Computer Standards & Interfaces 35 (2013) 529–535

Contents lists available at SciVerse ScienceDirect

Computer Standards & Interfaces

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

Quantities, units and computing

Marcus P. Foster

CSIRO Information Management and Technology, Private Bag 33 Clayton South, VIC 3169, Australia

ARTICLE INFO

Article history:
Received 9 August 2012
Received in revised form 26 November 2012
Accepted 8 February 2013
Available online 26 February 2013

Keywords:
Physical quantity
Units
Ontology
Notation
Precedence
Character set

ABSTRACT

Quantities and units are concepts central to our measurement and manipulation of the physical world, but their representation in information systems is barely codified and often ignored. The lack of formalization of metrological concepts, operations, symbols and characters has resulted in multiple reinvention (or more dangerously, omission) of these entities in informatics systems. At best, this creates ambiguity and inconvenience; at worst, the potential for an engineering disaster. The computer representation of quantities and SI units is reviewed at these four levels. Three implementations (languages, calculators and sensor data transfer) supporting units of measure are examined. Some suggestions for a hierarchy of metrological-informatics standards are given.

Contents

Abbreviations: BIPM, International Bureau for Weights and Measures; CIPM, International Committee on Weights and Measures; CGPM, General Conference on Weights and Measures; GUM, Guide to the expression of uncertainty in measurement; IEEE, Institute of Electrical and Electronics Engineers; ISO, International Organization for Standardization; ISQ, International System of Quantities; SI8, 8th edition of the 'SI Brochure'; VIM3, International vocabulary of measurement, 3rd edition.

E-mail address: marcus.foster@csiro.au.

0920-5489/$ – see front matter © 2013 Elsevier B.V. All rights reserved.
http://dx.doi.org/10.1016/j.csi.2013.02.001

1. Introduction

Quantities and units are concepts central to our measurement and manipulation of the physical world, but their representation in computational and information systems is barely codified and often ignored [1]. The lack of formalization of metrological concepts, operations, symbols and characters has resulted in multiple reinvention (or more dangerously, omission) of these entities in informatics systems. At best, this creates ambiguity and inconvenience; at worst, the potential for an engineering disaster. The most-cited example is the loss of the spacecraft Mars Climate Orbiter, which most commentators [2,3] attribute to a confusion between metric and US customary units. Even the NASA review summarized the root problem as "Failure to use metric units in the coding of a ground software file, 'Small Forces', used in trajectory models". While this is technically correct, with respect the root problem could be more usefully described as *failure to include a unit in a quantity expression*, since the thrust calculated for a trajectory correction in 'English units' was transmitted as a pure number to the spacecraft thruster software, which interpreted it as a metric unit.

This suggests that a more useful standard for calculating, storing or transmitting the value of a quantity is needed: include the magnitude, unit and kind-of-quantity ('impulse' in this case), so that the receiving computer can reject or convert it. Although this seems self-evident, it is not trivial. Thirty years ago, Finkelstein [4] described 'measurement' as "a special case of representation by symbols, with strong relation to description by language, computer data representation and the like", and optimistically predicted "there will be significant advances in the areas of database and intelligent knowledge-based systems and one can confidently expect that measurement theory will progress in strong relation with them". There has been scant codification of metrological 'computer data representation' since.

An overview of measurement standards and of possible reasons for the lack of metrological-informatics standards is given, before a suggested framework for analyzing and developing these is presented in Section 1.3.

1.1. Measurement standards

Humans have used local units of measurement for several thousand years. Trade, and increasingly science and technology, required rigorous, reproducible unit standards, and led to the creation of the metric unit platinum *etalons* kilogram and meter in 1799, the establishment of the BIPM in 1875 and the international system of units, the SI, in 1960. The current 8th edition of the SI [5] is referred to here as 'SI8'.

VIM3 [6] describes metrological entities and relationships in text and concept diagrams. It defines (1.13) a 'system of units' as "a set of base units and derived units, together with their multiples and submultiples, defined in accordance with given rules, for a given system of quantities". The SI, however, evolved from a system of practical metric units and its associated quantity system [7] was not formalized for another 20 years. The current version of this document, ISO 80000 [8] is sometimes called the international system of quantities, ISQ. None of these basic standards SI8, VIM or ISQ are presented in machine-readable form.

The ISQ uses the term 'quantity calculus' to describe the (trivial) algebra of quantities and units. Its first axiom (Maxwell's) is that the value of a quantity Q is the product of a numerical value $\{Q\}$ and a 'unit' $[Q]$ (i.e. a unit quantity): $Q = \{Q\} \cdot [Q]$.

Quantity calculus includes commutative and associative laws and scalar multiplication [9]. Different quantities can have the same dimension (e.g. torque and moment; entropy and heat capacity), and are said to be different 'kinds of quantity' (VIM3, 1.2). Quantity calculus has some limitations for practical measurement. It describes operations allowable on ratio scales, but does not apply to ordinal scales like Rockwell Hardness and to interval scales such as Celsius, Fahrenheit, etc.

1.2. Why so few metrological-informatics standards?

One reason for the neglect of units in informatics is because unit and quantity systems carry cultural, political and historical compromises, and are not as systematic, unambiguous or rigorous as computer science would expect. Some longstanding metrological discussions [10], not necessarily supported by the author, are:

- the utility of the concept 'dimension' is limited, e.g. the Hz and rad/s have the same dimension T^{-1} but are different quantities;
- if derived units are algebraically reduced, they become units of different quantities;
- the unit and dimension 'one' have multiple meanings in the SI;
- the SI base unit 'kilogram' contains a prefix, and should be renamed;
- the SI base unit (thermodynamic) mole is different to historical chemical usage and is redundant;
- angle is as tangible a geometric quantity as length and should be considered a base quantity;
- the SI should have separate units for temperature and temperature difference;
- the candela is a physiological unit defined for one wavelength, and should not be a base unit in the SI.

There are also different conceptual and philosophical understandings about measurement and measurement units:

- practical measurement has a social function, must be fit for purpose, and must concord with the theory of scales, conventions and uncertainty [11];
- the different definitions of measurement by standard bodies need to be resolved before VIM4 is drafted [12];
- the VIM focuses on ratio scales of measurement, and should be expanded to include non-numerical and ordinal measurement as described by Stevens' [13] nominal, ordinal and interval scales [14];
- the VIM concepts of 'quantity' and 'kind of quantity' overlap and require redefinition [15,16];
- the SI mole is not a 'true' measuring unit [17];
- the base quantity 'amount of substance' should be recast as 'numerosity' [18];
- the SI needs to distinguish true and 'parametric' quantities, units and dimensions [19];
- dimensionless quantities are misrepresented and should be called 'unitless' [19].

1.3. Quantities and informatics — a framework

Against this background, the representation of quantities, dimensions and SI units is reviewed at the character/keyboard, symbolic (plain and formatted text), operational and semantic levels. Fig. 1 is an informal representation of these informatics levels, and their relationship to the key metrological entities and operations. Using this framework, three implementations supporting units of measure (languages, calculators and data transfer) are examined. As the focus here is less on metrology and more on the parsing of character strings, the VIM3 concept 'value of a quantity' (e.g. '1.23×10^4 N·s')

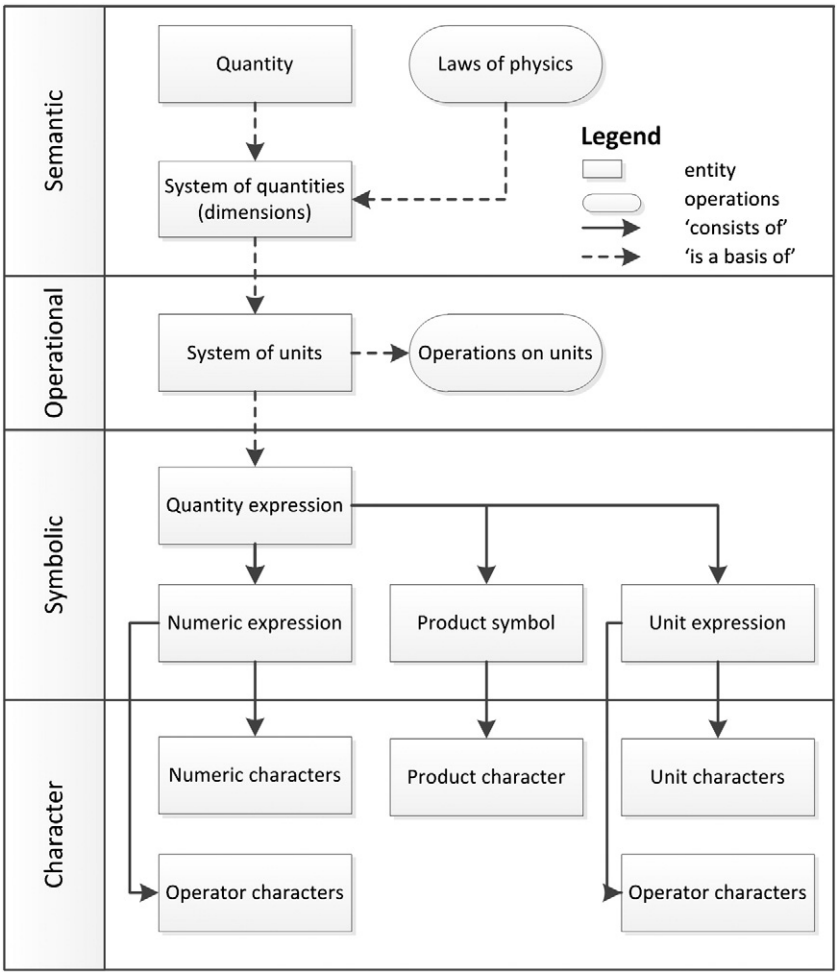

Fig. 1. Informatics levels for representing units of measurement.

is called here a 'quantity expression', and a string denoting a unit (e.g. s, ms, N, MJ/kg, $W \cdot m^{-2} \cdot K^{-1}$) is called a 'unit expression'.

2. Character representation

The SI specifies language-independent *symbols* for units, prefixes and operators, as distinct from a unit collection, such as the US customary units, which can have multiple *abbreviations* for a particular unit. The basic level of representation of SI units (and ISQ quantities) is the mapping of their symbols to a character set.

The SI Brochure and ISO 80000 print the symbols used for units, prefixes and operators but do not specify any character information. It is assumed that the characters are drawn from the Latin and Greek alphabets and from mathematical symbols for the units degree, arc minute and arc second and for the operators for precedence, multiplication, division and unary minus.

Lower and uppercase Latin characters are available in the ASCII, ISO-8859-1 (Latin 1) and Unicode character sets. Some candidate characters for the non-Latin letters are shown in Table 1. A similar table could be made for the hundreds of quantity symbols defined in ISO 80000.

The SI's unspecified symbols are perfectly suitable for visual communication, but not for digital communications. For example, should the SI symbol for unary minus be mapped to the keyboard 'hyphen' (-), the mathematical operator 'minus sign' (−), or the typographical characters em dash or en dash? Unicode provides characters 'OHM SIGN', 'DEGREES CELSIUS', 'MICRO SIGN', but states that these are

Table 1
Candidate characters for SI symbols and their hexadecimal codes.

SI symbol	7-bit ASCII	8-bit ISO-8859-1	Multi-byte Unicode
μ [micro]	✗	MICRO SIGN (00B5)	GREEK SMALL MU (03BC)
Ω [ohm]	✗	✗	GREEK CAPITAL OMEGA (03A9); OHM SIGN (2126)
°C [degree Celsius]	✗	DEGREE SIGN + 'C'	DEGREES CELSIUS (2103)
° [degree]	✗	DEGREE SIGN (00B0)	DEGREE SIGN (00B0)
′ [arc minute]	APOSTROPHE (0027)	APOSTROPHE (0027)	PRIME (2032)
″ [arc second]	QUOTATION MARK (0022)	QUOTATION MARK (0022)	DOUBLE PRIME (2033)
[unit product]	SPACE (0020)	SPACE (0020)	SPACE (0020)
· [unit product]	✗	MIDDLE DOT (00B7)	DOT OPERATOR (22C5)
/ [unit division]	SOLIDUS (002F)	SOLIDUS (002F)	DIVISION SLASH (2215)
× [number product]	✗	MULTIPLICATION SIGN (00D7)	MULTIPLICATION SIGN (00D7)
− [unary minus]	HYPHEN-MINUS (002D)	HYPHEN-MINUS (002D)	MINUS SIGN (2212)
() [precedence]	PARENTHESIS (0028,0029)	PARENTHESIS (0028,0029)	PARENTHESIS (0028,0029)

provided for backwards compatibility, and that the Greek and mathematical characters should be used in new documents.

ISO published a standard ISO 2955 [20] for coding SI units in plain text using the ASCII character set, which represented '°' as 'deg', '°C' as 'Cel', 'Ω' as 'Ohm' and 'μ' as 'u'. It contained several ambiguous symbols (a, cd, PEV), and was withdrawn in 2001 after the publication of the much larger Unicode character set that contains all the SI symbols. The absence of a current ISO or IEEE standard has led to developers choosing their own mappings. At least three proposals have been published. MIXF [21] specifies a 7-bit ASCII representation for SI quantity expressions. UCUM [22] and Frink [23] specify respectively, a 7-bit ASCII representation for unit expressions and a Unicode representation for quantity expressions, both with hundreds of unique symbols representing metric, customary and natural units, which have an associated dimension. This semantics of units allows implementations to determine unit conformance, and thus perform unit conversions and arithmetical operations.

3. Keyboard representation

The characters on a Western keyboard number fewer than 100, and similar to ASCII plain text, do not include the SI symbols dot operator, multiplication sign, degree, arc minute, arc second, mu and omega. This is inconvenient for authors, who must generate these characters via system software, key combinations or the use of special 'insert' functions in application software. Even in writing the simple quantity expression '1.234×10^5 N·m', there are no keyboard characters for two of the three multiplication symbols. It is recognized that the convenience of a unit affects its adoption [24]; similarly, authors often substitute these inconvenient symbols by a visual analog or an abbreviation, with consequent incompatibility in information systems.

4. Symbolic representation — formatted text

4.1. Ambiguity of SI symbols

SI8 contains two duplicate prefix/unit symbols 'm' (meter and milli) and 'T' (tesla and tera), with a further two 'h' (hour, hecto) and 'd' (day, deci) introduced by the "non-SI units accepted for use with the SI". SI8 has two syntactically ambiguous symbol combinations: 'cd' (candela, centiday) (the later breaks the semantic rule that prefixes cannot be attached to 'min' and 'd'), and the prefix 'da' (which breaks the semantic rule disallowing compound prefixes).

4.2. Ambiguity of unit expressions

Duplicate prefix/unit symbols are not necessarily ambiguous, depending on the rules for combining them in unit expressions [25]. Confusion about how to write a SI unit expression is widespread, because the SI Brochure and ISO 1000 have changed the specified product symbol (period, space, middle-dot, implicit) repeatedly over the last fifty years [26]. Consequently, without a stable agreed standard, it is difficult to guarantee correct parsing.

Even now, the two standards have different specifications:

- SI8 states "In forming products and quotients of unit symbols the normal rules of algebraic multiplication or division apply. Multiplication must be indicated by a space or half-high (centered) dot (·), N m or N·m, since otherwise some prefixes could be misinterpreted as a unit symbol"
- ISO 80000-1 withdrew the period as a product symbol 18 years after the SI Brochure and states "A compound unit formed by multiplication of two or more units shall be indicated in one of the following ways: N·m, N m. Note: The latter form may also be written without a space, i.e. Nm, provided that special care is taken when the symbol for one of the units is the same as the symbol for a prefix. This is the case for m, meter and milli, and for T, tesla and tera".

There are several problems with these critical rules:

- In both rules, the printed form of the product symbol shows 'space-centered dot-space', whereas the BIPM description specifies 'centered dot'
- In the ISO rule, it is not obvious that "provided that special care is taken" means 'except', and that not "one of", but only the leading unit symbol needs to be unambiguous. Further, this rule requires the reader to know all the SI unit and prefix symbols to determine whether there is overlap.
- The ISO rule does not address the duplicate prefix-unit symbols introduced by the 'non-SI units accepted for use with the SI'. Does ISO 80000 allow the ambiguous derived units 'dlm' (day-lumen or decilumen) and 'hW' (hour-watt or hectowatt)?

4.3. Quantity expressions

The SI8 specification (5.3) for a quantity expression is: "The value of a quantity is expressed as the product of a number and a unit, and a space is always used to separate the unit from the number, the space being regarded as a multiplication sign. Exceptionally, the unit symbols for degree, minute, and second of plane angle, °, ′, and ″, respectively, have no space left between the numerical value and the unit symbol". ISO 80000 contains the same rule (7.1.4).

Thus the space character in a quantity expression (e.g. '$6.673\ 84 \times 10^{-11}$ $m^3\ kg^{-1}\ s^{-2}$') has multiple meanings, representing the digit padding, scientific-notation padding, number–unit product symbol and unit–unit product symbol. This presents further challenges for parsing.

Despite Unicode containing all SI unit and operator symbols, it is insufficient for specifying formatted unit expressions, since these contain typographical conventions such as superscripts, line-shifting and underlining, to render unit expressions such as '$m \cdot s^{-1}$' or '$\frac{m}{s}$'.

4.4. Quantity names and symbols

In formatted text, the name or symbol of the quantity is usually explicitly stated near the unit of measurement. Quantity symbols are printed in italic type, and often have subscripts.

5. Symbolic representation — plain text

5.1. Unit expressions

The ISO 2955 specification for a unit expression used period (.) as the unit–unit product symbol, the solidus (/) as the division symbol, exponentiation as a trailing number, and parenthesis as precedence, e.g. '$kN.m^{-2}$' or 'kN/m^2' for the typeset '$kN \cdot m^{-2}$' or 'kN/m^2'. It disallowed the double solidus (e.g. m/s/s).

UCUM is similar, but allows the double solidus, which is parsed left to right.

MIXF is similar, but uses the circumflex '^' for the exponentiation operator, e.g. 'kN/m^2'. It disallows the double solidus.

Frink uses the space (the same as a formatted expression) for the product symbol, the solidus (/) as the division symbol, and the circumflex '^' for exponentiation, e.g. 'N m', 'W/(m^2 sr)'. It allows the double solidus. Frink supports Unicode, and has a configurable symbol table, thus supporting the SI symbols '·', '°', 'Ω' and 'μ', if required.

5.2. Quantity expressions

ISO 2955 and UCUM do not address the representation of quantity expressions in plain text. An obvious scheme is to combine the

numerical value and unit separated by a product symbol, as is used for formatted quantity expressions. MIXF specifies an ISO 6093 [27] numeric representation followed by a period and a unit expression, e.g. 1.234.W/(m^2.sr). Frink uses the space (the same as a formatted expression) as the value-unit product, e.g. 1.234 W/(m^2 sr).

6. Operational representation

6.1. Units and dimensions

In unit-aware computational software such as calculators and algebra systems, a set of base quantities (i.e. dimensions) is chosen and units are assigned a vector of dimensional exponents. Note that the base quantities and base units chosen for the SI, do not necessarily suit informatics needs. Often, software systems use gram (g) as the base unit of mass to simplify the handling of prefixes, charge rather than current as the electrical base quantity and support additional base units of angle and temperature difference for mechanics and thermodynamics computations respectively.

6.2. Quantity expressions

In the computational software listed above, the internal representation of a quantity expression is tripartite:

$$Q = \{Q\} \cdot [Q] \cdot (\dim\ Q), \text{ where } \dim\ Q \text{ is the dimension of } Q.$$

This enables four basic unit operations:

1. Multiplication of units of any dimension
2. Addition of units of the same dimension
3. Conformance of dimensional equality of unit expressions
4. Conversion between units of the same dimension

Of course, such software still allows the addition of a torque and energy variable, since they have the same dimension. There are two views about how this should be managed. The first [28], is that quantity expressions should have a quad-partite representation:

$$Q = \{Q\} \cdot [Q] \cdot (\dim\ Q)(\text{kind } Q),$$

$$\text{e.g. } M_1 = (1.234\text{E}5) \cdot (\text{N}\cdot\text{m}) \cdot \left(\text{ML}^2\text{T}^{-2}\right) \cdot (\text{torque}).$$

This would additionally enable four basic quantity operations:

1. Multiplication of quantities of any kind
2. Addition of quantities of the same kind
3. Conformance of 'kind' equality of quantity expressions
4. Conversion between quantities of the same kind.

(Note that this scheme could likely be expanded to differentiate scalar and vector quantities, quantities of specific bodies or substances and so on).

The alternative view is that quantities, kinds-of-quantities, dimensions and units are concepts which require a human mind, rather than computational mechanisms, to interpret and detect errors (see for example, 10-years of discussion threads in the Mathcad user forum [29]).

6.3. Precedence

In unit systems, a prefix is bound to a unit symbol with higher precedence than exponentiation, multiplication or division.

The SI prescribes incorrect operator precedence. SI8 (5.1) states "In forming products and quotients of unit symbols the normal rules of algebraic multiplication or division apply", but goes on to say that "A solidus must not be used more than once in a given expression without brackets to remove ambiguities". This later sentence contradicts the normal rules of algebra, in which operators of equal precedence are evaluated left-to-right, and thus deprecates the unambiguous and previously common unit expression for acceleration 'm/s/s'.

Information systems also need to deal with unit expressions which contain numerical factors instead of prefixes. A common example is the unit 'L/100 km' for automotive fuel consumption, in widespread legal use in Europe and Australia. Although humans easily understand this unit (by unconsciously binding the factor to the unit, as if it were a prefix), it does not meet the SI definition or style for a derived unit, and at least should be written as 'L/(100 km)' for mathematical correctness, as per NIST [30]. Calculators that use standard mathematical precedence to parse expressions containing these factored units, yield meaningless results. This precedence issue could be solved by declaring that a numerical factor is bound to the unit with precedence below a prefix. Thus the precedence table for parsing both unit and factor–unit expressions should be:

1. Prefix-unit binding
2. Exponentiation
3. Factor–unit binding
4. Parentheses
5. Multiplication and division

7. Semantic representation

Measurement ontologies are intended to represent the relationships and operations between quantities, kinds of quantities, dimensions and units. These ontologies formalize the entities and relationships shown in (parts of) the concept diagrams of VIM3, and are a foundation service for systems which measure, store, transfer, process or display data about the physical world. For example, Thirunarayan and Pschorr [31] list ten applications required to support a semantic sensor network, with observation and measurement (O&M) being a basic catalog web service.

Measurement ontology projects often seem to founder on the difficulties of representing vague or contradictory metrological concepts. Additional clarity might be realized if measurement ontologies were classified by the types of metrological entities that they represent. A suggested classification is:

Level 1 or 'character' ontologies are simple character-sets such as ISO 2955, which standardize characters for unit symbols and operators.

Level 2 or 'units of measure' (UOM) ontologies additionally represent the concepts of unit and dimension, and can thus support the operations of unit conversion and unit conformance. They may also include names for units. Some established 'units of measure' ontologies include EngMath [32] and SciUnits [33], and by this definition, self-described code-sets like UCUM.

Level 3, or 'quantity and units of measure' ontologies additionally include the concept of quantity, and system of quantities. Examples are QUOMOS [34] and QUDT [35].

Level 4 or 'measurement' ontologies are supersets of Level 3 ontologies. A recent, and possibly the sole, example 'OM' [36] proposes that most existing ontologies lack full descriptions of quantities, prefixes, ordinal and nominal measurement scales and systems of quantities and units. It formalizes these often-ignored concepts. The purpose of this ontology is to provide a foundation for building practical unit-aware software tools, and a web-tool, an engineering-application and a spreadsheet annotator are demonstrated.

Level 5, or 'metrological' ontologies are proposed as a superset of a measurement ontology, and would also incorporate the concepts (precision, accuracy, distribution etc.), relations and operations

associated with calculating and representing measurement uncertainty, as described in the GUM [37].

8. Practice case 1: programming languages

Representing quantities and units is an old problem in language design, and has cyclical popularity. Many early languages have no dimension/unit support, and a typical solution to provide this is to add a pre-processor [38–40].

More recently, Van Delft [41] developed a Java extension to represent dimensions (with units as constants). Allen et al. [42] proposed a method for representing quantities and units in object-oriented languages. They later [43] developed a specification for the language Fortress for scientific modeling, including support for dimensions and units. Unit support was never implemented, perhaps an indication of its underestimated complexity, or of its impact on compile and run-time performance. Other proposals include static- and inference-checking of units for F# [44] and C [45], and the language Frink [46], developed specifically for representing calculations of physical phenomena.

Much engineering and scientific computation is performed in spreadsheets, which support cell formulae and functions, as well as inbuilt programming languages, but which have no representation of units, dimensions and quantities. Two approaches to minimizing spreadsheet unit errors are to include a typing system as an add-on [47] or to use a reasoning system to infer and check dimensional information in each cell [48].

Similarly, there is a need to inspect program source code for dimensional correctness, especially for programs that control or measure the physical world. A solution for inspecting embedded control software has been proposed by Cook and Fidge [49].

9. Practice case 2: computational applications

Different computational applications have varying degrees of recognition of SI unit and prefix symbols, depending on the intended usage of the system. A handful of computational systems are examined below, the web calculators Wolfram|Alpha (WA), Google Calculator and Frink and algebra systems Mathcad and Mathematica 8.

9.1. Support for SI unit and prefix symbols

There is a finite number (580) of combinations of SI prefixes (20) with base units (7) and derived units with special names (22). Two ('L' and 't') of the seven 'non-SI units accepted for use with the SI' take a limited range of prefixes.

- The calculators use plain text input/output; WA and Frink recognize Unicode Greek letters μ and Ω. All calculators correctly assign a higher precedence to prefixes over exponentiation.
- Frink recognizes all combinations of SI symbols, based on recursive parsing to the basic prefix and unit atoms. It also recognizes SI prefix and unit names.
- WA did not recognize all combinations of SI prefix and unit symbols until about two years after release, which seems to indicate that the combinations are individually coded, rather than being represented as atoms. WA's parsing flexibility allows it to recognize some common but incorrect abbreviations for SI units such as 'Kg' and 'gm'. Google Calculator recognizes a few common combinations of SI symbols.
- Recent versions of Mathcad recognize all SI unit symbols, but not SI prefixes.
- Mathematica 8 does not recognize SI unit and prefix symbols, but can represent SI units with its own symbols (e.g. 'Kilo ∗ Gram' for 'kg').

9.2. Support for unit expressions

There are an unlimited number of SI unit expressions combining prefixes, units and operators.

- Frink recognizes the space as product operator, solidus as division operator, parenthesis as precedence operator, and circumflex as exponentiation operator, as well as other Unicode operators.
- Google Calculator recognizes the space, solidus, parenthesis and circumflex on its limited number of units.
- WA recognizes the middle dot and asterisk (*) as product operators.
- Mathcad and Mathematica allow the product symbol to be set as a preference; Mathcad automatically provides typesetting for unit expressions.

10. Practice case 3: sensor data transfer

Although array-based scientific data is usually stored in binary form (e.g. NetCDF), data transmission from sensors is increasingly organized in XML-based (plain) text. The lack of ISO or IEEE standards for representing units, quantity expressions and kinds-of-quantity is a problem for developers of these systems.

Data transfer formats for specific domains may require only a limited set of units, and thus may specify their own unit coding scheme. An example is the Water Data Transfer Format [50] (WDTF), where "uS/cm" represents microsiemens per centimeter.

Similarly, WDTF specifies the value of a quantity as:

```
<wdtf : Measurement >
    <wdtf : result uom = ″uS/cm″ > 90.00</wdtf : result >
</wdtf : Measurement > .
```

For complete disambiguation of a sensor measurement, the kind of quantity should be specified in data communications, so that quantities with the same units can be reliably differentiated. This clearly requires a common semantics of quantities, which may be standards-based (e.g. ISO 80000) or internally-specified. WDTF declares the quantity, unit of measure and numerical value thus:

```
<wdtf : Measurement >
    <om : observedProperty title = ″Electrical Conductivity@25C″/ >
    <wdtf : result uom = ″uS/cm″ > 90.00</wdtf : result >
</wdtf : Measurement >
```

11. Conclusions

The intersection of metrology and informatics is a domain neglected by international standards bodies. The lack of formalization of metrological concepts, operations, symbols and characters has resulted in confusion and reinvention of similar entities in informatics systems. In general, it is suggested that a hierarchy of standards is needed to codify the four levels of metrological entity.

1. Representation at the semantic level, based on VIM3.
2. Representation at the operational level — ISQ quantities, SI units and operations on quantities and units.
3. Representation at the symbolic level — rules for forming unit and quantity expressions and explicit metrological precedence rules for parsing unit expressions and factor-unit expressions.
4. Character mapping for ISQ quantities and operators and SI units, including character mapping for plain-text representation of quantity expressions.

In our digitally-connected world, a system of units like the SI has a decreasing utility in its paper form. In the interim, the steps needed

to improve the representation of SI notation in informatics systems are:

1. The SI symbols and the specification for unit expressions should be harmonized between the SI Brochure and ISO 80000.
2. The SI symbols should be chosen so as to generate unambiguous quantity and unit expressions.
3. The SI symbols should be specified as Unicode characters, or better still for simplicity changed to match 7-bit ASCII characters for keyboard convenience and for supporting legacy systems.
4. A standard should be developed for plain-text SI quantity and unit expressions that is visually close to the formatted expressions.

The organizations best placed to manage these standards are ISO and IEEE, both international organizations which already publish standards in areas such as nomenclature, symbols, quantities and character sets.

References

[1] B.D. Hall, Software support for physical quantities, 9th Electronics New Zealand Conference, Dunedin, 2002.
[2] J. Oberg, Why the Mars probe went off course, IEEE Spectrum 36 (1999) 34.
[3] A.G. Stephenson, Mars Climate Orbiter Mishap Investigation Board, ftp://ftp.hq.nasa.gov/pub/pao/reports/1999/MCO_report.pdf 1999, (accessed: 12 Mar 2012).
[4] L. Finkelstein, M.S. Leaning, A review of the fundamental concepts of measurement, Measurement 2 (1984) 25–34.
[5] BIPM, The International System of Units, 8th ed. BIPM, Sèvres, 2006.
[6] JCGM, International Vocabulary of Metrology — Basic and General Concepts and Associated Terms (VIM), Joint Committee for Guides in Metrology, Geneva, 2008.
[7] ISO, ISO 31-0, General principles concerning quantities, units and symbols, ISO, Geneva, 1981.
[8] ISO, ISO 80000-1, Quantities and Units — Part 1: General, ISO, Geneva, 2009.
[9] J. de Boer, On the history of quantity calculus and the International System, Metrologia 31 (1995) 405–429.
[10] M.P. Foster, The next 50 years of the SI: a review of the opportunities for the e-Science age, Metrologia 47 (2010) R41–R51.
[11] G. Price, On the communication of measurement results, Measurement 29 (2001) 293–305.
[12] R. Dybkaer, Definitions of 'measurement', Accreditation and Quality Assurance 16 (2011) 479–482.
[13] S.S. Stevens, On the theory of scales of measurement, Science 103 (1946) 677–680.
[14] R. White, The meaning of measurement in metrology, Accreditation and Quality Assurance 16 (2011) 31–41.
[15] L. Mari, On (kinds of) quantities, Metrologia 46 (2009) L11.
[16] R. Dybkaer, Generic division of 'quantity' and related terms, Accreditation and Quality Assurance 16 (2011) 649–651.
[17] I. Johansson, The mole is not an ordinary measurement unit, Accreditation and Quality Assurance 16 (2011) 467–470.
[18] R.C. Rocha-Filho, Reproposition of numerosity as the SI base quantity whose unit is the mole, Accreditation and Quality Assurance 16 (2011) 155–159.
[19] I. Johansson, Metrological thinking needs the notions of parametric quantities, units, and dimensions, Metrologia 47 (2010) 219–230.
[20] ISO, ISO 2955, Information Processing — Representation of SI and Other Units in Systems with Limited Character Sets, ISO, Geneva, 1983.
[21] A. Jaffer, MIXF: Representation of numerical values and SI units in character strings for information interchanges, http://people.csail.mit.edu/jaffer/MIXF/MIXF-1 2011.
[22] G. Schadow, C.J. McDonald, The Unified Code for Units of Measure, http://aurora.regenstrief.org/~ucum/ucum.html 2010, (accessed: 3 Mar 2012).
[23] A. Eliasen, Frink, http://futureboy.us/frinkdocs/ 2012, (accessed: 12 Mar 2012).
[24] J. Valdés, The unit one, the neper, the bel and the future of the SI, Metrologia 39 (2002) 543–549.
[25] M.P. Foster, Disambiguating the SI notation would guarantee its correct parsing, Proceedings of the Royal Society A 465 (2009) 1227–1229.
[26] M.P. Foster, Principles for constructing notation in unit systems and their application to the SI, Accreditation and Quality Assurance 17 (2012) 85–92.
[27] ISO, ISO 6093, Information Processing — Representation of Numerical Values in Character Strings for Information Interchange, ISO, Geneva, 1985.
[28] V.F. Ochkov, V.I. Lehenkyi, E.A. Minaeva, Physical quantities, dimensions and units in mathematical packages, Mathematical machines and systems 1 (2009) 78–90.
[29] PTC Corporation, Mathcad Community, http://communities.ptc.com/community/mathcad 2012, (accessed: 12 Mar 2012).
[30] A. Thompson, B.N. Taylor, The NIST Guide for the use of the International System of Units (SI). Appendix B.5, NIST, Gaithersburg, 2008.
[31] K. Thirunarayan, J. Pschorr, Semantic information and sensor networks, Proceedings of the 2009 ACM symposium on Applied Computing, Honolulu, Hawaii, 2009.
[32] T.R. Gruber, G.R. Olsen, An ontology for engineering mathematics, Fourth International Conference on Principles of Knowledge Representation and Reasoning, Bonn, 1994.
[33] NASA, SWEET SciUnits Scientific Units Ontology, http://sweet.jpl.nasa.gov/2.2/reprSciUnits.owl2011, (accessed: 15 Mar 2012).
[34] S. Ray, F. Olken, S.J.M. Mcrae, OASIS Quantities and Units of Measure Ontology Standard (QUOMOS), http://www.oasis-open.org/committees/tc_home.php?wg_abbrev=quomos 2010, (accessed: 7 Mar 2012).
[35] R. Hodgson, P.J. Kelle, QUDT — Quantities, Units, Dimensions and Data Types in OWL and XML, http://qudt.org/2011, (accessed: 03 Mar 2012).
[36] H. Rijgersberg, M. Wigham, J.L. Top, How semantics can improve engineering processes: a case of units of measure and quantities, Advanced Engineering Informatics 25 (2011) 276–287.
[37] JCGM, Evaluation of Measurement Data — Guide to the Expression of Uncertainty in Measurement, BIPM, Sèvres, 2008.
[38] M. Karr, D.B. Loveman, Incorporation of units into programming languages, Communications of the ACM 21 (1978) 385–391.
[39] N. Gehani, Units of measure as a data attribute, Computer Languages 2 (1977) 93–111.
[40] A. Dreiheller, M. Moerschbacher, B. Mohr, PHYSCAL — Programming Pascal with Physical Units, ACM SIGPLAN Notices 21 (1986) 114–123.
[41] A. van Delft, A Java extension with support for dimensions, Software-Practice & Experience 29 (1999) 605–616.
[42] E. Allen, D. Chase, V. Luchangco, J.W. Maessen, G.L. Steele, Object-oriented units of measurement, ACM SIGPLAN Notices 39 (2004) 384–403.
[43] E. Allen, D. Chase, J. Hallett, V. Luchangco, J.-W. Maessen, S. Ryu, G.L. Steele Jr., S. Tobin-Hochstad, The Fortress language specification version 1.0, http://labs.oracle.com/projects/plrg/fortress.pdf 2008, (accessed: 15 Mar 2012).
[44] A. Kennedy, Types for units-of-measure in F#, Proceedings of the 2008 ACM SIGPLAN workshop on ML, Victoria, BC, Canada, 2008.
[45] L. Jiang, Z. Su, Osprey: a practical type system for validating dimensional unit correctness of C programs, Proceedings of the 28th international conference on Software engineering, Shanghai, China, 2006.
[46] A. Eliason, Frink — a language for understanding the physical world, Lightweight Languages 2004 (LL4) Conference, Massachusetts, 2004.
[47] Y. Ahmad, T. Antoniu, S. Goldwater, S. Krishnamurthi, A type system for statically detecting spreadsheet errors, 18th IEEE International Conference on Automated Software Engineering, Montreal, 2003.
[48] C. Chambers, M. Erwig, Reasoning about spreadsheets with labels and dimensions, Journal of Visual Languages and Computing 21 (2010) 249–262.
[49] P. Cook, C. Fidge, Well-measuring programs, Australian Software Engineering Conference (ASWEC'06), Sydney, 2006.
[50] BoM, About Water Data Transfer Format (WDTF), http://www.bom.gov.au/water/regulations/wdtf/index.shtml 2011, (accessed: 03 Mar 2012).