Safety Relevant Design of a Fuzzy Controller

Gerhard H. Schildt
Senior member of IEEE
Daniela Kahn
Institute for Automation;Vienna University of Technology
Treitlstr. 1/183-1, A-1040 Vienna, Austria
Tel.: 0043-1-58801-18311, Fax: 0043-1-58801-18391

Keywords: Safety terms, , fail-safe system design, V&V aspects, fundamental safety principles

Abstract. After an introduction into safety terms a fuzzy controller for safety related process control will be presented. One can show that the size of necessary rules is relatively small. Thus, there exists a real chance for verification and validation of software due to the fact that the whole software can be structured into standard fuzzy software (like fuzzyfication, inference algorithms, and defuzzyfication), real-time operating system software, and the contents of the rule base. Furthermore, there is an excellent advantage due to real-time behavior, because program execution time is much more planable than for conventional PID-controller software. Additionally, up to now special know-how does exist to prove stability of fuzzy controller. Hardware design has been done due to fundamental principles of safety technique like watch dog function, dynamization principle, and quiescent current principle.

1. Introduction.

For vital prozess control safety critical devices and control systems are used. Up to now there is a certain threshold to apply software driven systems, because software will never be error-free. But, nevertheless all over the world engineers are looking for new approaches to use software driven systems for vital process control. To describe safety critical systems at first some terms of safety technique shall be introduced:

safety critical system: control system causing no hazard to people of material in case of environmental influence or system failure.
safety: property of an item to cause no hazard under given conditions during a given time; i.e. avoidance of undue fail conditions. (e.g. Undue fail conditions may be caused by technical system failures or malfunction of an electronic device interfered by electromagnetic noise).
hazard: state of a system that cannot be controlled by given means and may lead to damages to persons.
safe system state: property of a system state to cause no hazard to people or material
fail-safe: technical failures within an item may lead to fail states, which however have to be safe.

Because, up to now no fail-safe one-channel computer for vital process control is available, one has to choose a configuration of at least two computers running parallely. In this system configuration results of both channels are to be fed to a fail-safe comparator, whose output enables a safe gate in case of equivalent results, represented by corresponding command telegrams to be fed to the technical process. Because of availability aspects one normally applies a three-channelled system with (2 of 3)-voter, so that the system configuration normally runs with all three channels parallely, in case of failure or maintenance of one channel a *degraded mode of operation* is possible.

Additionally, *diversity principle* is applied in the field of vital process control. Diversity may be defined as follows:

"Existence of different means to perform a required function"

(e.g. different physical principles, different approaches to implement the same task, different algorithms) /1/.

Basically, there are typical problems due to diverse system design like

sufficient diversification within a n-version system (to be proved)
unplanable waiting times for results/command telegrams that are to be compared
certain tolerance zone management when comparing measured values, results, or command telegrams.

Therefore, it is an essential challenge to design a one-channelled software system running on a three-channelled hardware in order to manage hardware failures or electromagnetic interference.

2. Fuzzy Controller

We designed a fuzzy controller with an architecture as displayed in Fig. 1. At the input side, there is a *condition interface* producing *fuzzy equivalents* of several input variables. They are then fed to an inference engine cooperating with a rule base. The outputs from the *inference engine* are *fuzzy results* provided to an *action interface*, which finally performs defuzzyfication and process actuation.

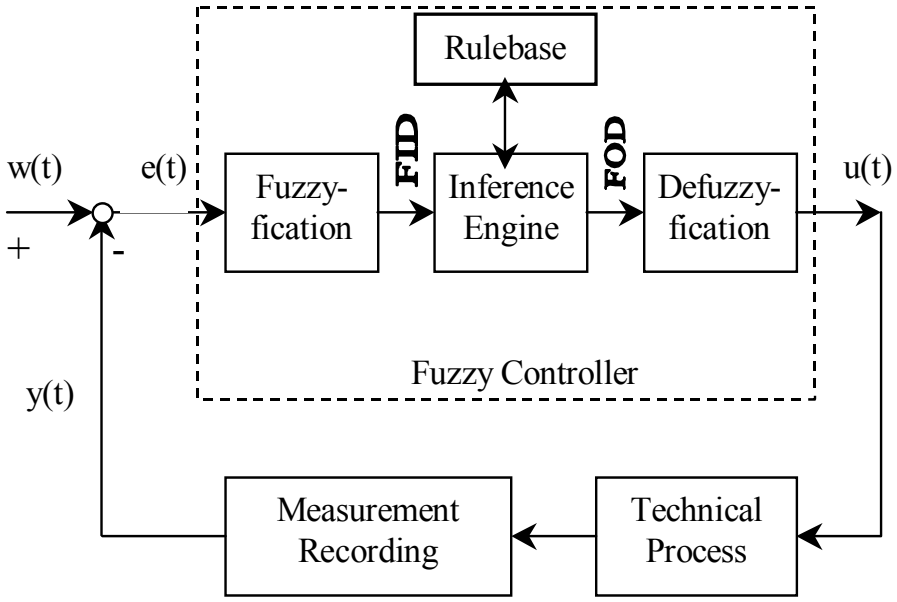

Fig.1 Fuzzy Controller

Analogue input signals, such as temperature, pressure, water level and so on are provided to the controller, however not directly, but as an *error* and *deviation of error*.

2.1 Fuzzyfication process

The variables error and deviation of error are then fed to the condition interface, whose function is to fuzzify the input values. Since mappings from these input data to values of fuzzy variables can be freely selected, we choosed triangular membership functions because of simple description in a ROM (Figure 2).

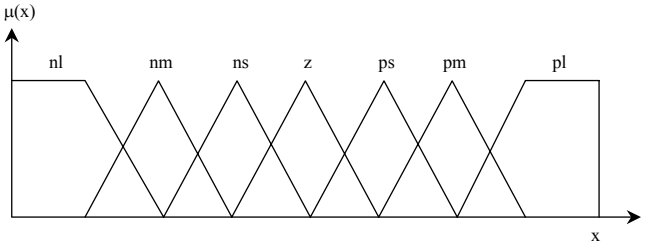

Fig.2: Membership Funktion

If one uses simple triangles, they are easy to describe in source code. For example , for P_LARGE the description can be written as (@0.6, 0, @1.0, 1, @1.4, 0), and for ZERO the description can be written as (@-0.3, 0, @0.0, 1, @0.3, 0), and so on. Note that all values of variables here are normalized into the range of /-1,1/ or /0,1/.

2.2 Inference engine

The main component of the controller is the *inference engine.* It is operated under a strictly cyclic regime, such as in a *programmable logic controller* (PLC). In contrast to the latter, however, each loop execution takes exactly the same time, because the same operations are carried out in every iteration. Thus, the controller´s real-time behaviour is fully deterministic and easily predictable. Every loop execution comprises three steps: (1) input data generation by analogue-to-digital conversion in the condition interface, (2) inference by determing appropriate control rules and (3) control actuation via digital-to-analogue converters in the action interface. These steps as well as the overall operation cycle are strictly synchronized with a system clock.

2.3 Rule base

The rule base contains a set of rules R_1, R_2, ..., R_n. These rules form the *expert knowledge* how to control the technical process and can be described as follows:

R_k : IF $p_k(e)$ THEN $c_k(u)$

with p_k premise
c_k conclusion
e error
u output value

Necessary adaptation of fuzzy controller towards technical process may be done by modifying the contents of rule base. Additionally, there is a real good chance of *tuning* the real-time behaviour of the controller by modifying defined membership functions. There is an essential advantage of applying a fuzzy controller because of its

planable real-time behaviour. Instead of calculation of high sophisticated differential equations some rules may fire. Due to a certain strategy conclusions can be derived easily.

Another essential advantage is the transparency of rule base, so that even an expert without any computer science is able to validate the rule set.

The rule set should be implemented as ROM or PROM for safety reasons. A special development platform was used to generate a *definition section* and *rule base section* /2/. Figure 3 shows both sections for an example of a combined temperature-steam pressure controller.

FIU Source Code

```
$ FILENAME:       temp/temp3.fil
$ DATE:                 09/18/1998
$ UPDATE:         09/23/1998

$ Temperature controller: Three inputs, two outputs
$       INPUT(S):               Error, Var(iationOf)_Error, Pressure
$       OUTPUT(S):  Var(iationOf)_Heater, Var(iationOf)_Cooling(Valve)


$ FIU HEADER
fiu tvfi (min max)*8;


$ DEFINITION OF INPUT VARIABLE(S)
invar Error " " : -1.0 () 1.0 [
        P_Large                 (@0.6, 0, @1.0, 1)
        P_Medium                (@0.3, 0, @0.6, 1, @1.0, 0)
        P_Small                 (@0.0, 0, @0.3, 1, @0.6, 0)
        Zero                    (@-0.3,0, @0.0, 1, @0.3, 0)
        N_Small                 (@-0.6,0, @-0.3,1, @0.0, 0)
        N_Medium                (@-1.0,0, @-0.6,1, @-0.3,0)
        N_Large                 (@-1.0,1, @-0.6,0)

        ];
invar Var_Error " " : -1.0 () 1.0 [
        P_Large                 (@0.6, 0, @1.0, 1)
        P_Medium                (@0.3, 0, @0.6, 1, @1.0, 0)
        P_Small                 (@0.0, 0, @0.3, 1, @0.6, 0)
        Zero                    (@-0.3,0, @0.0, 1, @0.3, 0)
        N_Small                 (@-0.6,0, @-0.3,1, @0.0, 0)
        N_Medium                (@-1.0,0, @-0.6,1, @-0.3,0)
        N_Large                 (@-1.0,1, @-0.6,0)

        ];
invar Pressure " " : 0.0 () 1.0 [
        Large                   (@0.5, 0, @1.0, 1)
        Medium                  (@0.0, 0, @0.5, 1, @1.0, 0)
        Small                   (@0.0, 1, @0.5, 0)
        ];


$ DEFINITION OF OUTPUT VARIABLE(S)
outvar Var_Heater " " : -1.0 () 1.0 * (
        P_Large                 =  0.8,
        P_Medium                =  0.4
```

```
        P_Small         =  0.2
        Zero            =  0.0
        N_Small         = -0.2
        N_Medium        = -0.4
        N_Large         = -0.8

        );
outvar Var_Cooling " " : -1.0 () 1.0 * (
        P_Large         =  0.8,
        P_Medium        =  0.4
        P_Small         =  0.2
        Zero            =  0.0
        N_Small         = -0.2
        N_Medium        = -0.4
        N_Large         = -0.8
        );
```

$ RULES

```
if Error is P_Small     and Var_Error is N_Large     and Pressure is Large     then Var_Cooling is Zero;
if Error is P_Small     and Var_Error is N_Medium    and Pressure is Large     then Var_Heater is H_Medium;
if Error is P_Small     and Var_Error is N_Medium    and Pressure is Large     then Var_Cooling is Zero;
if Error is P_Small     and Var_Error is N_Small     and Pressure is Large     then Var_Heater is N_Small;
if Error is P_Small     and Var_Error is N_Small     and Pressure is Large     then Var_Cooling is Zero;
if Error is P_Small     and Var_Error is N_Large     and Pressure is Medium    then Var_Heater is N_Small
if Error is P_Small     and Var_Error is N_Large     and Pressure is Medium    then Var_Cooling is Zero;
if Error is P_Small     and Var_Error is N_Medium    and Pressure is Medium    then Var_Heater is N_Small;
if Error is P_Small     and Var_Error is N_Medium    and Pressure is Medium    then Var_Cooling is Zero;
if Error is P_Small     and Var_Error is N_Small     and Pressure is Medium    then Var_Heater is Zero;
if Error is P_Small     and Var_Error is N_Small     and Pressure is Medium    then Var_Cooling is Zero;
if Error is P_Small     and Var_Error is N_Large     and Pressure is Small     then Var_Heater is Zero;
if Error is P_Small     and Var_Error is N_Large     and Pressure is Small     then Var_Cooling is Zero;
if Error is P_Small     and Var_Error is N_Medium    and Pressure is Small     then Var_Heater is Zero;

                                                     :

                                                     :

if Error is P_Large     and Var_Error is P_Medium    and Pressure is Large     then Var_Coolings is N_Large;
if Error is P_Large     and Var_Error is P_Small     and Pressure is Large     then Var_Heater is P_Medium;
if Error is P_Large     and Var_Error is P_Small     and Pressure is Large     then Var_Coolings is N_Large;
if Error is P_Large     and Var_Error is P_Large     and Pressure is Medium    then Var_Heater is P_Large;
if Error is P_Large     and Var_Error is P_Large     and Pressure is Medium    then Var_Coolings is N_Large;
if Error is P_Large     and Var_Error is P_Medium    and Pressure is Medium    then Var_Heater is P_Large;
if Error is P_Large     and Var_Error is P_Medium    and Pressure is Medium    then Var_Coolings is N_Large;
if Error is P_Large     and Var_Error is P_Small     and Pressure is Medium    then Var_Heater is P_Large;
if Error is P_Large     and Var_Error is P_Small     and Pressure is Medium    then Var_Coolings is N_Large;
if Error is P_Large     and Var_Error is P_Large     and Pressure is Small     then Var_Heater is P_Large;
if Error is P_Large     and Var_Error is P_Large     and Pressure is Small     then Var_Coolings is N_Large;
if Error is P_Large     and Var_Error is P_Medium    and Pressure is Small     then Var_Heater is P_Large;
if Error is P_Large     and Var_Error is P_Medium    and Pressure is Small     then Var_Coolings is N_Large;
if Error is P_Large     and Var_Error is P_Small     and Pressure is Small     then Var_Heater is P_Large;
                        and Var_Error is P_Small     and Pressure is Small     then Var_Coolings is N_Large;
```

Figure 3: Definition section and rule base section

We found out, that for sufficient process control the size of rule base is limited

to a bounded number of rules (e.g. 86 rules only). This is an essential advantage for V&V process.

2.4 Defuzzyfication Process

Because an actuator needs a discrete value for operation, a certain defuzzyfication strategy has to be applied. Usual methods of defuzzyfication are

MAX_HEIGHT
MEAN_OF_MAXIMA
CENTER_OF_GRAVITY

We decided to implement the *center of gravitiy strategy* because of its efficient control behaviour /3/.

3. Safety Features of Fuzzy Controller

In safety technique there exist certain fundamental principles like *dynamization principle, monitoring function, watchdog function, quiescent current principle.*

So, we decided to implement these principles into our design for a safety-critical fuzzy controller. After input values have been transformated to fuzzy values by *fuzzyfication process* certain scanner checks, if one or more rules fire. In case of an unplanned stop of the scanner the *dynamic monitoring component* consisting of a very simple and passive hardware disables a safe gate so that no command telegrams are fed to the technical process (figure 4).

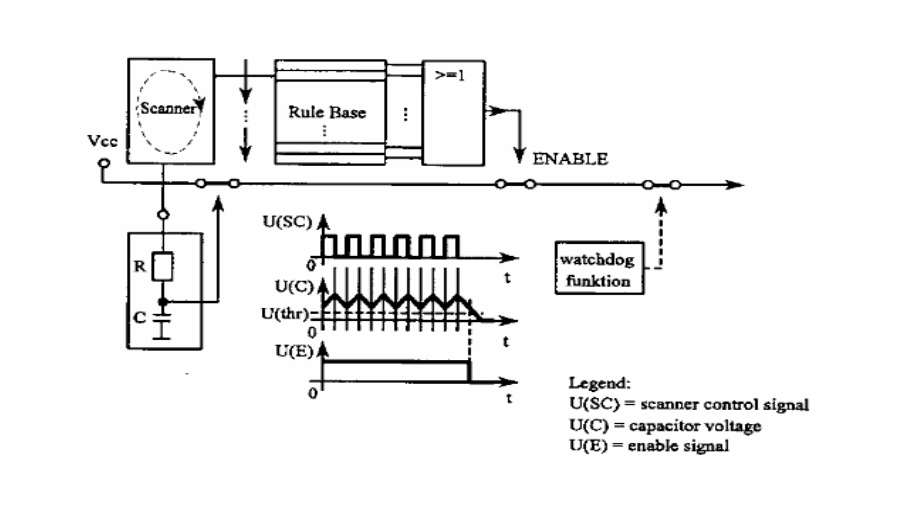

Figure 4: Safety components for a fuzzy controller

An additional *watch dog function module* disables a safe gate due to stopped scanner or because no rule has fired at all after a well-defined time interval. Thus, the technical process changes over to a well-defined safe system state *(shutdown)*. As much as possible detailled functions have to be implemented in simple and passive hardware because of V & V reasons.

4. V & V aspects

Because it is not possible to implement the whole functionality in hardware a one-channelled software remains that has to be validated. Figure 5 shows the necessary software structure comprising the *Operating System Software,* the so-called *Standard Fuzzy Software,* and a *Rule Base.*Operating system software and standard fuzzy software have to be validated only once, however for each new application one needs a V & V licensing process for the rule base.

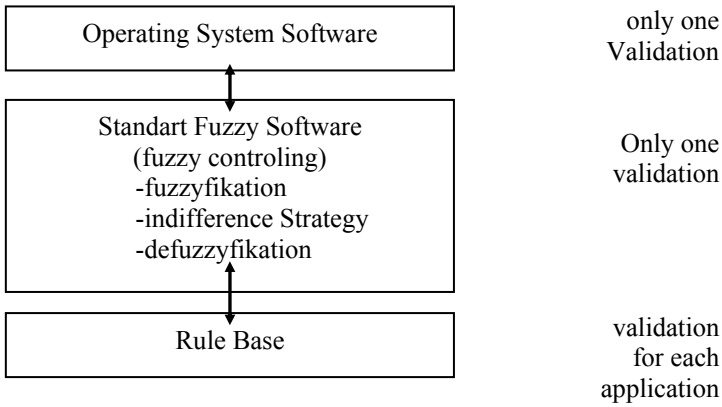

Figure 5: Software structure of safety critical fuzzy controller

5. Conclusions

A fuzzy controller for safety critical process control was described. We implemented fundamental principles of safety technique like *dynamization principle, monitoring function* and *watch dog function* into a special fuzzy controller design. Hardware and software aspects have been discussed. We recognize not only better chances for V & V process, but also a better real-time behaviour of such a knowledge based system. Up to now some theoretical knowledge for *stability proof* is available so that we see a real good chance for applying a fuzzy controller in the field of safety critical process control.

References:

/1/ G.H. Schildt: "On Diverse Programming for Vital Systems", IFAC - Proceedings on Safety of Computer Control Systems, 1989

/2/ APTRONIX INC.: “FIDE Application Note 006-980914”, San Jose, CA, USA 1998

/3/ G.H.Schildt, W. Kastner: „Prozessautomatisierung“, Springer-Verlag Wien New York, 1998