Embedding fuzzy variables into C++

Daniela Gotseva

Abstract *This paper introduced applying fuzzy sets theory into programming languages and logic. After short introduction the author present new data type in C++: Fuzzy. This is linguistic variable that help programmers to reduce time and cost estimation for software projects. COCOMO II is used for time and cost calculation. New data type is shown in EBNF and simple example is used for time estimation.*

Key words: *Computer Systems and Technologies, Fuzzy Sets, Fuzzy Logic, Programming Languages.*

INTRODUCTION

This paper introduced applying fuzzy sets theory into programming languages and logic. The author presents new programming language fuC++. This language is a C++ extension with embedding fuzzy constant and linguistic variables. New keywords are inserted for fuzzy variables and membership functions. Some important membership functions are applied and should be realised into new language. These functions are presented with their graphics. New language features are shown with EBNF. This new language represented Fuzzy Oriented Programming. Its application is important in some key areas as medicine, education, artificial intelligence. Business logic representation and understanding should be improved using fuC++. Time consuming for understanding and embedding business logic should be reduced as a result.

LAYOUT

Let's take a look at programming life cycle. At first point of view we can see:

- ✓ Human logic isn't binary one. This leads to hard coding and increases translation errors from human to binary logic;
- ✓ It's difficult to understand user requirements. Time for analyses and design enlarged as a result;
- ✓ Analyses and design errors are valuable. It will be a good idea to reduce them.

To increase programming capabilities, reduce time consuming for user requirements specification, the author proposed embedding linguistic variables into programming languages. As a result we can produce:

- Easily user requirement understanding and coding;
- Embedding human logic into programming;
- Adding "more or less", "extremely", "slightly", "very" to user notations.

C++ is used as a base, because this language is object oriented programming language with common purpose. It is modern programming language with fast compilation and very good optimization.

New keyword is added to express linguistic variables. Let's take a look at it:

fuzzy <type name> {
  <fuzzy values list>
  [<type cast definitions>]
} [<linguistic variables list,,,>];

fuzzy <type name> < linguistic variables list,,,>;

where bold face shown new keyword and ***<text>*** is metavariable, and [text] is optional.

<Fuzzy values list> is defined by:

"***<fuzzy constant>***"(***< μ function>***,<x1>[,<x2>,<x3>,<x4>]);

<fuzzy constant> is human term. There is no restriction on it.

< μ function> is membership function and can be one of:

- lineinc function with parameters x1 and x2 and formula:

$$f(x) = \begin{cases} 0, x < x1 \\ \dfrac{x - x1}{x2 - x1}, x1 <= x <= x2 \\ 1, otherwise \end{cases}$$

- linedec function with parameters x1 and x2 and formula:

$$f(x) = \begin{cases} 1, x < x1 \\ \dfrac{x2 - x}{x2 - x1}, x1 <= x <= x2 \\ 0, otherwise \end{cases}$$

- triinc function with parameters x1, x2 and x3 and formula:

$$f(x) = \begin{cases} \dfrac{x - x1}{x2 - x1}, x1 < x < x2 \\ \dfrac{x3 - x}{x3 - x2}, x2 <= x <= x3 \\ 0, otherwise \end{cases}$$

- trapinc function with parameters x1, x2, x3 and x4 and formula:

$$f(x) = \begin{cases} \dfrac{x - x1}{x2 - x1}, x1 < x < x2 \\ 1, x2 <= x <= x3 \\ \dfrac{x4 - x}{x4 - x3}, x3 < x < x4 \\ 0, otherwise \end{cases}$$

- trapdec function with parameters x1, x2, x3 and x4 and formula:

$$f(x) = \begin{cases} \dfrac{x2 - x}{x2 - x1}, x1 < x < x2 \\ 0, x2 <= x <= x3 \\ \dfrac{x - x3}{x4 - x3}, x3 < x < x4 \\ 1, otherwise \end{cases}$$

All of those functions are embedded into C++. The programmer takes responsibility of parameters value, which should be numbers (integer or real). They depend on human notation. Membership function shows crispy variable belonging to the fuzzy constant. At

the moment only real crispy values are used.

<type cast definitions> is C++ operator typecast function. It shows how to convert data from and to linguistic ones. It's optional.

RESULTS

I will be present the results using some example. It will be shown how to define new variables and the differences when using standard C++ constructions. Next I will show what happen with time and cost estimation for this simple example, using COCOMO (Constructive COst MOdel) II model.

Example: Let's define linguistic variable temperature with low, middle and high fuzzy constants. The followed rules are given:

- ✓ **the temperature is low below 10°C and between 10°C, and 20°C using linear dependency;**
- ✓ **the temperature is high above 30°C and between 25°C, and 30°C using linear dependency;**
- ✓ **the temperature is middle between 20°C and 30°C, and using linear dependency between 15°C and 20°C, and between 30°C and 35°C.**

There is no typecasting.

Let's define this variable, using new keyword Fuzzy:

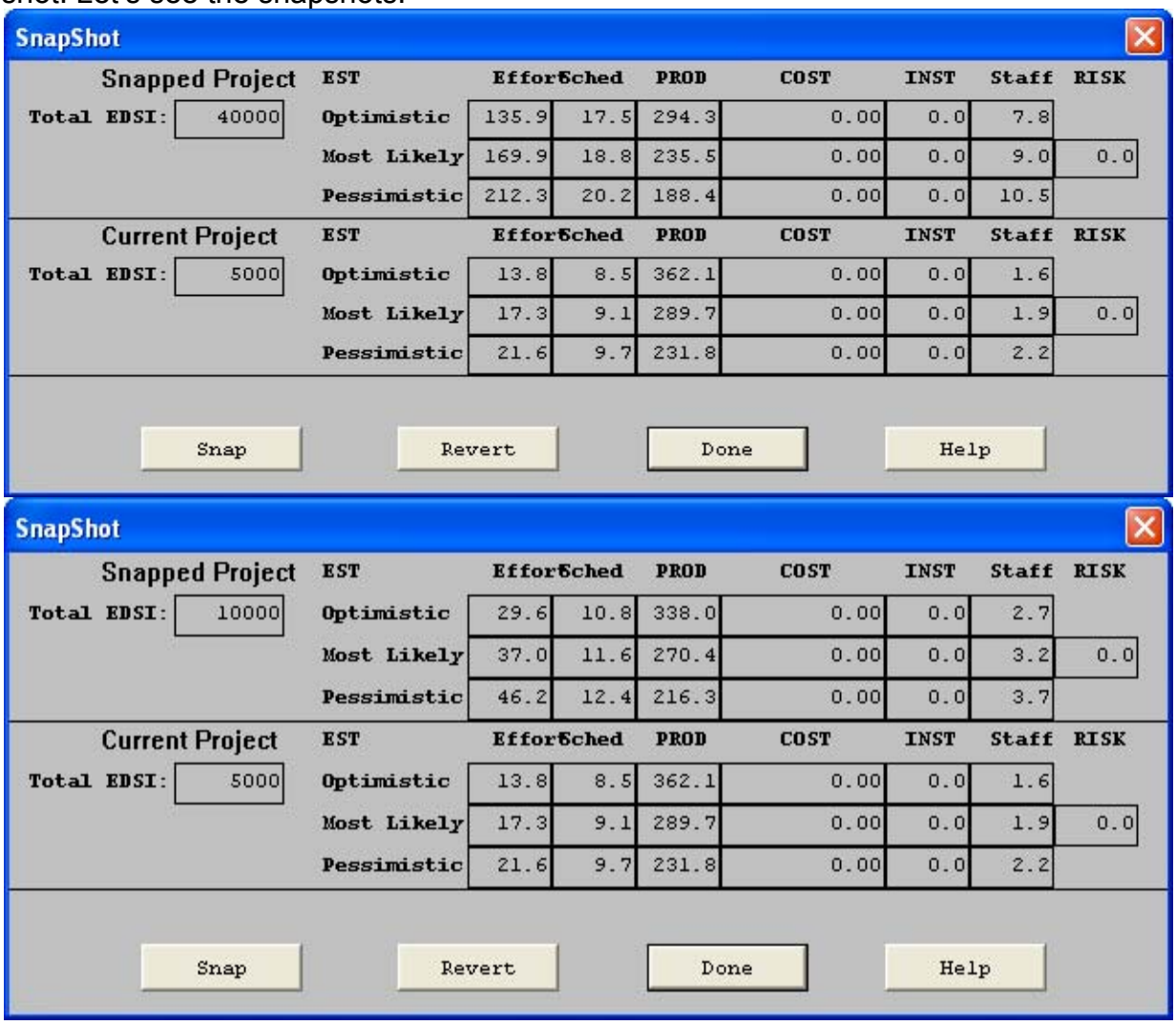
```
Fuzzy TTemp {
"high"(lineinc,25,30);
"middle"(trapinc,15,20,30,35);
"low"(linedec,10,20);
} temperature;
```

Let's try to define the same using standard C++ structure:

```
Class Tlineinc {
Double x1,x2;
Tlineinc(double x, double y): x1(x), x2(y) {};
Public:
      Double Y (double x) {
        if (x<x1) return 0;
        if (x>x2) return 1;
          return (x2-x)/(x2-x1);
};
};
Class Tlinedec {
Double x1,x2;
Tlinedec(double x, double y): x1(x), x2(y) {};
Public:
      Double Y (double x) {
        if (x<x1) return 1;
        if (x>x2) return 0;
          return (x-x1)/(x2-x1);
};
};
Class Ttrapinc {
Double x1,x2,x3,x4;
Tlineinc(double a, double b,double c, double d): x1(a), x2(b), x3(c), x4(d) {};
Public:
      Double Y (double x) {
        if (x>x1 && x<x2) return (x-x1)/(x2-x1);
```

```
        if (x>=x2 && x<=x3) return 1;
        if (x>x3 && x<x4) return (x4-x)/(x4-x3)
          return 0;
};
};
class TTemp {
Tlineinc high(25,30);
Ttrapinc middle(15,20,30,35);
Tlinedec low(10,20);
} temperature;
```

Next let's see what happen with cost estimation, using COCOMO II. For experimental reason I will use Module definition with 1000 variables because COCOMO II is calibrated for 2000 SLOC minimum. When using Fuzzy definition total line of code will be 5*1000 = 5000. This project is "current Project" in COCOMO II snapshot. In the other case the result will be: 40*1000 = 40000 in worst case (when the project needed of different membership function definition), and 10*1000 = 10000 in better one (when the programmer works with this membership functions only). The project named "snapped Project" in COCOMO II snapshot. Let's see the snapshots:



Here effort is time cost in Mah per Month, Sched is time cost for scheduling project, PROD is productivity. COST and INST is 0 because no Labor rate ($/month) is entered. Staff is the time estimation needed for other project support work RISK can help us to provide project risk analyses. I leave it 0 because example is very simple. All calculations

are done for Optimistic, Most Likely and Pessimistic cases. **The results shown that time estimation when using Fuzzy variables is between 2 and 10 times shorter than when using C++ classes.**

CONCLUSIONS AND FUTURE WORK

Embedding linguistic variables in C++ leads to creating new programming style: Fuzzy Oriented programming. This style collects Object oriented programming and using fuzzy variables with it. FOP will be useful to reduce analyze and design software project phases. The next step C++ extension will be adding fuzzy expressions.

REFERENCES

[1] Fukami, S., Mizumoto, M. and Tanaka, K. Some considerations of fuzzy conditional inference, Fuzzy Sets and Systems, 1980, 4, 243–273.
[2] Jantzen, J. Array approach to fuzzy logic, Fuzzy Stets and Systems, 1995, 70, 359–370.
[3] Kaufmann, A. Introduction to the theory of fuzzy sets, Academic Press, New York, 1975.
[4] Kiszka, J. B., Kochanska, M. E. and Sliwinska, D. S. The influence of some fuzzy implication operators on the accuracy of a fuzzy model, Fuzzy Sets and Systems, 1985, 5, (Part1) 111–128; (Part 2) 223 – 240.
[5] Mamdani, E. H. Application of fuzzy logic to approximate reasoning using linguistic synthesis, IEEE Transactions on Computers, 1977, C-26 (12): 1182–1191.
[6] Wenstop, F. Quantitative analysis with linguistic values, Fuzzy Sets and Systems, 1980, 4(2), 99–115.
[7] Zadeh, L. A. Fuzzy sets, Inf. And Control, 1965, 8 338–353.
[8] Zadeh, L. A. The concept of a linguistic variable and its application to approximate reasoning, Information Sciences, 1975, 8, 43–80.
[9] Zadeh, L. A. Fuzzy logic, IEEE Computer, 1988, 21(4): 83–93.
[10] Zimmermann, H.-J. Fuzzy Set theory and it's application, 1993, second ed, Kluwer, Boston (1. ed. 1991).
[11] http://www.softstarsystems.com/cocomo2.htm

ABOUT THE AUTHOR

Assist. Prof. Daniela Gotseva, Faculty of Computer system and control, Technical University Sofia, tel.: +359 2 965 2338, +359 88 7 630 162, E-mail: dgoceva@tu-sofia.bg.