A Unified Granular Fuzzy-Neuro Framework for Predicting and Understanding Software Quality

Mokhtar Beldjehem
Sainte-Anne’s University
1589 Walnut Street
Halifax, Nova Scotia, B3H 3S1, Canada

E-mail: mokhtar.beldjehem@usainteanne.ca

Abstract

We propose herein a novel unified framework that uses a developed hybrid fuzzy-neuro system in order to evaluate the impact of inheritance aspects on the evolvability of a class library, and to study the relevance of using inheritance as indicator of class interface stability with respect to version change. To this goal, we propose a novel computational granular unified framework that is cognitively motivated for learning if-then fuzzy weighted rules by using a hybrid neuro-fuzzy or fuzzy-neuro possibilistic model appropriately crafted as a means to automatically extract or learn software fuzzy prediction rules from only input-output examples by integrating some useful concepts from the human cognitive processes and adding some interesting granular functionalities. This learning scheme uses an exhaustive search over the fuzzy partitions of involved variables, automatic fuzzy hypotheses generation, formulation and testing, and approximation procedure of Min-Max relational equations. The main idea is to start learning from coarse fuzzy partitions of the involved metrics variables (both input and output) and proceed progressively toward fine-grained partitions until finding the appropriate partitions that fit the data. According to the complexity of the problem at hand, it learns the whole structure of the fuzzy system, i.e. conjointly appropriate fuzzy partitions, appropriate fuzzy rules, their number and their associated membership functions.

Keywords: software quality prediction and understanding, possibility theory, fuzzy sequence, if-then fuzzy weighted rules, level of stability, hybrid fuzzy-neuro possibilistic model, approximation of Min-Max relational equations.

1. Introduction and Related Works

Designing a software system so that it can easily evolve as the operating environment changes, as user requirements are modified, as new requirements appear, and as errors (which inevitably occur in large scale software systems) are uncovered, is a difficult task and it remains an ultimate goal and a challenge for software engineering. Building software systems that can evolve rapidly and gracefully in response to changing needs is the most important challenge facing the software industry today. We believe that what is known to be under the umbrella of intelligent software engineering in general and more specifically data-driven and simulation-based software engineering could constitute a good candidate and a promising avenue of research in this field and could dramatically cut software evolution costs.

Pressman [1] estimated to 60% the part devoted to maintenance in the total effort of a software development project, of which 80% is devoted directly or indirectly to software evolution (adaptive and perfective maintenance). It is widely accepted that developing easily reusable and maintainable software is of great interest to the software industry, however this
still remains an endeavors for the managers and software practitioners alike. Current object-oriented (OO) software systems must satisfy non-functional requirements including quality and performance aspects. These, in contrary to functional requirements, are difficult to assess during the test phase of the life cycle development, i.e., that it is no longer enough that a system meet the functional specification; it has to be also easily adaptable to future changes. In other terms it has to meet non functional requirement such as those of maintainability, evolvability, and s.o. Quality prediction models could constitute an interesting solution to such a problem by providing some preventive maintenance layer on software at its early stages of its development life cycle.

Granular and/or soft computing models as in Zadeh [2-4]; Yager and Zadeh [5]; Pedrycz [6]; Yao [7]; Gupta et al. [8]; Liu et al. [9]; Beldjehem [10-12] combining several paradigms in general and hybrid fuzzy-neuro models as in Beldjehem [13-19], Sinha and Gupta [20] in particular could effectively contribute to the building of next-generation intelligent reliable software quality prediction models. The benefits from adopting such hybridization is its capacity to account conjointly for both empirical input-output historical data as well as heuristic software domain knowledge when it is available from software engineers, design patterns, good practices and body of knowledge of the software engineering field in general.

In addition to resolve the boundary problem, such hybrid fuzzy-neuro models are transparent, tolerant and could effectively ensure accuracy, performance and interpretability. The main idea stems from the possibility to use a hybrid fuzzy-neuro system to generate (tune or extract) a knowledge base (KB) or more specifically a rule base (RB) in terms of fuzzy IF-THEN production rules and thus to generate automatically a fuzzy rule-based quality prediction model by supervised learning from I/O examples. Besides integrating non-linearities directly from the learning examples (training set), the additional advantages of such an approach are emergent properties such as the inherited property of value approximation and robustness which are of paramount importance in exhibiting generalizations necessary to process unseen actual situations in the field (in addition to the testing and validation sets).

On one hand, this has led to intensive research for the development of accurate quality prediction models using various paradigms ranging from Bayesian approach, neural networks, clustering algorithms, conventional ID3 and C4.5 inductive approach of Quinlan [21] to Fuzzy and inductive fuzzy binary trees approaches as in Sahraoui et al. [22-23]; Boukadoum et al. [24]. However to the best of our knowledge, due to (1) the lack of good practices in software quality and the unavailability of accurate historical data in Software Engineering, (2) the incapacity to account conjointly for both historical data and available explicit linguistic knowledge (3) the lack of interpretability, understandability and explanation facilities; most developed “proof of concept” prototypes are of low accuracy and thus fail to fulfill their “raison d’être”, i.e. helping the managers and software engineers in practice in regards to decision making for the purposes of prediction, rework and/or refactoring. On the other hand due to the Black-box nature of the conventional naive or the classical Bayesian models in the context of software quality estimation, they are of limited interest from the decision making viewpoint; by consequent they should be progressively replaced with intelligent predictive computational models incorporating explicit formal cause-effect relationships expressed in terms of fuzzy IF-THEN rules that are built or rather machine learned automatically.

2. A Novel Learning Methodology
2.1. Motivations for our learning methodology
We want to conduct herein a case study in connection with OO class libraries; which have to preserve, as much as possible, the compatibility among versions in order to maintain software backward compatibility. A software is stable in the face of a change to requirements if we do not need to modify it at all. We can sensibly talk of software being more or less stable, depending on the level of change required. Thus stability is indeed a fuzzy concept and is a matter of degree. A software is flexible if it can be readily extended to accommodate likely new requirements with only minimal impact on the existing structure. We propose to develop hybrid soft computing tools that allow the prediction of class evolvability through the symptomatic detection of potential instabilities during the design phase of such libraries. This may help avoid later problems. Evolvability might be defined as the ease with which a software system or a component can evolve while preserving its design as much as possible. In the context of the OO paradigm we might restrict the preservation of the design to the preservation of the library interface. Preserving the library interface must be a continuous goal starting from the initial design. Intuitively, a good design must allow the improvement of existing functionalities and the addition of new functionalities while preserving the library interface. This leads to the problems of assessing the goodness of the design from the perspective of evolvability, and of identifying the internal attributes (coupling, cohesion, size and complexity, inheritance, etc.) that could be used as evolvability indicators. Our current focus is on the validation of the hypothesis that the inheritance aspects of an OO class library might constitute good indicators of its capacity to evolve. To this end, we propose to develop hybrid fuzzy-neuro approach for automatic building and assessing evolvability prediction models. More specifically, we will use our developed hybrid fuzzy-neuro model in order to evaluate the impact of inheritance aspects on the evolvability of a class library, and to study the relevance of using inheritance as indicator of class interface stability with respect to version change. It is worth mentioning that the concept of stability itself (or conversely instability) is a matter of degree and indeed is a fuzzy concept. This is due in part to the fact that intuitively; quality is of intrinsic character and it is inherently a qualitative not quantitative concept. As a result, empirical investigations of measurable internal attributes and their relationship to external quality characteristics are a crucial issue for improving the assessment of software product quality. In these context large measures (known as metrics) have been proposed in the literature and have been used in predicting the fault-proneness of classes during design, and for predicting the maintenance effort. In particular it has been shown that size and inheritance metrics are good indicators for the stability of a framework. We will focus our attention on how inheritance aspects can be good indicators of the interface evolution of an OO class library. More specifically we will investigate the possibility to learn and understand causal relationships between some inheritance metrics, and the stability of OO library interfaces. Moreover, we will propose the interpretation of the results in terms of weighted fuzzy IF-THEN production rules and the relative importance of the variation of the metrics in relation with the stability of the class library. In our modeling we use fuzzy metrics that are linguistic variable defined over term sets (or label of fuzzy sets) and represented in terms of membership functions (MFs) and/or possibility distributions. When the possible values for a metric variable are symbolic rather numeric, approximations can be represented in terms of a fuzzy set with a corresponding membership function (MF). The stability itself too is modeled as a linguistic variable, this allows coping with several levels of stability (or instability). In
our modeling we use Zadeh’s possibility theory as a basis for approximate reasoning [25-26] and more specifically possibility/necessity measures which enables us to accurately estimate how much it is possible that a class is stable (or instable), and how much it is necessary that a class is stable (or instable). We believe that our approach will definitely open the door for intelligent next generation quality prediction systems. Besides machine learning the causal relationships between the inheritance metrics and the stability, they allow the detection of the importance or relevance of each metric to stability which is of paramount importance for an empirical approach of studying and understanding software quality aspects and hence providing justification facilities for the metrics validation issues. Thus enables the understanding of relative importance of each inheritance metric and its influence in the (in) stability of class libraries.

On larger scale, the key mechanisms of object technology (encapsulation, inheritance and polymorphism), offer many opportunities for quality improvement. In fact, in practice the use of objects does not, in it self, improve the quality of software. If anything, object technology introduces new opportunities to introduce defects. A prime example is the misuse of inheritance. A single change in a badly designed class hierarchy can wreak havoc throughout a software system by producing unintended side effects on numerous subclasses and countless instances in a running system. However, inheritance allows a local modification to a single class to produce large scale changes in a system without requiring individual changes to every affected object.

Zadeh’s fuzzy sets and fuzzy logic [27-29] may be considered as a basis for knowledge and meaning representation and is particularly suited for dealing with natural language and software quality issues. We believe that it is the concept of possibility/necessity distributions Zadeh [25-26] rather than the truth that will play the primary role in manipulating such knowledge for the perspective of drawing conclusions. Possibility theory (Zadeh [25-26]; Yager [30]; Dubois and Prade [31]; Olaf [32]) provides a formal framework for representing and dealing with ignorance, and uncertainties prevalent in modeling real world problems in a flexible computerized manner straightforwardly. It allows handling uncertainty in a rather coherent qualitative way. Two measures of uncertainty called possibility and necessity are associated with a possibility distribution. These measures turn out to be a convenient tool for modeling of uncertainty, which allows for the representation of imprecise pieces of information, gradual properties, flexible constraints (expressing preferences), incomplete state of information or partial states of ignorance. However it is well accepted that crafting manually fuzzy systems to resolve complex large scale real-world problems is a difficult task that is not always obvious for both the designer (the knowledge-engineer) and the domain expert. This is due partly to the cognitive limits of the human being as advocated by Miller [33], but also to the difficulty of understanding the intricacies of dimensionality and inherent complexities and peculiarities of large scale real world and software problems, and in particular when dealing with complex large scale software systems. Not to mention the lack of precision in the human-human interaction and communication that affects significantly the knowledge acquisition process during the tandem knowledge-engineer/domain expert relationship. Furthermore once it is undertaken it is labor-intensive, costly, error prone, time-consuming, and done on a trial-and-error basis in an adhoc manner and hence need to be totally or partly automated. This is known as the knowledge acquisition bottleneck problem or the Feigenbaum bottleneck and is a common problem for all AI approaches. Soft computing
as an automated knowledge acquisition hybrid methodology aims at remedying such a bottleneck problem, and is a good candidate in mining very large software repositories.

Various soft computing (SC) methods and techniques have been developed and used to tackle this learning problem from various points of views. However they are based on some idealizing assumptions and no one adopts a holistic approach to resolve such a problem globally, i.e., finding conjointly appropriate fuzzy partitions, fine tuning the membership functions of the labels used in the rules as well as identifying the structure of the fuzzy system (both the required number of rules and rules themselves explicitly) simultaneously. In practice the required number of rules of the system is not known in advance. Indeed learning fuzzy if-then rules is a difficult multi-parameter optimization problem! We have previously devised, developed, formally validated and deployed a granular hybrid fuzzy-neuro system called Fennec (Beldjehem [8-19]) that was successfully applied to a difficult problem of biomedical diagnosis on Proteins/Biological Inflammatory Syndromes (B.I.S), to image processing and vision engineering as well as to a complex handwriting pattern recognition problem (Beldjehem [34]). Based on our previous work, we propose herein an integrated framework to modify the model, accommodate it and extend its ability and scope of applicability for dealing with software quality prediction by integrating some useful concepts from the human cognitive processes and adding some interesting granular functionalities and knowledge of the software domain. In general, software engineering activities are knowledge intensive and software design is a good application area since the knowledge is generally heuristic in nature and software engineers tend to think on terms of rules and more specifically on terms of fuzzy rules.

Figure 1. From a coarse fuzzy partition to a fine-grained fuzzy partition

The basic idea underlying our framework stems from the following interesting remarks about human cognition: Let us first focus our attention on the human problem
solving process. In solving problems the human starts from a coarse description but if needed iterates and goes gradually to a fine-grained description or in-depth details enabling more understanding of the underlying problem until reaching a point where one can effectively find a solution and so stops and does not need any more details. At this point, an excess of precision is not needed (is not necessary) because a certain satisfying trade-offs between precision (level of details) and generality of description has been reached and is sufficient and enough for finding a satisfactory approximate solution to the specified problem. Thus after each iteration (increment) a gain of information is obtained enabling more in-depth and more understanding of the underlying situation. Thus, the human converges to a solution gradually by leveraging the level of details. See Figure 1 for more details in connections with a granular soft computing (GrSC) setting. Low levels of details allow coarse or general descriptions reflecting crude approximations whereas high levels of details allow specific descriptions reflecting more or less relatively precise approximations (crisps at the extreme). It is appealing and convenient to mimic mechanically or to emulate computationally such a cognitive process in order to automatically build faithfully by learning an appropriate “good” fuzzy prediction system that exhibits both a high accuracy and a good performance for any problem at hand. This motivates us in building a learning system able to use such abstraction and granulation mechanisms in a fashion that is akin to the way humans achieve problem solving process. In general the required levels of details necessary in describing rules as well as the required number of rules for solving a problem depends to the degree of complexity of the problem at hand and are unknown and hence we propose to detect and determine them by learning within our framework. The rational behind using levels of granularity in modeling software metrics and stability is obvious for the software practitioner and the reader.

3. The Statement of the Learning Problem
3.1. Modeling of the software quality prediction problem

It is worth mentioning that the concept of stability itself (or conversely instability) is a matter of degree and indeed is a fuzzy concept. This is due in part to the fact that intuitively; stability is an external characteristic of intrinsic character and it is inherently a qualitative not quantitative concept. Empirical investigations of measurable internal attributes and their relationship to external quality characteristics are a crucial issue for predicting software product quality; as pointed out by several authors [35-44] among others. In these context large measures (known as metrics) have been proposed in the literature and have been used in predicting the fault-proneness of classes during design, and for predicting the maintenance effort. More specifically we will investigate the possibility to learn causal relationships between some inheritance metrics, and the stability of OO library interfaces. Moreover, we will propose the interpretation of the results in terms of weighted fuzzy IF-THEN production rules and the relative importance of the variation of the metrics in relation with the stability of the class library. In our modeling we use fuzzy metrics that are linguistic variables defined over term sets (or labels of fuzzy sets) and represented in terms of membership functions (MFs) or possibility distributions. Referring to figure 2, the (in) stability itself as a concept is modeled as a linguistic variable, this allows coping with several levels of stability (or instability). As illustrated in figure 2, the transition from STABLE to UNSTABLE is gradual rather than abrupt, in contrary to the definition of conventional
crisp concept of stability. Two output neurons are needed in order to represent the two concepts of stability and instability as illustrated in figure 2. Stability is allowed herein to be graduated, that is, be a matter of degree.

In our modeling we use Zadeh’s possibility theory [25-26] and more specifically possibility/necessity measures which enables us to accurately estimate how much it is possible that a class is stable (or instable), and how much it is necessary that a class is stable (or instable). We believe that our approach will definitely open the door for intelligent next generation quality prediction systems. Besides learning the causal relationships between the inheritance metrics and the stability, they allow the detection of the importance and/or relevance of each metric to stability which is of paramount importance for an empirical approach of studying for understanding software quality aspects and hence providing justification facilities for the metrics validation issues. Thus enables the understanding of relative importance of each inheritance metric and its influence in the (in) stability of class libraries. Ultimately, this enables us to determine the minimal subset of metrics allowing to predict the stability (or instability) of library class accurately. We assume herein that we are dealing with the stability issues of either a Java or C++ library or any other object-oriented language. The inputs neurons are herein a certain number of structural metrics (Inheritance metrics of class hierarchy, cohesion, and coupling), our aims is at predicting the stability of class or equivalently to capture the degree of stability of the class interface in respects to the evolution of the library from version Vk to an new version Vk+1.

Conventionally, a software quality metric is defined as a function which inputs software data and outputs a single value interpretable as the degree to which software possesses an attribute that affect quality. In departure of conventional methods, we assume herein that a software quality metric is a linguistic variable that might have linguistic values represented by labels of fuzzy sets (such EXTEREMELY SMALL, VERY SMALL, MORE OR LESS SMALL, MEDIUM, MORE OR LESS LARGE, VERY LARGE, EXTEREMELY LARGE) and interpreted by MFs as illustrated in Figure 3. Thus each metric is interpreted by a fuzzy partition or equivalently a fuzzy sequence. Graduation and granulation are used conjointly in modeling a fuzzy metric.

For the sake of simplicity, we assume dealing with the case of simple inheritance. The inheritance metrics that are assumed to capture the evolution of a class interface are (1) those related to the location of the class in the inheritance tree (2) those related to
the ancestors and descendants of the class (3) those connected to the addition, inheritance and overriding of methods, as illustrated in Table 1.

![Figure 3. A fuzzy partition of granularity c=5 that is a superposition of two wave functions representing a fuzzy metric.](image)

**Table 1. Class metrics used and their descriptions**

<table>
<thead>
<tr>
<th>Metric</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>DIT</td>
<td>Depth of inheritance Tree. Measure the size of the longest path from a class to a root class within the same inheritance tree.</td>
</tr>
<tr>
<td>CLD</td>
<td>Class to leaf Depth. Measures the size of the longest path from a class to a leaf class within the same inheritance tree.</td>
</tr>
<tr>
<td>PLP</td>
<td>Position in the longest path. DIT/(CLD+DIT)</td>
</tr>
<tr>
<td>NMA</td>
<td>Number of method added New methods in a class</td>
</tr>
<tr>
<td>NMI</td>
<td>Number of methods inherited. Methods inherited and not overridden</td>
</tr>
<tr>
<td>NMO</td>
<td>Number of methods overridden. Methods overridden</td>
</tr>
<tr>
<td>NOM</td>
<td>Number of methods. NMA+NMI+NMO</td>
</tr>
<tr>
<td>PMA</td>
<td>Percentage of methods added. NMA/NOM</td>
</tr>
<tr>
<td>PMI</td>
<td>Percentage of methods inherited. NMI/NOM</td>
</tr>
<tr>
<td>PMO</td>
<td>Percentage of methods overridden. NMO/NOM</td>
</tr>
<tr>
<td>NOC</td>
<td>Number of children</td>
</tr>
<tr>
<td>NOD</td>
<td>Number of descendants</td>
</tr>
<tr>
<td>NOP</td>
<td>Number of parents. NOP ∈ {0, 1}, which is the case of simple inheritance</td>
</tr>
</tbody>
</table>

3.2. Description of the Learning Process

The learning is parametric as well as structural. It has to deal with the complexity of the problem and to discover appropriate knowledge chunks, and approximation heuristics for the prediction problem at hand. Taking into account the degree of complexity of the problem at hand as well as the empirical knowledge contained in the training set, the learning subsystem:

- Identify explicitly the appropriate fuzzy partition for each variable by learning. They are used only as references to generate fuzzy hypotheses. For each variable
the appropriate number of granules and the slopes of which will be determined during learning. This information could be either kept or thrown away once the learning is completed without loss of information for the system. As they constitutes only means for generating appropriate membership functions of fuzzy rules and are not used during inference.

- Find the appropriate membership functions for both the antecedents and consequents of every potential rule that is needed to model the problem at hand.

- Ultimately, build the appropriate “good” collection of if-then fuzzy rules (the rule base or knowledge base that consists of a set of linguistic rules), that fits “best” the data that consists of I/O pairs of the training set.

In order to build an automatic workable computational multi-pass learning model some design assumptions are made:

- At each cycle for each input variable \( X_i \) the system generates dynamically a fuzzy partition of \( c \) granules (starting with \( c=2 \), and incrementing \( c \) by 1 or 2 at each cycle until reaching a satisfying point). This point constitutes the stopping criterion of our learning mechanism and it reflects the accuracy level required for the system. It is worth mentioning that increasing \( c \) alone does not affect the algorithmic computational complexity of the learning process! It is the number of input variables (\( m \)) of the system when it is very large that affects it significantly. We assume to have a reasonable value for \( m \) which is almost the case in most classes of real world problems.

An output variable may be deal with as an input one, but for the sake of simplicity and programmability we assume that a fuzzy partition is given (known a priori for each output variable) and prepared cautiously by the domain expert. In general, for a given output variable the actions (or classes) are well categorized (the number and names of granules are known) by the domain expert even thought the slopes of associated MFs have to be questioned during learning.

4. Formulating of the Learning Problem
4.1. Hypothesis Generation, Formulation and Testing

How to characterize and to represent a fuzzy partition? What operators are needed in manipulating a fuzzy partition? During learning-time, only one operator is needed to create a fuzzy partition having the required known granularity \( c \). It is the re-partitioning operator. It consists to divide dynamically during learning-time the universe of discourse into \( c \) overlapping granules. It works from scratch, i.e., there is no need for splitting, or fusion or expanding. A partition is used as reference only and its granules do not necessarily constitute MFs for actual rules as they are only used for formulation of initial fuzzy hypotheses during the generation by the systematic exhaustive search algorithm and they are both scale-dependents and context-dependents. We have no other assumption about the fuzzy partition and we are not interested to argue in such matters like “good” partition. The learning will be done at the rule level rather than at the partition level and hence learning a “good” rule is indeed a crucial issue of utmost importance. A fuzzy partition is illustrated in Figure 3 (observe how the rightmost and the leftmost granules are shaped); it is a parameterized family (sequence) of membership functions that cover the universe of discourse for every variable either input or output. It is created dynamically by the execution of the re-partitioning operator...
of granularity equals to c during learning-time. In fact, it is obtained by superposition of two wave functions defined over the same universe of discourse \( X \) ranging in the interval \([a_{\min}, a_{\max}]\). Thus, it is straightforward to extract parameters of granules (MFs) from a given fuzzy partition, as each granule may be considered as an indexed term of the family (or sequence).

A fuzzy partition is represented by vector of \( c \) parameters, where \( c \) is the granularity level. A computationally more efficient way to characterize it is to use a parametric representation of the MFs of its constituents (called fuzzy members). A fuzzy partition might be thought of as a sequence of granules, each of which is represented by an indexed term. This makes sense as they are computed and manipulated easily like ordinary crisp terms during learning-time. In general as illustrated in Figure 3, every value \( x \) of the universe of discourse corresponds to at most two granules. \( A_1, A_2 \ldots A_i \ldots A_c \) are just synthetic linguistic labels interpreted by fuzzy sets of normalized MFs. A fuzzy partition might be thought of as a synthetic alphabet that the system create by learning for future hypotheses generation. Thanks to this flexible scale-dependent representation, regardless the range of the universe of discourse of an input variable, the terms of the fuzzy partition sequence are explicitly expressed straightforwardly as follows:

The first term (or granule)

\[
\mu_{A_1}(x)=\begin{cases} 
\frac{(a_2-x)(a_2-a_1)}{(a_2-a_1)}, & \text{if } a_1 \leq x \leq a_2 \\
1 & \text{otherwise}
\end{cases}
\]

For i=2, 3 \ldots c-1, where \( c \) is the granularity of the partition or the i-th term

\[
\mu_{A_i}(x)=\begin{cases} 
\frac{(x-a_{i-1})(a_i-a_{i-1})}{(a_i-a_{i-1})}, & \text{if } a_{i-1} \leq x \leq a_i \\
\frac{(a_{i+1}-x)(a_{i+1}-a_i)}{(a_{i+1}-a_i)}, & \text{if } a_i < x \leq a_{i+1} \\
0 & \text{otherwise}
\end{cases}
\]

And finally the last term

\[
\mu_{A_c}(x)=\begin{cases} 
\frac{(x-a_{c-1})(a_c-a_{c-1})}{(a_c-a_{c-1})}, & \text{if } a_{c-1} \leq x \leq a_c \\
1 & \text{otherwise}
\end{cases}
\]

4.2. Learning by Hybrid Min-Max Fuzzy-Neuro Network

Fuzzy rules attempt to capture the “rules-of-thumb” approach generally used by software engineers for decision-making and problem solving. Fuzzy (weighted) rules have been advocated, used, studied and interpreted by many authors initially Zadeh [28]; Cayrol et al. [45]; Dubois et al. [46]; Beldjehem [13-18]; Yager[47]) and originally machine learned automatically by Beldjehem[13-18] We will focus in dealing with a multi-input single-output (MISO) system as any multiple-input multiple-output (MIMO) system could be converted to a certain number of MISO systems. Let us start with a model overview: As in Beldjehem [13-18] we consider herein to design a fuzzy-neural possibilistic network according to the scheme Fuzzy to Neural (or to switch from fuzzy systems to neural networks). We use fuzzy if-then weighted rules that are herein of the control type instead of the classification type as in Beldjehem [13-18] and such a rule looks like:
If \((X_1 \text{ is } w_{k1}, c_{k1}) \text{ and } (X_2 \text{ is } w_{k2}, c_{k2}) \text{ and } (X_3 \text{ is } w_{k3}, c_{k3}) \text{ and } (X_5 \text{ is } w_{k5}, c_{k6})\)

Then \(Y_k = V_{ki}\)

\(c_{kj}\) is a weight that represents the grade of importance of “\(X_j \text{ is } w_{kj}\)” in relation with the output \(Y_k\). Thus, conversely the weight \(a_{kj} = 1 - c_{kj}\) represents the grade of unimportance of “\(X_j \text{ is } w_{kj}\)” in relation with the same output \(Y_k\).

Referring to Figure 4, we propose herein a feed-forward fuzzy-neural possibilistic network. We begin with a brief description of the model: two types of weights are associated with the connections. Thus the fuzzy-neuro possibilistic network might be thought of as a transparent learning device of any non-linear mapping of inputs into an output.

![Figure 4. Schematic representation of the hybrid fuzzy-neuro possibilistic Min-Max model used](image)

In fact beyond predicting quality, we are more interested herein by building a class of software prediction tools that justifies and explains its reasoning so that the knowledge and problem solving process is remembered and mimicked by the software engineer in order to tackle the software quality validation and understanding issues. Simply put a system which not only solve the problem of the software prediction but also is able to construct a transparent model for the software engineer and the software project manager (the human problem solver) alike towards understanding those software quality issues under scrutiny and empirically validate novel metrics, ideas and techniques.

Type 1: Direct connections between input cells \((X_j)\) and output cell \((s_k)\) with only synthetic linguistic weights \((w_{kj})\), interpreted as labels of fuzzy sets, characterizing the variations of the input cells (“\(X_j \text{ is } w_{kj}\)” with the output cell \((s_k)\), in this case we have \(a_{kj} = [0,0]=0\). Thus \((\prod(X_j; w_{kj}) \lor 0) = \prod(X_j; w_{kj}).\) Thus the connection between a hidden cell and output cell simply disappears from the graph allowing direct connection.

Type 2: Connections between input cells \((X_j)\) and output cells \((s_k)\) via intermediate cells \((H_{ki})\), weights associated to connections between input cells \((X_j)\) and intermediate cells \((H_{ki})\), are herein artificial or synthetic linguistic \((w_{ki})\), weights associated to connections between intermediate cells \((H_{ki})\), and output cell \((s_k)\) are herein numerical intervals \((a_{kj} \subseteq ([0,1]), \text{ instead of a scalar value ranging in the interval } [0,1])\) \((a_{kj} \in [0,1]).\)
\(w_{kj}\) are unknown artificial or synthetic linguistic weights and \(a_{kj}\) are unknown confidence interval that reflects a domain of possible values of unimportance for the corresponding connections. Thus provides much more flexibility for the network.

A learning session starts with a “blank” fully connected hybrid fuzzy-neuro network without a priori information concerning the weights, i.e. the weights might be thought of as “placeholders” only. Let us consider now cell activation for an arbitrary output cell \((s_k)\), as illustrated in Figure 4, where only connections used in activation of \(s_k\) appear. From the semantic point of view, such a figure reflects a neural representation of an if-then fuzzy weighted rule of control type. Let \(\Pi(X_j; w_{kj}) = \text{Sup} \ [w_{kj} \cap X_j]\) be possibility measure associated to fuzzy sets \(w_{kj}\) and \(X_j\). And let \(\text{N}(X_j; w_{kj}) = \text{Inf} \ [w_{kj} \cap \text{Not} \ X_j]\) be necessity measure associated to fuzzy sets \(w_{kj}\) and \(X_j\). In general our model is governed by the three abstract fuzzy approximate equations as shown below. Thus allowing the manipulation of fuzzy I/O examples and enabling approximate learning reflecting soft mapping, this in fact is a departure from conventional learning algorithms.

\[
\pi_k = \bigwedge_{j \in \{1,2,3,5\}} (\Pi(X_j; w_{kj}) \lor a_{kj}) \\
\eta_k = \bigwedge_{j \in \{1,2,3,5\}} (\text{N}(X_j; w_{kj}) \lor a_{kj}) \\
s_k = [\eta_k, \pi_k] 
\]

Obviously, each output variable will be assigned an interval as illustrated in equation 3; the inputs of the fuzzy-neuro networks represent the software metrics used in predicting the software class (in)stability and two output variables are required in order to represent the two dual fuzzy concepts of stability and instability respectively in terms of possibility/necessity measures. The interpretation by the means of linguistic approximations of the output either stable or instable is as illustrated in Table 2. The process of linguistic approximation consists of finding a label whose meaning is the same or the closest (according to some metric) to the meaning of unlabeled MF (representing either a fuzzy set or an interval) generated by some computational model (learning in our current study).

Observe that Maximum \((\lor)\) limits lower amplitudes of inputs, we have \((\Pi(X_j; w_{kj}) \lor a_{kj}) = a_{kj}\) if \(\Pi(X_j; w_{kj}) \leq a_{kj}\), and amplifies higher ones \((\Pi(X_j; w_{kj}) \lor a_{kj}) = \Pi(X_j; w_{kj})\), if \(\Pi(X_j; w_{kj}) \geq a_{kj}\), so the Min-Max composition indicates a somewhat excitatory character. It is worthwhile to notice that Min-Max composition as containing Min and Max operations is strongly nonlinear. Furthermore, in connections with relational and rule-based fuzzy systems setting such model has been formally validated and it has been shown recently Beldjhem [10-11] that Min-Max composition preserves the value approximation property, i.e. it is a robust system with respect to small perturbations, which is not always the case for the conventional neural networks or the classical Bayesian models. Observe that when \(a_{kj} = 1\), the term \(\Pi(X_j; w_{kj}) \lor a_{kj}\) (respectively \(\text{N}(X_j; w_{kj}) \lor a_{kj}\)) is deleted in the application of Minimum \((\land)\). Thus ensures the interpretability and transparency of the model. It is now clear that \(a_{kj}\) reflects a notion of unimportance, we point out herein that it is strongly hard if not impossible to make values assignment to grades of unimportance in practical applications, we will propose a mechanism to learn such grades of unimportance. See Table 3, which reflects the metric’s effect in relation with the (in) stability of a given class in our framework. In connection with our problem of software quality prediction, semantically, missing edge
reflects the non-influence of the input (of the corresponding metric) in the appearance of the output (either the stability or instability) of the class. This enables us to determine the minimal subset of metrics allowing an accurate and effective prediction.

Table 2. The linguistic approximations of certainty values

<table>
<thead>
<tr>
<th>Certainty value $S_i$</th>
<th>Linguistic approximation</th>
</tr>
</thead>
<tbody>
<tr>
<td>[0, 0]</td>
<td>IMPOSSIBLE</td>
</tr>
<tr>
<td>[0, 0.05]</td>
<td>ALMOST IMPOSSIBLE</td>
</tr>
<tr>
<td>[0, 0.1]</td>
<td>SLIGHTLY IMPOSSIBLE</td>
</tr>
<tr>
<td>[0, 0.65]</td>
<td>MODERATELY POSSIBLE</td>
</tr>
<tr>
<td>[0, 1]</td>
<td>POSSIBLE</td>
</tr>
<tr>
<td>[0.35, 1]</td>
<td>QUITE POSSIBLE</td>
</tr>
<tr>
<td>[0.9, 1]</td>
<td>VERY POSSIBLE</td>
</tr>
<tr>
<td>[0.95, 1]</td>
<td>ALMOST SURE</td>
</tr>
<tr>
<td>[1, 1]</td>
<td>SURE</td>
</tr>
</tbody>
</table>

Table 3. The metric’s effect in relation with the instability of a given class

<table>
<thead>
<tr>
<th>The magnitude of the metric’s effect</th>
<th>Non-importance</th>
</tr>
</thead>
<tbody>
<tr>
<td>Does not effect</td>
<td>[0.9, 1]</td>
</tr>
<tr>
<td>Minimal effect</td>
<td>[0.8, 0.9]</td>
</tr>
<tr>
<td>Small effect</td>
<td>[0.6, 0.8]</td>
</tr>
<tr>
<td>Medium effect</td>
<td>[0.4, 0.6]</td>
</tr>
<tr>
<td>Large effect</td>
<td>[0.1, 0.4]</td>
</tr>
<tr>
<td>Very large effect</td>
<td>[0, 0.1]</td>
</tr>
<tr>
<td>Partial ignorance</td>
<td>[0, 1]</td>
</tr>
</tbody>
</table>

5. Resolution of the Learning Problem
5.1. The Learning Algorithm and Implementation Issues

During a learning session the same learning algorithm is used for each output variable $Y_j$. Let us briefly describe the learning algorithm that is composed of many cycles, each of which is executed as follows: For each output variable $Y_j$ and for each granule belonging to the fuzzy partition that corresponds to $Y_j$, iteratively, an initial fuzzy hypothesis corresponds to a combination of certain number of MFs (each of which corresponds to granule of an input variable) is created (formed) by a systematic exhaustive search procedure. Once a fuzzy hypothesis is formed it is loaded or incorporated in the hybrid fuzzy-neuro network weights for test purposes, its components (elements) will be adjusted to fit the training data. Such hypothesis is considered as a potential candidate to be a rule and then is questioned and adjusted during learning by the means of a hybrid fuzzy-neuro possibilistic network using a successive approximation algorithm of systems of Min-Max relational equations. This adjustment is repeated until finding the ones that minimize the signal error. Hence another new combination is then generated and we repeat the same procedure. Thus the obtained adjusted hypotheses that minimize the cost over all possible combinations and that were embedded in the weights of the hybrid fuzzy-neuro possibilistic network are kept in a temporary learning table.
The algorithm proceeds by increasing the granularity and repeats the same cycle, until reaching a satisfying point. In general the learning is stopped when either a certain level of accuracy has been reached or it is impossible or it is computationally worthless to seek minimizing the error much more, i.e. this situation means that increasing the granularity is no more interesting. In general this point constitutes a trade-offs between tractability and low cost solution. Learning need to find an approximate solution that is not necessarily precise (or crisp) optimal one but at the same time it builds a model that do manage to resolve the problem at hand effectively. At the end one or more of the obtained adjusted hypotheses that minimize the cost (over all considered granularity levels) constitutes a valid hypothesis and is transferred and stored in a knowledge base (KB) of the system as it consists effectively of a new learned rule. The system check whether or not a rule is new, i.e. whether or not it is already included the KB, and if necessary, transmits it to the KB, in an intelligible form for the storage (hash table data structure). Assume the system get two or more valid hypotheses, after checking each one, each one is eventually added to the KB as a new rule. The advantage is that by construction (learning) we build a production system with no contradictory rules and thus giving a high satisfactory performance. This is in fact a built-in quality attribute.

Thanks to these granular functionalities, this novel learning algorithm constitutes a departure from the conventional ones, in that it conjointly determine dynamically during the learning-time the required satisfying number of rules necessary to model the problem as well as the rules themselves explicitly. Intuitively, this number is proportional to the degree of complexity of the problem at hand.

The resolution of fuzzy relations equations constitutes a good tool in fuzzy modeling especially for dealing with inverse problems. The fuzzy relational calculus theory (Di Nola et al. [48]; Beldjehem [13-18]) provides us with a set of analytic formulas expressing solutions for some types of equations and their systems. However, the existence of solutions of the system is not known in advance. This makes any preliminary analysis rather tedious if not impossible. We reformulate the problem of solving a system of Min-Max from interpolation-like format to approximation-like one. This means that instead of trying to find exact solution, we try to find the best approximate solution. Any scalar and any element of vectors or matrices are assumed to have its value in the interval [0, 1]. Formally; our problem can be stated as follows: "Given an \( m \times n \) matrix \( R \) and an \( n \) vector \( b \), find an \( m \) vector \( a \) such that \( (a \bigtriangleup R \supseteq b) \) where \( \bigtriangleup \) is the Min-Max composition and \( \supseteq \) denotes the fuzzy inclusion operation. Let us consider the case when there is no solution for the system (it does not satisfy the necessary condition, i.e. \( a \bigtriangleup R \supseteq b \)). This can be also reflected by only computing a distance. Let \( A, A' \) be fuzzy subsets of \( U \) and \( a, a' \) be the corresponding grades of membership vectors. By \( ||a-a'|| \) we denote the number \( \max_i (|a_i-a'_i|) \), i.e. the maximum of the absolute values of the differences between all element of \( a \) and \( a' \). It might be interpreted as the signal error subject to be minimized. Equivalently by using this distance rather than the fuzzy inclusion concept we get the same results; and for this reason we use such a distance \( ||a \bigtriangleup R \setminus b|| \) in our implementation of the system. It corresponds to minimal distance, hence \( a \) is the best approximator. Thus, since our algorithm is valid for both interpolation-like and approximation-like formats, it allows to resolve the more general following problems: "Given an \( m \times n \) matrix \( R \) and an \( n \) vector \( b \), find all \( m \) vectors such that \( a \bigtriangleup R \supseteq b \)". This algorithm is used as approximation procedure by the learning algorithm in our system. The learning consists
mainly in crunching (approximating) systems of Min-Max equations while manipulating abstract synthetic linguistic concepts (labels, hypotheses). It can be shown that the best approximator (from the fuzzy inclusion point of view) corresponds to the lower bound \( a \) of the inf-semi-lattice. It can be computed straightforwardly using the \( \varepsilon \) resolution operator only. It has been shown in Beldjehem [13] by a worst-case analysis that our computing algorithm has a linear complexity of \( \Theta (m \times n) \) with respect to \( n, m \) is a constant. In order to illustrate the functioning and the behavior of our approximation algorithm let us hand-execute it on the following example, \( R \) and \( b \) are known. The \( \varepsilon \) resolution operator has initially been defined in Beldjehem [13] as follows:

\[
\begin{align*}
 x \in \varepsilon y = \begin{cases} 
 y & \text{if } x < y \\
 0 & \text{otherwise}
\end{cases}
\end{align*}
\]

\[
R = \begin{bmatrix}
0.5 & 0.6 & 0.1 & 0.3 & 0.6 \\
0.7 & 0 & 0.8 & 0.4 & 0.7 \\
0.8 & 0.3 & 0.5 & 0.7 & 0.6 \\
0.4 & 0.8 & 0.6 & 0.8 & 0.7 \\
0.4 & 0.4 & 0.7 & 1 & 0.6 \\
0.9 & 1 & 1 & 1 & 0.8
\end{bmatrix}
\]

\[
b = \begin{bmatrix} 0.3 & 0.3 & 0.5 & 0.4 & 0.5 \end{bmatrix}
\]

Firstly, we compute the lower bound \( a \) of the inf-semi-lattice

\[
a = \vee (R \vdash b), \text{ where } \vee \text{ stands for } \text{MAX}
\]

\[
a = [0.5 \ 0.3 \ 0 \ 0 \ 0 \ 0]
\]

Then, by performing \( \Delta \) the Min-Max composition, we obtain \( a \Delta R \)

\[
b = [0.3 \ 0.3 \ 0.5 \ 0.4 \ 0.5] \text{ (the target vector)}
\]

\[
a \Delta R = [0.4 \ 0.3 \ 0.5 \ 0.4 \ 0.6]
\]

\[
\| a \Delta R - b \| = 0.1 \text{ (the error signal)}
\]

Compare \( b \) and \( a \Delta R \), observe the surprising remarkable approximating power of \( a \)

### 5.2. Abstract Computational Model of a Learning Session

The computational abstract model of learning implements a kind of successive approximation of Min-Max system process, and find weights of the hybrid fuzzy-neuro networks that fits “best” the data that consists of pairs I/O of the training set. Formally, from the computational point of view, for each output \( s_k \), a learning session consists to resolve or rather to approximate \((r+1)\) systems of Min-Max equations, as follows:

\[
a \Delta R^{(0)} \supseteq b
\]

\[
a \Delta R^{(1)} \supseteq b
\]

...
\[ a \Delta R^{(i)} \supseteq b \]

Learning consists to prefer (generate and validate) the configuration (the fuzzy hypothesis) of the best approximate solution (from the fuzzy inclusion point of view), i.e. which minimizes the local cost function and hence the corresponding deep structure. In other terms the learning process finds incrementally the "best" deep structure which corresponds to the following matrix \( R^{(i)} \): \( l \in [0, r] \) such that:

\[ a \Delta R^{(i)} \supseteq a \Delta R^{(j)} \supseteq b, \quad \forall \ j = 0 \ldots r \]

Or equivalently,

\[ \| a \Delta R^{(i)} - b \| \geq \| a \Delta R^{(j)} - b \|, \quad \forall \ j = 0 \ldots r \]

Learning tries progressively by successive approximation to minimize the local cost function by the generation and the approximation of a new system. Thus, this approximation algorithm constitutes the mathematical machinery of learning. It has been shown that this system is a universal approximator Beldjehem [10-11]. In general the value of the local cost function may be seen as a quality index for a learning session or a performance index for the system. Learning has high speed due to its simplicity and analytic nature. Indeed the fuzzy learning process may be thought of as a new kind of algorithmic fuzzy optimization or rather an algorithmic fuzzy approximation.

### 6. Concluding Remarks and Future Works

We have developed a cognitively motivated granular computational framework for learning fuzzy systems from software repositories and have illustrated how to apply it properly and effectively in order to resolve a software quality prediction problem. This allows the automatic learning of fuzzy if-then quality prediction rules of object-oriented software systems which are large scale, with very large software repositories, too complex or too ill-defined to admit of precise quantitative analysis, description or quality control strategy. Such a framework integrates conjointly both the perceptual and the cognitive aspects of the human problem-solving process and ensures a granular processing of the underlying input from different granularity levels. It is the first attempt in the field. During learning-time the system finds automatically the adequate levels of details (granularities) for the problem at hand. It is possible using a linguistic approximation to learn automatically completely a true linguistic fuzzy prediction system. The similar promising alternative that constitutes another candidate solution is to use an evolutionary algorithms (EA) as in Pedrycz [49]; Falkenauer [50]; Cordon et al. [51]. EAs can easily handle the multi-parameter problems of software quality prediction and are certain to grow in visibility, importance and scope of applicability.

Besides learning the causal relationships between the inheritance metrics and the stability, it allows the detection of the importance and/or relevance of each metric to stability which is of paramount importance for an empirical approach of studying for understanding software quality aspects and hence providing justification facilities for the metrics validation issues. The understanding of relative importance of each inheritance metric and its influence in the (in) stability of class libraries, will enables us to determine the minimal subset of metrics allowing to predict the stability (or instability) of library class and thus might implement an effective software tool as a front end assistant enabling software designers, programmers and project managers to
predict the software quality and to handle, to rework, to refactor, to plan, to monitor and control early the related software development activities especially those connected to “programming in the large.” of complex large scale software projects.

Our framework could accommodate and tackle easily three similar and related problems that consist to measure (1) reusability (2) class fault-proneness, and (3) cost estimation. We propose also to extend our investigation to comparing and assessing our approach with related work in terms of accuracy, performance-interpretability tradeoffs and finally to draw conclusions and propose future promising directions for research. We are exploring to use our framework for the purposes of rework and/or refactoring in order to improve the design of existing code as in Fowler [52].

Granular soft computing, software engineering, machine learning, simulation, performance evaluation have to learn from each other, and could be integrated or fused synergistically (not competitively) and melded seamlessly in order to build next generation of intelligent software (quality, effort and cost) prediction systems. Such systems are needed acutely and even required in mining very large software repositories and thereby coping with the evolution and maintenance of complex large-scale software systems and dealing with their ever changing requirements and dynamic environments.

7. References

Authors

Dr. Mokhtar Beldjehem research interests include fuzzy & soft computing, computational intelligence, software design, data-driven & simulation-based software engineering, mining software repositories, decision analysis & support, machine learning, image processing, pattern recognition, information retrieval, Web-centric computing, E-learning, E-Health, ALife.