Computer Standards & Interfaces 000 (2017) 1–4

Contents lists available at ScienceDirect

Computer Standards & Interfaces

journal homepage: www.elsevier.com/locate/csi

Architecture for software-assisted quantity calculus☆

David Flater

National Institute of Standards and Technology, 100 Bureau Dr, Gaithersburg, MD 20899, USA

ARTICLE INFO

Keywords:
SI
Quantity
Unit
Uncertainty
Value
Unit 1

ABSTRACT

A quantity value, such as 5 kg, consists of a number and a reference (often an International System of Units (SI) unit) that together express the magnitude of a quantity. Many software libraries, packages, and ontologies that implement "quantities and units" functions are available. Although all of them begin with SI and associated practices, they differ in how they address issues such as *ad hoc* counting units, ratios of two quantities of the same kind, and uncertainty. This short article describes an architecture that addresses the complete set of functions in a simple and consistent fashion. Its goal is to encourage more convergent thinking about the functions and the underlying concepts so that the many disparate implementations, present and future, will become more consistent with one another.

Published by Elsevier B.V.

1. Introduction

In the scientific conventions of the early 19th century, mathematical operations applied only to numbers, and units of measure were just information that described what the numbers represented. This belief gave way to the practice of including units of measure within the scope of the mathematical operations, thereby formalizing the method of working with combinations of units. The resulting *quantity calculus* methodically determines the units of derived quantities and protects us from the error of computing nonsensical combinations of quantities that have different dimensions [2]. For example, if a property line is moved by 3 m, we are allowed to add 3 m to the width of the lot, but we cannot add 3 m to the lot size that is measured in m^2. We must multiply 3 m by the depth of the lot (x m) to obtain the change to the lot size ($3x$ m^2).

Many software libraries and packages that implement "quantities and units" (Q&U) functions are available; for example:[1]

- Mathematica, quantities and units in the Wolfram language [3];
- Maxima package ezunits [4];
- Modelica, physical variables and SIunits library [5];
- LabVIEW unit labels [6];
- MathCad units [7];
- GNU Units [8];
- UDUNITS [9];
- R packages for units of measure (e.g., units and udunits2 [10]);
- Ruby gems for units of measure (e.g., ruby-units [11], phys-units [12], and unitwise [13]; there are at least 10);
- C++ libraries for units of measure (e.g., Boost.Units [14]);
- F# units of measure (a built-in language feature) [15];
- Python package numericalunits [16]; and
- Julia packages SIUnit [17] and Unitful [18].

A comparable number of formal models and ontologies related to quantities and units exist; for example:

- Conceptual model of the International Vocabulary of Metrology (VIM) [19, Annex A];
- Dybkaer's Ontology on Property [20];
- Unified Code for Units of Measure (UCUM) [21];
- Quantities and Units of Measure Ontology Standard (QUOMOS) [22];
- Units Markup Language (UnitsML) [23];
- Quantities, Units, Dimensions, and Data Types Ontologies (QUDT) [24];
- Quantities, Units, Dimensions, Values (QUDV) in Systems Modeling Language (SysML) [25]; and
- Ontology of units of Measure and related concepts (OM) [26].

☆ This article is an abbreviated and revised extract from a National Institute of Standards and Technology (NIST) technical report with the same title [1]. Official contributions of NIST are not subject to copyright in the United States.

E-mail address: david.flater@nist.gov

[1] Certain commercial entities or software are identified in order to describe a concept adequately. Such identification is not intended to imply recommendation or endorsement by NIST, nor is it intended to imply that the entities or software are necessarily the best available for the purpose.

https://doi.org/10.1016/j.csi.2017.10.002
Received 9 August 2017; Received in revised form 27 September 2017; Accepted 10 October 2017
Available online xxx
0920-5489/Published by Elsevier B.V.

Although all of the above libraries, packages, models, and ontologies begin with the International System of Units (SI) [27] and associated practices, they differ in how they address issues such as *ad hoc* counting units, ratios of two quantities of the same kind, and uncertainty. These issues and others that have undermined the functionality and consistency of software for metrology have been discussed in related work [7,28].

This article describes an architecture that addresses the complete set of functions for Q&U software in a simple and consistent fashion. The goal is not to produce *yet another* implementation or framework to compete in an already overcrowded arena, but to identify the major architectural features that have proven to be important or sorely needed in this author's experience, and thereby encourage more convergent thinking and improved compatibility among different implementations in the future.

The building blocks of the architecture are:

- A catalog of recognized units and numerical prefixes;
- Values (in the measurement sense), including derived values and compound values;
- Probability distributions to characterize uncertain values, with automated propagation of distributions to derived values; and
- An extensible type system for specializations of the SI unit 1.

Section 2 through Section 5 address each of these in turn. Additional recommendations concerning numeric data types are given in Section 6. Section 7 summarizes.

2. Units

Quite simply, one cannot begin to implement quantity calculus without first having a catalog of units that the software will recognize and refer to. We need not belabor the point, but all Q&U software must at least recognize the standard base units, derived units, and numerical prefixes (k, M, etc.) that are identified in the SI brochure [27] and be capable of converting "non-coherent" units to their standard equivalents.

3. Values

The following terms are defined by the 3rd edition of the VIM [19].

quantity: property of a phenomenon, body, or substance, where the property has a magnitude that can be expressed as a number and a reference.

[The "reference" is typically an expression in terms of SI units.]

quantity value: number and reference together expressing magnitude of a **quantity**.
measurement result: set of **quantity values** being attributed to a **measurand** together with any other available relevant information.

[The "set of quantity values" is intended to accommodate uncertainty, given that a single true quantity value generally cannot be determined.]

A *value* in a software implementation of quantity calculus represents one or more measurement results with one or more **quantity value** variables. Those with only one such variable (the usual case) are *simple values*; the rest are *compound values*.

A *base value* represents immediate results of a measurement. A *derived value* is the result (output) of a function of other values (the inputs).

Figure 1 illustrates the concepts. On the left side, two base values, a speed expressed in meters per second and a duration expressed in seconds, are input to a function that multiplies them together, producing a derived value that is expressed in meters. On the right side, polar coordinates are given as an example of a compound value, because in most applications it is necessary to keep the radial coordinate and the angular coordinate together.

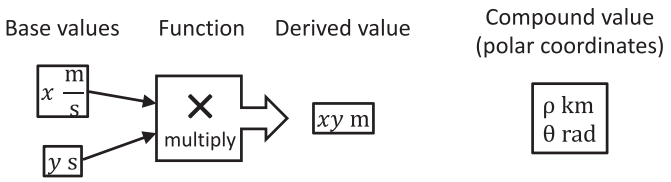

Fig. 1. *Values* in a software implementation of quantity calculus.

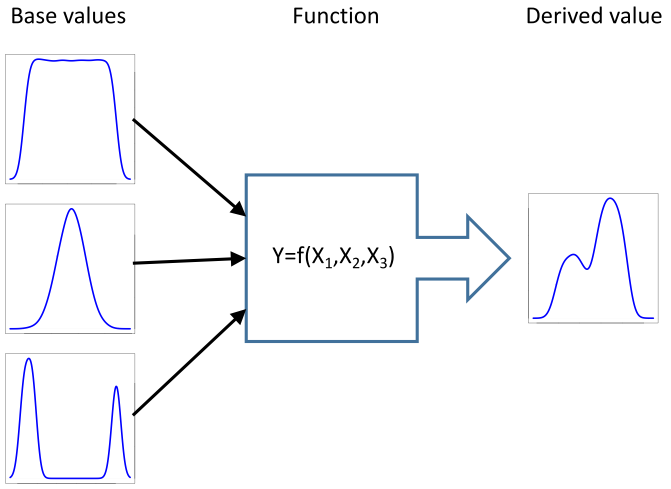

Fig. 2. Propagation of the distributions of base values through a function to the distribution of a derived value.

Like the catalog of SI units, simple values and value functions are essential to all software implementations of quantity calculus. Compound values, on the other hand, are not universally supported.

In the architecture described by this article, values are represented by distributions, as explained in the next section.

4. Characterizing and propagating uncertainty

Uncertainty too often is completely ignored or is handled in a limited way in available Q&U software.

Expressing uncertainty using standard deviations and multiples thereof involves simplifying assumptions that are unnecessary when quantity calculus is automated. The more general approach is to propagate probability distributions (see Figure 2) [29,30]. Probability distributions replace both a point estimate and its standard uncertainty. They may be continuous (for dimensional quantities and ratios) or discrete (for counted quantities). They may be univariate (for simple values) or multivariate (for compound values). They may be unimodal (for simple estimates) or multimodal (for when a single estimate would be misleading). And if a user wishes to have a quantity value with zero uncertainty, such as a trivial count or a defined constant, it can be assigned a degenerate (deterministic) distribution that has only one possible value.

Propagation of distributions can be implemented in two distinct ways:

1. A distribution is a black box software function that exposes only the interface to get sampled quantity values. A quantity value is characterized by a histogram or kernel density plot derived from a sufficiently large sample, and an interval is produced using statistical methods. When a derived value is sampled, its value is calculated using samples from other values as inputs to its function.
2. A distribution is a symbolic mathematical function that is either represented directly by the software or suitably approximated (e.g., with Chebfun [31]). Sampling is not required; instead, a quantity value is plotted directly, and intervals and derived values are derived by operating algebraically on the functions themselves.