12th IMEKO TC4 International Symposium
Electrical Measurements and Instrumentation
September 25–27, 2002, Zagreb, Croatia

A SOFTWARE SOLUTION FOR DEVELOPING SYMBOLIC TRANSDUCERS

Radu Varbanescu, Ana Lucia Varbanescu

POLITEHNICA University of Bucharest, Faculty of Automation and Computers
Bucharest, Romania

Abstract – The solution presented here enables any transducer to become symbolic. The development of this software application includes the engine that converts the measurement result and a simulator that allows the user to verify the settings before transferring them to the transducer.

The application needs two types of resources: basic software resources (required for setting the numeric–symbolic conversion application) and hardware/software resources (that will host and execute the application). If the transducer is symbolic by itself, then the resources will be mainly hardware (RAM, ROM, microprocessor, micro-controller etc.). Otherwise, if only the user interface will be symbolic, the required resources are mainly software (execution capabilities, displaying routines etc.).

Keywords: symbolic transducer, intelligent measurements, software added intelligence

1. INTRODUCTION

The main objective is to develop an application that can make a numeric – symbolic conversion that will allow the user to obtain the measurement results in a form that is direct and significant. Home appliances are good examples. How would you like your refrigerator indication: “Temperature OK” or “1ºC”? “Too warm!” or “5ºC”? The first form (directly significant) is preferred instead of the second one (exact, but not as significant, because the user needs to know the temperature specifications for the refrigerator).

A second objective for this application is to be delivered by the transducers suppliers as an add-on for their products. Used like this, it will allow the customer to choose between using the transducer with its standard behaviour and setting a symbolic behaviour, suitable for his/her application.

This application is developed based on a complex database structure that allows both the user and the supplier to add data needed for various configurations. Using this database system, the user should be able to set the symbolic behaviour suitable for the transducer in any particular situation. The system has its own learning engine, so that setting a new configuration will automatically add the new data into the databases. The user interface is graphical, simple and intuitive, so that the configuration is easy to made. The software application includes also a simulator component that allows the user to preview the measurement results for random generated values, or for user supplied values. This simulator can be turned off during the real process, but it’s better to be turned on for you to see how the measurement data are interpreted.

2. SYMBOLIC TRANSDUCERS

The symbolic transducers represent the intelligent transducers class that can talk with the user by non-numeric symbols. The most used symbols are the linguistic ones (words, expressions, word groups) [1], [2].

The symbolic approach relies on a functional model based on the human operator qualitative observations about the measuring process. The measured quantity is now represented by a quantitative/numeric model and by a qualitative/symbolic one. The models are not competitive and a good cooperation between numeric and linguistic processing methods determines the coherence of the two result types. So, it can be said that a symbolic transducer means a functional association between a “classical” transducer and a numeric to symbolic conversion block [3].

2.1. Symbols, symbolism

The section presents some definitions and theoretical aspects concerning the symbolic representations of data [4].

Symbol – representation of a physical entity with the purpose of identifying it;

Symbolism – relation between the “real world” and the symbolic world expressed by the following relationship:

$$\mathcal{C} = < \mathcal{Q}, \mathcal{S}, \mathcal{M}, \mathcal{R}_Q, \mathcal{R}_S, \mathcal{F} >$$

Where:

$\mathcal{Q}$ – the real world elements set;

$\mathcal{R}_Q$ – the set of relations between the real world elements;

$\mathcal{S}$ – the symbols set;

$\mathcal{R}_S$ – the set of relations between symbols;

$\mathcal{M}$ – the relation from $\mathcal{Q}$ to $\mathcal{S}$; it determines the symbolism representation mechanism;

$\mathcal{F}$ – the relation from $\mathcal{R}_Q$ to $\mathcal{R}_S$; it determines the symbolism relation mechanism.

$\mathcal{C}$ is a symbolism if and only if

For $q_1, q_2, \ldots, q_k \in \mathcal{Q}$, $s_1, s_2, \ldots, s_k \in \mathcal{S}$, $k = 2, \ldots, n$

$\mathcal{R}_Q(q_1, \ldots, q_k) \Leftrightarrow \mathcal{R}_S(s_1, \ldots, s_k)$

Where $\mathcal{R}_S = \mathcal{F}(\mathcal{R}_Q)$ and $s_1 = \mathcal{M}(q_1)$, $s_2 = \mathcal{M}(q_2)$, …, $s_k = \mathcal{M}(q_k)$.

The measuring scales represent a good example of a symbolism. In this case the symbols are numbers. So, there are three scale types [5]:

- nominal scale, defined by : $q_1 \sim_Q q_2 \Leftrightarrow \mathcal{M}(q_1) \sim_S \mathcal{M}(q_2)$
- ordinal scale, defined by: $q_1 \geq_Q q_2 \Leftrightarrow \mathcal{M}(q_1) \geq_S \mathcal{M}(q_2)$
- linear scale, defined by:

$$q_1 \sim_Q q_2 \Leftrightarrow \mathcal{M}(q_1) \sim_S \mathcal{M}(q_2)$$
$$q_1 \, o_Q \, q_2 \sim_Q q_3 \Leftrightarrow \mathcal{M}(q_1) \, o_S \, \mathcal{M}(q_2) \sim_S \mathcal{M}(q_3)$$

2.2. Linguistic symbolism

The aim of the linguistic symbolism is to represent the measured quantity by words, expressions or word groups, starting from numerical measurement results. To do such a numeric – linguistic conversion, we have to define three symbolism types:

1. The numerical symbolism $C_1 = < Q, \mathcal{N}, \mathcal{M}_1, \mathcal{R}_Q, \mathcal{R}_\mathcal{N}, \mathcal{F}_1 >$ where $\mathcal{N}$ is the numeric set.
2. The linguistic symbolism $C = < \mathcal{N}, \mathcal{L}, \mathcal{M}, \mathcal{R}_\mathcal{N}, \mathcal{R}_\mathcal{L}, \mathcal{F} >$ where $\mathcal{L}$ is the set of words and $\mathcal{M}$ depends on the measurement context.
3. The human symbolism $C' = < Q, \mathcal{L}', \mathcal{M}', \mathcal{R}_Q, \mathcal{R}_{\mathcal{L}'}, \mathcal{F}' >$ that directly connects the real world to the linguistic world. $\mathcal{M}'$ depends on the general context and its efficiency is determined by the usage of all the knowledge about the measurement to be accomplished.

The three representations forms are connected by the relation $C' = C_1 \, \Delta \, C$, where C must be configured to determine a good resemblance between the transducer and human operator behaviours.

2.3. Meaning and description

Meaning of a lexical value is defined as an injective mapping from $\mathcal{L}$ to $\mathcal{N}$[6]:

$$\tau : \mathcal{L} \to \mathcal{P}(\mathcal{N}),$$
$$\forall a \in \mathcal{L},\ \tau(a) = \{x \in \mathcal{N} \mid x\mathcal{M}a\}$$

Description of a numeric value is defined by lexical values that characterize it:

$$\theta : \mathcal{N} \to \mathcal{P}(\mathcal{L}),$$
$$\forall x \in \mathcal{N},\ \theta(x) = \{a \in \mathcal{L} \mid x\mathcal{M}a\}$$

Fuzzy meaning is defined using the set of fuzzy subsets of $\mathcal{N}$, $\mathcal{F}(\mathcal{N})$

$$\tau : \mathcal{L} \to \mathcal{F}(\mathcal{N}),$$
$$\forall a \in \mathcal{L}, \forall x \in \mathcal{N}\ \ \mu_{\tau(a)}(x) = \mu_\mathcal{M}(a, x)$$

Fuzzy description is defined using the set of fuzzy subsets of $\mathcal{L}$, $\mathcal{F}(\mathcal{L})$:

$$\theta : \mathcal{N} \to \mathcal{F}(\mathcal{L}),$$
$$\forall a \in \mathcal{L}, \forall x \in \mathcal{N}\ \ \mu_{\theta(x)} = \mu_\mathcal{M}(a, x)$$

2.4. Fuzzy measuring scales

There are two fuzzy scales we have to define.

1. Nominal scale is defined by:

$$a, b \in \mathcal{L},\ a = b \Leftrightarrow \mu_{\tau(a)}(x) = \mu_{\tau(b)}(x),\ \forall x \in \mathcal{N}$$
$$\Leftrightarrow \mu_{\theta(x)}(a) = \mu_{\theta(x)}(b),\ \forall x \in \mathcal{N}$$

2. Ordinal scale is defined by:

$$a \leq_\mathcal{L} b \Leftrightarrow \mu_{\tau(a)}(x) \leq \mu_{\tau(b)4}(x),\ \forall x \in \mathcal{N}$$

where $\mu_{\tau(b)4}$ is an operator defining an order relation on $\mathcal{L}$ [4].

2.5. Linguistic concepts

Linguistic concept = association between a lexical value and its meaning.

In an exact approach creating a linguistic concept implies the definition of a partition on the numeric domain. The operation may be extended to the fuzzy case, using a partition with the following properties:

1. $\inf\left(\max_{a \in \mathcal{L}} \mu_{\tau(a)}(x)\right) > 0,\ \forall x \in \mathcal{N}$, that means that $\mathcal{L}$ covers $\mathcal{N}$;
2. $\forall a, b \in \mathcal{L}\ \sup_x \left(\min(\mu_{\tau(a)}(x), \mu_{\tau(b)}(x))\right) < 1$, that determines the linguistic concepts to be different;
3. $\forall x \in \mathcal{L}\ \ \Sigma\mu_{\tau(a)}(x) = 1$, that expresses the orthogonality of the concepts.

The most used methods for creating fuzzy linguistic concepts are:

a. methods based on semantic relations between symbols;
b. methods using interpolation between particular lexical values meanings.

3. DATA AND DATA STRUCTURES

The keywords defining the data structures used by the application are:

Transducer – instance of the transducers class. This virtual object is specified by some properties: type, name, range, measuring unit and accuracy. In fact, the transducers class defines a hierarchical structure of subclasses more and more specialized.

Application – a particular situation that uses a certain transducer. In such a situation the transducer works on a limited range and it is characterized by application dependent specific properties. This case is a good example for the inheritance concept in OOP (Object Oriented Programing) [7] because the transducer associated with the specific application represents a specialization of its generic ancestor. So it's obvious why the application belongs to the transducer and not inverse (as the structured programming asks).

Linguistic symbol (linguistic estimation) – the word or the group of words which express in a symbolic way a particular numeric value.

Central value – the central concept, assigned to the central value of the range the data belong.

Adaptation function – function used to modify the concept forms.

Dictionary – database containing the linguistic symbols that can be associated to an application.

Function (linguistic concept form) – mathematical functions defining the concept forms. For the well known, but complicated, forms the program builds a special forms library.

The data structures represented by their associated databases are:

3.1 Transducers

The transducer collection is organized as a database having the structure presented in Table 1

Table 1 The transducers database structure

Field name	Data type	Explanations
Code*	Numeric	An identifying code assigned to the transducer
Name	Character	The transducer name
Class	Character	The transducer type (ex. temperature transducers class)
Lower range limit	Numeric	The measuring range limits
Upper range limit	Numeric	
Accuracy	Numeric	
Measuring unit	Character	

*The program automatically assigns the transducer code.

3.2. Applications

The applications data base structure is presented in the Table 2.

Table 2 The applications database structure

Field name	Data type	Explanations
Transducer code*	Numeric	The code that links the application to a certain transducer
Application Code	Numeric	Internal code of the application
Name	Character	Application name
Lower range limit	Numeric	The application range limits
Upper range limit	Numeric	
Central value**	Numeric	The central concept code
Concepts***	Numeric	The concepts number
Form code****	Numeric	The code of function giving the concepts forms
Adaptation code	Numeric	The adaptation function code

The structure contains a field specially created to link this database to the transducers database. This relation is necessary because the applications are properties of the transducers (from the programming point of view).
*Transducer code is automatically specified, due to the existing relation between the two databases;
**The central value is selected from the existing dictionary;
***The number of linguistic concepts is limited to 9;
****The concept form code and the adaptation function code are directly related to the function collection;

3.3. Functions

The functions database structure is presented in Table 3. Each database record defines the function on the first fuzzy subset. The defined function will then be replicated for all the subsets, changing only the definition domain.

Each function has to satisfy several mathematical criteria. Defining a new function, the user has to verify these criteria first and then he can type the new record.

Table 3 The functions database structure

Field name	Data type	Explanations
Function code*	Numeric	
Name**	Character	The name associated to the function. It is useful for the predefined functions.
Branch 1***	Character	The mathematic expression of the first function branch.
Lower 1****	Numeric	The first branch of the function is defined on the interval determined by those two limits
Upper 1	Numeric	
There are 5 branches, and five upper/lower limits for them		

*The function code is assigned automatically;
**The names are only identifiers;
***Branch 1, branch 2, … are mathematical expressions representing the function evolutions on intervals;
****Lower 1/upper 1, … are the limits of the sub ranges for branch 1, …

3.4. Linguistic symbols, concepts and dictionary

The dictionary is the data structure storing words, expressions and groups of words that represent in a symbolic way the measurement numerical results. In the dictionary there are two word types: words defining the central concepts and words defining the secondary (derived) concepts. The words of the first category are the only ones that can be assigned to the central concepts. The "central" words generate the words/groups of words assigned to the rest of concepts. Usually, the concept associated to the central value is named concept 0. So, the string of concepts will be:

concept –4<concept –3<…<concept 0<concept 1<…< concept 4

There are two methods to develop the words dedicated to the secondary concepts. The first method associates comparative degrees (much more, more, less) to the central word. Using this method the string of words can be generated automatically, starting from the central word. The second method uses special words to be assigned to the secondary concepts. These special words are parts of word families used in the fluent language. A good example is representing temperature by *cold, warm, hot*. Such a string cannot be obtained automatically and the user must generate it concept by concept. The data structure is presented in the Table 4:

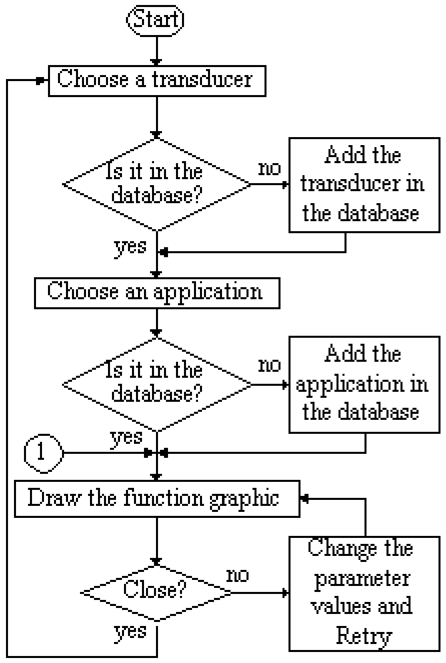
Table 4 The linguistic concepts database

Field name	Data type	Explanations
Code*	Numeric	The central concept code
Text	Character	The word representing the central concept
Grad_k	Character	The words before the central concept
Gradk	Character	The words after the central concept

*The internal code of each central concept has to be unique, because the same central concept can occur in different situations. This means that two or more gradual strings can start from the same central concept. So, each symbol string will have two identification keys: the concept 0 and the internal code.

4. HOW THE PROGRAM WORKS

This section presents the program from the user point of view, emphasizing operations like setting, configuring, adding [8]. The steps required to build up a symbolic transducer are illustrated in the figure 1. If this application is used for a master transducer, the user must insert in place of the label "1" a new command like "*Send configuration to the transducer*". The rest will remain the same.



Start
Choose a transducer
Is it in the database?
no
Add the transducer in the database
yes
Choose an application
Is it in the database?
no
Add the application in the database
1
yes
Draw the function graphic
Close?
no
Change the parameter values and Retry
yes

Fig. 1 The flowchart for building up a symbolic transducer

4.1. Transducer selection

The user has to select a certain transducer from the transducer collection. The transducer can be selected by name or by internal code. There are two special situations that request other information. At first, the expected transducer is not present in the collection; secondly, there are two or more transducers with the same name. In the first case the user has to add the new transducer to the collection. The program has a dedicated routine to do this. In the second case the user has to access more information about the existing transducers. The "Info" command presents the features for the selected transducer. The features analysis will determine the selection.

4.2. Application selection

After you chose the transducer, you have to select the application. The program provides an application list, already defined, for the selected transducer. Each application is specified by name and by code. The user can meet the same special situations as in the transducers case and the solutions are the same. Concerning the application database it is important to emphasize that the selection is made in two steps. The first step is a filtering operation, which selects those applications that are useful for the specified transducer. The filter, using a single filtering condition applied to the code field has the form:

Filter ← (Application.Field("Transducer_Code")= Chosen_Transducer.Code)

In the second step the application will be selected from the filtered database.

Now, we have the transducer and its particular application. The next step is to transfer the selected features to the transducer (for a "master" one) or to the control system (for a "slave" one).

4.3. How the program can learn

The program can "learn" by adding new components and a more accurate specification of the parameters. Databases content can be increased using configuration operations. The transducers catalogue can be updated by adding new transducers and by modifying (editing) the existing ones. The applications catalogue can be updated too.

4.4. Graphical representation

After all the parameters were set, it is important to see how the transducer acts in various situations. This is the main reason for ending the set-up process with a graphical simulation. If the transducer can pass the tests, the configuration is saved and it will be used for the current application. If the test cannot be passed, the parameters should be adjusted in order to obtain a correct configuration.

The idea for this simulation is very simple: a value should be entered – representing the measured value – and the program will translate it into the appropriate expression – representing the symbolic transducer output. The user should compare the results of the program with the expected value: if a match is found, the test is passed.

The graphical interface shows the representation of the function that was defined for the application. The "position" of the entered value is shown on the picture by an animated cursor. That tool is useful for an inverse check-up: the user chooses a value *on the picture* and if the numerical value is correct, the test is approved.

This tool is a powerful simulator, enhanced with tuning skills. In most cases, in real-life situations the transducer will pass any of the tests passed here. Anyway, as the laws of testing say, there are no guarantees for 100% correctness in translation by random tests.

4.5. Master and slave transducer

There is yet another important issue to be discussed. From resources point of view, we split the transducers class in two parts: master transducers and slave transducers. What is the difference?

A *master transducer* is a transducer that has sufficient hardware resources so that it can memorize the configuration and use it locally. This means that it will output the measurement data using the symbolic representation. The computer is used only to create the data frame that will be exported to the transducer memory (ROM, RAM or Flash Card).

A *slave transducer* is sort of a dummy component. Computer only implements its symbolic behaviour. Basically, the transducer will output the numeric value, but the application will convert it into a symbolic one.

Remember that the user will not notice the difference in behaviour, but he/she can feel the performance difference (a master transducer will be faster and more reliable than a slave one).

4. CONCLUSIONS

First of all, it is important to understand that the main engine of this application is the conversion algorithm from numeric data to linguistic expressions, and it is fully functional. There are still some supplemental features to be added, so that the user can easily configure and re-configure any device. The database system should also be improved for faster access to data.

This software application was build for enhancing transducers by adding symbolic behaviour, a feature that can be very useful for many kinds of supervising applications. It works for both master transducers (which will become stand-alone symbolic transducers) and slave transducers (they will become computer-aided symbolic transducers).

In present, this application is tested with several specific cases.

5. REFERENCES

[1] L. Finkelstein, D. Hofmann, “Intelligent Measurement-A View of the State of the Art and Current Trends”, *Measurement*, no. 5/1987 pp. 151-153

[2] D. Hofmann , “Intelligent Measurements - New Solutions for Old Problems”, *Measurement*, no. 13/1994

[3] E. Benoit, L. Foulloy, “Symbolic Sensors: One Solution to the Numerical-Symbolic Interface”, *Proc. IMACS DSS,* Toulouse, 1991 pp. 321-326

[4] G. Mauris, E. Benoit, L. Foulloy, “Fuzzy Symbolic Sensors - From Concept to Applications”. *Measurement*, no. 12/1994 pp. 357-384

[5] L. Finkelstein, M.S.Leaning, “A review of the fundamental concepts of measurement”. *Measurement*, no. 2/1984 pp.25-34

[6] E. Benoit, L. Foulloy, “Symbolic Sensors”, *Proc. 8th IMEKO TC7*, Kyoto, 1991 pp. 131-136

[7] B. Meier, “Object Oriented Software Construction”, *Prentice Hall*, EngleWood Cliffs, 1988

[8] L. Finkelstein, J. Huang, “Using Software Application Methods for Measurement Instrument Systems”, *Measurement*, no. 10/1992

Authors:

Prof. Dr. Radu Varbanescu, POLITEHNICA University of Bucharest, Faculty of Automation and Comptuers, Dpt. of Control Systems, 313 Spl. Indpendentei, Bucharest, Romania, +401 7693079, b30ara@hotmail.com

Eng. Ana Lucia Varbanescu, POLITEHNICA University of Bucharest, Faculty of Automation and Comptuers, Dpt. of Computer Science, 313 Spl. Indpendentei, Bucharest, Romania, +401 7693079,analucia@acasa.ro