

Chapter 1

REPRESENTATION AND CONVERSION OF DIMENSION UNITS IN CAD DATA MODELS

D. Roller, O. Eck, B. Rieg, and D. Schäfer

Institut für Informatik, Universität Stuttgart, Breitwiesenstr. 20-22, 70565 Stuttgart, Germany

Abstract This paper presents a knowledge-based approach to represent the dimension units of numerical values in CAD data models. Although numerical quantities are associated with dimension units in typical engineering data, these units are missing in most product data models. Some extended data models allow to store information about dimension units, but this information is limited to a fixed and predefined number of units. The presented approach helps to model parts of the semantics of product data by extending numerical values to semantic entities that can be converted automatically. Further, the evaluation of simple units (e.g. mm und cm) or more complex units (e.g. $N=kg \cdot m/s^2$) in arithmetical operations (e.g. addition, multiplication or division) is described. The approach works on any set of units with any complexity. Users are able to define new units by so-called reduce and substitute rules. In order to evaluate dimension units in arithmetical operations, all units are converted first into a standard form, using the defined reduce and substitute rules. Next, the dimension unit of the complete operation is calculated by reducing fractions, combining units, calculating numerical values, and converting the result of the complete operation into the desired unit. This paper also describes an implementation of the presented approach that integrates the concept of representing and evaluating dimension units in a shared CAD database system.

Keywords: Representation of Dimension Units, Knowledge-Based Management Systems, Shared CAD Databases

1. INTRODUCTION

Most rule-based concepts in the area of “intelligent CAD” are not used in product design in real life because most rules are either unknown at design time or too complex for a complete and formal specification. This paper presents a knowledge-based approach which leads to an efficient support of product design. It extends numerical values in CAD data models to semantic entities that include numerical values and their dimension units. Dimension units are part of the semantics of product data and their representation provides the following advantages in collaborative product design:

- Increased reusability of product data
- Decreased rate of design errors
- Increased design efficiency

Mixing up the dimension units of numerical values is an error that often occurs in product design and which can be avoided by applying the presented approach. Especially in distributed design processes, where designers of different countries are using their national dimension units, an automatic conversion of units allows designers to concentrate on design work instead of dealing with data exchange problems. Therefore, this approach provides significant support to avoid design errors. The representation of dimension units additionally increases the reusability of product data and supports sharing of knowledge. The representation of the semantics of numerical values is a first step to provide an intelligent design support by representing knowledge instead of plain engineering data.

2. RELATED WORK

Compared with related approaches, the concept presented here shows a complete solution for representing dimension units that can be implemented even in traditional database systems. This section provides a short survey of related work in this field.

In (Diederich 1991), domain-based metadata is discussed that captures and characterizes the semantics of attributes in knowledge bases. One type of metadata is the unit of atomic values. Since the system does not know how to interpret and convert this

metadata, the semantics of units is comprehensible only for the users but not for the representation system itself.

In several approaches, knowledge sharing across disciplines is possible due to a priori ontological agreements about the meaning of terms (Gruber 1992), (Olsen 1994). That means that even the units of measures are subject to sharing agreements. In (Gruber 1994), an ontology for mathematical modeling in engineering is described ("EngMath"). This ontology also includes conceptual foundations for physical dimensions and units of measure and is designed explicitly for knowledge sharing purposes. The ontology is used as a communication language among cooperating engineering agents as well as a foundation for other engineering ontologies.

The presented approach is not based on ontological agreements. Each user has its own view to the database and is able to use his own measures, as long as he explicitly specifies the definitions of his dimension units. The conversion between dimension units is performed automatically by the system, so the explicit representation of a dimension unit is transparent to the user. He has not to care about the units used by his colleagues, and there has to be no agreement of shared units of measures.

Another type of systems that are used to represent dimension units are expert systems (Gero 1987). These systems have extensive representation structures and inference capabilities. Regarding the domain of product modeling in CAD, these systems hardly have not been accepted in practice for product modeling. Product data is usually stored in product database systems and therefore, extended representation and reason methods are needed that can be integrated in these database systems and provide efficient runtime performances.

3. REPRESENTATION OF DIMENSION UNITS

This section presents an approach to represent dimension units in CAD models. In order to be able to represent dimensions, information about units has to be stored together with the corresponding numerical values. An extended representation structure is used to store both. A numerical value together with its dimension unit is called a term. Additional to information about units, the data model has to represent rules which define the relations among dimension units. When this information is modelled, an inference system is able to evaluate it and convert terms automatically according to the defined rules (see figure 1). The presented inference system consists of two different components: the conversion system which converts dimension units into equivalent units and an evaluation system which evaluates arithmetical operations on terms. Since the evaluation of operations also requires to convert units, the evaluation system uses the conversion system internally.

3.1 Representation of dimension units

Representing dimension units requires syntactical structures that define a unique representation of units. A unit of measures may consist of a single unit (e.g. kg), or several units combined with arithmetical operations (e.g. m/s^2). In the presented representation structure, units of measure are specified in square brackets, e.g. [m] or [m/sec²]. The following specification in the Backus-Naur Form (BNF) shows the formal definition of dimension units.

$$\begin{array}{l} \text{UnitSpec} \\ \rightarrow \text{'[' UnitExpr ']} \end{array} \quad (1)$$

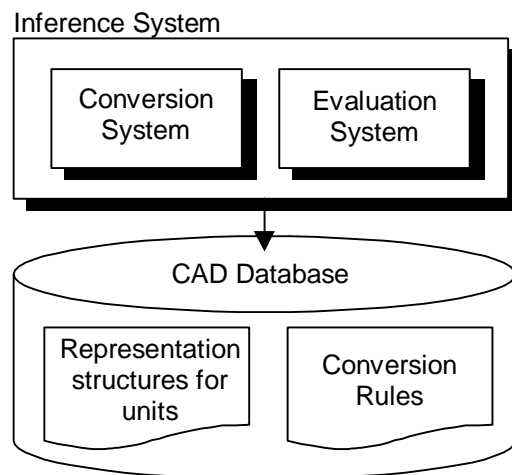


Figure 1. Architecture of a system for representing dimension units

- UnitExpr
 - UnitExpr '*' Factor
 - UnitExpr '/' Factor
 - Factor
- Factor
 - Factor '^' Number
 - Identifier
 - Number
 - '(' UnitExpr ')'

Figure 2 clarifies the syntax of dimension units by some examples. The left column shows correct terms. In the first incorrect example, the brackets are missing, in the second example, the operation is missing since Nm is not regarded as an independent unit.

Correct:	Incorrect
3 [cm]	43 cm
4.5 [N*m]	4.5 [Nm]
22 [kg*m/sec^2]	22 [kg*m/sec2]
22 [kg*m*sec^-2]	

Figure 2. Examples of correct and incorrect dimension units

3.2 Conversion of dimension units

There are two types of conversion rules: *Reduce rules* and *substitute rules*. Reduce rules define how dimension units can be converted that represent the same dimension. An example is the dimension length, which can be measured using the units mm, cm, km or inch. All units of a given dimension build up a so-called *dimension class*.

Reduce and substitute rules build up a *dimension script*. A dimension script consists of all defined rules and therefore represents the instructions for the conversion system. A dimension script has to be parsed in order to be able to evaluate it. The syntax of a dimension script is defined in (2):

UnitScript (2)
 → ReduceRules SubstituteRules

The syntax of reduce rules is given in the following definition (3):

ReduceRules (3)
 → 'Reduce' ReduceList
 ReduceList
 → ReduceList Identifier ':' {' UnitList '
 → epsilon
 UnitList
 → UnitList ',' SimpleUnit
 → SimpleUnit
 SimpleUnit
 → Number '*' Identifier
 → Identifier

The following example shows reduce rules for the dimension classes length and weight. In the example, unit m is called a *basic unit*, because all other units of dimension length are defined related to it.

Reduce (4)
 m: {1000*mm, 100*cm, 0.001*km, 3.937*inch}
 g: {0.001*kg}

Substitute rules define relations between units of different dimensions. The definition of substitute rules is given by definition (5):

```

SubstituteRules
  → 'Substitute' SubstituteList          (5)
  → epsilon
SubstituteList
  → SubstituteList Identifier ':' '{ UnitList2 }'
  → epsilon
UnitList2
  → UnitList2 ',' UnitExpr
  → UnitExpr
UnitExpr
  → see (1)

```

The following example (6) shows substitute rules that define the dimension units N and J. According to these definitions, unit N can be transformed into the units kg, m, and sec: $1 N = kg \cdot m / sec^2$.

```

Substitute
  N: { kg*m/sec^2, W*sec/m }          (6)
  J: { kp*m, N*m, W*sec }

```

4. THE INFERENCE SYSTEM

This section presents the concept of an inference system that converts dimension units and evaluates arithmetic operations. A precondition of all inferences on dimension units is their unique representation which is defined in section 3. Evaluating reduce and substitute rules, an inference system is able to evaluate them for the following types of inferences:

- Conversion of dimension units
- Evaluation of arithmetical operations
- Conversion into a desired unit

The first inference converts different dimension units. An example for this inference is:

$$614\text{cm}^2 + 614\text{mm}^2 = 61400\text{mm}^2 + 614\text{mm}^2 = 62014\text{mm}^2 \quad (7)$$

The evaluation of arithmetical operations defines addition, subtraction, multiplication and division among terms. It has to be noticed that operations (e.g. addition) are not defined for all combinations of dimension units:

$$100\text{cm}^2 + 50 \text{ kg} = \textit{undefined} \quad (8)$$

A conversion into a desired unit converts a term into a dimension the user demands for. A conversion is also not defined for all units, e.g. 10 cm^2 can be converted to mm^2 but not to kg . Therefore, it has to be checked whether a conversion is possible or not.

4.1 Conversion of dimension units

Only a *standard form* allows an inference system to compare different terms and to find out that the following equation is correct: $1 * \text{cm} * \text{m} = 1 * \text{m} * \text{cm}$. After transforming all involved terms into the standard form, the conversion and evaluation of operations can be executed.

In the standard form, all units are basic units. In a first step, all units are transformed into their basic units by applying reduce and substitute rules. The following example shows how a unit is transformed into the standard form using rules (4) and (6). The exponents of each dimension unit are written explicitly and all divisions are transformed into a multiplication with negative exponent. Next, all units are sorted in lexicographical order, and multiple units of the same kind are combined by adding their exponents.

$$\begin{aligned} & 1 \text{ N} \\ \Rightarrow & 1 \text{ kg}^2 * \text{m} / (\text{sec}^2) \\ \Rightarrow & 1000 * \text{g}^2 * \text{m}^1 * \text{sec}^{-2} \\ \Rightarrow & 1000 * [(\text{m},1), (\text{g},2), (\text{sec},-2)] \end{aligned} \quad (9)$$

Next, the *length* of a term is introduced. This attribute of terms is needed to compare the complexity of different terms when a term is being converted to a more simple form. The length of term t , written as $|t|$, is defined as:

$$\begin{aligned}
 |t| &= z * |(m,i)| = i, \text{ if } t = [(m,i)], i \in \mathbb{N}, z \in \mathbb{R} & (10) \\
 |t| &= \sum_i |i_i|, \text{ if } t = z * [(m_1,i_1), (m_2,i_2), \dots, (m_n,i_n)], n \in \mathbb{N}, z \in \mathbb{R}
 \end{aligned}$$

Evaluating substitute rules, it is possible that two or more substitute rules are available for a substitution. Figure 3 shows a graphical representation of substitute rules. In this example, there are two alternative substitutions for dimension unit N: $N = \text{kg} * \text{m} / \text{sec}^2$ and $N = W * \text{sec} / \text{m}$. In order to find the "right" substitution, a conversion to a desired unit has to be repeated with the other alternative if the conversion fails with the first substitution. A heuristics to find the right alternative at first is to choose this substitute rule that substitutes a dimension unit to the most simplest combination, which means that the length of the unit is the smallest.

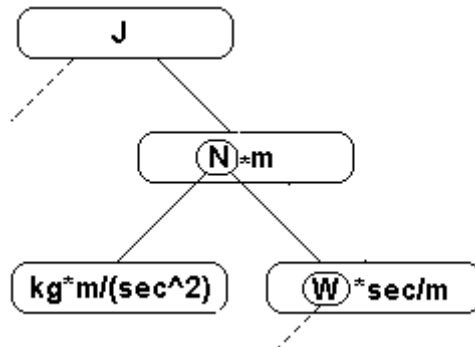


Figure 3. A graphical representation of substitute rules

4.2 Evaluation of arithmetical operations

The calculation of arithmetical operations (e.g. addition, subtraction, multiplication and division) are also initiated by converting all terms into their standard form. Addition and subtraction of terms is only possible, when their units belong to the same dimension

class. Transforming to the standard form, the inference system detects that adding $10\text{J} + 20\text{N}\cdot\text{m} + 15\text{W}\cdot\text{sec}$ yields a valid result (according to the substitute rule of \mathcal{J}) while adding $10\text{m} + 20\text{sec}$ is undefined. Testing this condition prevents from the error of adding values with different dimension units.

In contrast to addition and subtraction, the operations multiplication and division are defined for all combinations of dimension units. Again, the factors of the operation are transformed to the standard form first. A division is transformed into a multiplication with negative exponent. Next, the numerical values and the units are multiplied and the result is transformed again to the standard form. The following example calculates the result of the operation $1\text{N} / 12\text{km}/\text{min}$.

$$\begin{aligned}
 & 1 [\text{N}] / 12[\text{km}/\text{min}] \\
 & = 1000 * [(\text{g}^1), (\text{m}^1), (\text{sec}^{-1})] \\
 & \quad / 12 * 1000 / 60 * [(\text{m}^1), (\text{sec}^{-1})] \\
 & = 1000 / 12 / 1000 * 60 * [(\text{g}^1), (\text{m}^1), (\text{sec}^{-1})] \\
 & \quad * [(\text{m}^{-1}), (\text{sec}^1)] \\
 & = 5 * [(\text{g}^1), (\text{m}^0), (\text{sec}^0)] \\
 & = 5 [\text{g}]
 \end{aligned} \tag{11}$$

As the example shows, the dimension units m and sec are reduced in the multiplication and therefore the result of the division is 5g .

Additional to arithmetical operation, comparability operations can be defined that check if a term is equal, greater or smaller than another. The evaluation of comparability operations, which also need to convert the involved terms to their standard form first, can be defined very easily on the basis of the subtraction operator and the comparability operator of numerical values:

$$\begin{aligned}
 & t_1 > t_2 \text{ if } t_1 - t_2 > 0 \\
 & t_1 < t_2 \text{ if } t_1 - t_2 < 0 \\
 & t_1 = t_2 \text{ if } t_1 - t_2 = 0
 \end{aligned} \tag{12}$$

4.3 Conversion to a desired unit

As described before, a conversion to a desired unit is not always possible. [cm²] can be converted to [mm²], but obviously not to [m/s]. A conversion therefore has to check first if the desired task is defined at all. Otherwise, an error message has to be generated by the inference system. (13) defines a condition that checks if the conversion of dimension unit t_1 to t_2 is feasible.

$$t_1 \text{ is convertible to } t_2 \Leftrightarrow |t_1/t_2| = 0 \quad (13)$$

Equation $|t_1/t_2|=0$ in (13) defines a division of the dimension units and defines that the units are convertible, if all non-numerical elements can be reduced in the division of these terms. The numerical value as the result of this division is equal to the requested value in the desired unit.

The following example shows the conversion of the term 200[m/sec] into the desired unit [km/min]. The conversion is being done by dividing the term 200[m/sec] by the requested unit [km/min]. The units of this division are reduced to zero, therefore the conversion is defined. The calculated numerical value is 12, which means that 200[m/sec] is equal to 12[km/min].

$$\begin{aligned} & 200 \text{ [m/sec] / [km/min]} && (14) \\ & = 200 * [(m^1), (sec^{-1})] * 1000 * 60 * [(m^{-1}), (sec^1)] \\ & = 200 / 1000 * 60 [(m^0), (sec^0)] \\ & = 12 \end{aligned}$$

5. REALIZATION IN THE "ACTIVE SEMANTIC NETWORK"

This section presents a realization that shows how the presented approach can be put into practice. The shown realization is based on an intelligent database system which provides an inference system that is able to realize the presented inferences. A shared

database is an important component in modern CAD environments, because it helps designers to combine their design results into a common solution. Additionally, a shared database is able to serve as a communication medium for the designers of a product. A distributed database system allows to share data among a geographically distributed community of designers. The integration of the presented approach in a shared CAD database system for combining design results of different persons is therefore a promising approach to support product development processes with interdisciplinary, distributed design teams.

The presented database system is called "Active Semantic Network" (ASN). The ASN is an intelligent database that adapts conventional database functions to the particular requirements of modern cooperative product design. The goal of the ASN is to support design teams by providing a common workspace to combine their ideas and exchange partial results. More detailed information about the ASN can be found in (Roller, 1999) and (Schäfer 1999).

The ASN is based on an object-oriented database system whose representation structures are redesigned using a meta model. The ASN provides a programming interface in order to access objects stored in the database. Inference facilities are added on top of the object-oriented database system in order to provide an active behaviour to the system. Beneath a rule-based mechanism, the ASN uses so-called slot demons for inferences. In a slot demon, an attribute of an object cannot only be a simple value, such as a numerical value, but also a evaluation script. This script is parsed and evaluated at runtime. The following evaluation slot shows an example that calculates the surface of a cylinder by its radius:

```
Cylinder.Surface    [PI.Value * this.Radius^2]    (15)
```

In order to represent dimension units in slot values, the unit is represented in an evaluation slot together with its numerical value. The following evaluation slot shows the radius of a cylinder.

```
Cylinder.Radius    [10 [cm]]                    (16)
```

A slot demon is realized by an object that is able to evaluate the contents of evaluation slots. An object-oriented method allows users to evaluate dimension scripts. The

following method call of object `Demon` analyses a dimension script which is stored in the string variable `DimensionScript`.

```
Demon.AnalyzeMeasures(DimensionScript); (17)
```

A slot demon also provides methods to read or write the contents of an evaluation slot. When a desired unit is given, this unit is specified in braces. The following example reads the slot value of slot `Volume` in the desired unit cm^3 . The variable `Trans` refers to the database transaction in which the presented operation is being executed.

```
Demon.GetValue(Trans, "Cyl1.Volume {cm^3}"); (18)
```

Analogously, the dimension unit itself can be read by the programming interface. In the following example, the return of the function call is the dimension unit stored in a regular string.

```
char* Mass = Demon.GetMeasure(Trans); (19)
```

Beside these operations of the ASN programming interface for reading information from a database, operations to modify, insert, or delete dimension units are provided in the same way.

6. SUMMARY

This paper shows a knowledge-based concept to extend numerical values to terms including a numerical value and its dimension unit. Using a general approach, users are able to define the units they need for their specific applications by simple rules. An inference system is able to interpret these rules and to convert units automatically into each other. Additionally, the arithmetical operations addition, subtraction, multiplication and division are extended from numerical values to terms. That means, dimension units are also considered in the execution of these operations. The paper also describes an implementation of the presented approach, which demonstrates the feasibility of the presented concepts. In the presented Active Semantic Network, the inference system is

integrated in a database system, building a knowledge base for intelligent, shared product data models.

As the implementation shows, the approach can be realized with a relatively efficient performance. An integration into a engineering database system is possible without using the complete and extensive inference facilities of typical expert systems. Compared to traditional data structures of database systems, the representation and evaluation of dimension units of course effects an increase in the size of the data models and a decrease in the calculation performance. Compared to the impact of design errors and the lower degree of reusability for the entire design process, these disadvantages appear rather insignificantly.

7. REFERENCES

- Diederich, J., and Milton, J. (1991). Creating Domain Specific Metadata for Scientific Data and Knowledge Bases, *IEEE Transactions on Knowledge and Data Engineering*, Vol. 3, No. 4, pp. 421-434.
- Gero, J.S. (1987). *Expert Systems in Computer-Aided Design*, Elsevier Science Publishers, Amsterdam.
- Gruber, T.R., and Olsen, G.R. (1994). An Ontology for Engineering Mathematics, Fourth International Conference on Principles of Knowledge Representation and Reasoning.
- Gruber, T.R., and Tenenbaum, J.M. (1992). Towards a Knowledge Medium for Collaborative Product Development, Gero, J.S. (ed.): *Proceedings of the Second International Conference on Artificial Intelligence in Design*, Kluwer Academic Publishers, pp. 413-432.
- Olsen, G.R., Cutkosky, M., Tenenbaum, J.M., and Gruber, T.R. (1994). Collaborative Engineering based on Knowledge Sharing Agreements, *Proceedings of the 1994 ASME Database Symposium*.
- Roller, D., and Eck, O. (1999). Knowledge-based techniques for product databases, *Int. Journal of Vehicle Design* Vol. 21, No. 2/3, pp. 243-265.
- Schäfer, D., Eck, O., and Roller, D. (1999). A Shared Knowledge Base for Interdisciplinary Parametric Product Data Models, Lindemann, U., Birkhöfer, H., Meerkamm, H., and Vajna, S. (eds.): *Proceedings of the 12th International Conference on Engineering Design - ICED '99*, 3, pp. 1593-1598.