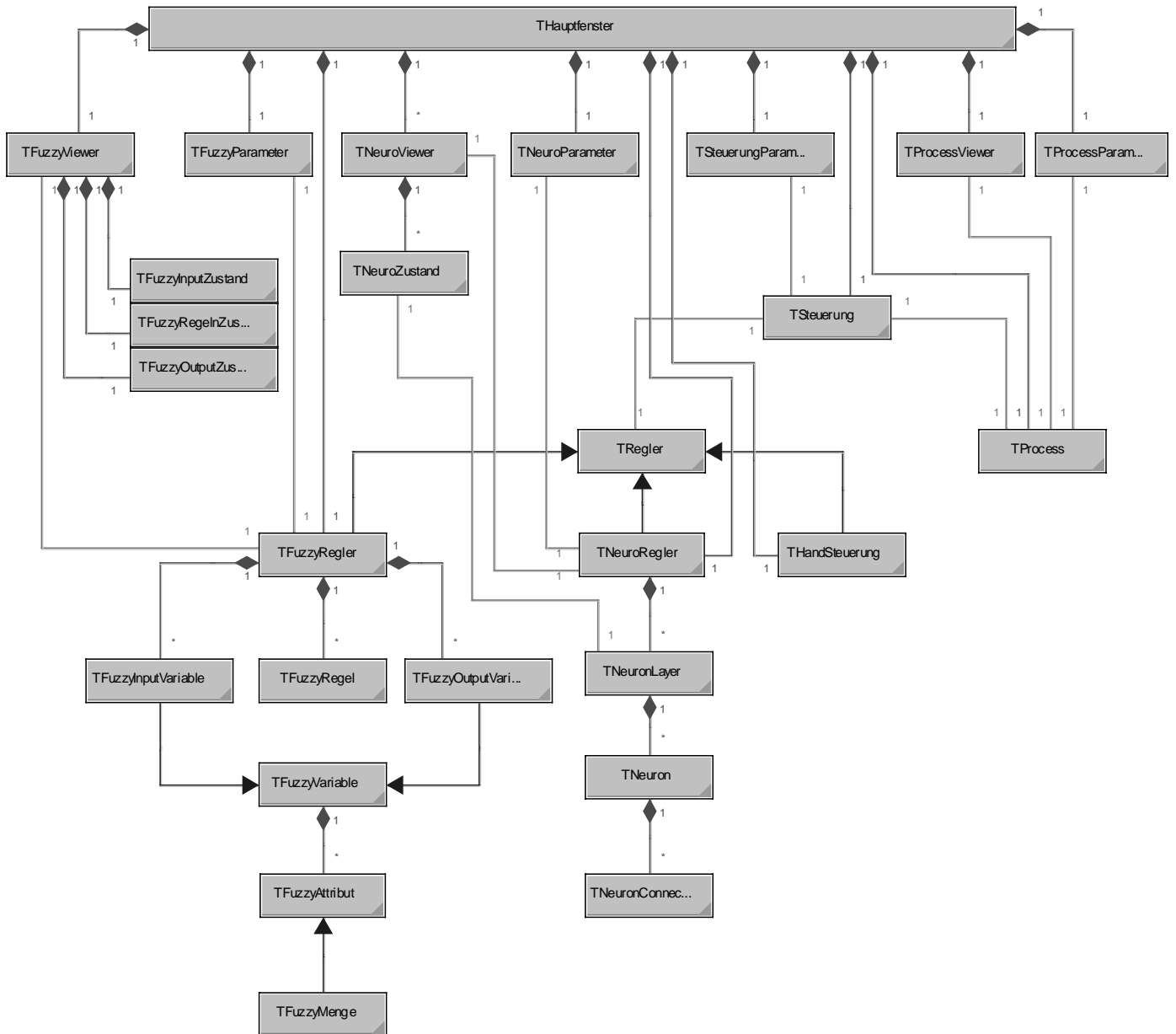


## Struktur des Simulationsprogrammes

Das Programm ist komplett in Delphi 4 programmiert unter Verwendung folgender Struktur.



Dazu kommen noch 2 Hilfsklassen (TFuzzyNorm, TFunktion), die in den Reglern selbst verwendet werden und deshalb später noch erklärt werden.

Die oberste Ebene ist das Hauptfenster, welches die Instanzen aller wesentlichen Objekte anlegt und verwaltet. Die nächste Ebene enthält Viewer-Klassen, die den Zustand eines zugeordneten Objektes (Regler, Process) anzeigen können und Parameter-Klassen, die für die Parametrisierung und Strukturanpassung des zugeordneten Objektes verantwortlich sind. Einige Anzeige- und Parametrisierungen für die Steuerung sind bereits im Hauptfenster integriert. Auch das Laden und Speichern der Objekte ist über das Hauptfenster organisiert. Bei irgendwelchen Änderungen der Objekte werden

die Viewer- und Parameterklassen informiert (Funktionen ViewUpdate() und StructureUpdate().) Die Viewerklassen benutzen zum Teil weitere Klassen/Fenster (z.B. für jede Schicht eines Neuronalen Netzes ein Fenster.)

Die Klasse TSteuerung besitzt Referenzen auf ein TRegler-Objekt und ein TProcess-Objekt. Das Process-Objekt wird zyklisch nach seinem Zustand gefragt, dieser dann dem Regler übergeben, und dessen Resultate wieder als Steuergrößen an den Process zurückgegeben. Die Taktdauer wird in Millisekunden im Steuerung-Objekt eingestellt, außerdem signalisiert es wenn eine Auffrischung der Bildschirmdarstellung notwendig ist. Die maximale Bildauffrischrate kann wieder im Millisekunden eingestellt werden, die beiden Takte werden entsprechend synchronisiert (Es werden dazu 2 Threads benutzt.) Weiterhin enthält die Klasse einen Puffer in dem alle Ein- und Ausgabedaten der Regler aufgezeichnet und auf Festplatte gespeichert werden können.

Die Klasse TProcess repräsentiert den eigentlichen Process und kann entweder intern simuliert werden oder auch echte Hardware steuern.

Die Klasse TRegler ist die Basisklasse für alle Regler, wenn neue Regler eingebaut werden sollen, müssen sie von dieser Klasse abgeleitet werden.

### Fuzzy-Regler

Die Klasse TFuzzyRegler enthält mehrere Listen, die die Ein- und Ausgabevariablen sowie die Regeln speichern, außerdem enthält sie die Standardeinstellungen für alle im Regler verwendeten Normen in Variablen vom Typ TFuzzyNorm.

Der Regelvorgang wird folgendermaßen durchgeführt:

1. Anlegen der Eingangswerte an die Variablen (Intern werden auch die Attribute gesetzt.)
2. Ausführen der Regeln (Die Attributwerte werden selbständig geholt und wieder gesetzt.)
3. Auslesen der Werte der Ausgangsvariablen (Intern werden dabei die Attributpolygone zusammengefaßt und nach der Schwerpunktmethod ein Wert berechnet.)

Den Regel-Objekten wird der linke bzw. rechte Teil einer Regel als Text übergeben und dann intern in eine Liste von Operationen compiliert. Außerdem können noch Name, Bemerkung und Sicherheitsfaktor gesetzt werden. Die Norm für den Sicherheitsfaktor ist im Normalfall intern auf die entsprechende Norm des Reglers umgelenkt.

Die Variablen-Objekte unterscheiden sich im wesentlichen nur durch ihre Schnittstellen und die Unterstützung einiger Normen bei Ausgabevariablen. Die gemeinsame Elternklasse wird durch TFuzzyVariable realisiert. Diese Klasse enthält eine Liste mit Fuzzy-Attributen, den Namen der Variablen, Bemerkungen u.ä.

Die Fuzzy-Attribute sind von der Klasse TFuzzyMenge abgeleitet und ergänzen hier Felder für Name, Bemerkung u.ä.

Die Klasse TFuzzyMenge realisiert eine durch einen Polygonzug beschriebene Fuzzymenge. Sie enthält auch Funktionen zum Zugriff auf Zugehörigkeitswerte an beliebigen Stellen, Fläche und Schwerpunkt des Polygons und die Vereinigung zweier Mengen unter Beachtung einer Fuzzynorm.

Der Typ TFuzzyNorm ist selbst keine Klasse sondern ein Record, der den Code für die Norm und die eingestellten Parameter enthält. Die Wahl eines Records liegt daran, das Delphi immer mit Referenzen arbeitet und keine Kopien von Objekten bei Zuweisungen anlegt, was in diesem Falle aber gewünscht ist. Die „Methoden“ sind deshalb als Klassenfunktionen der Klasse FuzzyNorm realisiert, der 1.Parameter der Methoden ist

dort immer vom Typ TFuzzyNorm. Die meisten Normen kann man mit 2 Werten berechnen lassen, aber bestimmte Funktionen auch mit kompletten Parameterarrays. Den Typ und Namen der Funktionen kann man über weitere Klassenfunktionen erfahren.

Die Initialisierung beim Programmstart steht in der Datei Fuzzyinitialisierung.pas.

### Neuro-Regler

Die Klasse TNeuroRegler verwaltet mehrere Layer, außerdem kontrolliert sie die komplette Zuordnung der Nummern zu den einzelnen Neuronen. Der erste Layer wird als Eingabeschicht benutzt, der letzte als Ausgabeschicht.

Die Regelung läuft folgendermaßen:

1. Rücksetzen der aktuellen Werte der Neuronen.
2. Anlegen der Werte an die Eingangsneuronen (dabei werden die Werte transformiert und der aktuelle Wert der Neuronen gesetzt.)
3. Abfragen der Werte der Ausgangsneuronen (bei gleichzeitiger Transformation) (Intern werden dabei alle Neuronen zu denen eine Verbindung besteht, rekursiv neu berechnet. Verbindungen innerhalb des Layers oder zu weiter hinten gelegenen Neuronen benutzen des Wert des letzten Zyklus.)
4. Speichern der neuen Werte in die Felder für die alten Werte.

Wenn Verbindungen innerhalb der Layer notwendig sind müßte der komplette Ablauf mehrfach durchgeführt werden, bis ein stabiler Zustand erreicht ist.

Der Neuroregler legt auch die Standard-Input-, Transfer- und Outputfunktionen fest. (Ähnlich realisiert wie die Fuzzynormen.)

Die Neuronlayer beinhalten die Neuronen der Schicht und legen wieder die Standardfunktionen der Schicht fest. (Zeigen normalerweise auf die des Reglers.)

Die Neuronen haben ihre eigenen Transferfunktionen, welche aber normalerweise denen des Layers entsprechen. Weiterhin enthalten sie eine Liste mit Verbindungen zu anderen Neuronen.

Diese Verbindungen sind Objekte der Klasse TNeuronConnection und enthalten neben einer Referenz auf das Zielneuron noch das Gewicht und ein Flag, ob die Verbindung aktiv ist. (Man kann damit Verbindungen vorläufig deaktivieren.)

Die Transfer- und anderen Funktionen werden durch einen TFunktion-Record dargestellt, die Gründe sind dieselben wie bei den Fuzzynormen. Die zugehörige Klasse heißt Funktion, und bietet neben der einfachen Funktion auch noch die zugehörige Ableitung.

Die Initialisierung beim Programmstart steht in der Datei Neuroinitialisierung.pas.

### XXXXXViewer-Klassen

Die Viewer-Klassen benutzen derzeit Listviews, in denen der Status der Elemente durch wechselnde Icons angezeigt wird. Alternativ kann man auch eine Listendarstellung einschalten um die genauen Werte zu sehen.

Die Windows-interne Verarbeitung von Änderungen in Listviews ist allerdings relativ langsam, so daß der Rechner schnell zu 100% ausgelastet ist. Die Simulation und Regelung wird davon übrigens nicht beeinflusst, weil sie in einem eigenen Thread läuft.

### XXXXXParameter-Klassen

Die Reglerstruktur wird links im Dialog in einem Treeview dargestellt und die rechte Seite stellt dann die Eigenschaften des selektierten Objekts dar.

## Bekannte Probleme

- Die Verriegelung der Threads untereinander mittels Locks ist derzeit noch nicht ausreichend ausgebaut, so dass bisweilen Zugriffskonflikte auftreten zu scheinen. Vor allem müsste der Zugriff bei Strukturänderungen (z.B. Laden eines neuen Reglers) besser abgesichert werden.
- Man kann die interne Struktur noch nicht über die Parametrisierungsklassen ändern.
- Das Datenformat beim Laden/Speichern ist zu inflexibel bei Erweiterungen, evt. wäre auch der Übergang zu einem textbasierten Datenformat sinnvoll.

## Erweiterungen

- Neue Regler müssen von TRegler abgeleitet sein und im Hauptfenster eingebunden werden.
- Neue Simulationsobjekte/Prozesse müssen die Klassen TProcess, TProcessViewer und TProcessParameter ersetzen.