Sicherheitsrelevanter Fuzzy Controller

Gerhard H. Schildt
Senior member of IEEE
Daniela Kahn
Institut für Rechnergestützte Automation
Technische Universität Wien
Treitlstr. 1/183-1, A-1040 Wien, Austria
e-mail: schi@auto.tuwien.ac.at
dani@auto.tuwien.ac.at

Schlüsselwörter: Sicherheitstechnische Begriffe, Fail-safe-Entwurfsprinzipien, V&V Aspekte, grundsätzliche Sicherheitsprinzipien, Fuzzy Logik für Controller

Abstract: Nach einer Einführung in sicherheitstechnische Begriffe wird ein Fuzzy Controller für sicherheitsrelevante Aufgaben der Prozesssteuerung vorgestellt. Es lässt sich zeigen, dass der erforderliche Umfang der Regelbasis relativ gering ist, um eine zufriedenstellende Arbeitsweise des Controllers sicherzustellen. Die Besonderheit dieses Entwurfes besteht darin, dass sich der Inhalt der Regelbasis für eine Validierung besonders gut eignet, da erfahrene Prozessoperateure (Anlagenfahrer), die über keine vertieften Kenntnisse der Informatik verfügen, durchaus in der Lage sind, den Inhalt der Regelbasis in ihrer Eigenschaft als Experten zu validieren. Für die überigen Softwarekomponenten des Fuzzy-Controllers (Fuzzyfizierung, Inferenzalgorithmus, Defuzzyfizierung sowie Echtzeitbetriebssystemsoftware) ist eine sog. Standard-Fuzzy-Software zu entwickeln und einmalig zu validieren. Darüberhinaus ergibt sich bei einem Fuzzy Controller ein recht gutes Echtzeitverhalten im Vergleich zum herkömmlichen PID-Controller. Der Entwurf ist weiter dadurch gekennzeichnet, dass grundsätzliche sicherheitstechnische Prinzipien implementiert wurden, so z.B. eine taktgesteuerte Überwachung des Scanners, ob dieser deterministisch die Regelbasis bearbeitet. Sollte der Scannvorgang „hängen bleiben“, geht das Gesamtsystem durch eine Watchdog-Funktion in den sicheren Systemzustand über

1. Einführung

Für sicherheitsrelevante Aufgaben der Prozeßsteuerung sind validierbare Hard- und Softwaresysteme erforderlich. Bisher bestehen noch erhebliche Bedenken, für sicherheitsrelevante Prozesse softwarebasierte Systeme einzusetzen, da der Grundsatz gilt, dass Software niemals fehlerfrei sein wird (*„software will never be error-free“* [GRA01]. Dennoch suchen überall Entwickler nach neuen Wegen, um auch softwarebasierte Systeme zur Steuerung sicherheitsrelevanter Systeme einzusetzen. Zunächst sollen einige sicherheitstechnische Begriffe eingeführt werden
Sicherheitsrelevantes System = ein System, das keine Gefährdung von Menschen und

Material im Fall von Umwelteinflüssen oder systembedingten Ausfällen bewirkt.

Technische Sicherheit = Eigenschaft einer Komponente, die unter gegebenen Umständen für eine vorgegebene Zeit keine Gefährdung von Menschen und Material verursacht (solche Fehlzustände können durch technisch bedingte Ausfälle oder Fehlfunktionen elektronischer Einrichtungen z.B. durch elektromagnetische Störbeeinflussung bewirkt werden).

Gefährdung: Zustand eines Systems, der nicht mehr mit den im System gegebenen Mitteln beherrschbar ist und zu Personen- oder Sachschäden führen kann.

Sicherer Systemzustand: ein Systemzustand, aus dem heraus keine Gefährdung für Personen und Sachen entstehen kann, und der von selbst nicht mehr verlassen werden kann.

Fail-safe : technische Ausfälle innerhalb einer Systemkomponente dürfen zu Fehlzuständen führen (*„fail“*), die jedoch sicher sein müssen (*„safe“*).

Da es bis heute keinen einkanaligen fail-safe arbeitenden Computer für sicherheitsrelevante Prozesssteuerungen gibt, kann man stattdessen einen zweikanaligen Mikrocomputer (SIMIS) einsetzen, der zwar gegenüber Hardwareausfällen gesichert ist, nicht jedoch gegenüber Softwarefehlern. Um auch diese Fehlzustände zu vermeiden, kann man das Diversitätsprinzip (redundante Software; engl.: *n-version programming*) anwenden, wobei das Diversitätsprinzip bedeutet, dass eine gewünschte Funktion auf unterschiedlichen Entwicklungswegen realisiert wird (engl.: *"Existence of different means to perform a required function"*) [BIS86]. Dennoch ist festzustellen, dass man bei einem diversitären Systementwurf immer noch mit erheblichen Problemstellungen konfrontiert wird:

1. Es muß sichergestellt sein, daß hinreichende Diversifizierung vorliegt (was nicht anderes bedeutet als ein „inverser“ Sicherheitsnachweis).

2. Es treten nicht-planbare Wartezeiten für miteinander zu vergleichender Resultate auf. Das ist gerade für Echtzeitanwendungen ein entscheidender Nachteil !

3. Bei diversitärem Systementwurf benötigt man einen Vergleicher für zu vergleichende Resultate, der jedoch nicht nur ein einfacher Vergleicher ist, sondern auch noch über die Eigenschaft einer Toleranzfensterbearbeitung für zu vergleichende Resultate verfügen muß (engl.: *tolerance zone management*). Außerdem ist diese Eigenschaft auch noch fail-safe zu realisieren.

Angesichts solcher Problemstellungen ist ein einkanaliger Systementwurf für eine sicherheitsrelevante Prozesssteuerung eher anzustreben, der die Möglichkeit einer

Validierung und Verifikation bietet (engl.: *V & V process*). Hierzu bietet sich ein Controller-Konzept auf der Basis der Fuzzy-Logik an, da ein fuzzy-basierter Controller ein regelbasiertes System ist, dessen Regelbasis inhaltlich transparent und von daher leichter prüfbar ist. Daher wird im folgenden ein solcher Controller vorgestellt.

2. Fuzzy Controller

Es wurde ein Fuzzy Controller entsprechend Abb.1 entworfen. Eingangsgröße des Reglers sind wie beim herkömmlichen PID-Regler die Regelabweichung e(t) sowie die Änderungsgeschwindigkeit der Regelabweichung de(t)/dt, die jedoch anschließend durch eine Fuzzyfizierungskomponente über Zugehörigkeitsfunktionen (engl.: *membership function*) in Fuzzy-Äquivalente umgesetzt werden. Die Inferenzkomponente enthält eine implementierte Strategie, nach der vorzugehen ist, wenn eine oder mehrere Regeln aus der Regelbasis zutreffen (*„feuern“*). Dazu wird die Regelbasis (engl.: *rule base*) abgescannt, um festzustellen, welche Regeln feuern. Die Inferenzkomponente erstellt daraus ein fuzzyfiziertes Ergebnis (engl.: *fuzzy result*), das jedoch in dieser Form als Stellgröße für den technischen Prozeß ungeeignet ist. Es ist daher durch eine sog. Defuzzyfizierungskomponente wieder in einen scharfen Wert (engl.: *crisp value*) umzuwandeln. Dabei ergibt sich eine analoge Stellgröße, die dem technischen Prozeß zugeführt wird.

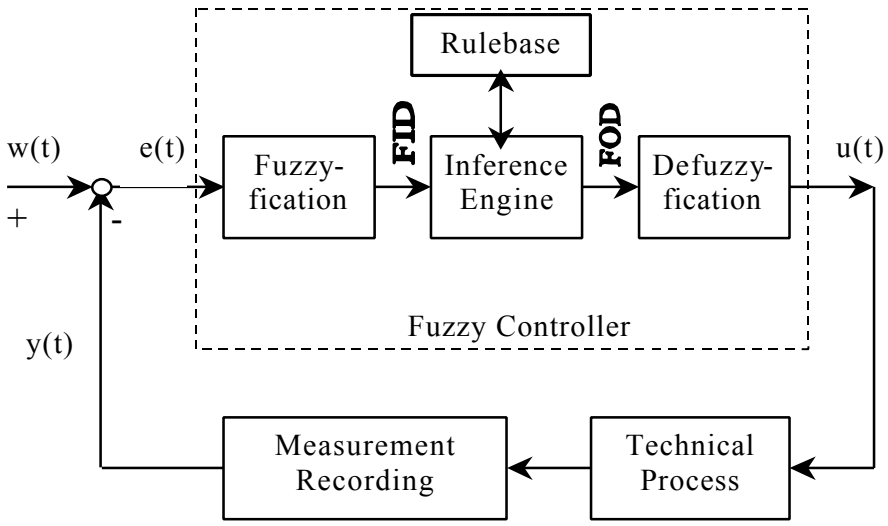

Abb.1: Fuzzy Controller

2.1 Fuzzyfizierungskomponente

Eingangsgrößen des Fuzzy-Controllers sind üblicherweise die Regelabweichung e(t) sowie die Änderungsgeschwindigkeit der Regelabweichung de(t)/dt. Diese wert- und

zeitkontinuierlichen Größen werden nun über Zugehörigkeitsfunktionen (engl.: *membership functions*) in Fuzzy-Äquivalente umgesetzt. Dabei ist man in der Wahl der Ausgestaltung der Zugehörigkeitsfunktionen nicht beschränkt; ohne Einschränkung der Funktionalität kann man z.B. dreiecksförmige Zugehörigkeitsfunktionen auswählen. Wählt man nämlich einfache Dreiecksfunktionen aus, so sind diese einfach im Sourcecode zu beschreiben: Z.B. kann eine Zugehörigkeitsfunktion für einen hohen Wert der Regelabweichung einfach beschrieben werden als (@0.6, 0, @1.0, 1, @1.4, 0) oder die Zugehörigkeitsfunktion für verschwindend kleine Werte der Regelabweichung als (@-0.3, 0, @0.0, 1, @0.3, 0) als normalisierte Beschreibung. Das bietet zugleich den Vorteil der einfachen Abspeicherung im Rechner, indem man für jede Zugehörigkeitsfunktion allein drei Wertepaare hinterlegt, zwischen denen dann linear interpoliert werden kann. Dadurch entsteht eine relativ einfache Software für die Fuzzyfizierungskomponente. Abb. 2 zeigt ein Fuzzy-Set mit den zugehörigen Zugehörigkeitsfunktionen.

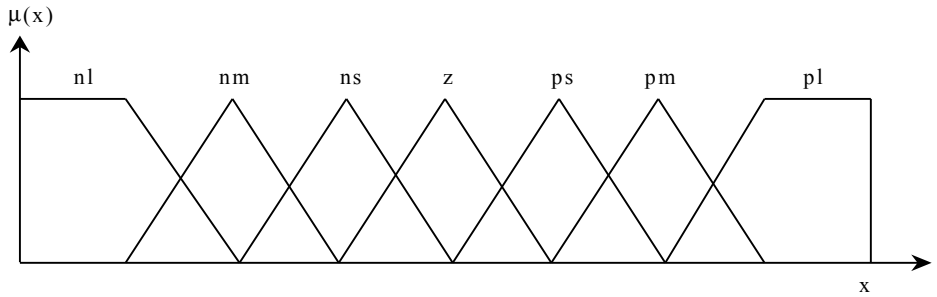

Abb.2: Fuzzy set mit den Zugehörigkeitsfunktionen

2.2 Inferenzkomponente

Die zentrale Komponente des Fuzzy-Controllers ist die Inferenzkomponente (engl.: *inference engine*). In ihr ist die Vorgehensweise hinterlegt, wie zu verfahren ist, wenn eine oder benachbarte Regeln aus der Regelbasis feuern. Will man den Fuzzy Controller für sicherheitsrelevante Zwecke entwerfen, so muß sichergestellt sein, dass die Regelbasis zyklisch abgeprüft („gescannt“) wird und dieser Vorgang nicht „hängen bleibt“, da sonst der technische Prozeß nicht mehr echtzeitgemäß geführt wird. Dabei benötigt dieser Scanner-Vorgang stets dieselbe Zeit, wodurch das Echtzeitverhalten vorhersagbar wird. Dieser Vorgang wird von einem zentralen Taktgenerator synchronisiert. Außerdem ist durch eine geeignete Schaltung festzustellen, ob bei dem Scann-Vorgang überhaupt eine oder mehrere Regeln gefeuert haben.

2.3 Regelbasis

Die Regelbasis enthält einen Satz von Regeln $R_1, R_{2,} \ldots\ldots R_n$. Die Menge dieser Regeln bildet die Regelbasis und damit ein Abbild der Verhaltensbeschreibung eines technischen Prozesses. Solche wissensbasierten Beschreibungen werden vor allem bei solchen technischen Prozessen genutzt, bei welchen eine geschlossene Beschreibung mit Differentialgleichungen als mathematisches Modell praktisch nicht möglich ist.

Damit kann also das Wissen eines Experten für die Steuerung eines technischen Prozesses nachgebildet werden.
Ein wesentliches Merkmal solcher regelbasierten Systeme ist aus sicherheitstechnischer Sicht noch die Transparenz der Regeln: Sie bestehen nämlich aus *IF ... THEN ... ELSE – Konstrukten* der folgenden Form:

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

mit p_k Prämisse
c_k Konklusion
e Regelabweichung
u Stellgröße

Weiter lässt sich diese Struktur noch erweitern, indem man z.B. Regel folgendermaßen strukturiert:

R_k : IF error is large AND deviation of error is middle THEN conclusion

Dabei ergibt sich eine weitere Möglichkeit, den Fuzzy Controller möglichst optimal an die Anforderungen des technischen Prozesses anzupassen, indem man durch einfaches Umeditieren der Regelbasis das Verhalten des Controllers bei relativ geringem Aufwand dem technischen Prozess anpassen kann. Eine weitere Maßnahme des *tunings* ist dadurch gegeben, dass man bei Fuzzyfizierung die Fuzzy sets bestehend aus Zugehörigkeitsfunktionen modifiziert. Auf diese Weise kann ebenfalls ein geplantes Echtzeitverhalten erreicht werden. Statt komplexe Differentialgleichungen für die Applikation zu lösen, wird die Wissensbasis genutzt. Hier bietet sich die Möglichkeit, ein entsprechendes Programm möglichst einfach zu schreiben und darüber hinaus hat man die Möglichkeit, den Inhalt der Wissensbasis von einem Experten auf Richtigkeit (und Zeitigkeit) prüfen zu lassen. Der besondere Vorteil liegt darin, dass ein Operator („Anlagenfahrer“) ohne informatisches Fachwissen als Experte am besten die Richtigkeit der Wissensbasis beurteilen kann, besser noch als ein Informatiker, der die Handlungen des Operators in informatische Aussagen umzusetzen bemüht ist und dabei eher Fehler machen kann sowohl hinsichtlich der Vollständigkeit als auch der Widerspruchsfreiheit in der Wissensbasis.
Der Inhalt der Regelbasis soll für sicherheitsrelevante Anwendungen in einem ROM gespeichert werden. Im folgenden wird ein Beispiel für den Aufbau einer solchen Regelbasis für einen thermischen Reaktor mit Heizungs- und Kühlungseinrichtung gezeigt [APT98].

FIU Source Code

```
$ FILENAME:       temp/temp3.fil
$ DATE:                 09/18/2001
$ 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 N_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             and Pressure is Large     then Var_Heater is P_Medium;
if Error is P_Large     P_Medium                     and Pressure is Large     then Var_Cooling is N_Large;
if Error is P_Large     and Var_Error is             and Pressure is Large     then Var_Heater is P_Medium;
if Error is P_Large     P_Medium                     and Pressure isLarge      then Var_Cooling 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_Cooling 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_Cooling is N_Large;
if Error is P_Large     and Var_Error is             and Pressure is Medium    then Var_Heater is P_Large;
if Error is P_Large     P_Medium                     and Pressure is Medium    then Var_Cooling is N_Large;
if Error is P_Large     and Var_Error is             and Pressure is Small     then Var_Heater is P_Large;
if Error is P_Large     P_Medium                     and Pressure is Small     then Var_Cooling is N_Large;
if Error is P_Large     and Var_Error is P_Small     and Pressure is Small     then Var_Heater is P_Large;
if Error is P_Large     and Var_Error is P_Small     and Pressure is Small     then Var_Cooling 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_Cooling is N_Large
                        and Var_Error is
                        P_Medium
end                     and Var_Error is
                        P_Medium
                        and Var_Error is P_Small
                        and Var_Error is P_Small
```

Abb. 3: Definitionsbereich und Regelbasis

Bei der Implementierung solcher fuzzy-basierter Regelungen findet man, dass die Anzahl der erforderlichen Regeln zur Regelung eines technischen Prozesses durchaus nicht über alle Grenzen wächst sondern eher begrenzt ist. Erfahrungen mit dem vorgenannten Beispiel haben gezeigt, dass eine befriedigende Prozessbeschreibung in diesem Fall schon mit insgesamt 86 Regeln möglich ist. Damit ist auch der Aufwand für eine Validierung der Regelbasis überschaubar.

2.4 Defuzzyfizierung

Der technische Prozess benötigt eine analoge Stellgröße, dagegen liefert die Inferenzkomponente ein fuzzyfiziertes Ergebnis (engl.: *fuzzy result*). Daher ist eine entsprechende Umsetzung des unscharfen Ergebnisses in eine analoge Stellgröße unabdingbar. Diese Umsetzung wird durch die Defuzzyfizierungskomponente bewirkt. Übliche Methoden der Defuzzyfizierung sind:

MAX_HEIGHT (Maximum der Zugehörigkeitsfunktion der Stellgröße)
MEAN_OF_MAXIMA (Mittelwert der Maxima)
CENTER_OF_GRAVITY (Schwerpunktskoordinate der Zugehörigkeits-funktion der Stellgröße).

Die besten Ergebnisse bei der Regelung technischer Prozesse wurden bisher mit der dritten Methode erreicht, wobei sich die Stellgröße u für den technischen Prozess aus der Schwerpunktskoordinate der Zugehörigkeitsfunktion der Stellgröße bestimmt [SCH98].

3. Sicherheitsaspekte des Fuzzy Controllers

In der Sicherheitstechnik existieren folgende grundlegende Sicherungsverfahren:

- das *Dynamisierungsprinzip* von Signalen
- ein *Überwachungskonzept*
- *Watchdog Funktion* und das
- *Ruhestromprinzip.*

In das vorliegende Konzept des Fuzzy Controllers wurden diese Konzepte wie folgt integriert: Nachdem die Eingangsvariablen e(t) und de(t)/dt fuzzyfiziert worden sind, überprüft eine Hardwareschaltung, ob der Scann-Vorgang periodisch abläuft. Aus dem Abtastvorgang wird eine rechteckig verlaufende Kontrollspannung abgeleitet (*Dynamisierungsprinzip*), solange der Scann-Vorgang läuft. Über einen R-/C-Tiefpaß entsteht eine annähernd gleichgerichtete Kontrollspannung, die einen Schwellwertschalter ansteuert.
Parallel dazu entscheidet ein ODER-Gatter, ob mindestens eine Regel aus der Regelbasis gefeuert hat.
Das Ausgangssignal des Schwellspannungsschalters und das Ausgangssignal des ODER-Gatters werden durch Konjunktion miteinander verknüpft.
Solange das UND-Gatter eine log.“1“ liefert, wird eine Watchdog-Schaltung im Zustand log.“1“ gehalten (*watchdog function*). Ist die UND –Funktion nicht erfüllt, so trennt die Watchdog-Schaltung nach einer voreingestellten Zeit die generierte Stellgröße u vom technischen Prozeß (*Ruhestromprinzip*), wodurch der technische Prozeß automatisch in den sicheren Systemzustand überführt wird (engl.: *shutdown*).
Bei der Realisierung der erforderlichen Hardware ist zu beachten, dass weitestgehend passive elektronische Schaltungen Verwendung finden, um die Führung eines Sicherheitsnachweises zu ermöglichen. Grundsätzlich gilt, dass eine spezielle, passive Hardware stets einfacher zu validieren ist als Software (Abb. 4).

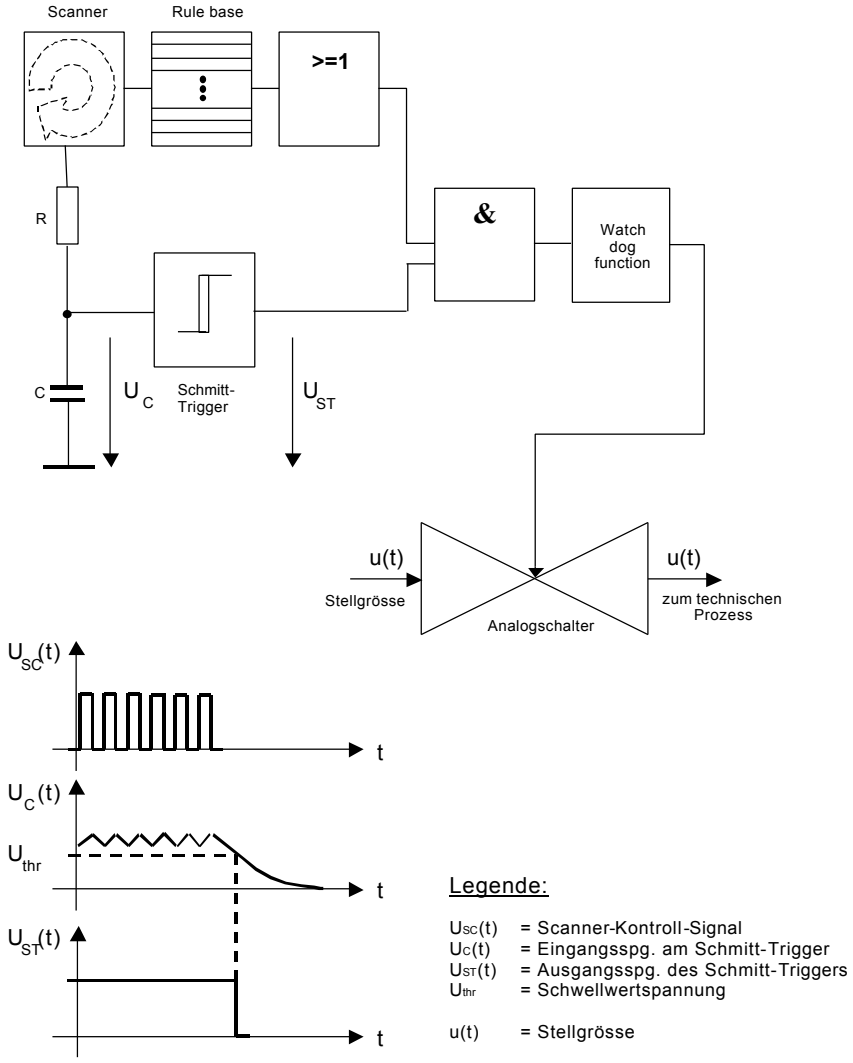

Abb. 4 : Schaltung des sicherheitsrelevanten Fuzzy Controllers

4. V & V-Aspekte

Da es nicht möglich ist, die gesamte Funktionalität eines sicherheitsrelevanten Fuzzy Controllers in Hardware zu realisieren, müssen verbleibende Softwarekomponenten validiert werden. Abb. 5 zeigt die Softwarestruktur des Fuzzy Controllers bestehend aus Betriebssystemsoftware, Standard Fuzzy Software für die Fuzzyfizierung, Inferenz und Defuzzyfizierung sowie die Regelbasis. Betriebssystemsoftware und Standard Fuzzy

Software bedürfen nur einer einmaligen Validierung, dagegen muß der Inhalt der Regelbasis für jede Anwendung neu validiert werden. Der herausragende Vorteil dieses Regelsystems besteht darin, dass das transformierte Prozessabbild als Regelbasis von einem Experten („Anlagenfahrer“) validiert werden kann.

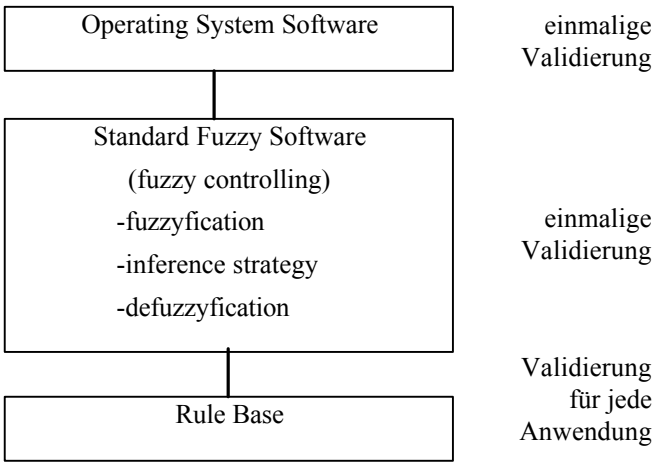

Abb. 5: Softwarestruktur des sicherheitsrelevanten Fuzzy Controllers

5. Zusammenfassung

Ein sicherheitsrelevanter Fuzzy Controller wurde vorgestellt. Dem Know-How der Sicherheitstechnik folgend wurden grundsätzliche Prinzipien wie die Dynamisierung der Signale, eine Überwachungsfunktion, die Watchdog-Funktion und das Ruhestromprinzip implementiert. Hierzu ist passive Hardware als Elektronik einzusetzen, um den Prozeß des Sicherheitsnachweises zu unterstützen. Auf diese Weise soll der Softwareanteil möglichst gering gehalten werden. Betriebssystem und Standard Fuzzy Software bedürfen einer einmaligen Validierung, dagegen stellt der Inhalt der Regelbasis ein transformiertes Abbild des technischen Prozesses dar, der für jede Applikation erneut zu validieren ist. Die Herausforderung dieses Systementwurfes lag darin, dass erfahrene Anlagenfahrer in der Lage sind, den Inhalt der Regelbasis zu validieren, um so Fehler (Unvollständigkeiten und/oder Widersprüche) bei der Wissensakquisition durch einen Informatiker zu vermeiden.

Literaturverzeichnis

[GRA01]: Grams,T.: Grundlagen des Qualitäts- und Risikomanagements Zuverlässigkeit, Sicherheit, Bedienbarkeit“, Vieweg Verlag 2001, ISBN 3-528-039450

[BIS86]: Bishop P. G.: PODS – A Project on Diverse Software. IEEE Transactions on Software Engineering (9), S. 929 – 940, 1986

[APT98].: FIDE Application Note 006-980914”, San Jose, CA, USA 1998

[SCH98]: G.H.Schildt, W. Kastner: „Prozessautomatisierung“, Springer-Verlag Wien New York, 1998, ISBN 3-211-82999-7