Fuzzy Model-View-Controller Pattern

Rasool Karimi
Department of Electrical and Computer
University of Tehran
IRAN
r.karimi@ece.ut.ac.ir

Caro Lucas
Department of Electrical and Computer
University of Tehran
IRAN
lucas@ipm.ir

***Abstract*--There are a lot of patterns for software development which Model-View-Controller (MVC) pattern is one of them. In this pattern, the software is divided into three separate layers: View, Controller, and Model. This model is similar to open loop model of control system. By applying fuzzy control theory to MVC pattern (model) we can develop a new model that we refer to it as Fuzzy MVC (FMVC) pattern. By this pattern, we can develop softwares that have fuzzy graphical user interface.**

***Index Terms*-- Fuzzy Controller, Fuzzy Graphical User Interface (FGUI), Fuzzy Model-View-Controller (FMVC) pattern.**

I. INTRODUCTION

The most important way for communication between people is linguistic communication. So, people are interested in communication with artificial systems such as: computers, automobiles, washing machines, and similar systems in this way. However that is not possible most of the times. So, people must communicate with these systems via their interfaces. The most important feature of linguistic communication is its fuzzy property. That means, information which is transmitted is not completely defined and has ambiguity. This is one of the reasons that attract people to linguist communication. So, if interfaces of artificial systems are be fuzzy, people can communicate with them better. For example an automobile driver controls speed and direction by the gas pedal and steering wheel interfaces. These interfaces are analog and the driver is not forced to determine speed and direction precisely and digitally. So as we can see, people live in an analog world. Sensing parts of the body receive this analog information and send them to brain. Brain processes this information in an analog way and sends appropriate commands to body. In the 20^{th} century, people decided to make artificial brain to do computing, faster, easier, and more exact. This artificial brain was named “computer”. Because of progress in digital electronic technology, computer was made based on this technology. Thus, information is received, processed and saved digitally in the computer. While these operations are done in the people's brain in analog way. Computer programmer had to write computer programmer codes with 0 and 1 language for computer to understand the program. This was a very difficult task to do. The solution that was introduced to solve this problem was that the programmer generates program's code in a high level language that is more similar to the natural language. Then before execution, the high level code is given to a middle layer which is called “compiler”. Compiler translates this program to the machine language. One of the benefits of this solution is that programming becomes easier while the program is still understandable for the computer. So programmer can communicate with computer easily. But the programmer is not the only user of the computer.

The program might be used by many different users. Form user’s point of view, the software is a virtual computer. As the software dose its work and the user does not know anything about what is happening behind the software. User communicates with software via its user interface. Today this user interface is graphical. Graphical User Interfaces (GIU) usually are such that user has to enter information exactly and this is not proper in some occasions. Seems that as compiler improves the communication between programmer and computer, if the GUI cans get fuzzy information from user, changes them to exact information and give them to software, then the communication between user and computer improves. For this aim, the software architecture must be modified.

The remainder of this paper is organized as follows: First we describe Model-View-controller (MVC) pattern that is an important pattern in software architecture. Then similarity between MVC pattern and open loop model of control system is described. At last, we modify MVC pattern and develop the new pattern with name of Fuzzy MVC (FMVC) pattern to developing softwares with Fuzzy Graphical User Interface (FGUI).

II. MVC PATTERN

Before building a house, it must be designed. Without a good design, house is not strong enough, windows and doors will be in wrong places, walls are not fixed many other problems may be happened. Computer software is more complex than the house. So it must be well designed [4]. One goals of the software design is to build software architecture. Software architecture is a blueprint of software that describes most important parts of it and addition their relation [5]. One important part of the software architecture is pattern. Pattern is a good idea that has been good in one practical context and will probably be useful in other areas as well [8]. There are many patterns that can be used in software development.

MVC pattern is one of these patterns. In this pattern,

software is divided into 3 independent layers [7].

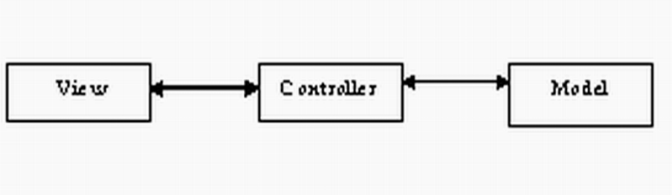

Figure 1. The Model-View-Controller Pattern

1) Model layer

This layer may be a database that maintains data. Also, functions and logics that software is based on them are in this layer. These functions are collected in "function library". In some applications database and functions library both exist but in other applications one of them exists. In this paper we assume that only one of them is in this layer.

2) View layer

This layer interacts with the user. In fact it is (GUI) of the software. User enters necessary data and parameters for software execution in this layer and observes result of it.

3) Controller Layer

This layer is an interface between view and model layers. It receives requests from user and transfers them to model layer. Then receives result of model layer and transfers them to view layer.

Some softwares are developed for searching databases. Model layer in them is a database and user searches the database by entering information tin the user interface. This information is called "data parameters". Applications that model layer of them consists of functions library, are developed for doing other tasks. Assume that this task can be executed in several ways. We refer to each of these ways as a "software behavior". Each of these behaviors is implemented as a function. So functions library contains all behaviors of software. For each of these functions or behaviors there is one option in GUI. User can customizes output or behavior of software by selecting these options. For example consider a user that uses "WinZip" program for compressing files. This program has two options for file compression: One option is for compression with fast speed and other option is for compression with best efficiency. For each of these options there is one function that implements it. "Visual Studio" is another example. In Visual Studio software, the programmer has two options for compiling the source code. The first option is that the generated code has minimum size and the second is that the generated code will be executed faster. For each of these options there is one function that compiles source code.

These options that determine behavior of software are called “control parameters”. Regarding control parameters the control layer selects a function of model layer and passes the data parameters to it. Then it receives the result of function and sends back it to the view layer for user observation.

A. Disadvantages of MVC pattern

A software that is developed based on MVC pattern has crisp GUI. Crisp GUI has two major properties. The one is that the user has to select only "one" of the control parameters. For example in winzip software user may wants to compress a file in the following way:

"Compress the file almost fast, but generated archive is no very big."

Or in Visual Studio software the programmer may want to compile the code in this way:

"Compile the source code such that it will be executed fast but has not a very big."

The second reason is that the user has to enter most of the data parameters digitally and exactly while what the means is something not exact. For example consider the software that is used in police office for diagnosis of guilty people. The method for using this software is that he witness who was present in the crime scene and has observed the guilty person gives properties of him or her to the software. There is a database that contains profiles of previous recorded guilty people. Then software searches this database and finds the guilty person if he or she has had profile in the database. The information about the guilty person that the witness has is vague and imprecise. For example he knows that guilty person is rather short, is middle aged and is rather fat, which means that his/her weight has a rather high value. The software will not be proper if it requires the precise values of size, age, and weight of the guilty person. Because the witness has to change vague information to precise information. In addition to this task is difficult for the user, the accuracy of the software might also become less, as when vague information is changed to precise information some of their concepts might be lost.

These are some of the cases that can not be done with a crisp GUI, but they are possible in a fuzzy GUI. because of similarity between MVC model and open loop model of control system, we can use ideas of control system to modify MVC model.

III. OPEN LOOP CONTROL SYSTEM

The block diagram of an open loop control system is as follows [2]:

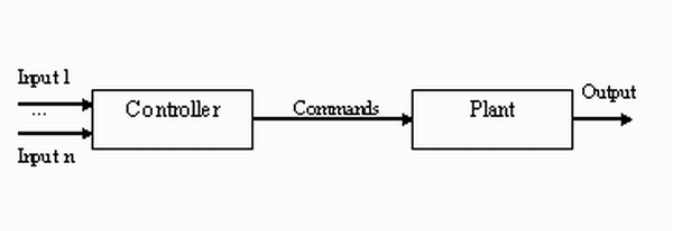

Figure 2. Open loop model of control system.

The controller considers the inputs, and sends appropriate commands to the plant to reach the desired output. The plant cans any system such as: a DC motor, a robot manipulator, a

society of people, or a "software". In fuzzy control, controller is contained of a set of "if-then" rules. If the input is crisp, it must be fuzzified first. It means that the possibility of it must be determined in first part of any rule. Second part of any rule is fired according to the first part. Then all of these fired values are gathered and after defuzzification, the crisp value is calculated and applied to the plant [3].

If we consider the plant as a software, we must define other part of control system such as: inputs, controller, commands, and output.

A. Similarity between a MVC pattern and an open loop control system

MVC model and open loop model of control system are similar in some aspects. Model layer is plant that must be controlled. Inputs of the system are parameters that user enters via user interface (view layer). Output of the plant is result of function call that may be: a number, a file, a network packet and etc. Control layer is also controller of the control system and its commands are function names of model layer that must be executed. In fact controller layer controls model layer via inputs that user enters in user interface.

The control layer of MVC model is a set of "if-then" rules in following way:

If option 1 is selected then function 1 must be invoked.
If option 2 is selected then function 2 must be invoked.
…
…
…
If option n is selected then function n must be invoked.

If view layer is modified such that it is possible to select several options with deferent possibilities, then several rules will be active in control layer. That means several functions must be executed. Then result of each of them must affect output with considering the possibility of corresponding option. This is similar to fuzzy control method.

This similarity helps us to extend some concepts from control domain to software domain. Two instances of them are described below.

B. Robust System

In control engineering, if the system has bounded output for bounded input then it is robust. If software crash is considered as a boundless output of the system and interaction of user with software is considered as the system input, then this definition of robust system is equal to robust software definition in software engineering domain. As in software engineering the software is robust if it dose not crash where interacting between user and itself. That means that has bounded output for bounded input. In control science we can prove robustness of system mathematically. If we can define a transfer function for software, then we can prove robustness of software mathematically. This is very important subject for software engineering.

C. Definition of system's precise model

In Control engineering, when the plant is very complex, definition of complete mathematically model of it becomes difficult and even impossible. On the other hand, if we insert more elaborations of plant in the model, the control system will not work better necessarily. As the system will be sensitive in opposite with disturbances and noises and it may be fault. To solve this problem, some parts of plant are neglected to define a simpler model. Neglected parts of plant are considered as disturbances. In the other hand in software engineering before the software is implemented, a model of user requirements is designed. More part of required cost and time for developing a software is spent for design of this model. Despite all, this model will not be complete and will not match all user requirements. For solving this problem we can use methods which are used in control engineering. That means it is not necessary to define a exact model of user requirements and we instead of it, identify the critical requirements of user and define an approximate model based on these requirements and implement them. Then in runtime of software life, controller part of the software is responsible for adapting the software with user requirements. Controller must identify user requirements from interaction between he/she and software and changes view, model, and even itself (controller) part of software to satisfy new requirements. This work is done with auto code generation that is one of the wishes of software programmers. Other concepts such as linearity and causality can be extended to software world and solve some of the problems there.

IV. Fuzzy MVC Pattern

As mentioned above, in MVC pattern, software is separated into three different layers: view, controller, and model. In order to develop software with Fuzzy Graphical User Interface (FGUI), we must change MVC pattern. These changes are done in view and control layers, but model layer is not changed. SOYPEST is an example of softwares that have FGUI, but it focuses on data parameters and has no any solution for fuzzification of control parameters [8].

A. View Layer

Information that user inserts in this layer, is divided into to two categories: data parameters and control parameters.

1) Data Parameters

Data parameters are related to concepts that the user knows. This knowledge is some times fuzzy, and it is difficult to change it to digital values. For this parameter, several fuzzy labels must be defined and user selects one of them. These labels must be carefully selected to cover user requirements.

For of the most times selecting five labels in the following way will be enough:

1. One label for minimum value.
2. One label for maximum value.
3. One label for middle value.
4. One label for a value between middle and minimum values.
5. One label for a value between middle and maximum values.

For example short, tall, intermediate, rather short, and rather tall labels are good for size definition. If it is necessary define more labels, a slider bar must be created. Minimum and maximum values must be placed in two ends of slider bar. User moves handle of slider bar between these two ends and determines his or her desired values.

2) Control Parameters

These parameters determine behaviors of software. If there is no clear boundary between these behaviors, forcing user to select only one of them is not proper. User is interested in selecting a combination of them. For this, there must be a slider bar for each option, and user by moving the handle, can determine how much this behavior affects output of software. Nothing and Full labels must be placed on two ends of slider bar and Half label on the middle. If the user wants a behavior not to affect output completely, he or she should locate the handle in Nothing label, otherwise he or she locate the handle on Full label. By moving this handle, the effect of this behavior on output becomes more or less. The effect value of behavior is called possibility of behavior. Total sum of possibilities of these behaviors must be one.

B. Controller Layer

The responsibility of this layer is changing received fuzzy parameters from user to crisp parameters that data layer requires. It processes data and control parameters in different ways.

1) Data Parameters Processing

In FGUI, there are several linguistic variables for each data parameter. User selects one of these linguistic variables instead of entering digital values for these parameters. These parameters must be passed to model layer. If model layer is a data base, then this parameter must be translated to fuzzy set of that label. For example suppose that user wants to search the peoples who are rather tall. In control layer “rather tall” label translated to 180-190 range and this range is used in query from data base. If the model layer is the functions library with following conditions:

1. Each function requires one parameter from real type.
2. Result of each function is a real number.

Following steps are done for the selected function:

1. Several samples are selected from fuzzy set of the selected label. Each sample is a real number. Total possibility of samples must be 1.
2. Sum variable is initialized with zero.
3. for each sample

3.1 Function is called with this sample parameter.

3.2 Result of function is multiplied into membership order of sample in the fuzzy set.

3.3 Calculated value is added to the sum variable.

4. At last, value of sum variable is the output of the function.

It is not proper to call function with only sample that has membership order 1. Because what is in the mind of user, is ambiguity and is not an exact value. Other values of fuzzy set are important, corresponding with their membership order. The above algorithm has considered this point.

2) Control Parameter Processing

As mentioned before, user with these parameters customizes behavior of the software. Corresponding to each behavior, there is one option in view layer and one function in model layer. Because several options can be selected, several functions must be executed, and result of each function must affect output of software corresponding to its membership order.

If the output of functions is a number, then for each option, the following steps are done in controller layer:

1. Sum variable is initialized with zero.
2. for each option:

2.1 First membership order of this option must be calculated (Fuzzification [1]). This is done with the help of the handle location of the slider bar and fuzzy set that is defined for it.

2.2 Corresponding function is called.

2.3 Result of function is multiplied to membership order that is computed in step 1.

2.4 Calculated value is added to sum variable.

3. At last, sum variable contains output of the software.

If input parameter and result of functions are a file or data structure, following algorithm is suitable:

1. Sum variable is initialized with zero.
2. for each option:

2.1 First membership order of this option must be calculated (Fuzzification). This is done with the help of the handle location of the slider bar and fuzzy set that is defined for it.

2.2 part of file or data structure is separated with considering to membership order. For example if membership order is .3, then 30% of file is separated and function is called with this parameter.

2.3 result of function that is a file or a data structure is stored.

3. Files or data structures that are produced in previous step, fused with each other to produce output of software.

Fusion must be done in a way that the output remains consistent.

For example, suppose that the user wants to compress a file with help of winzip program. This software has two behaviors. First behavior is file compression with maximum speed and second behavior is file compression file with smallest size of generated archive. GUI of software is changed to let user select output of software with combination of these behaviors. User sets possibility of first behavior 30% and second behavior 70%. Control layer divides input file to two parts. First part is 30% of the total file and second part is 70% of the total file. It gives the first part to first function and the second part to the second function. Then fuses results of these two functions and final archive is generated.

The above algorithms are not complete, and we must refine them corresponding to the program that uses FMVC pattern.

V. CONCLUSION AND NEXT WORK

What is introduced in this paper is an introduction for using ideas of control engineering in software engineering that must be completed in future. It seems that software engineering can uses from control engineering to produce better software. In order to do that, we must redefine concepts like: linearity, causality, robustness and other concepts like these for software and design transition function. This work seems to be one of important parts next step.

ACKNOWLEDGMENT

The authors would also like to thank Mr. Roozbeh Daneshvar for him assistance and edition feedback.

REFERENCES

[1] J. J. Buckly, E. Esfandiary, “An introductions to fuzzy logic and fuzzy set theory”, Pysica-Verlog, Germany, 2002.
[2] K. M. Passino, S. Yurkovich, "Fuzzy Control", Addison Wesly, 1997.
[3] "S. S. Farinwata, D. P. Filew, R. Langari. "Fuzzy Control: Synthesis and Analisys", John & Wiley, 2000.
[4] R. S. Pressman, “Software Engineering”, 5th edition, McGrawHill, Boston, 2001.
[5] I. Sommerville, "Software Engineering", 6th edition, Addison Wesly, 2000
[6] E. Gamma, R. Helm, R. Johnson, J. Vilssides, "Design Patterns", Addison Wesley, 1995.
[7] P. Kuchana, "Software Architecture Design Patterns in java", Auerbach, 2004.
[8] http://www.ncst.ernet.in/kbcs/vivek/issues/10.4/soypest/node3.html

Rasool Karimi was born in Tehran, Iran in 1980. He receives the B.S. degree in computer engineering from university of Azad, Tehran, Iran in 2003. Now he is studying in university of Tehran, in field of AI & Robotics. His main research interest is agent based manufacturing.

Caro Lucas: He received the Ms.c degree from the University of Tehran, Iran, in 1973, and the Ph.D degree from the university of California, Berkeley, in 1976. He is a professor in the Department Of Electrical and Computer Engineering, University of Tehran, Iran, as well as researcher at the School of Intelligent Systems (SIS), Institute for studies in Theoretical Physics and Mathematics (IPM), Tehran, Iran. He has served as the Director of SIS (1993-1997), Chairman of the ECE Department at the University of Tehran (1986-1988), Managing Editor of the Memories of the Engineering Faculty, University of Tehran (1979-1991), Reviewer of Mathematical Reviewers (since 1987), Associate Editor of Journal of Intelligent and Fuzzy systems (1992-1999) , and Chairman of the IEEE, Iran section (1990-1992). He was also a Visiting Associate Professor at the University of Toronto (summer, 1989-1990), University of California, Berkeley (1988-1989), an Assistant Professor at Garyounis University (1984-1985), University of California, Los Angeles (1975-1976), a Senior Researcher at the International Center for Theoretical Physics and the International Center for Genetic Engineering and Biotechnology, both in Trieste, Italy, the Institute of Applied Mathematics Chinese Academy of Sciences, Harbin Institute of Electrical Technology, a Research Associate at the Manufacturing Research Corporation of Ontario, and a Research Assistant at the Electronic Research Laboratory, University of California, Berkeley. He is the holder of Patent on “Speaker Independent Farsi Isolated Word Neurorecognizer”. His research interests include biological computing, computational intelligence, uncertain systems, intelligent control, neural networks, multiagent systems, data mining, business intelligence, financial modeling and knowledge management. Professor Lucas has served as the chairman of several International Conferences. He was the founder of the SIS and has assisted in founding several new research organizations and engineering disciplines in Iran. He is
the recipient of several research grants at the University of Tehran and SIS.