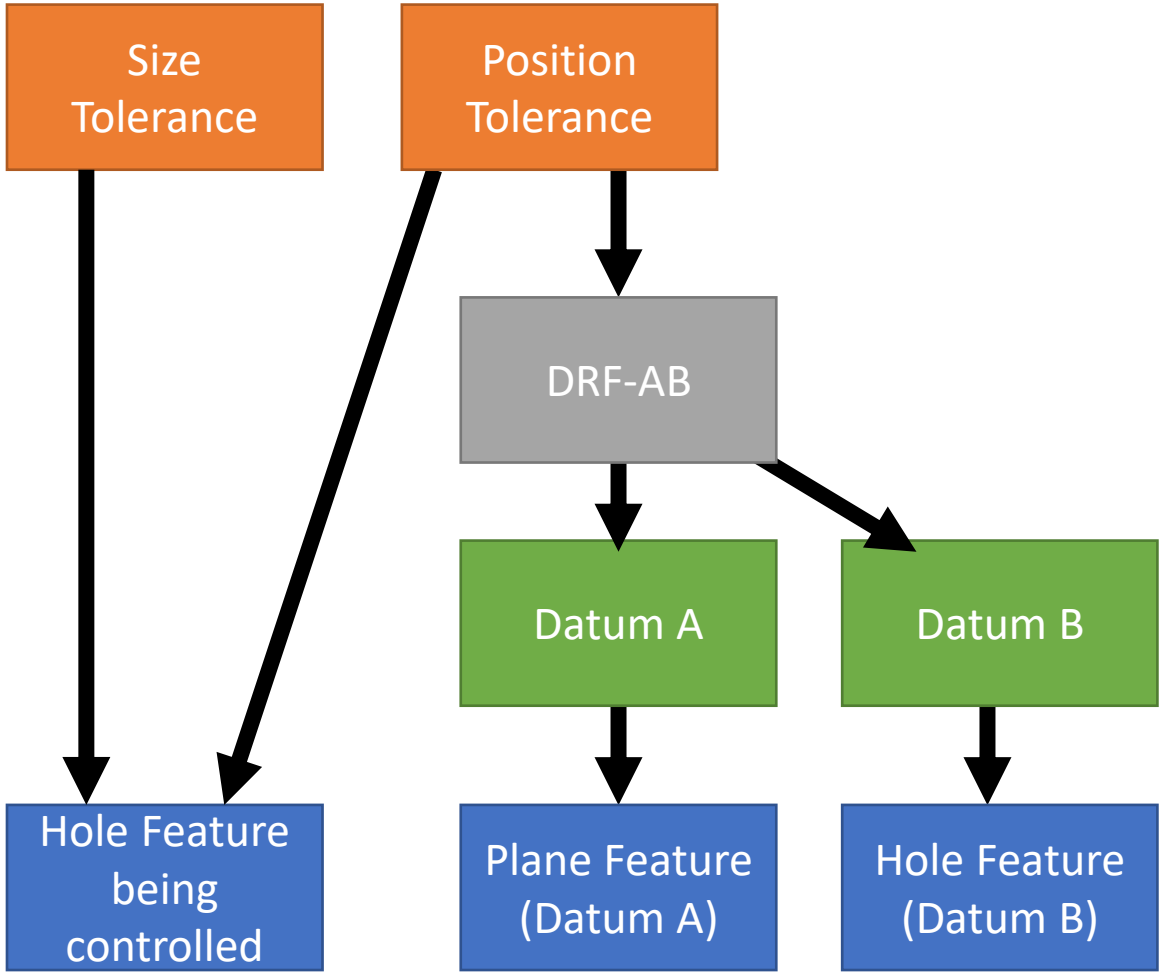
A Characteristic points to a Feature, and a Feature points to CAD geometry.



Major QIF Elements
Library: Datums & DRFs

Datums and DRFs are data structures used to help define the geometric controls implied by a GTol. This is how Features, Datums Features, and GTols are linked.



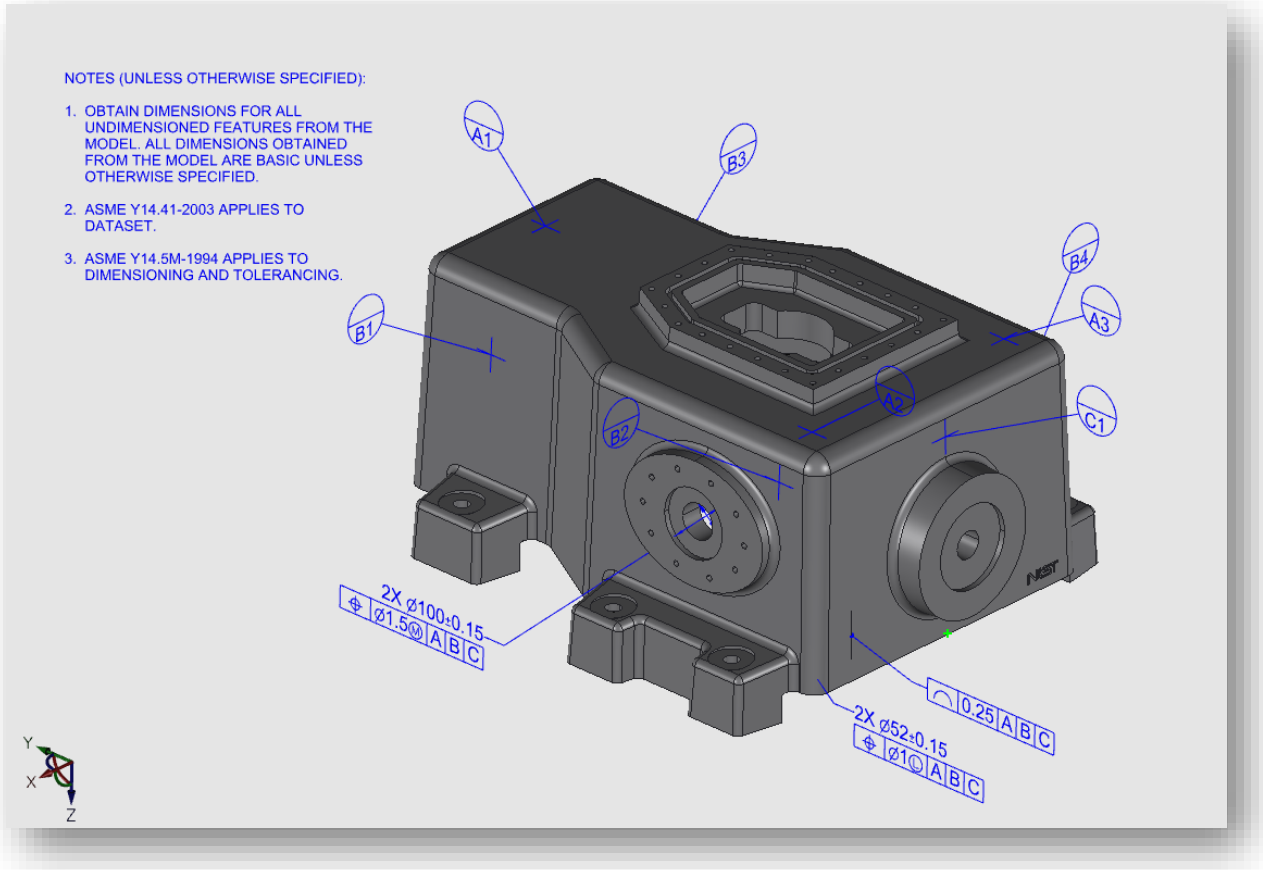


Major QIF Elements
Applications: Product (aka, MBD)

The “Product” is the name of the element for QIF MBD data.

This is how Feature/Characteristic Nominals and Feature/Characteristic Measurements are tied to the MBD model.

This is how **Digital Twin** is done.

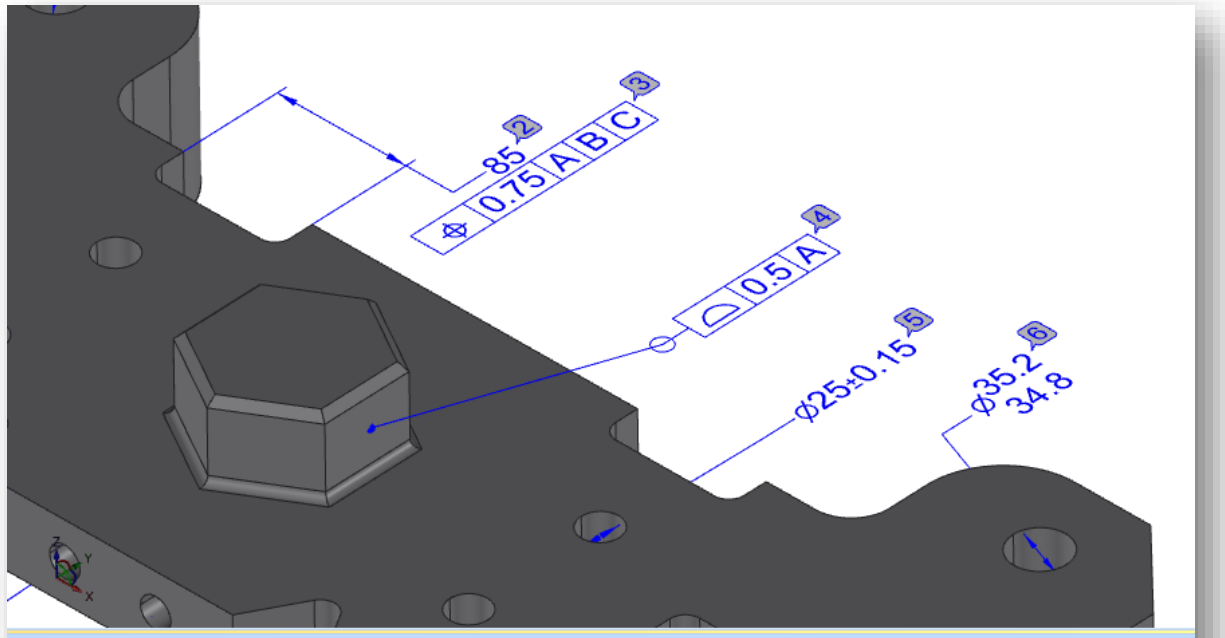


Major QIF Elements
Applications: Plans

Measurement Plans can be described in various levels of detail. Can range anywhere between:

High level plan: just the Bill of Characteristics – *what is being measured*?

Detailed plan: What is being measured, what equipment is being used to do it, where are sampling points being taken, etc.

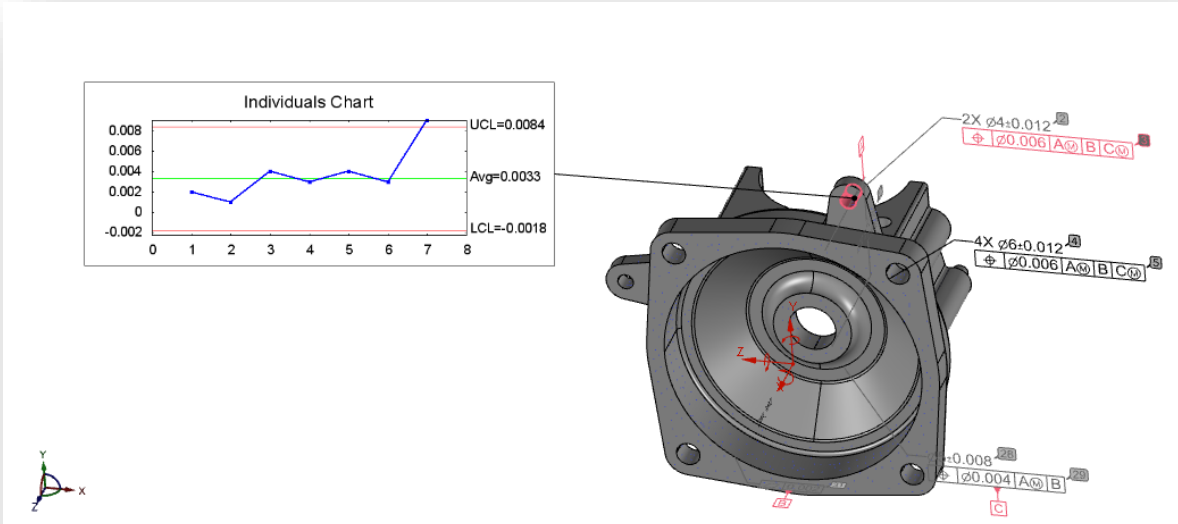


Major QIF Elements
Applications: Results

Tolerances are placed on a product definition to be measured.

QIF Measurement Results is how Feature and Characteristic measurements are stored and linked to the corresponding product definition entity.

This allows for the mapping between: Measurement ↔ Product Definition



Bill of Characteristics

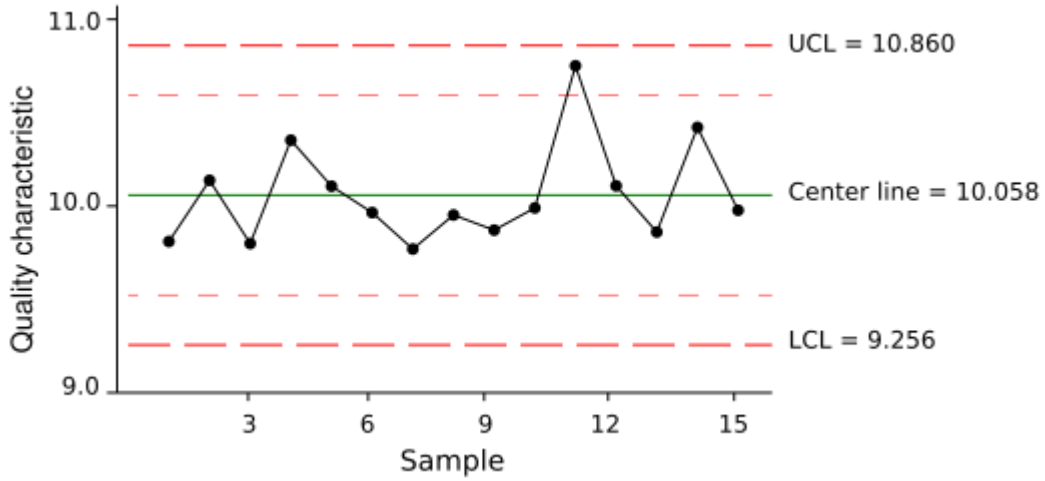
AS9102 (Report Type) | Reset | Decolorize | Re-Balloon | Export | Import | Bind | HTML Report | PDF Report | Publish | Import | Individuals Chart (Measurement Visualization) | QPID: 5054e4a1-

T	Saved View	Feature Name	Annotation ...	GD&T	(+)	/	(-)	DRF	Critica...	SN#1	SN#2	SN#3	SN#4	SN#5
Ø 2	FRONT_VIE...	Cylinder 4652	DRV_DIM_D...	Ø4 ±0.012	0.012	4	-0.012	-		4.000	4.010	4.011	4.012	4.010
Ø 2	FRONT_VIE...	Cylinder 4749	DRV_DIM_D...	Ø4 ±0.012	0.012	4	-0.012	-		4.010	4.011	4.012	3.999	4.000
⌖ 3	FRONT_VIE...	Cylinder 4652	AE_GTOL3	⌖ Ø0.006 AⓂ B CⓂ	0.006	-	-	AⓂBCⓂ		0.002	0.001	0.004	0.003	0.004
⌖ 3	FRONT_VIE...	Cylinder 4749	AE_GTOL3	⌖ Ø0.006 AⓂ B CⓂ	0.006	-	-	AⓂBCⓂ		0.004	0.003	0.004	0.003	0.001
Ø 4	FRONT_VIE...	Cylinder 4720	AE_DRIVEN...	Ø6 ±0.012	0.012	6	-0.012	-		6.010	6.020	6.030	6.012	5.990
Ø 4	FRONT_VIE...	Cylinder 4751	AE_DRIVEN...	Ø6 ±0.012	0.012	6	-0.012	-		6.012	5.990	6.012	5.990	5.998
Ø 4	FRONT_VIE...	Cylinder 4752	AE_DRIVEN...	Ø6 ±0.012	0.012	6	-0.012	-		5.990	5.998	5.990	5.998	6.000
Ø 4	FRONT_VIE...	Cylinder 4750	AE_DRIVEN...	Ø6 ±0.012	0.012	6	-0.012	-		5.998	5.990	5.998	6.000	5.990

Major QIF Elements
Applications: Statistics

QIF Statistics can be used in one of two ways:

- **Specify statistical method**: define the SPC rules that should be applied to measurement data, when it is acquired
- **Show SPC Results**: bundle up a set of results data, and show the result of the SPC analysis of that dataset

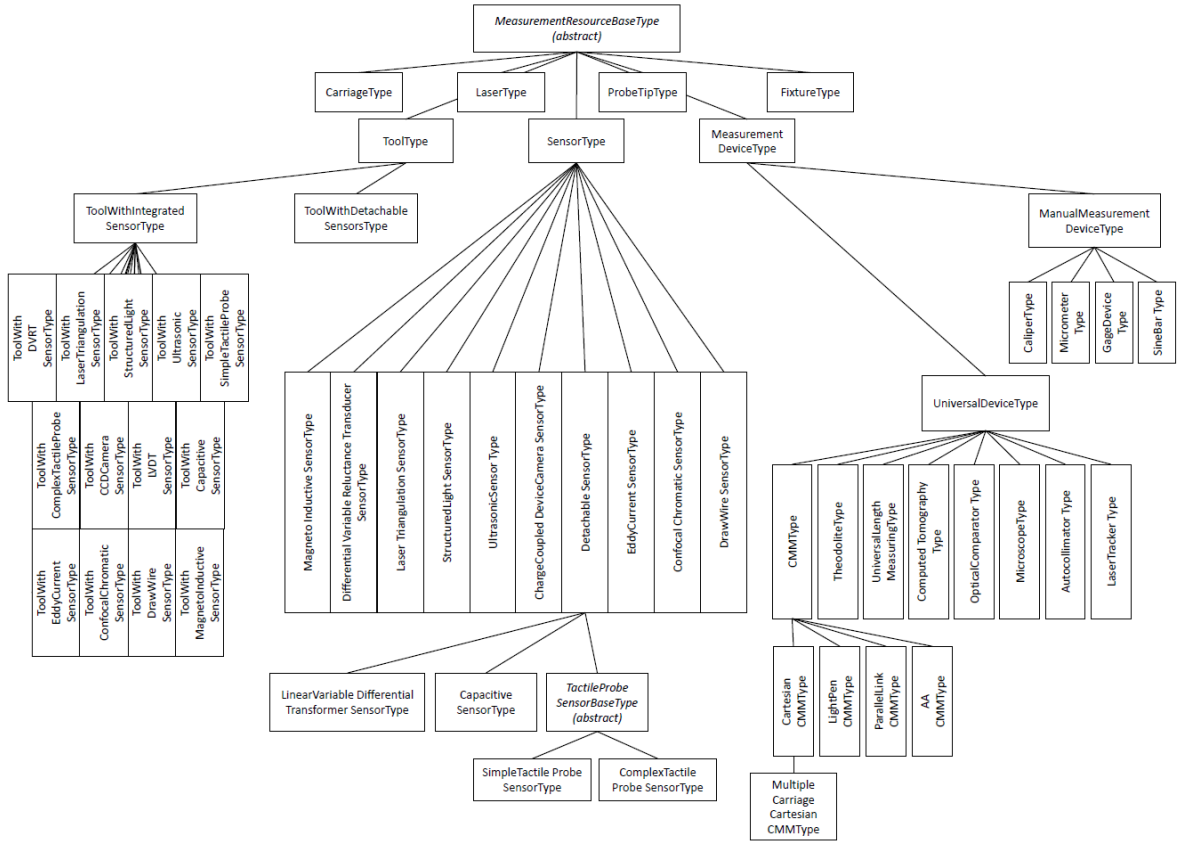


Major QIF Elements
Applications: Resources

What measurement equipment will be used to measure something?

QIF Resources is used to specify this equipment.

A huge range of devices are described in QIF 3.0, including CMMs, calipers, gages, theodolites, lots of sensors, and more!



Major QIF Elements
Applications: Rules

Provides a mechanism to specify things like:

- Measurement best practices
- Measurement templates
- Corporate standards for measurement

How does it work?

```
If ("Boolean condition") then
{
   Corresponding "action"
}
```

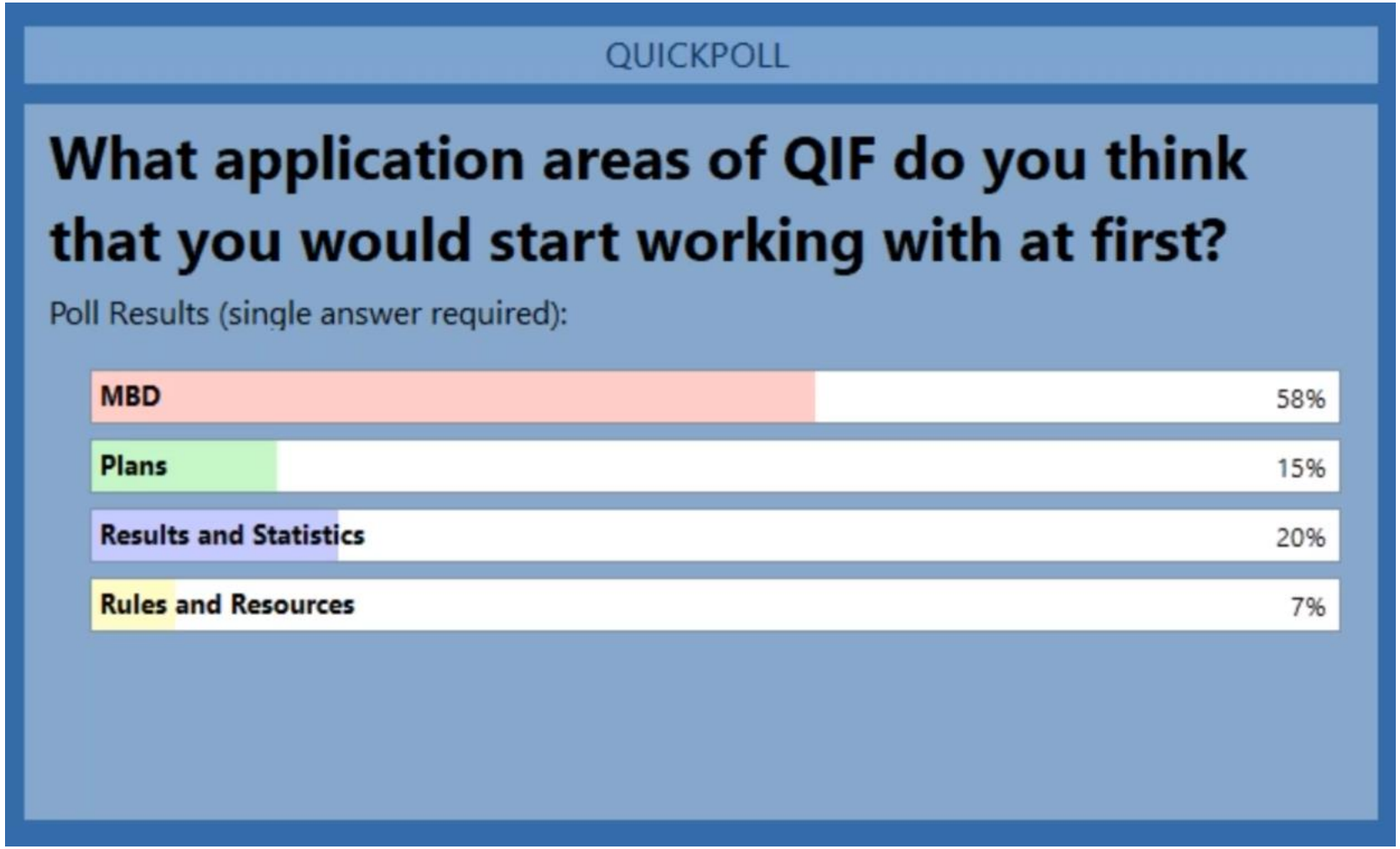
Boolean conditions that can be checked:

- Tolerance type, value, etc.
- Feature type, size, etc.
- Is feature a datum?
- If feature internal/external?
- User defined sampling category
- Anything else can be evaluated to a Boolean results, e.g., "surface area >= 150"

Actions that can be taken:

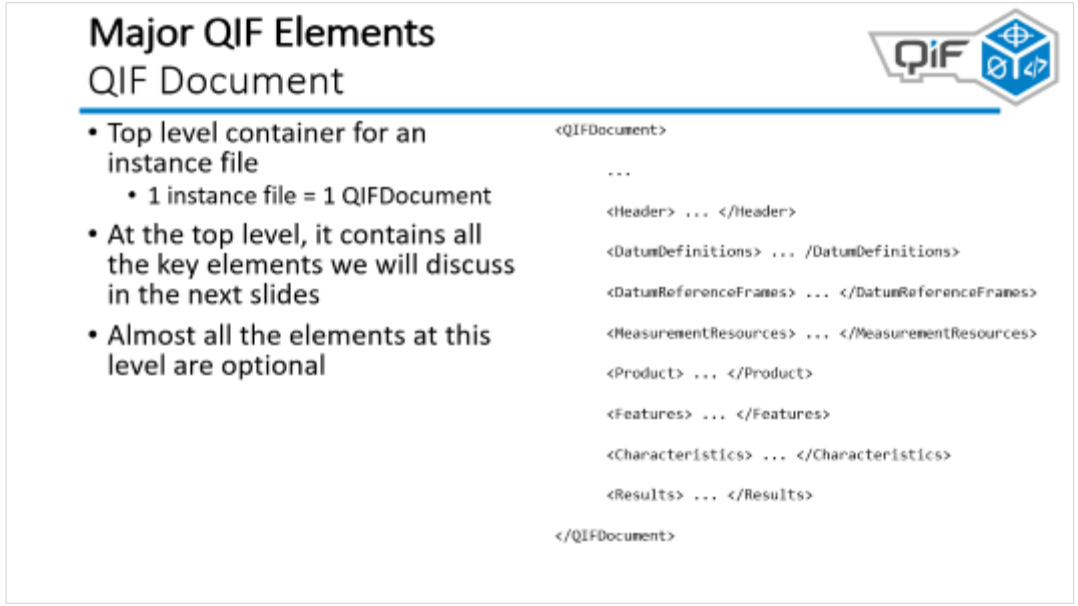
- Sampling point count, density, and sampling strategy
- Fitting algorithm
- Measurement equipment: required, allowed, prohibited

Poll


QUICKPOLL
What application areas of QIF do you think that you would start working with at first?
Poll Results (single answer required):
MBD
58%
Plans
15%
Results and Statistics
20%
Rules and Resources
7%

Modularity of QIF Data

Back to QIF Document…



A single QIF instance file is a QIF Document which might contain:

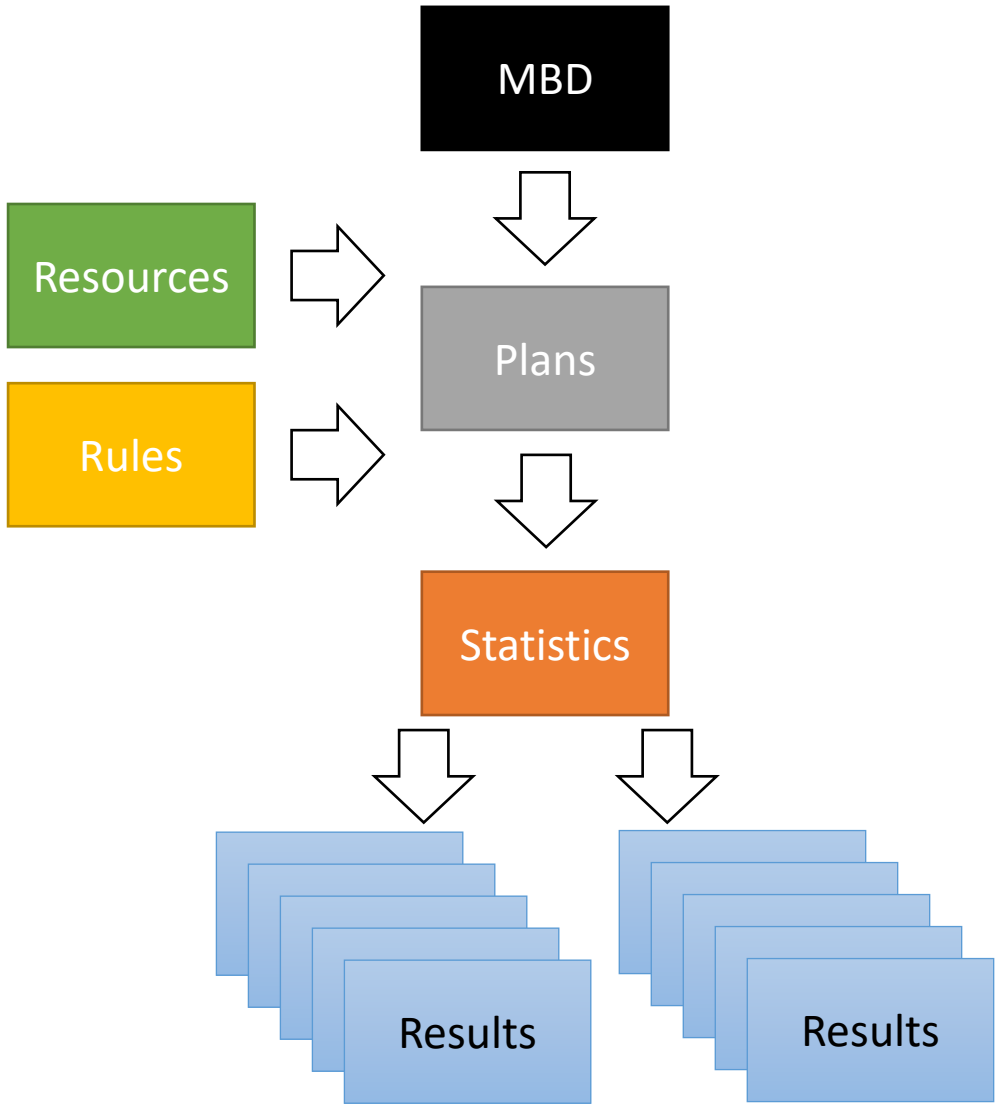
- Product (MBD)
- Plans
- Results
- Statistics
- Resources
- Rules

A single part made by a manufacturing enterprise will typically have 1 MBD product definition, **but may have multiple sets of plans, results, statistics**. Also, the resources or rules used on this part might be used on thousands of others.

This is why QIF is designed to support a federated data model, where a single mapped bulk of QIF data can be stored across multiple instance files.

Modularity of QIF Data

QIF data can be split up among various instance files. For example, the diagram on the right shows a possible group of interrelated QIF instance files, where one box corresponds to one instance file.

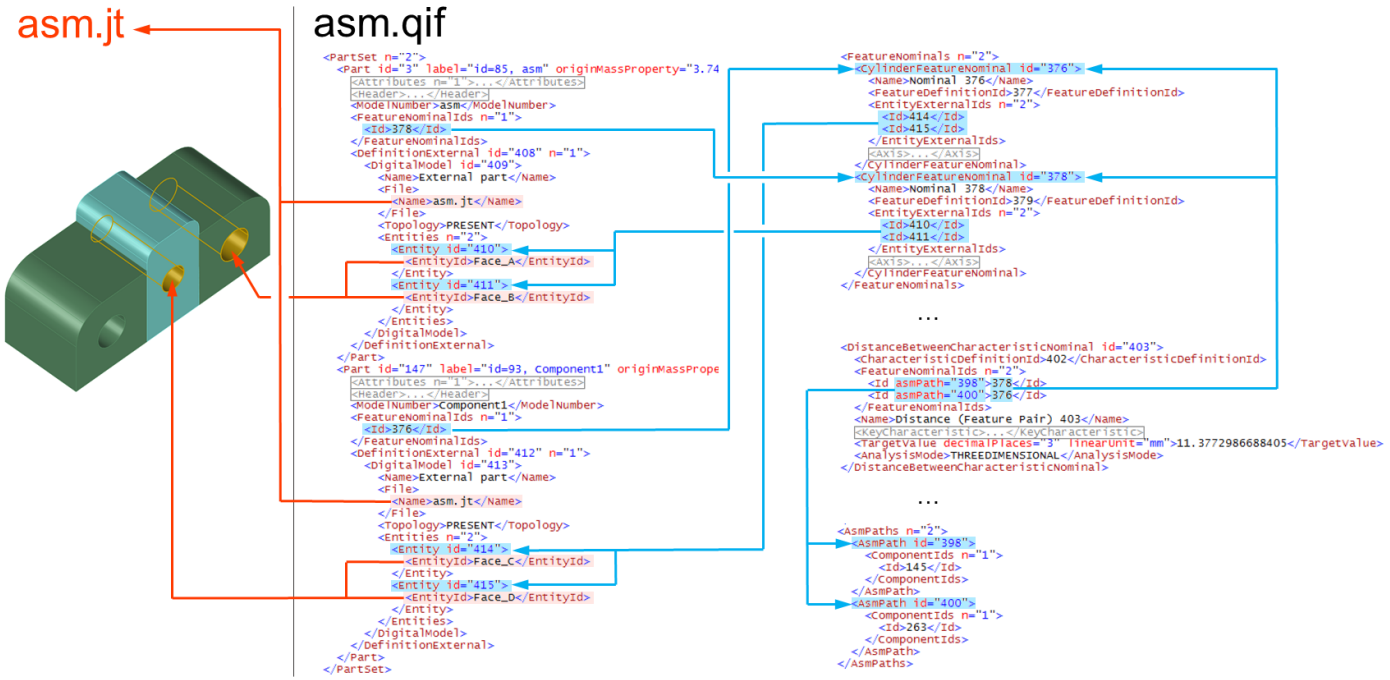


But remember:
A QIF instance file usually *does* carry more than one type of data. So, it doesn't typically make sense to ask "*is this a QIF MBD file, or a QIF Plans file?*" It might be both!

Modularity of QIF Data

QIF can also externally reference a non-QIF file.

For example, the diagram on the right shows how a QIF assembly can externally reference JT geometry.

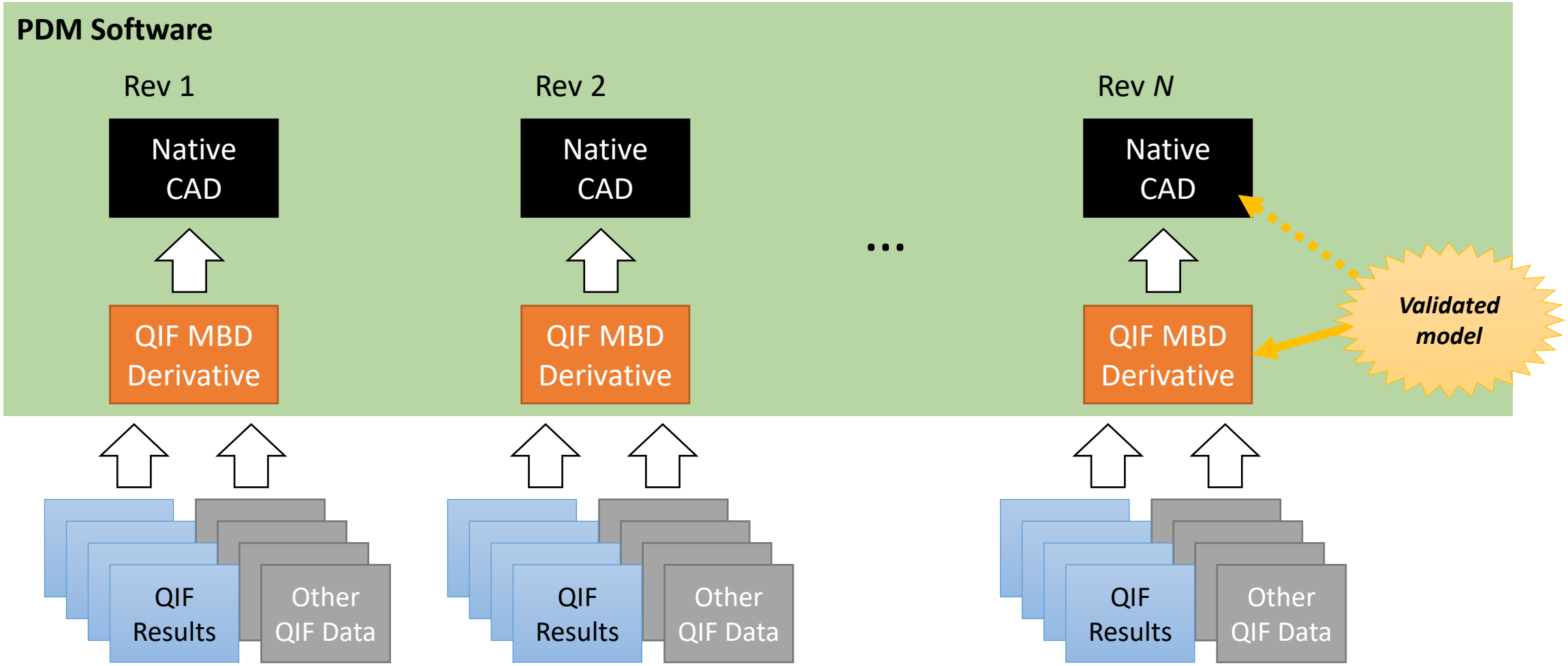


QIF in a PLM world

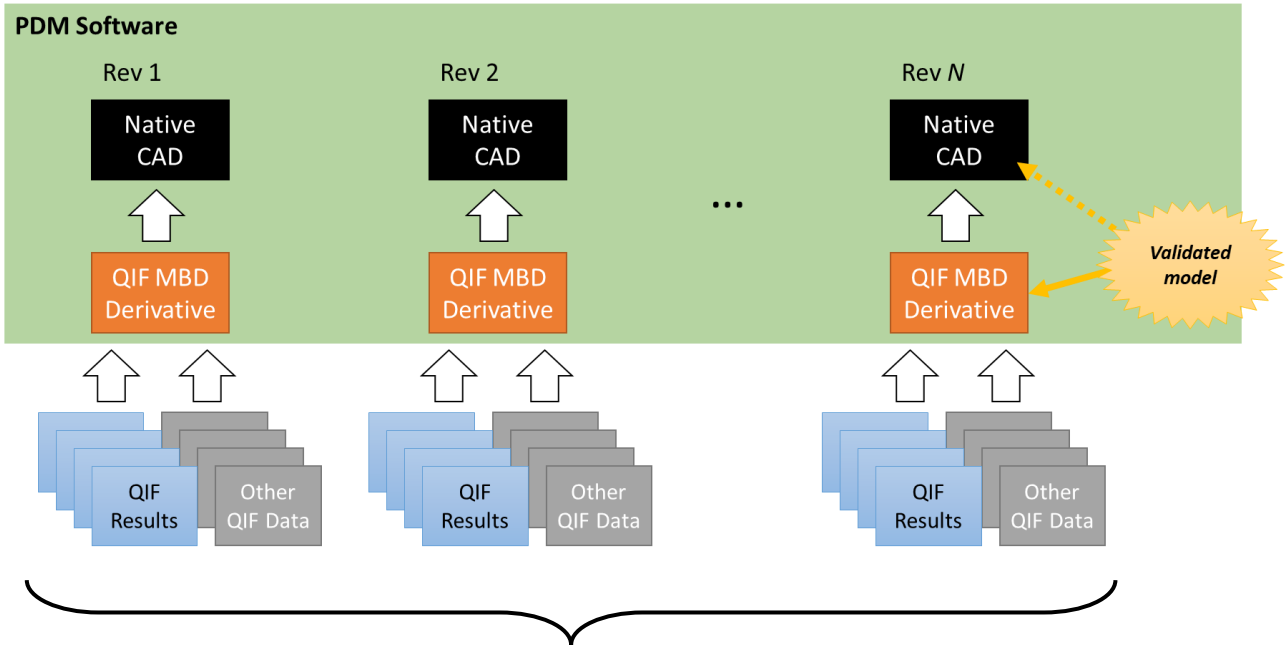
PDM software is used by manufacturing to organize all product data, centered on the product definition.
(E.g., Teamcenter, Windchill, ENOVIA, etc.).
This is how a product's lifecycle is systematized.

How does QIF fit into this paradigm?

QIF in a PLM world



QIF in a PLM world



Features/Tolerances which are not changed between revisions can be easily correlated.

Mapping maintained from "**single source of truth**" (i.e., CAD model in PDM) and **all measurement data**

Very high data resolution – the measurement data for

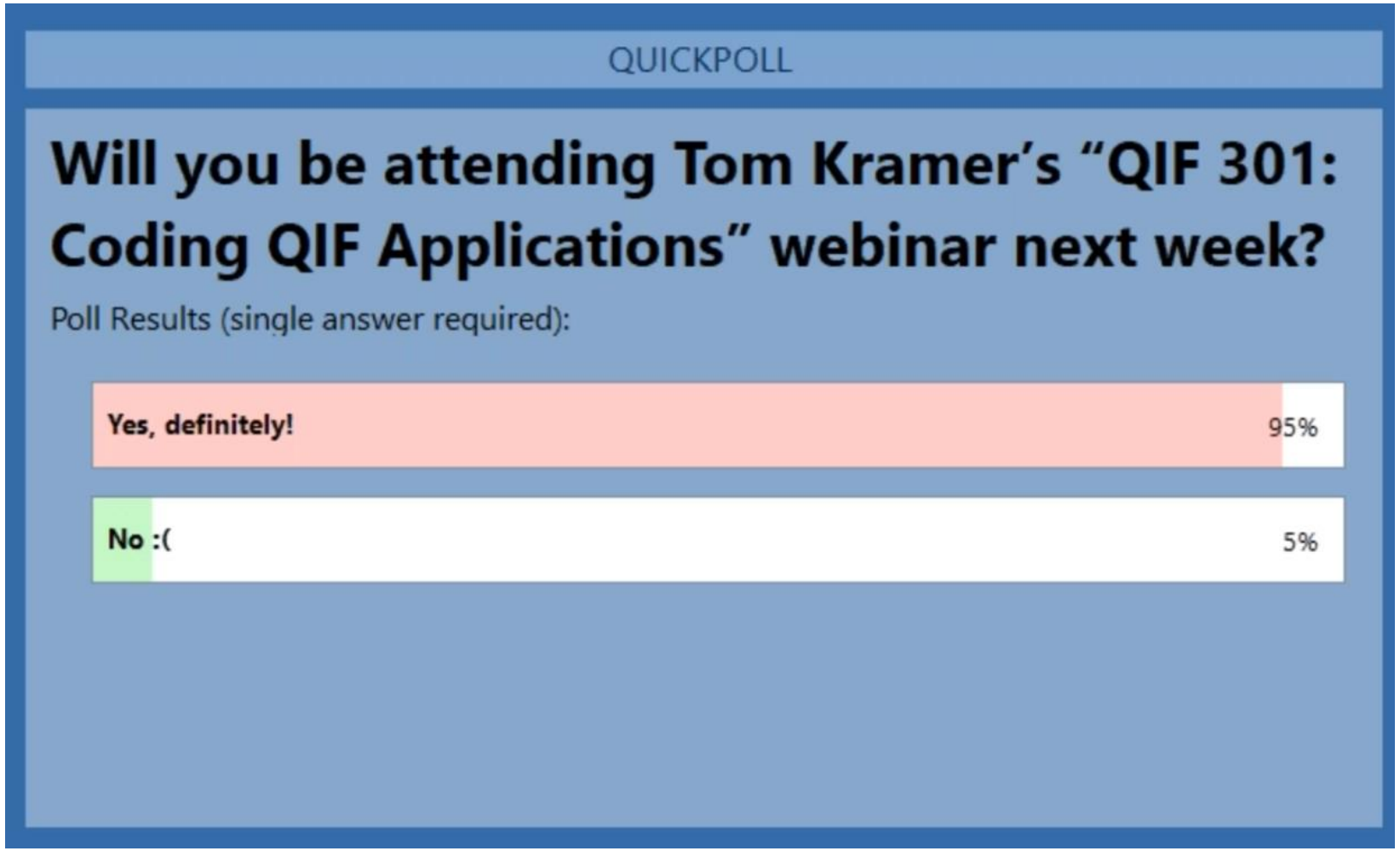
- "abc" tolerance on
- Part with "123" serial number

corresponds to a specific annotation on our authoritative product definition.

Remember: your metrology department gathers **more data** about your product & process than any other part of your enterprise. Are you leveraging this information?

This is Digital Twin.

Poll


QUICKPOLL
Will you be attending Tom Kramer's "QIF 301: Coding QIF Applications" webinar next week?
Poll Results (single answer required):
Yes, definitely!
95%
No :(
5%

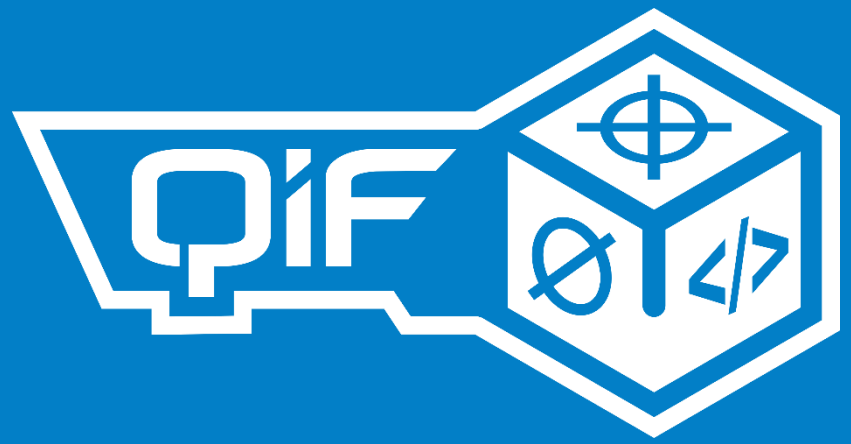
Questions

Check out our online community:

https://qualityinformationframework.github.io/

Schema Browser: If you want to dig into the schemas in detail, you should check out this schema browser. It allows you to easily navigate the structure and relationships of the QIF data model.

www.qifstandards.org



Digital Metrology Standards Consortium DMSC

DMSC's QIF Tutorials

May 7, 2020

QIF Tutorials

Basics, Technology, and **Applications**

QIF 101: Understanding QIF Basics –
Curtis Brown, Honeywell FM&T (April 23, 2020)

QIF 201: Reviewing QIF Technology –
Daniel Campbell, Capvidia (April 30, 2020)

QIF 301: Coding QIF Applications –
Tom Kramer, NIST Guest Researcher (May 7, 2020)

A Beginner's Guide to QIF3.0 Implementation

Thomas (Tom) Kramer
NIST Guest Researcher
Research Associate of Catholic University
QIF Developer & Software Developer

Thanks to Bob Stone

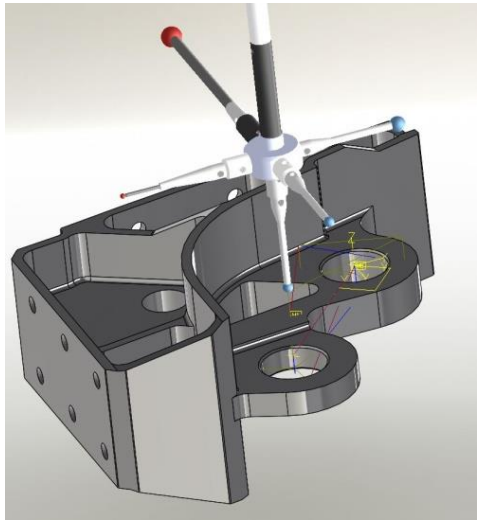
DMSC member with Origin International
QIF developer
Software developer
Alter ego

See also
http://info.originintl.com/resources/news-events/hello-world-guide-to-qif-by-bob-stone

Who is the DMSC?

Digital Metrology Standards Consortium

- A **non-profit**, cooperative sponsorship, **consortium** organization. Conceived & sponsored in 1983; Separate legal entity 2005, dba Digital, 2018.
- Dedicated to identifying, promoting, fostering, and encouraging the **development** and **interoperability** of information standards that benefit the digital metrology community.
- Preparing standards that **impact** digital **model-based quality** enterprise.
- A professional **group** of manufacturing metrologists, software developers, and innovators worldwide. Note: 500+ years of experience contributed to the QIF.
- Maintainers of **Dimensional Measuring Interface Standard** (**DMIS**) standard.
- Developers & maintainers of **Quality Information Framework** (**QIF**) standard.
- **ANSI accredited** standards making organization
- **A-Liaison** member of **ISO / TC 184 / SC 4** (allows for harvesting ANSI standards)



What is QIF3.0 …

from a software developer's perspective?

- It's a Standard – downloadable, documented, stable.
- Its scope is Quality: results, statistics, resources, plans, rules, MBD, visualization.
- It's a data model defined in XML Schema Definition Language (XSDL).
- Its data files are XML governed by the XSDL model and XSLT constraints.
- It's BIG.

How does a software developer support QIF?

- “Supporting QIF” means your software will read or write QIF documents in XML format.
- All QIF documents must be valid against the QIF XSD schema files and the QIF XSLT.

Getting started

- STEP 1:

  Download the (freely available) standard from qifstandards.org (includes text doc, XML schema files, XSLT, example XML instance files).

  Study the text doc.

  Study the HTML at qualityinformationframework.github.io/qif3-browser/qif3.html start at the QIFDocument *element*.

- STEP 2:

  Pick your approach to dealing with QIF.

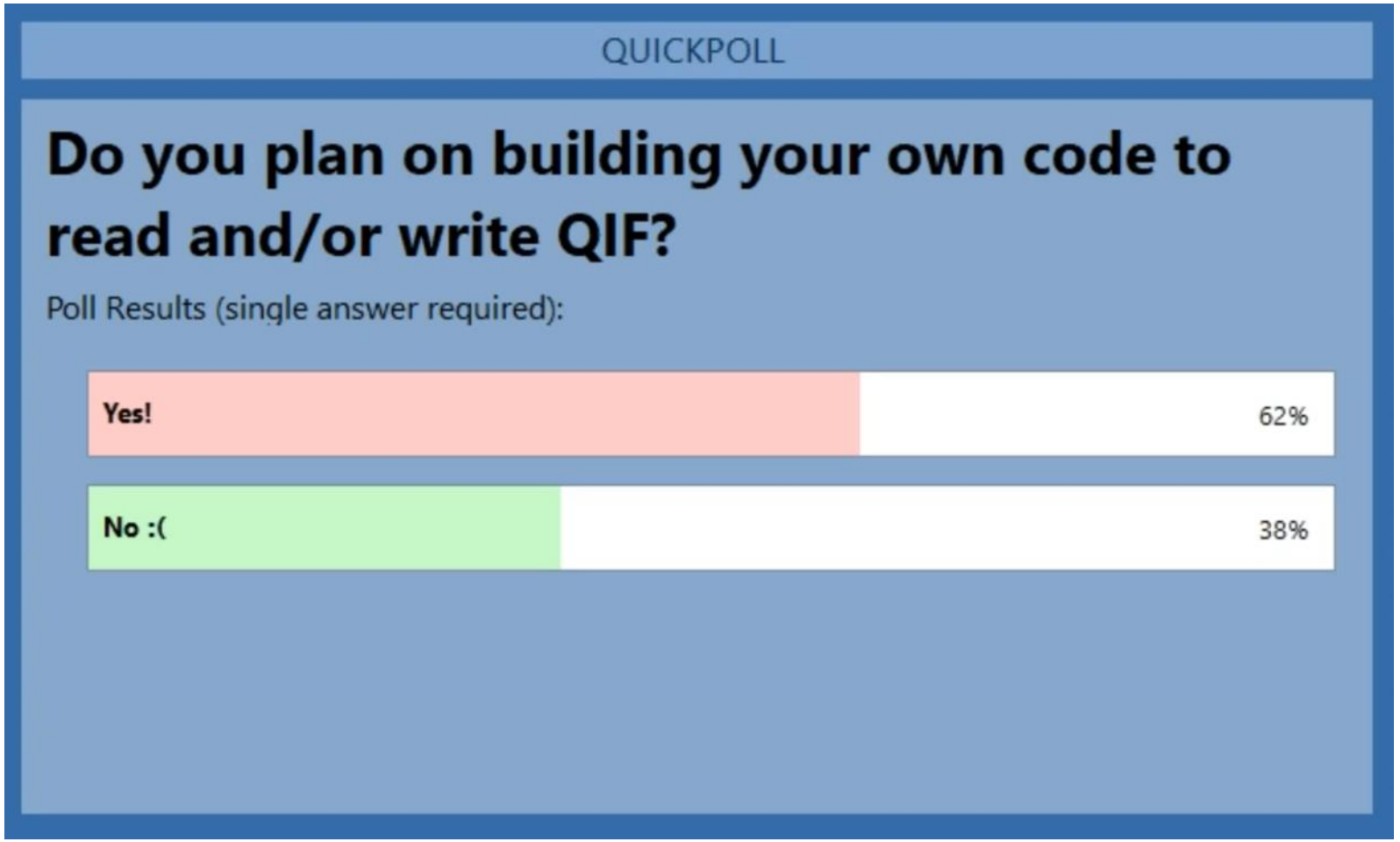
  print?

  Xpath?

  DOM?

  Source code bindings? ✓

Poll


QUICKPOLL
Do you plan on building your own code to read and/or write QIF?
Poll Results (single answer required):
Yes!
62%
No :(
38%

Finding source code binding generators

Check the Internet.

For example, search for “xsd to code”, or see https://en.wikipedia.org/wiki/XML_data_binding

Dimensional Metrology Standards Consortium DMSC

What's available (1):

- **"QIF_CPlusPlus"**
  C++ (native)
  CodeSynthesis XSD 3.3

- **"QIFdotNET"**
  C# (.NET, managed)
  Microsoft XSD

- **"demoQIFout"+ "demoQIFin"**
  Python
  PyXB (pxybgen)

What's available (2):

- Download QIF3.0DocumentBundle2020Feb14.zip from https://github.com/QualityInformationFramework/qif-community/tree/master/bindings
- Includes:
  - C++ classes for all of QIF 3.0 - header (hh) and implementation (cc) (but the "Signature" is modeled as a string)
  - C++ model-independent helper classes for primitive data
  - parsing a QIF 3.0 instance file into an abstract syntax tree (uses a YACC/Lex parser)
  - writing an abstract syntax tree to a QIF 3.0 instance file
  - a "read file in, write file out" minimal application
  - a Makefile.

But be careful

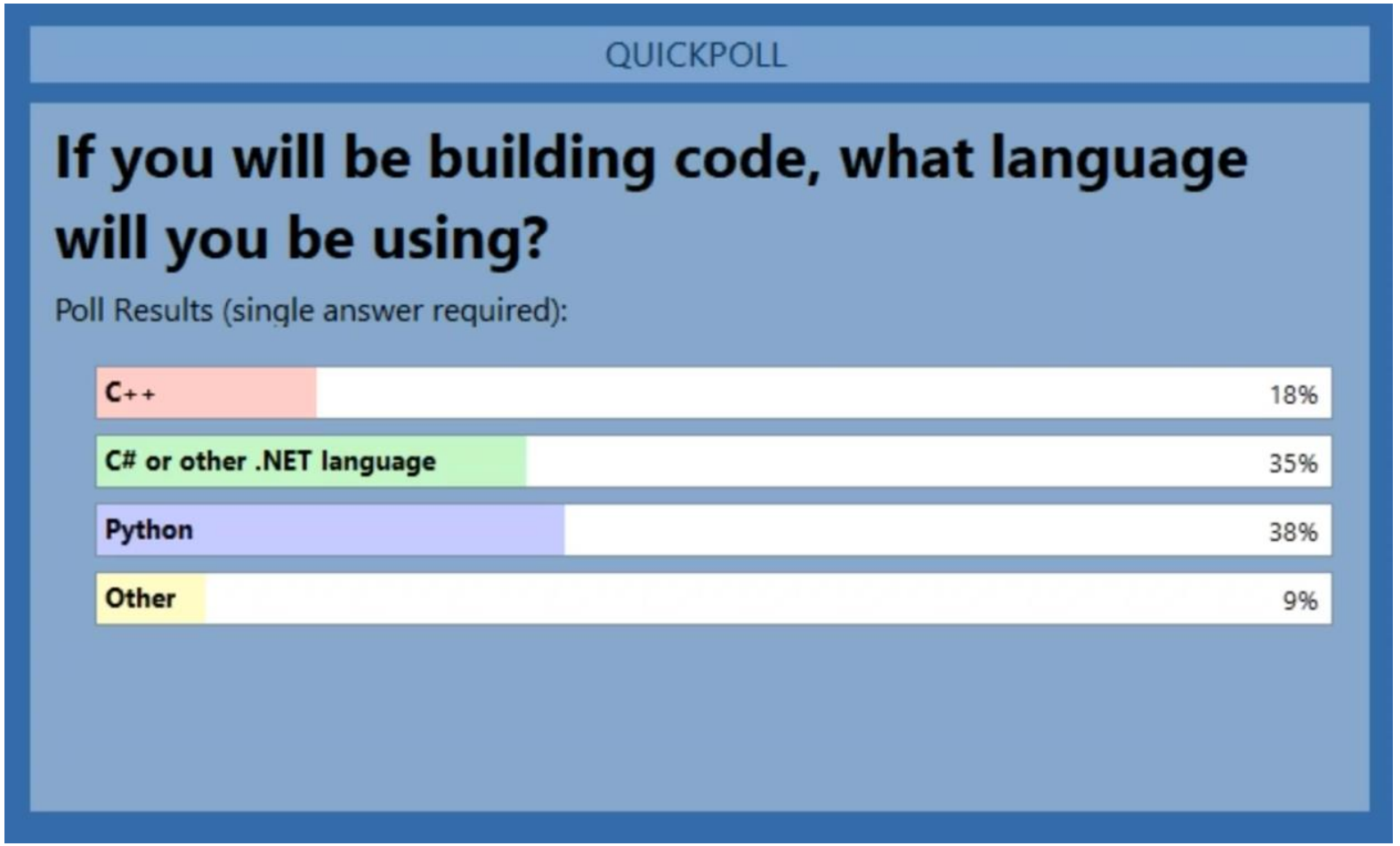
LEGAL DISCLAIMERS

- The free, open source software from DMSC, is provided “as is” and without any warranty.
  READ OUR “LICENSE.TXT” FOR DETAILS

- The DMSC does not provide the source code generators, or the code generated by them.
  The relationship between you and the source code generator is covered by THEIR software license.
  READ THEIR SOFTWARE LICENSE THOROUGHLY

- The DMSC neither endorses nor recommends any particular source code generator

Poll


QUICKPOLL
If you will be building code, what language will you be using?
Poll Results (single answer required):
C++
18%
C# or other .NET language
35%
Python
38%
Other
9%

Source code binding generators

What they do well:

Automatically create classes from schema files

From schema definition:

```
<xs:simpleType name="PointSimpleType">
  <xs:restriction base="ListDoubleType">
    <xs:length value="3"/>
  </xs:restriction>
</xs:simpleType>
```

To C++ source code:

```
class PointSimpleType: public ::xsd::qif2::ListDoubleType
{
  public:
  // Constructors.
  //
  PointSimpleType ();

  PointSimpleType (const ::xsd::qif2::ListDoubleType&);

  PointSimpleType (const xercesc::DOMElement& e,
                   ::xml_schema::flags f = 0,
                   ::xml_schema::container* c = 0);
```

What they don't do well:

- key/keyref checks
  Code implementing key/keyref checks is not generated.

The free open-source apps from DMSC

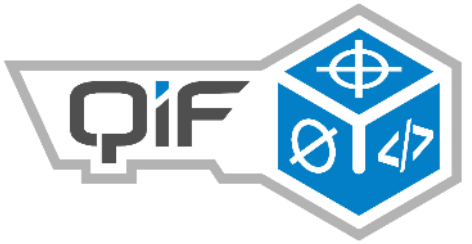
Three of the apps:

- use QIF binding code you generate using a code generator
- build the same QIF instance file using C++, C#, and Python
- also produce DMIS
- run in Windows.

The fourth app:

- uses the QIF binding C++ code provided in the qif-community site
- builds the same QIF instance file as the others
- does not produce DMIS
- is not tied to any particular OS
- has two versions to show different coding approaches.

Apps using free open source binding



Objects are connected by pointers in both apps.

One version uses “new” to make objects

```
FeatureItemsType * featItems = new FeatureItemsType();
featItems->setn(new NaturalType("1"));
```

The other uses automatic variables for objects (recommended) - more lines, fewer characters, memory management easier

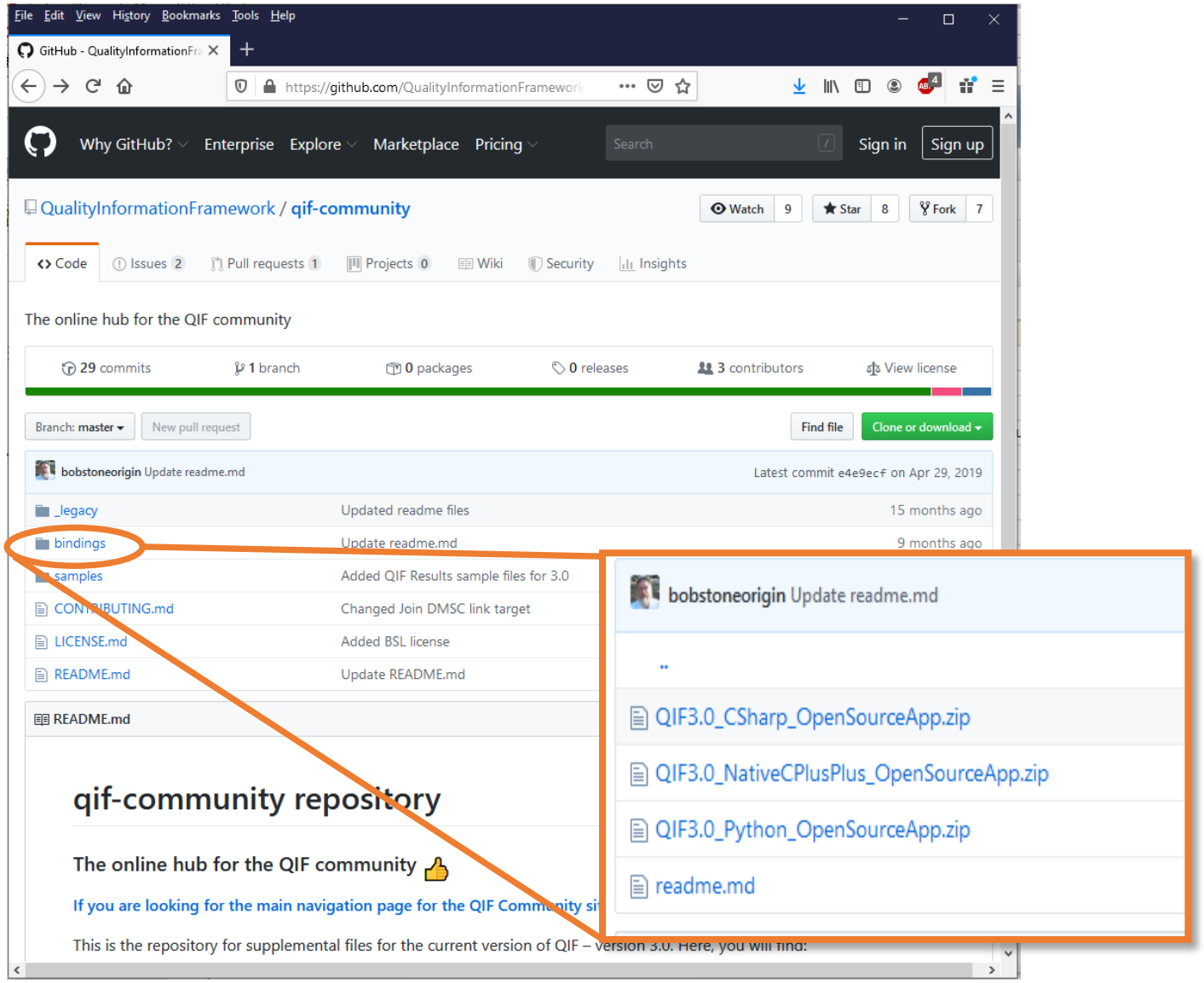
```
FeatureItemsType featItems;
NaturalType featItemsN("1");
featItems.setn(&featItemsN);
```

Where to find the apps:

qif-community

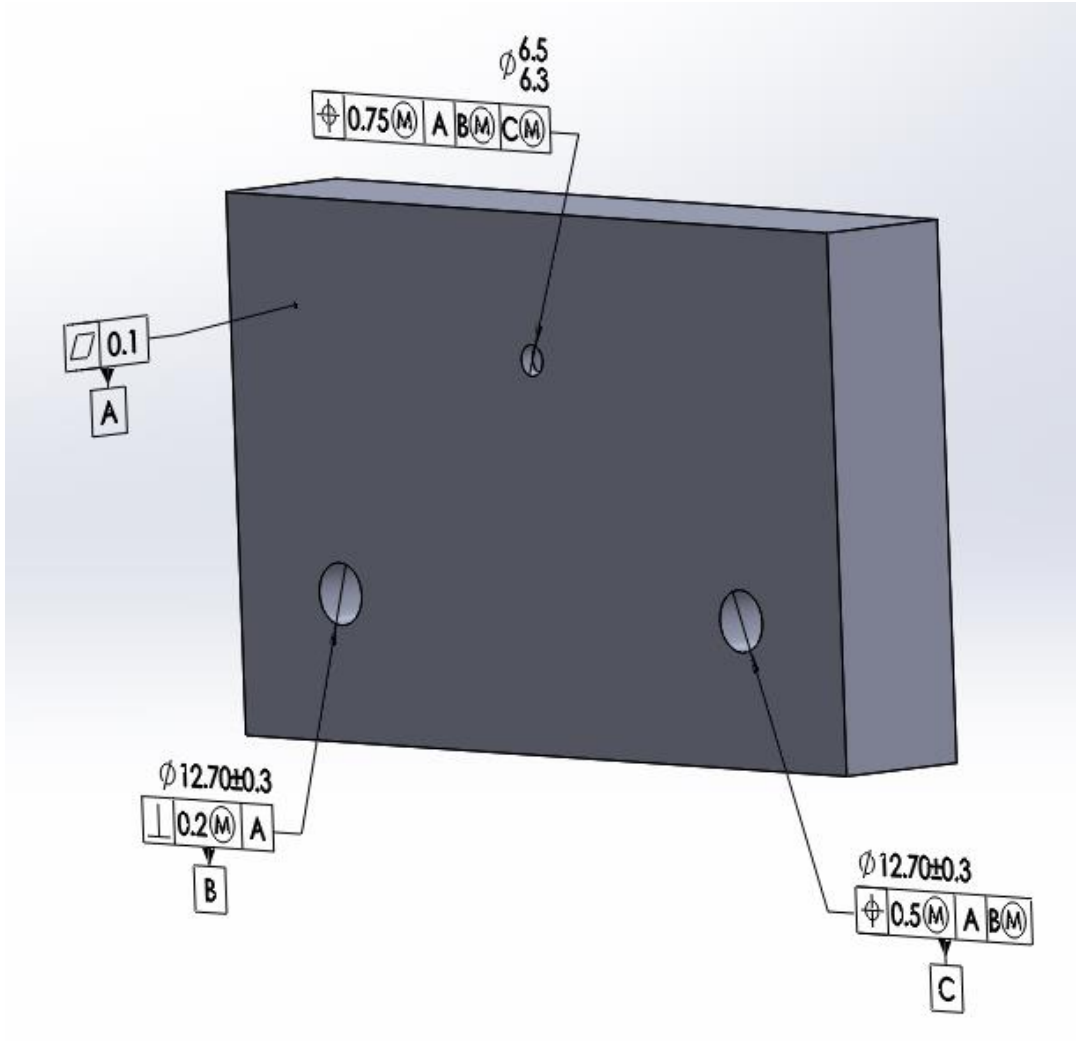
on GitHub

https://github.com/QualityInformationFramework/QualityInformationFramework.github.io



What the apps do (OUTPUT side):

Encode this:



To produce this:

```
<?xml version="1.0" encoding="UTF-8" standalone="no" ?>
<QIFDocument xmlns="http://qifstandards.org/xsd/qif3" idMax="51" versionQIF="3.0.0"
             xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
             xsi:schemaLocation="http://qifstandards.org/xsd/qif3 ../QIFApplications/QIFDocument.xsd">

  <QPId>906c4d97-5a81-4ccb-b328-2bab6b800765</QPId>

  <Header>
    <Application>
      <Name>QIF_CPlusPlus open source QIF application</Name>
      <Organization>Digital Metrology Standards Consortium (DMSC)</Organization>
    </Application>
  </Header>
...
  <DatumDefinitions n="3">
    <DatumDefinition id="1">
      <DatumLabel>A</DatumLabel>
...
  <DatumReferenceFrames n="3">
...
  <MeasurementResources>
    <MeasurementDevices n="2">
...
  <Features>
    <FeatureDefinitions n="3">
      <PlaneFeatureDefinition id="9"/>
...
  <Characteristics>
    <FormalStandardId>50</FormalStandardId>
    <CharacteristicDefinitions n="6">
      <FlatnessCharacteristicDefinition id="13">
        <ToleranceValue>0.1</ToleranceValue>
...
  <Plan>
...
  <Results>
    <MeasurementResultsSet n="1">
      <MeasurementResults id="51">
        <MeasuredFeatures n="4">
...
          <MeasuredCharacteristics>
...
</QIFDocument>
```

(OUTPUT) Producing QIF

The open source applications:

- Set up units
- Define datums
- Define datum reference frames
- Give feature definitions, nominals, items, and measurements
- Hook features to datums
- Give characteristic definitions, nominals, items, and measurements
- Define measurement resources
- Give a trivial measurement plan
- Generate a valid QIF document

Your data goes here

```
# plane location
planLoc = QIFDocument.PointType([0.0, 0.0, 0.0])
planANom.Location = planLoc

# plane normal
planNor = QIFDocument.UnitVectorType([0.0, 0.0, 1.0])
planANom.Normal = planNor
```

What the apps do (INPUT side):

Consumes this:

```
<?xml version="1.0" encoding="UTF-8" standalone="no" ?>
<QIFDocument xmlns="http://qifstandards.org/xsd/qif3" idMax="51" versionQIF="3.0.0"
             xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
             xsi:schemaLocation="http://qifstandards.org/xsd/qif3 ../QIFApplications/QIFDocument.xsd">

  <QPId>906c4d97-5a81-4ccb-b328-2bab6b800765</QPId>

  <Header>
    <Application>
      <Name>QIF_CPlusPlus open source QIF application</Name>
      <Organization>Digital Metrology Standards Consortium (DMSC)</Organization>
    </Application>
  </Header>
...
  <DatumDefinitions n="3">
    <DatumDefinition id="1">
      <DatumLabel>A</DatumLabel>
...
  <DatumReferenceFrames n="3">
...
  <MeasurementResources>
    <MeasurementDevices n="2">
...
  <Features>
    <FeatureDefinitions n="3">
      <PlaneFeatureDefinition id="9"/>
...
  <Characteristics>
    <FormalStandardId>50</FormalStandardId>
    <CharacteristicDefinitions n="6">
      <FlatnessCharacteristicDefinition id="13">
        <ToleranceValue>0.1</ToleranceValue>
...
  <Plan>
...
  <Results>
    <MeasurementResultsSet n="1">
      <MeasurementResults id="51">
        <MeasuredFeatures n="4">
...
        <MeasuredCharacteristics>
...
</QIFDocument>
```

To produce this:

```
FILNAM/'cpp.qif',5.3
$$ Digital Metrology Standards Consortium (DMSC))
$$ This DMIS results file produced from QIF document:
$$   C:/Users/Public/Documents/QIF-Community/QIF3.0_Open_Source_Apps/cpp.qif
$$ with DMSC's open source demoQIFin.py application written in Python
$$ using XML schema source code bindings created with PyXB (Pyxbgen)
UNITS/MM,ANGDEC

$$ Plane nominal DAT_A
OUTPUT/F(DAT_A),T(FLAT1)
F(DAT_A)=FEAT/PLANE,CART,0.0,0.0,0.0,0.0,0.0,1.0
T(FLAT1)=TOL/FLAT,0.1
$$ Plane actual DAT_A
OUTPUT/FA(DAT_A),TA(FLAT1)
FA(DAT_A)=FEAT/PLANE,CART,0.0,0.0,0.0,0.0,0.0,1.0
TA(FLAT1)=TOL/FLAT,0.023,INTOL
DATDEF/FA(DAT_A),DAT(A)

$$ Cylinder nominal DAT_B
OUTPUT/F(DAT_B),T(DIAM1_B),T(PERP1)
F(DAT_B)=FEAT/CYLNDR,INNER,CART,30.0,0.0,0.0,0.0,0.0,-1.0,12.7
T(DIAM1_B)=TOL/DIAM,-0.3,0.3
T(PERP1)=TOL/PERP,0.2,MMC,DAT(A)
$$ Cylinder actual DAT_B
OUTPUT/FA(DAT_B),TA(DIAM1_B),TA(PERP1)
FA(DAT_B)=FEAT/CYLNDR,INNER,CART,30.0,0.0,0.0,0.051,0.0,-0.9987,12.699
TA(DIAM1_B)=TOL/DIAM,12.699,INTOL
TA(PERP1)=TOL/PERP,0.07,INTOL,MMC,DAT(A)
DATDEF/FA(DAT_B),DAT(B)

$$ Cylinder nominal DAT_C
OUTPUT/F(DAT_C),T(DIAM1_C),T(POSN1)
F(DAT_C)=FEAT/CYLNDR,INNER,CART,150.0,0.0,0.0,0.0,0.0,-1.0,12.7
T(DIAM1_C)=TOL/DIAM,-0.3,0.3
T(POSN1)=TOL/POS,3D,0.5,MMC,DAT(A),DAT(B),MMC
$$ Cylinder actual DAT_C
OUTPUT/FA(DAT_C),TA(DIAM1_C),TA(POSN1)
FA(DAT_C)=FEAT/CYLNDR,INNER,CART,150.0,0.0,0.0,-0.0099,0.0099,-0.9999,12.72
TA(DIAM1_C)=TOL/DIAM,12.720000000000001,INTOL
TA(POSN1)=TOL/POS,3D,0.102,INTOL,MMC,DAT(A),DAT(B),MMC
DATDEF/FA(DAT_C),DAT(C)

$$ Circle nominal CIRC1
OUTPUT/F(CIRC1),T(DIAM2),T(POSN2)
F(CIRC1)=FEAT/CIRCLE,INNER,CART,90.0,50.0,-1.0,0.0,0.0,1.0,6.35
T(DIAM2)=TOL/DIAM,-0.1,0.1
T(POSN2)=TOL/POS,3D,0.75,MMC,DAT(A),DAT(B),MMC,DAT(C),MMC
$$ Circle actual CIRC1
OUTPUT/FA(CIRC1),TA(DIAM2),TA(POSN2)
FA(CIRC1)=FEAT/CIRCLE,INNER,CART,90.015,49.973,-1.0,0.0,0.0,1.0,6.2
TA(DIAM2)=TOL/DIAM,6.2,OUTOL
TA(POSN2)=TOL/POS,3D,0.0618,INTOL,MMC,DAT(A),DAT(B),MMC,DAT(C),MMC
ENDFIL
```

(INPUT) Consuming QIF, producing DMIS

The applications do this:

- Extracts units
- Feature/characteristic relationship
- Feature nominal
- Characteristic nominal
- Feature actual
- Characteristic actual
- Datum definitions
- Generates DMIS results document
- Views DMO document in NotePad

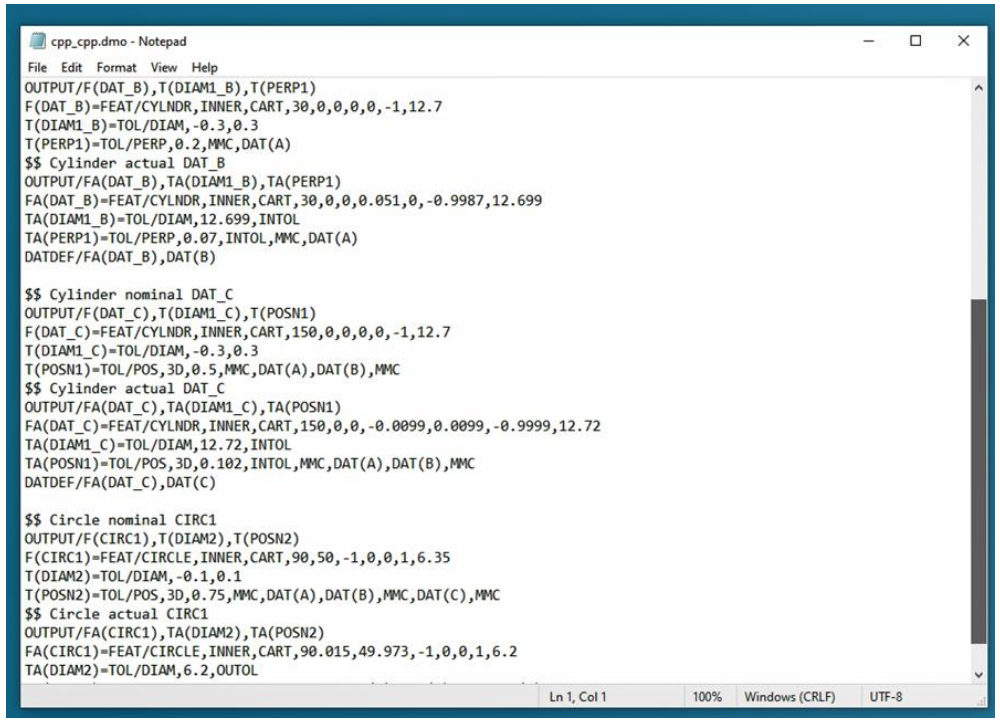
```
UNITS/MM,ANGDEC

$$ Plane nominal DAT_A
OUTPUT/F(DAT_A),T(FLAT1)
F(DAT_A)=FEAT/PLANE,CART,0,0,0,0,0,1
T(FLAT1)=TOL/FLAT,0.1
$$ Plane actual DAT_A
OUTPUT/FA(DAT_A),TA(FLAT1)
FA(DAT_A)=FEAT/PLANE,CART,0,0,0,0,0,1
TA(FLAT1)=TOL/FLAT,0.023,INTOL
DATDEF/FA(DAT_A),DAT(A)
```

```
<DatumDefinition id="1">
  <DatumLabel>A</DatumLabel>
  <FeatureNominalIds n="1">
    <Id>10</Id>
  </FeatureNominalIds>
</DatumDefinition>
```

```
<PlaneFeatureNominal id="10">
  <FeatureDefinitionId>9</FeatureDefinitionId>
  <Location>0 0 0</Location>
  <Normal>0 0 1</Normal>
</PlaneFeatureNominal>
```

Watch them run

QIF_CPlusPlus

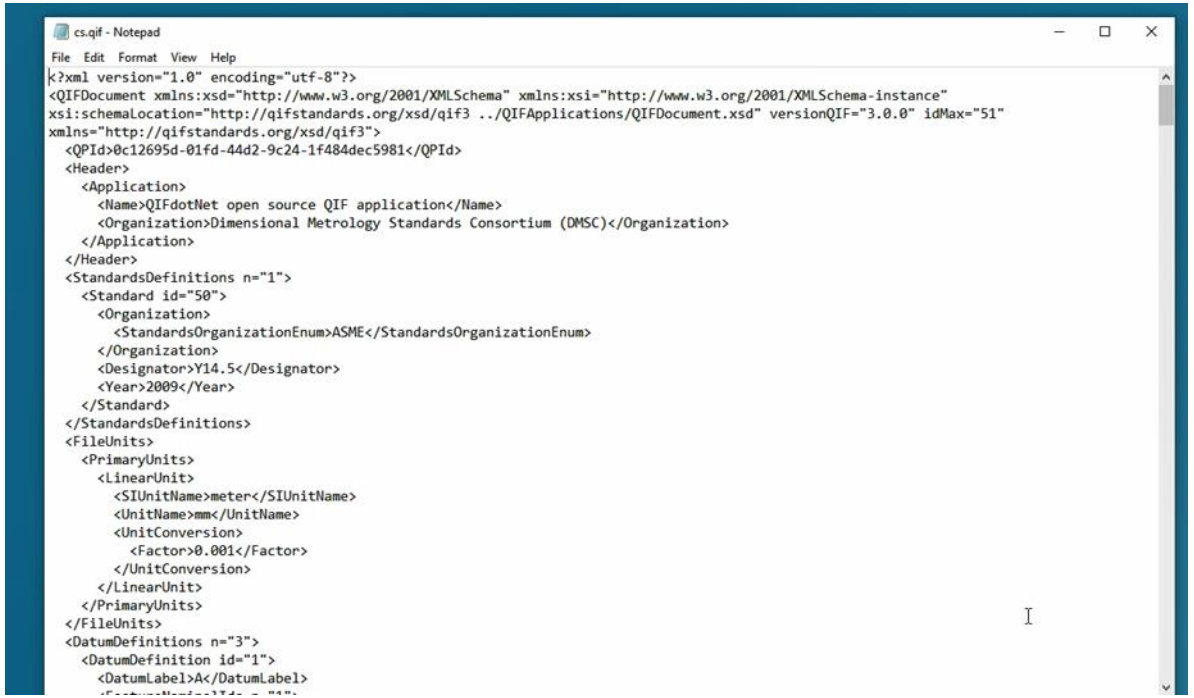


Click here to watch the video!

Watch them run

QIFdotNET

Microsoft® Visual C#®

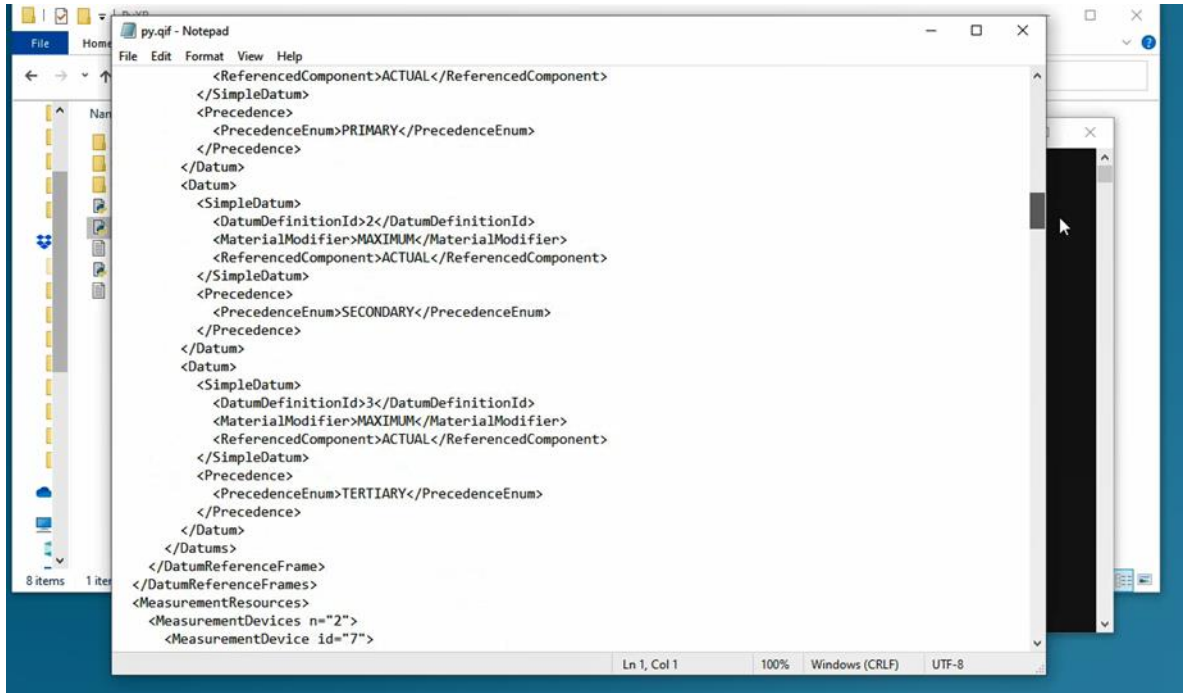


Click here to watch the video!

Watch them run

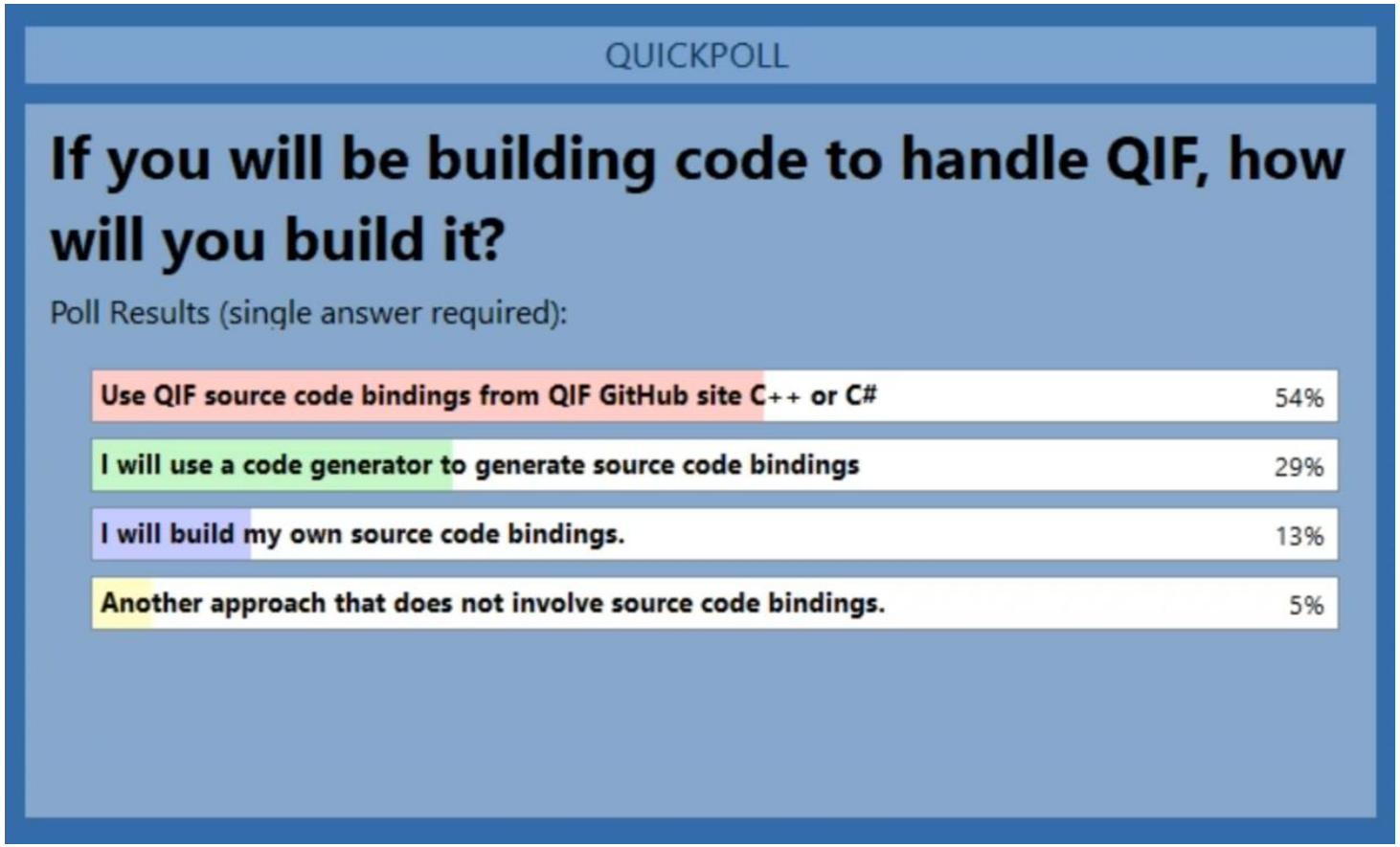
demoQIFout

demoQIFin



Click here to watch the video!

Poll


QUICKPOLL
If you will be building code to handle QIF, how will you build it?
Poll Results (single answer required):
Use QIF source code bindings from QIF GitHub site C++ or C# 54%
I will use a code generator to generate source code bindings 29%
I will build my own source code bindings. 13%
Another approach that does not involve source code bindings. 5%

From soup to nuts with DMSC QIF 3.0 C++

1. Download the C++ code (from qif-community on GitHub).
2. READ OUR LICENSE AGREEMENT.
3. Read the “ReadMe.txt” and follow the instructions.
4. Build and run the *read input file, write output file* app.
5. Build and run the two *generate tree, write output file* apps.

From soup to nuts with code generators

1. Download QIF schema files (from QIFStandards.org).
2. Download the App (from qif-community on GitHub).
3. READ OUR LICENSE AGREEMENT.
4. Read the “ReadMe.txt” and follow the instructions.
5. Download the source code generator.
6. READ THEIR LICENSE AGREEMENT.
7. *Replace troublesome schema files.
8. Make the source code bindings.
9. *Massage the source code bindings.
10. Build and run the app.

The source code:

- Annotated
- All 4 apps have parallel implementations
- Write to lists (n)
- Check for lists
- Traverse lists
- Handle substitute elements
- For these 3, code is regionalized


```python
#region Datum definitions
datumDefs = QIFDocument.DatumD
datumDefs.n = 3

# define ABC datums
# define datums ABC (as simple

datA = QIFDocument.DatumDefini
datA.id = qifdata.idMax
qifdata.idMax += 1
datA.DatumLabel = "A"
datumDefs.append(datA)

datB = QIFDocument.DatumDefini
datB.id = qifdata.idMax
qifdata.idMax += 1
datB.DatumLabel = "B"
datumDefs.append(datB)

datC = QIFDocument.DatumDefini
datC.id = qifdata.idMax
qifdata.idMax += 1
datC.DatumLabel = "C"
datumDefs.append(datC)

qifdata.DatumDefinitions = dat
#endregion

#region Datum reference frames
drfDefs = QIFDocument.DatumRef
drfDefs.n = 3

#  nd ABC datum

#make a datum reference frame
```

```cpp
#pragma region Datum definitions
///define ABC datums

//define datums ABC (as simpl
DatumDefinitionType datA;
datA.id(qifid++);//required
datA.DatumLabel(_T("A"));
//datumDefs.DatumDefinition().
//datumDefs.n(datumDefs.n()+
DatumDefinitionType datB;
datB.id(qifid++);//required
datB.DatumLabel(_T("B"));
//datumDefs.DatumDefinition().
//datumDefs.n(datumDefs.n()+
DatumDefinitionType datC;
datC.id(qifid++);//required
datC.DatumLabel(_T("C"));
//datumDefs.DatumDefinition().
//datumDefs.n(datumDefs.n()+
#pragma endregion Datum definiti

#pragma region Datum reference f
///define A, AB, and ABC datu

#pragma region A
//make a datum reference fram
DatumReferenceFrameType drfA;
drfA.id(qifid++);//required
//datums list
 sType drfAdats;
ats.n(0);

is simple datum with no m
```

```
#region Datum definitions
/// define ABC datums

// define datums ABC (as simple datum letters, we
DatumDefinitionType datA = new DatumDefinitionTyp
datA.id = qifid++; // required id
datA.DatumLabel = "A";
datumDefs.Add(datA);
DatumDefinitionType datB = new DatumDefinitionTyp
datB.id = qifid++; // required id
datB.DatumLabel = "B";
datumDefs.Add(datB);
DatumDefinitionType datC = new DatumDefinitionTyp
datC.id = qifid++; // required id
datC.DatumLabel = "C";
datumDefs.Add(datC);
#endregion

#region Datum reference frames
/// define A, AB, and ABC datum reference frames

#region A
// make a datum reference frame A
DatumReferenceFrameType drfA = new DatumReference
drfA.id = qifid++; // required id
// datums container
drfA.Datums = new DatumsType();
// array of 1 datum
tum = new DatumWithPrecedenceType[1
= 1;

 datum with no material condition m
A = new DatumType();
```

Details, details:

The source code gets you past these hurdles
(that you didn't even know existed):

- Getting a valid header to the QIF document
- By reference or deep clone?
- C# — list elements, substitution groups
- C# — elements of same type
- C# — enumerations are never null
- Python — substitution groups
- C++ — resolving namespace conflicts

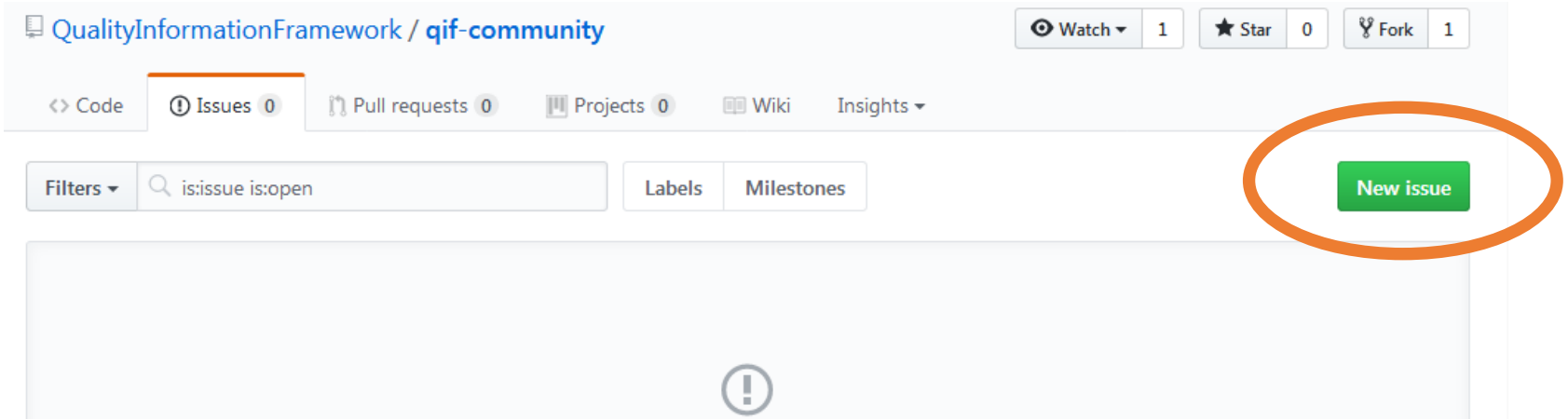
Best of luck:

Take-aways (we hope):

- Free, open source C++ code is available for QIF 3.0.
- QIF works with XSD source code binding generators.
- There are DMSC open source QIF apps available.
- These can be used as implementation templates for other source code generators.

Support:

- Enter an issue. qif-community on GitHub



Contribute:

Do an app in a different language:

On a different OS:

Mac OS

With a different compiler:

Or with a different source code generator:

CODE SYNTHESIS 4.0

Liquid XML Data Binding

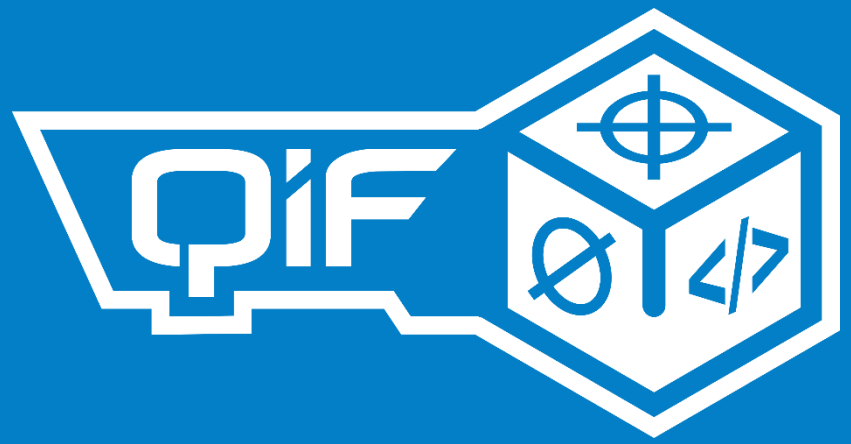
ALTOVA® XMLSpy® 2017

And share*

* DO NOT SHARE AUTOMATICALLY GENERATED SOURCE CODE

Questions

www.qifstandards.org



Digital Metrology Standards Consortium DMSC

DMSC's QIF Tutorials

May 7, 2020

QIF Tutorials

Basics, Technology, and Applications

QIF 101: Understanding QIF Basics –
Curtis Brown, Kansas City NSC (April 23, 2020)

QIF 201: Reviewing QIF Technology –
Daniel Campbell, Capvidia (April 30, 2020)

QIF 301: Coding QIF Applications –
Tom Kramer, NIST Guest Researcher (May 7, 2020)