

QUDT: An OWL Ontology for Measurable Quantities, Units, Dimension Systems, and Dimensional Data Types

James “Chip” Masters (TopQuadrant, Inc.)

Ralph Hodgson (TopQuadrant, Inc.)

Paul Keller (NASA)



Presentation Overview

- Introductory Remarks
- Definitions of Main Concepts Modeled
- The Main Classes in the QUDT Ontology (with examples)
- Functions for Unit Conversion and Computing Products/Quotients of Quantity Kinds
- Current and Future Work
- References and Credits



Introductory Remarks

- Sponsored by NASA's Constellation Program
 - 30+ year space exploration program to return to the moon and eventually send humans to Mars
 - Investing in Semantic Technologies to meet short term and long term '-ilities'
 - Affordability; Sustainability; Operability; etc.
- One of a set of related ontologies designed to support the Constellation Program Lifecycle, as well as domain specific areas, including
 - Telemetry and command
 - Modeling and simulation



Main Concepts Modeled (I)

- Quantity Kind - an observable and measurable property of objects, systems, or events that is independent of any particular measurement or expression of magnitude
 - Mass, Velocity, Power, Torque, etc.
- Quantity - the magnitude of a quantity kind for a particular object, system, event, or class of that can be expressed as a numerical value with respect to a chosen scale
 - Mass of a hydrogen atom, Escape velocity of the Earth, Duration of this teleconference, etc.



Main Concepts Modeled (II)

- Unit of Measure - a chosen scale for expressing the magnitude of a quantity as a numerical value; units are associated with the quantity kinds they quantify.
 - Meter, Kilogram, Volt, etc.
- Quantity Value - the numerical value of a quantity's magnitude with respect to a chosen unit of measure for the corresponding quantity kind
 - 2 Meters, 5 Kilograms, 1000 Volts, etc.



Main Concepts Modeled (III)

- Quantity System - a collection of quantity kinds comprised of a base set and a derived set such that each member of the collection can be expressed as a product of members of the base set raised to a rational power.
- Unit System - a collection of units in which each member is a measurement scale for a quantity kind in a corresponding quantity system.

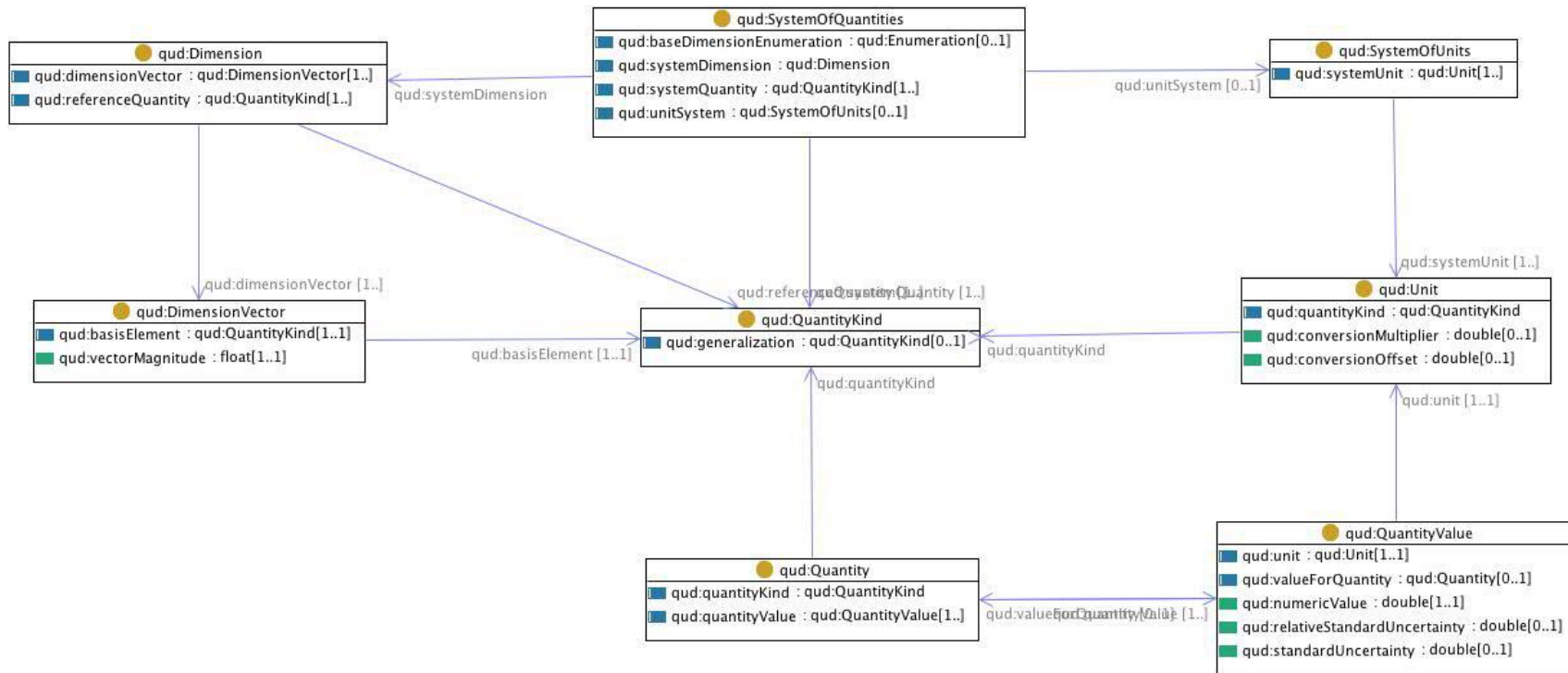


Main Concepts Modeled (IV)

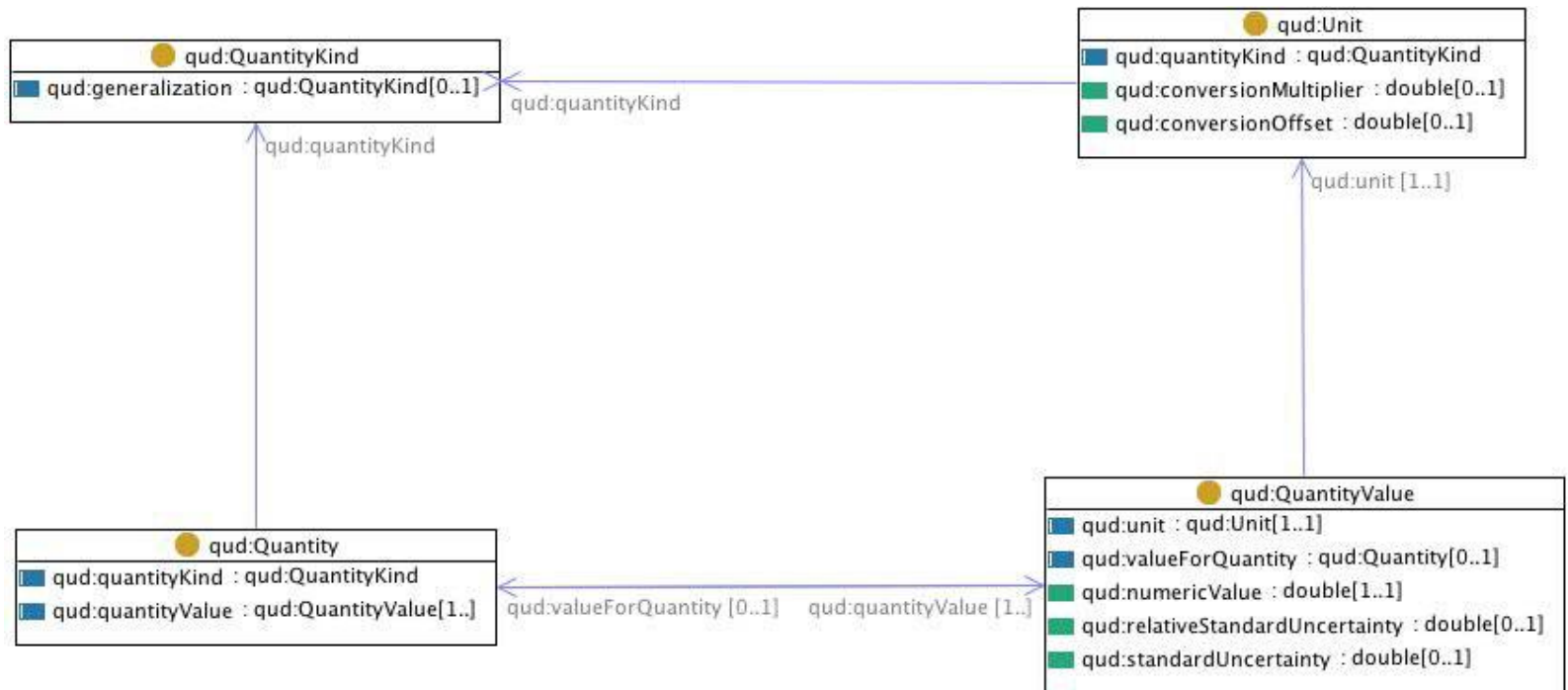
- System Dimension - a mapping from a tuple of rational numbers to a product of base quantity kinds such that the tuple members correspond to the exponents of the base quantity kinds; each quantity kind in a system corresponds to exactly one system dimension; multiple quantity kinds may correspond to the same dimension.



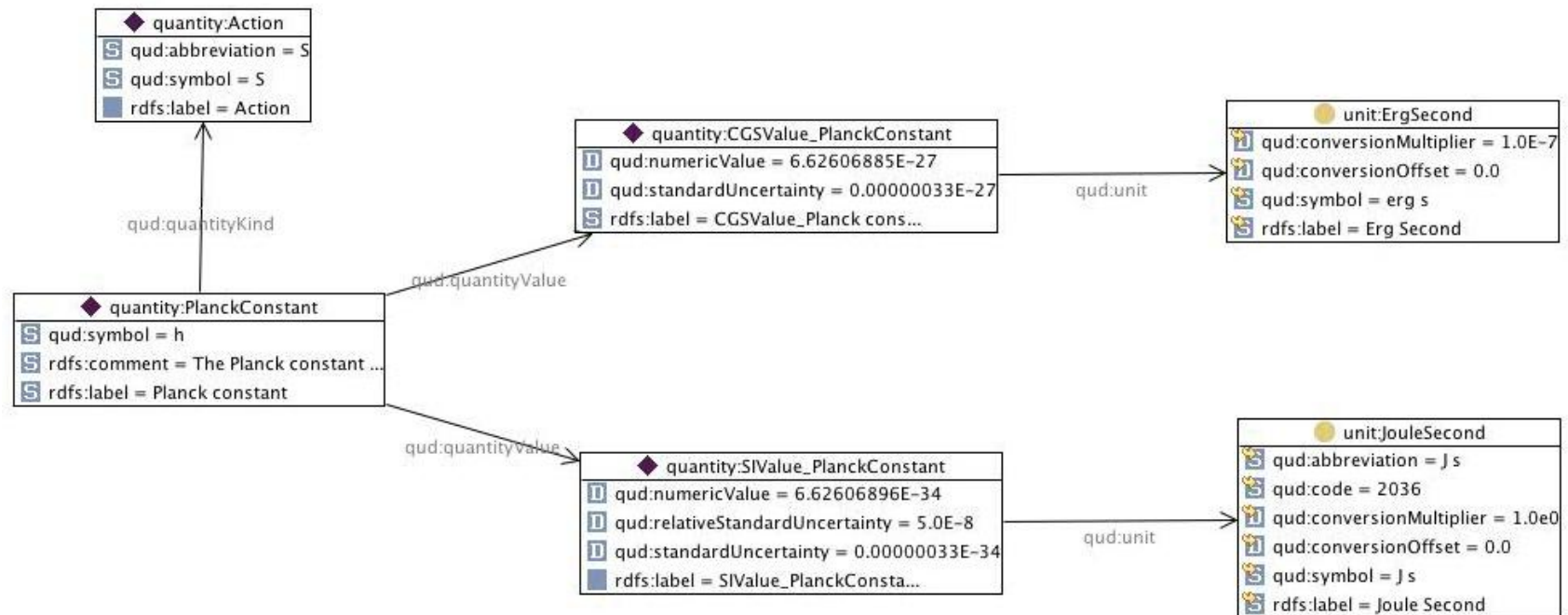
Main QUDT Class Diagram



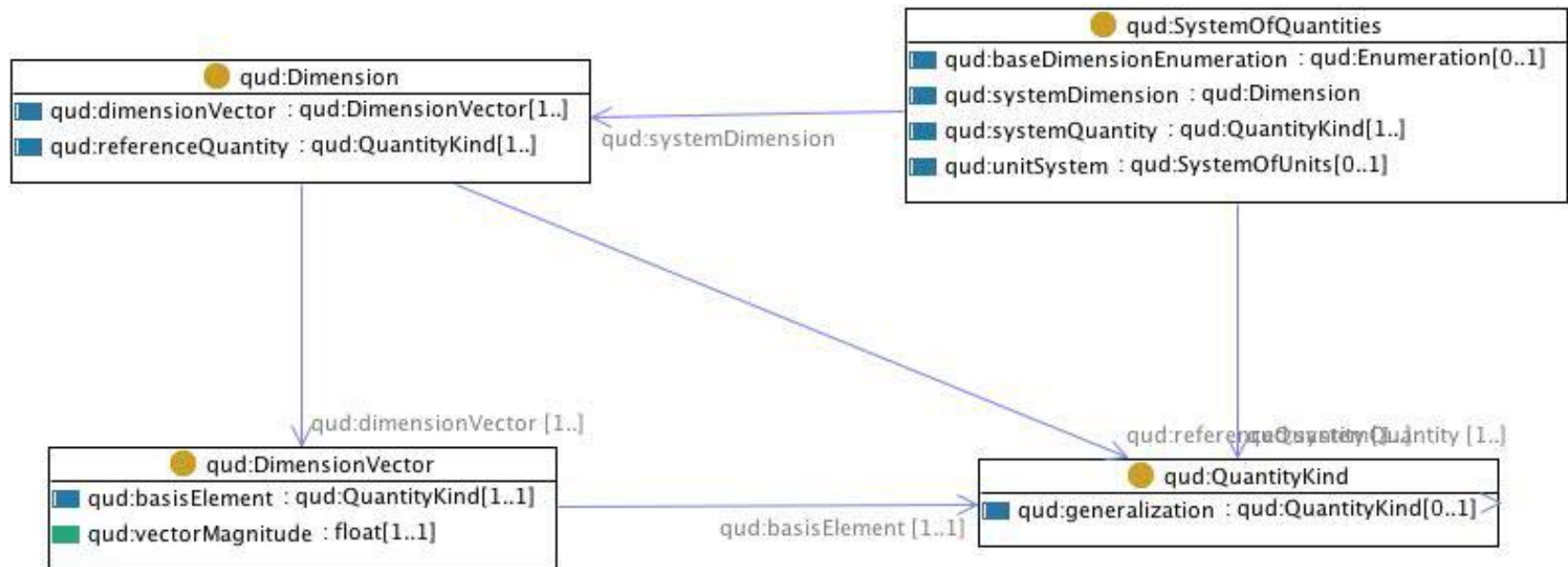
Expressing Quantity Values



Example: Value of Planck's Constant in SI and CGS Units



System Dimensions for Quantity Kinds



Example: Dimensions for Permittivity



SPIN Functions Using QUDT

- SPARQL Inference Notation (SPIN) -- RDF vocabularies enabling the use of SPARQL to define constraints and inference rules on Semantic Web models
- QUDSPIN -- A library of SPIN functions and templates utilizing the QUDT ontology to convert between units, calculate products and quotients of quantities, etc.



Unit Conversion Function

The screenshot shows the Eclipse IDE interface for developing an ontology. The main window displays the 'Class Form' for the class `qudspin:convert`. The class is annotated with `rdfs:comment` and `rdfs:label`. It is a subclass of `qudspin:UnitFunctions`. The class has three arguments: `sp:arg1` of type `xsd:double`, `sp:arg2` of type `qud:Unit`, and `sp:arg3` of type `qud:Unit`. The return type is `xsd:double`. The class is also constrained by `spin:constraint`.

The `spin:body` property contains the following SPARQL query:

```
SELECT ?value
WHERE {
  ?arg2 qud:conversionMultiplier ?M1 .
  ?arg2 qud:conversionOffset ?O1 .
  ?arg3 qud:conversionMultiplier ?M2 .
  ?arg3 qud:conversionOffset ?O2 .
  LET (?value := (((?arg1 * ?M1) + ?O1) - ?O2) / ?M2)) .
}
```

The `spin:constraint` property contains the following SPARQL query:

```
ASK WHERE {
  ?this sp:arg2 ?arg2 .
  ?this sp:arg3 ?arg3 .
  FILTER (!qudspin:comparableUnits(?arg2, ?arg3)) .
}
```

The `spin:returnType` property is set to `xsd:double`. The `rdfs:type` property is set to `spin:Function`.

The left sidebar shows the Navigator view with the following tree structure:

- cc:Requirement (6)
- cc:Work
- owl:AllDifferent
- owl:DataRange
- owl:Ontology (12)
- owl:Thing
- rdf:List (1)
- rdf:Property (123)
- rdf:Statement
- rdfs:Class (72)
- rdfs:Container
- rdfs:Literal
- sp:SystemClass
- spin:ConstraintViolation
- spin:Modules
 - spin:Functions
 - glyph:Functions
 - qudspin:Functions
 - qudspin:DeprecatedFunctions
 - qudspin:DimensionFunctions
 - qudspin:getDimensionSymbol
 - qudspin:InternalFunctions
 - qudspin:QuantityFunctions
 - qudspin:UnitFunctions
 - qudspin:comparableUnits (3)
 - qudspin:convert (1)
 - qudspin:convertLiteral
 - qudspin:getCoherentUnit (2)
 - qudspin:getUnitProduct
 - qudspin:getUnitQuotient
 - spl:BooleanFunctions
 - spl:MathematicalFunctions
 - spl:MiscFunctions
 - spl:OntologyFunctions
 - sol:StringFunctions

The right sidebar shows the Properties view with a list of properties including `cc:jurisdiction`, `cc:legalcode`, `cc:morePermissions`, `cc:permits`, `cc:prohibits`, `cc:requires`, `oec:attributedSource`, `owl:differentFrom`, `owl:disjointWith`, `owl:inverseOf`, `owl:sameAs`, `qud:baseDimensionEnum`, `qud:basisElement`, `qud:dimensionInverse`, `qud:dimensionVector`, `qud:element`, `qud:exactMatch`, `qud:generalization`, `qud:quantityCategory`, `qud:quantityKind`, `qud:quantityOfSystem`, `qud:quantityValue`, `qud:referenceThing`, `qud:specialization`, `qud:systemDimension`, `qud:systemQuantity`, `qud:systemUnit`, `qud:unit`, `qud:unitFor`, `qud:unitOfSystem`, `qud:unitSystem`, and `qud:valueForQuantity`.

Unit Conversion Example

The screenshot displays the Eclipse IDE interface for TopBraid. The main workspace is divided into several panes:

- Left Pane:** A tree view of classes, with `unit:ElectronVoltSecond` selected.
- Top Left Pane (Class Form):** Shows the configuration for `unit:ErgSecond`.
 - Name: `unit:ErgSecond`
 - Annotations: `rdfs:label` is set to `Erg Second`.
 - Class Axioms: `rdfs:subClassOf` is set to `rdfs:Resource`.
 - Other Properties:
 - `qud:conversionMultiplier`: `1.0E-7`
 - `qud:conversionOffset`: `0.0`
 - `qud:quantityKind`: `quantity:AngularMomentum`
 - `qud:symbol`: `erg s`
 - `rdf:type`: `qud:AngularMomentumUnit` and `qud:DerivedUnit`
- Top Right Pane (Class Form):** Shows the configuration for `unit:ElectronVoltSecond`.
 - Name: `unit:ElectronVoltSecond`
 - Annotations: `rdfs:label` is set to `Electron Volt Second`.
 - Class Axioms: `rdfs:subClassOf` is set to `rdfs:Resource`.
 - Other Properties:
 - `qud:abbreviation`: `eV s`
 - `qud:conversionMultiplier`: `1.6021765314E-19`
 - `qud:conversionOffset`: `0.0`
 - `qud:quantityKind`: `quantity:AngularMomentum`
 - `qud:symbol`: `eV s`
- Bottom Left Pane (Query Editor):** Contains a SPARQL query:

```
SELECT
?result
WHERE
{
LET (?result := qudspin:convert(6.62606885E-27, unit:ErgSecond, unit:ElectronVoltSecond))
}
```
- Bottom Middle Pane (Query Results):** Shows the result of the query: `[result]` with the value `4.135667150367048E-15`.
- Bottom Right Pane:** A list of available classes, including `qudspin:getDimensionSymbol`, `qudspin:getDimensionSymbol-Int`, `qudspin:getQuantityKindDimensio`, and `qudspin:getQuantityKindDimensio`.



Quantity Kind Product Function

The screenshot displays the Eclipse IDE interface for editing an OWL class. The main window shows the 'Class Form' for the class `qudspin:getQuantityKindProduct`. The interface is divided into several panes:

- Navigator:** Shows a tree view of the project structure, with `qudspin:Functions` expanded to show `qudspin:QuantityFunctions`, where the current class is selected.
- Class Form:** Contains the following sections:
 - Name:** `qudspin:getQuantityKindProduct`
 - Annotations:**
 - `rdfs:label`: `get dimension product`
 - `rdfs:comment`: `This function computes the product of two quantity kinds that are defined within the same system; arg1 - a quantity system; arg2 - a quantity kind; arg3 - a quantity kind; returns the product of arg2 and arg3`
 - Class Axioms:**
 - `rdfs:subClassOf`: `qudspin:QuantityFunctions`
 - Other Properties:**
 - `spin:body`:

```
SELECT ?result
WHERE {
  ?pred rdfs:subPropertyOf* qud:systemQuantity .
  ?arg1 ?pred ?result .
  FILTER qudspin:isQuantityKindProduct(?arg1, ?arg2, ?arg3, ?result) .
}
```
 - `spin:constraint`:
 - Argument `sp:arg1` : `qud:SystemOfQuantities`
 - Argument `sp:arg2` : `qud:QuantityKind`
 - Argument `sp:arg3` : `qud:QuantityKind`
 - Inter Argument 1-2 : `qud:systemQuantity`
 - Inter Argument 1-3 : `qud:systemQuantity`
 - `spin:returnType`: `qud:QuantityKind`
 - `rdf:type`: `spin:Function`
- Properties:** A list of available properties on the right side of the editor.

The bottom status bar includes tabs for Error Log, Instances, SPARQL, Imports, and References, along with SVN Repositories and Basket icons.



Quantity Kind Product Example

The screenshot displays the TopBraid Eclipse Platform interface. The main window shows two 'Resource Form' panes for the resource 'dim.Dimension_SI_L2TI'. The left pane shows the 'Other Properties' section with 'qud:dimensionVector' expanded, listing various vector types (e.g., dim:Vector_I1, dim:Vector_J0, etc.). The right pane shows the same resource form but with 'qud:referenceQuantity' expanded, listing 'quantity:ElectricQuadrupoleMoment'. Below the resource forms is a 'Query Editor' with a SPARQL query:

```
SELECT
?result
WHERE
{
  LET (?result := qudspin:getQuantityKindProduct(
    quantity:SystemOfQuantities_SI,
    quantity:ElectricDipoleMoment,
    quantity:Length))
}
```

The query results pane shows a single result: 'quantity:ElectricQuadrupoleMoment'. On the right side, a list of methods is visible, including 'qudspin:getDimensionSymbol', 'qudspin:getDimensionSymbol-Int', 'qudspin:getQuantityKindDimensic', and 'qudspin:getQuantityKindDimensic'.



Current and Future Work

- Dimensional Data Type Systems
- Equations (Physical Laws) Relating Quantity Kinds
- Structural Type of Quantity Kinds (scalar, vector, tensor, etc.)
- Association Between Measurement Events, Measured Quantities, and Expression in Engineering Units
- Dimensionless Numbers



References and Sources

- [Bureau International des Poids et Mesures](#)
- [The NIST Reference on Constants, Units, and Uncertainty](#)
- [VIM 3: International Vocabulary of Metrology](#)
- [Eric Weisstein's World of Science](#)
- Classical Electrodynamics, J.D. Jackson, Wiley & Sons, 3rd Edition (1998)



Links

- The QUDT Ontology is available at <http://www.qudt.org>. Includes:
 - QUDT Schema and Instance Data
 - Physical Constants published in the NIST Reference on Constants, Units and Uncertainty
 - Links between QUDT URIs and DBPedia URIs (uses SKOS ontology for linking)

