GNU Fuzzy

Detlef D. Nauck

***Abstract*— Neural networks and fuzzy systems are both part of what we call soft computing or computational intelligence. Both approaches can be applied to similar classes of problems. Although fuzzy approaches have been successful in control applications in the 1990s this success has not noticeably spread to other domains. Neural networks in contrast enjoy a more steady and widespread commercial success be it in credit card fraud detection or as modules in almost every large data mining software system. Fuzzy systems have not yet enjoyed widespread success in the domain of business applications. It seems software vendors lack incentive for implementing fuzzy software because the benefits have not been made clear by the research community and because fuzzy software is more difficult to implement and use than, for example, neural networks. In order for fuzzy systems to succeed in business applications I suggest to look into building a fuzzy tool kit in a community process to provide an open source reference implementation. This paper looks at some of the key considerations that are important for building a fuzzy tool kit that can support the take-up of fuzzy systems in business applications.**

I. Introduction

In the 1990s fuzzy systems enjoyed a brief high with their success in control applications. Since early success stories like the Danish cement kiln (1982) and the Sendai Subway (1986) we have grown used to fuzzy features in appliances and consumer electronics like fuzzy washing machines and "fuzzy" cameras that amazingly produce crisp pictures.

The development of fuzzy controllers has been supported by a number of tools like the Fuzzy Programming Language FPL, the Fuzzy C-Compiler and the TIL Shell by Togai InfraLogic, Inc. [1]. The latter was once marketed as SIEFUZZY by Siemens. Other well-known tools are the Fuzzy Inference Development Environment FIDE by Aptronix [2] or fuzzyTECH by Inform [3]. There are the the Fuzzy Logic Toolbox for Matlab [4] developed by the ANFIS inventor Roger Jang [5] and a Fuzzy Logic add-on for Mathematica [6]. Both offer functions like fuzzy systems modeling, fuzzy control, and fuzzy c-means clustering.

In the middle of the 1990s other areas like data mining, pattern recognition and cluster analysis became more prevalent in the fuzzy community than control and new approaches like neuro-fuzzy combinations became fashionable research. This led to another class of tools that were more general purpose and not just focused on control application. The DataEngine from MIT GmbH [7] offers neural networks and fuzzy systems for data analysis tasks. The latest version of the Fuzzy Logic Toolbox is still marketed by MathWorks in the domain control system design and analysis, but actually it is more versatile with its offering of the neuro-fuzzy approach ANFIS and a fuzzy C-means implementation.

There are also many tools, programs and software fragments provided by academic researchers. Some of them we will mention in this paper. However, it is not the purpose of this paper to provide an overview about available fuzzy software implementations. Rather this paper considers the lack thereof and suggests that if the fuzzy research community does not look into fuzzy software in anger, fuzzy systems will not reach a satisfactory level of application in the foreseeable future.

After the initial success of fuzzy controllers there has been no major success of fuzzy approaches in applications and fuzzy methods are still shunned by industry. The story is quite different for neural networks. Not only are large scale applications like the credit card fraud detection system Falcon [8] well-established, we also see neural network algorithms in major data mining or data analysis platform provided by big names like, for example, SAS, IBM or SPSS. Oracle Data Mining provides support vector machines instead, but considers them a superset of neural networks [9].

I am convinced that fuzzy technology will not be picked up by industry unless useful software is widely available. The data mining hype helped transferring technologies like machine learning and neural networks into business applications. Business intelligence and customer relationship management systems have adapted these approaches because data mining software demonstrated their effectiveness. No such thing has happened for fuzzy approaches except for fuzzy control, which can be considered a niche application.

In order to support technology transfer I suggest that fuzzy system researchers engage in a community process and develop a fuzzy tool kit (FTK) that serves as a library for reference implementations of several fuzzy techniques.

II. Context Considerations

The type of fuzzy software that has been successful in the past has been aimed at the rather technical audience of control engineers. The software has a clearly defined benefit, i.e. helping you designing a controller much easier and faster than you used to be able to. The software actually enables you to design controllers for domains where you have only limited formal/mathematical knowledge of the controlled system. This sort of approach is convincing because it focuses on business value.

The success of fuzzy control was also helped by the enthusiasm of the Japanese industry to try out this new technology. A way of describing the development of a new technology is to use the so called "hype cycle" as coined by Gartner where a new technology after becoming visible

Detlef D. Nauck is with the Intelligent Systems Research Centre, Research & Venturing, BT Group, Orion pp1/12, BT Adastral Park, Ipswich, IP5 3RE UK (phone: 44 1473 605861; email: detlef.nauck@bt.com).

1-4244-1210-2/07/$25.00 ©2007 IEEE

progresses from inflated expectation over disillusionment and then enlightenment to productivity [10], [11]. In this sense the fuzzy control boom was the peak of inflated expectation and today fuzzy technology as a whole is probably still traveling through the trough of disillusionment.

The promises fuzzy technology has delivered to control applications have not been kept for other areas so far. In order to leave the phase of disillusionment we have to move fuzzy technology out of the engineering community and develop easy to use software that addresses a more generic decision support domain. With that I am referring to systems that provide outputs that are taken into account when making human-based or machine-based decisions. The system itself would be based on knowledge representation which is created manually, learned from data or a mixture both. The system would be able to process different types of data (e.g. numeric, symbolic, fuzzy) and not be restricted to a particular technological domain. This is obviously so generic that it also covers fuzzy control software. But the point is that the software must be domain-independent in order to be suitable for different business applications. To make it appeal to developers and users of business applications it also needs to be presented in a non-technical, benefit-oriented way and underpinned by a number of domain-specific demonstrators.

We can ask ourselves why the research community should get involved in software development. Isn't that something software vendors should do, who, after all, will reap the benefits; and didn't it work that way for technologies like decision trees, neural networks or, recently, even for Bayesian Networks in the domain of operational risk management?

Software manufacturers provide tools for solving business problems, i.e. tools that have a clearly identified benefit. Except for control applications, the research community has not yet managed to express a convincing benefit for providing fuzzy technology in business-oriented software. Approaches like decision trees and neural networks found their way into business software by riding the wave of the data mining hype where the proposed business benefit was supposed to be the valuable insight into your business you would gain from mining your business data. Although oversold as any new technology software vendors were quick to take up decision trees and neural networks, because they are essentially easily configurable algorithms that produce a good solution most of the time. Fuzzy systems in contrast have a lot of exposed parameters that are supposedly intuitive (rules, fuzzy sets), but are actually difficult to configure for non-experts. Currently, there is no technology wave in sight on which fuzzy systems could be swept into applications on a grander scale and beyond control applications. Therefore there is no incentive for software vendors to look at fuzzy systems.

I believe it is in the interest of our research community to step in and initiate the creation of open source fuzzy software as a reference implementation. This will not only facilitate the application of fuzzy technology in businesses, but will also serve as a valuable research platform.

III. Available Free Software

In this section we look at some of the freely available fuzzy software that can be accessed on the Web. The large majority has been produced by academics. The licenses range from GNU open source software to packages with undisclosed source code that are only free for research and education.

A. Libraries

Fuzzy software that is available as an open source library would be ideal for both research and application, because it could be conveniently extended and used for different purposes. Unfortunately, there is not much around.

The Free Fuzzy Logic Library FFLL has been written in C++ and has been used as the basis of a Fuzzy Logic Editor by a company specialized on video game software [12]. The library is free and the source code is available on SourceForge, but FLLL has not been updated since 2003.

The FuzzyJ Toolkit [13] is a Java library developed by Robert Orchard from the National Research Council of Canada. FuzzyJ provides a rich API [14] and has grown out of the work on FuzzyCLIPS [15] an extension to the CLIPS expert system shell [16]. The toolkit has been used to extend the Java Expert System Shell Jess [17] into FuzzyJess [18]. FuzzyJ is distributed without source code and is only free for education and research purposes. Commercial use requires a license. The FuzzyJ API provides 12 types of fuzzy sets, 3 types of fuzzy rules, fuzzy values, fuzzy variables and linguistic modifiers.

B. Tools

The vast majority of free fuzzy software comes in the shape of tools and demonstrators. This type of software is typically of limited use and not fit for large scale business applications. This is not surprising, because the tools have been developed by small research groups and only a few of them have enjoyed sustained development.

The author of this paper has himself developed several neuro-fuzzy algorithms [19] that have resulted in a number of neuro-fuzzy tools between 1995 and 2000. The tools NEFCON, NEFCLASS and NEFPROX provide learning algorithms for fuzzy control, fuzzy classifiers and fuzzy function approximation. The latest NEFCON incarnation [20] is available for Matlab and requires the Matlab Fuzzy Toolbox. The most recent implementation of NEFLCASS [21] has been written in Java, but the source code is not available. The C++ and Pascal source code of earlier versions is available, but out of date. NEFPROX [22] is available with its C++ source code, but has only limited functionality. The neuro-fuzzy software is available from Rudolf Kruse's research group [23] and is free for research and educational use. The focus of this software was always to demonstrate neuro-fuzzy algorithms and therefore it is not very suitable as the foundation of a new development process.

Fril [24], [25] is a for first-order predicate calculus that includes Prolog and provides support for fuzzy sets. Fril is free for educational, research or private use. It is distributed

without source code. A preliminary Java library is available that allows using Fril within Java applications. The source code of the library is also not available. The latest available updates of Fril date back to 2003.

FuzzyClips [15], [26] is an extension to the expert System shell CLIPS [16]. It allows to use fuzzy terms within reasoning processes and can handle uncertainty and fuzziness concepts. The software is free for research purposes and the source code is not available. The last update is from 2004.

XFuzzy 3.0 [27], [28] is a development environment for fuzzy inference systems. It consists of number of tools for inferencing, simulation, learning, graphical representation, editing and program synthesis. The tools are based on a common specification language XFL3. Xfuzzy has been written in Java and is free software under the GNU Public License (GPL). XFuzzy is probably the most advanced free fuzzy software package that is available today. A very appealing feature of XFuzzy is that it can generate source code in C, C++ and Java which means it can generate stand-alone applications of fuzzy systems designed in XFuzzy. The last update is from 2003 and it is not obvious from the team's web site, if the development still continues. Because XFuzzy is already quite an extensive toolkit, has been written in Java and is published under the GPL it could serve as the foundation of a larger community process. However, the source code was not available at the download site.

IV. Requirement Analysis

With the advent of Web 2.0 [29] and Web Services [30] it is nowadays more common to think about software services delivered via the network instead of using monolithic software platforms sitting on a desktop computer. This is one trend we should consider when assessing the requirements of potentially successful fuzzy software in this age.

Because we want fuzzy software to serve as a driver of technology transfer and encourage the use of fuzzy technology in business applications we also must consider how businesses use software. Traditionally, fuzzy software was developed by technical experts for technical experts. The focus was mainly on the algorithms and much less on usability and extensibility.

In this section we look at selected requirements and consider some implementation details for illustration purposes.

A. Core Fuzzy Engine

The core fuzzy engine must provide the central concepts like fuzzy sets, fuzzy values, fuzzy variables, types of fuzzy rules, fuzzy relations, linguistic hedges and a multitude of operators (t-norms, s-norms, defuzzification etc). The core should be designed for speed which means we have to take implementation details of fuzzy sets and operators into account. For example, a quick test in Java reveals that computing the membership degree for an exponential membership takes about 10 times longer than computing the membership degree for trapezoidal or triangular membership function. While the latter requires at most four comparisons, two subtractions and a division the former requires a call to the implementation of the exponential function. A test on a fast modern PC required about 30 seconds for computing 10^8 membership degrees of a Gaussian membership function.

That sounds reasonably fast, but consider a situation where we are going to learn a fuzzy system with 100 rules and 10 variables from 100.000 sample records. Assume further we have to iterate a 1.000 times over the data set. That results in 10^{11} computations of a membership degree (assuming we don't optimize and stop evaluating a rule once we encounter a membership degree of 0). In case of Gaussians we have an overhead of 30.000 seconds, i.e. over 8 hours of computing time just for membership computation alone. In case of trapezoidals we would require less than an hour. This problem can be addressed by tabulating the exponential function for the domain of a variable with sufficient density and select the closest entry or if necessary interpolate between two entries. Converting the argument of the function into an array index would result in one array access, one subtraction, two multiplications and one division bringing us close to a the effort required for trapezoidals. Assume we have a bell-shaped membership function centered at c:

$$\mu_{(c,\sigma)}(x) = \exp\left(-\left(\frac{x-c}{\sigma}\right)^2\right).$$

For this type of function we have

$$\begin{aligned}\mu_{(c,\sigma)}(x) &= \mu_{(0,1)}((x-c)/\sigma)) \text{ and}\\ \mu_{(0,1)}(-x) &= \mu_{(0,1)}(x)\end{aligned}$$

We can therefore simply store n values of $\mu_{(0,1)}(x)$ in an array as follows:

$$\begin{aligned}A[i] &= \exp\left(-x^2\right), \text{for } x \in [0,s], \text{ where}\\ i &= \left\lfloor n \cdot \frac{x}{s} \right\rfloor.\end{aligned}$$

We set $\mu_{(c,\sigma)}(x) = 0$ for $x \notin [c - s\sigma, c + s\sigma]$. For practical purposes it should be sufficient to select $s = 3$ because $\mu_{(c,\sigma)}(c \pm 3\sigma) \approx 10^{-}4$. This set-up allows us to compute $\mu_{(c,\sigma)}(x)$ approximately by the following pseudo code (example without interpolation):

$pos = x - c;\ spread = s \cdot \sigma;$
if $((pos < -spread)$ or $(pos > spread))$
    $y = 0.0;$
else if $(pos == 0)$
    $y = 1.0;$
else{
    if $(pos < 0)$ $pos = -pos$
    $i = \text{trunc}(n \cdot pos/spread);$
    $y = A[i];$
}

This approach is about six times faster than using a call to an exponential function. A test in Java reveals that for $n = 10.000$ and testing values for $\mu_{(1,2)}(x)$ with $x \in [0,1]$the maximum absolute error is about 0.00033 and the average

absolute error is about 0.00013. That means we have at least three significant digits for a membership degree.

It is probably not required for the fuzzy core to produce highly accurate membership degrees anyway. We may want to question if it is appropriate to distinguish between membership degrees of 0 and 0.001, for instance. Fuzzy technologies may not be suitable for applications that require that degree of accuracy. Therefore it should be possible to further increase the speed of computation by refraining from using floats and using integer arithmetic only.

B. Data Management

For any large-scale business application it is absolutely essential to provide access to databases. Consider, for example, a business with millions of individual customers that wants to determine the propensity to churn for each customer. The customer data will be stored in a database or data warehouse. Expecting the user to export the data into a file for use with the fuzzy software would pose a serious acceptability risk.

We also have to consider that many fuzzy applications will not be fully knowledge-based, but will have to be derived from data at least to some extent. That means we have to implement learning algorithms (see below). That also means that the software will have to loop over large data samples for learning and will have to score possibly even larger samples in an application mode. It is not realistic to restrict the amount of data to the amount of main memory. The software will have to provide efficient data management strategies implemented on database connections.

Consider for example, that we want to select a number of records from a large table where we test if a particular attribute has a membership degree above a certain threshold. It will be inefficient to load the table into memory, process the attribute and delete records that don't meet the criterion. It would also be inefficient to only load the column with the attributes in question and then fetch all rows passing the test via row identifiers or keys. Ideally we should be able to create a bespoke database statement and retrieve a result set containing only the records we need. Databases have been built to be very efficient in scanning tables and retrieving records. It would be too expensive to replicate this behavior in the FTK. Assume that we have a connection to an SQL database like MySQL or Oracle and that we want to test if the membership of attribute x with a triangular membership function with parameters *left*, *center* and *right* is above a threshold θ. We can dynamically create and execute an SQL statement like

```
SELECT * FROM MyTable
WHERE    CASE WHEN x = center THEN 1.0
         ELSE    CASE WHEN x < left OR x > right
                 THEN    0.0
                 ELSE    CASE WHEN x < center
                         THEN (x − left) / (center − left)
                         ELSE (right − x) / (right − center)
                         END
                 END
         END > threshold
```

This approach can also be followed by storing parameters of fuzzy sets or discretised membership functions in database tables and joining them via suitable conditions to data tables. An objective should be to move as many simple operations on large amounts of data as possible into a database system. If an object/relational mapping (ORM) approach like Hibernate for Java [31] is used the generation of SQL code can be done automatically.

An alternative would be to consider an implementation of a fuzzy query language like Fuzzy SQL [32], [33], [34] that directly supports fuzzy terms in database queries. There are some commercial [35] and free [36] implementations of Fuzzy SQL available. However, in their common form they don't seem to be suitable for inclusion into an FTK, because they either are stand-alone applications or scripts for particular database management systems. It appears to be more likely that an FTK could serve as platform for a generic Fuzzy SQL implementation.

C. Learning

As mentioned previously we have to assume that many applications require generation of fuzzy systems from data. That means fuzzy application software should provide a number of learning algorithm as they are known, for example, from neuro-fuzzy systems [5], [19], fuzzy decision trees [37], fuzzy cluster analysis [38] or genetic fuzzy systems [39].

Different learning algorithms have different requirements on the representation of fuzzy systems. Many learning approaches manipulate parameters of membership functions and generate or change fuzzy rules and therefore need access to the implementation details of the fuzzy core engine.

This poses a problem, because the core should be as small as possible and achieve a stable software version quickly. Learning in fuzzy system is still and active research area and we can expect that many researchers want to contribute algorithms. They should not need to change the core in order to integrate a new learning procedure. It would be more efficient if a clean core API would be available that enables fast and convenient implementation of learning algorithms.

For example, consider learning algorithms that use some variation of backpropagation. They would compute an output error, distribute it over the available fuzzy rules to compute rule errors, which would be used to update parameters of fuzzy sets. A core API could provide a number of methods that allow changing fuzzy set parameters based on error values. In backpropagation the change of a parameter p of a membership function μ is done by

$$\Delta p = -r\frac{\delta E}{\delta p} = -rd\frac{\delta \mu}{\delta p}$$

where r is a learning rate, E is the overall error and d is a product of a number of factors created by the chain rule, typically involving computation of outputs and rule fulfillments which depend on the particular type of fuzzy system. The fuzzy core can provide implementations of $\delta\mu/\delta p$ for each type of fuzzy set to be used by implementers of learning algorithms. Similarly this can be done for types of fuzzy rules etc. It is also possible to provide simplified

modification methods like *shift* or *scale* to accommodate learning algorithms that use simpler heuristics [19].

Fuzzy rule learning is easier to accommodate, because it essentially means combining resources (fuzzy sets and operators) provided by the fuzzy core. However, it must be taken into account that different learning algorithms may have different requirements for representing rules. Some algorithms may expect that rules share a common set of membership functions [19], while others may create fuzzy sets on the fly while creating fuzzy rules [40]. The fuzzy core should therefore provide a rule implementation that accommodates both types of rule learning approaches.

Fuzzy clustering, which is a form of unsupervised learning, requires different services from the fuzzy core. Clustering algorithms manipulate only matrices of membership degrees and may require methods for transforming clusters into rules by projection. To enable simple implementation of clustering algorithms the core should provide suitable methods for matrix representation and manipulation.

Providing resources for genetic fuzzy systems or any form of evolutionary computation in order to create or optimize fuzzy systems is a bigger challenge, because implementing evolutionary algorithms as such is a big task. It may be more appropriate to accommodate this type of approach via interfaces and external APIs. A fuzzy system could be transformed into a form that can be used by some evolutionary software system and the core could provide an API to execute a fuzzy system on test data to compute a performance or fitness measure.

D. Interfaces

In order to be successful it is important for a fuzzy software system to provide standard interfaces to existing software, but without relying on the user having access to any external commercial package. The Fuzzy Toolbox for MATLAB is one obvious candidate. It should be possible to import/export fuzzy systems from/into the Fuzzy Toolbox format. The same approach could be taken for Mathematica.

However, these package are used mainly by the engineering and research community and are therefore of limited reach if we want to address a larger business community and enable them to use fuzzy software.

An important way of integrating into existing and future business platforms are web services [30]. A fuzzy software system should be Java EE compliant [41], [42] so it can be deployed via an application server like BEA WebLogic, JBoss, or Apache Tomcat etc [43]. This enables the development of web applications and web services [44], but would not prevent the use in stand-alone applications if required.

Another important interface standard to look at is PMML – the Predictive Model Markup Language defined by the Data Mining Group DMB [45]. The DMG is an independent, vendor-led group that develops data mining standards. At the moment the DMG works on the specification of PMML 3.2. The currently available PMML Version 3.1 does contain, for example, neural networks and tree models, but no fuzzy models. The DMG reports that PMML 2.1 has been implemented so far by IBM in its Intelligent Miner Software, by SPSS in SPSS version 13 or later and in Clementine, by SAS Enterprise Miner and by CART from Salford Systems [46]. Some models described in PMML can also be read or written by Oracle 10g Data Mining [9]. SAS and IBM, for example, support PMML since 2004 [47].

We also should consider that a fuzzy system once created by the toolkit should be able to integrate into application without some type of runtime environment having to be executed as a separate application. That means that it must be possible that a fuzzy system is either exported as some form of software library that applications can dynamically link to or as a stand-alone executable (cf. XFuzzy [28]).

E. GUI

Traditionally, fuzzy software has been written within academic circles and the results were therefore mainly research demonstrators with very technical user interfaces. All commercial implementations the authors is aware of still have a research flavor, which tends to put off business users.

An FTK will require some GUIs that allow the convenient specification or viewing of fuzzy sets and fuzzy rules. However we must also become creative and design interfaces that appeal to non-technical users.

We cannot expect all users to provide fuzzy partitions for variables, decide about types of membership functions and select operators. Users should be offered support in making this kind of decisions whenever possible. Where that is not possible, e.g. because the user wants to represent knowledge, the specification of parameters must be heavily supported and simplified. For example, instead of drawing functions to specify fuzzy sets we can use a dialog-based approach that focuses on the linguistic terms used to describe knowledge.

It is likely that most fuzzy applications will be data driven. This is simply due to the nature of fuzzy systems like Mamdani or Takagi-Sugeno-Kang systems being essentially function approximators. That means beyond control applications they are typically suitable for classification or regression and therefore interesting for areas typically described as data mining, data analysis, business intelligence or predictive analytics. In these areas parameters are derived in some learning processes. Here we have the problem of communicating the derived knowledge to the user. Fuzzy systems still have a large deficit in this area and simply plotting a membership function is not a suitable way of conveying the meaning of a fuzzy term.

V. Conclusions

The purpose of this paper is to stimulate discussion around an open source, community-based software development process for a fuzzy tool kit. The context of a conference paper does not provide enough room to discuss all aspects sufficiently and so we have only looked at selected issues like scalability, data management and interfaces. The most popular types of fuzzy systems can only be used for modeling rather simple domains, because they directly relate input and output variables, typically via one step of fuzzy interpolation

[48]. We have not discussed support for chaining of fuzzy systems or graphical models like possibilistic networks. This will be required for supporting more complex applications.

We have also not looked at applications beyond function approximation. A promising area for future growth of fuzzy system research lies in text-based problems like text mining, search engines and question answering systems and they should also be supported by a future FTK.

The IEEE Computational Intelligence Society has initiated a task force on Fuzzy Sets and Systems Software (FSSS) chaired by Plamen Angelov. The web site of the task force will be at http://www.fuzzysoftware.org and will hopefully serve as a focus point of a future community process.

REFERENCES

[1] "Togai InfraLogic [online]," www.ortech-engr.com/fuzzy/togai.html, 1995, accessed 26 January 2007.

[2] "Aptronix [online]," http://www.aptronix.com, 2000, accessed 26 January 2007.

[3] Inform GmbH, "FuzzyTech [online]," http://www.fuzzytech.com, 2007, accessed 26 January 2007.

[4] The MathWorks, "Fuzzy toolbox 2.2.4 [online]," http://www.mathworks.com/products/fuzzylogic, 2007, accessed 26 January 2007.

[5] J.-S. Jang, C. Sun, and E. Mizutani, *Neuro Fuzzy and Soft Computing*. Upper Saddle River, NJ: Prentice Hall, 1997.

[6] Wolfram Research, "Mathematica Fuzzy Logic [online]," http://www.wolfram.com/products/applications/fuzzylogic, 2007, accessed 26 January 2007.

[7] MIT GmbH, "Dataengine [online]," http://www.dataengine.de, 2001, accessed 26 January 2007.

[8] FairIsaac, "Falcon fraud manager [online]," http://www.fairisaac.com/Fairisaac/Solutions/Product+Index/Falcon+Fraud+Manager, 2007, accessed 02 February 2007.

[9] Oracle, "Oracle 10g Data Mining FAQ [online]," http://www.oracle.com/technology/products/bi/odm/odm_10g_faq.html, February 2005, accessed 02 February 2007.

[10] Gartner, "Understanding hype cycles [online]," http://www.gartner.com/pages/story.php.id.8795.s.8.jsp, 2007, accessed 05 February 2007.

[11] Wikepedia, "Hype cycle [online]," http://en.wikipedia.org/wiki/Hype_cycle, 29 January 2007, accessed 05 February 2007.

[12] "Free fuzzy logic library [online]," http://ffll.sourceforge.net/, 2003, accessed 25 January 2007.

[13] R. A. Orchard, "FuzzyJ toolkit web site [online]," http://www.iit.nrc.ca/IR_public/fuzzy/fuzzyJToolkit.html, 2005, accessed 25 January 2007.

[14] ——, *NRC FuzzyJ Toolkit for the Java Platform User's Guide [online]*, Institute for Information Technology, National Research Council Canada, http://www.iit.nrc.ca/IR_public/fuzzy/fuzzyJDocs, September 2006, accessed 23 January 2007.

[15] ——, *FuzzyCLIPS Version 6.04A Users Guide, ERB-1054*, National Research Council Canada, 1998.

[16] G. Riley, "CLIPS 6.24: A tool for building expert systems [online]," http://www.ghg.net/clips/CLIPS.html, 2006, accessed 25 January 2007.

[17] E. Friedman-Hill, "JESS – the rule engine for the java platform [online]," http://herzberg.ca.sandia.gov/jess/, 2006, accessed 25 January 2007.

[18] R. A. Orchard, "Fuzzy reasoning in Jess: The Fuzzy J Toolkit and Fuzzy Jess," in *Proceedings of the Third International Conference on Enterprise Information Systems (ICEIS 2001)*, Setubal, Portugal, July 2001, pp. 533–542.

[19] D. Nauck, F. Klawonn, and R. Kruse, *Foundations of Neuro-Fuzzy Systems*. Chichester: Wiley, 1997.

[20] A. Nürnberger, D. Nauck, and R. Kruse, "Neuro-fuzzy control based on the NEFCON-model: Recent developments," *Soft Computing*, vol. 2, no. 4, pp. 168–182, 1999.

[21] D. Nauck and R. Kruse, "NEFCLASS-J – a Java-based soft computing tool," in *Intelligent Systems and Soft Computing: Prospects, Tools and Applications*, ser. Lecture Notes in Artificial Intelligence, B. Azvine, N. Azarmi, and D. Nauck, Eds. Berlin: Springer-Verlag, 2000, no. 1804, pp. 143–164.

[22] ——, "Neuro–fuzzy systems for function approximation," *Fuzzy Sets and Systems*, vol. 101, pp. 261–271, 1999.

[23] R. Kruse, "Neural networks and fuzzy systems [online]," http://fuzzy.cs.uni-magdeburg.de/, 2007, accessed 26 January 2007.

[24] J. Baldwin, T. Martin, and B. Pilsworth, *Fril – Fuzzy and Evidential Reasoning in Artificial Intelligence*. Research Studies Press, 1995.

[25] T. P. Martin, "Fril resources [online]," http://www.enm.bris.ac.uk/ai/martin/downloads/FrilResources.html, 2003, accessed 26 January 2007.

[26] R. A. Orchard, "Fuzzy extension to the CLIPS expert system shell (FuzzyCLIPS)[online]," http://www.iit-iti.nrc-cnrc.gc.ca/projects-projets/fuzzyclips_e.html, 2004, accessed 26 January 2007.

[27] F. J. M. Velo, I. Baturone, S. S. Solano, and A. Barriga, "Rapid design of fuzzy systems with XFuzzy," in *Proc. IEEE Int. Conf. on Fuzzy Systems 2003*, vol. 1. St. Louis: IEEE, 2003, pp. 342–347.

[28] "XFuzzy 3.0 [online]," http://www.imse.cnm.es/Xfuzzy/, 2003, accessed 26 January 2007.

[29] T. O'Reilly, "What is web 2.0 – design patterns and business models for the next generation of software [online]," http://www.oreillynet.com/pub/a/oreilly/tim/news/2005/09/30/what-is-web-20.html, Sept. 2005, accessed 26 January 2007.

[30] W3C, "Web services activity [online]," http://www.w3.org/2002/ws/, Nov. 2006, accessed 26 January 2007.

[31] C. Bauer and G. King, *Hibernate in Action*, 2nd ed. Greenwich, CT: Manning Publications Co., 2004.

[32] J. Galindo, J. Medina, O. Pons, and J. Cubero, "A server for fuzzy sql queries," in *Flexible Query Answering Systems*, ser. Lecture Notes in Artificial Intelligence (LNAI), T. Andreasen, H. Christiansen, and H. Larsen, Eds. Springer, 1998, no. 1495, pp. 164–174.

[33] E. Cox, "Fuzzy sql," Scianta Intelligence, http://scianta.com/pubs/AR-PA-008.htm, Online article, 1999, published online, accessed 02 February 2007.

[34] J. Galindo, A. Urrutia, and P. M., *Fuzzy Databases: Modeling, Design and Implementation*. Hershey, PA: Idea Group Publishing, 2006.

[35] E. Cox, "Fuzzy sql [online]," http://scianta.com/products/fuzzysql.htm, 2005, accessed 02 February 2007.

[36] J. Galindo, "FSQL (Fuzzy SQL) – a fuzzy query language [online]," http://www.lcc.uma.es/~ppgg/FSQL.html, 2007, accessed 02 February 2007.

[37] C. Z. Janikow, "Fuzzy decision trees: Issues and methods," *IEEE Trans. Systems, Man & Cybernetics. Part B: Cybernetics*, vol. 28, no. 1, pp. 1–14, 1998.

[38] F. Höppner, F. Klawonn, R. Kruse, and T. Runkler, *Fuzzy Cluster Analysis*. Chichester: Wiley, 1999.

[39] O. Cordón, F. Herrera, F. Hoffmann, and L. Magdalena, *Genetic Fuzzy Systems. Evolutionary Tuning and Learning of Fuzzy Knowledge Bases*. World Scientific, 2001, vol. 19.

[40] M. Berthold and K.-P. Huber, "Constructing fuzzy graphs from examples," International Computer Science Institute, Berkeley, Tech. Rep., 1997.

[41] Sun Microsystems, Inc., "Java EE at a glance [online]," http://java.sun.com/javaee, 2007, accessed 02 February 2007.

[42] K. Mukhar and C. Zelenak, *Beginning Java EE 5*. Berkeley, CA: Apress, 2005.

[43] Wikepedia, "Application server [online]," http://en.wikipedia.org/wiki/Application_server, January 2007, accessed 02 February 2007.

[44] *Developing Java Web Services*. Indianapolis, IN: Wiley Publishing, Inc., 2003.

[45] Data Mining Group, "Data Mining Group [online]," http://www.dmg.org, 2006, accessed 02 February 2007.

[46] ——, "PMML Products [online]," http://www.dmg.org/products.html, 2006, accessed 02 February 2007.

[47] SAS Institute Inc., "SAS and IBM demonstrate industry leadership in data mining interoperability [online]," http://www.sas.com/news/preleases/042704/news3.html, 27 April 2004, accessed 02 February 2007.

[48] R. Kruse, J. Gebhardt, and F. Klawonn, *Foundations of Fuzzy Systems*. Chichester: Wiley, 1994.