IEEE SSCI2011

APRIL 11-15, 2011
PARIS, FRANCE

SYMPOSIUM SERIES ON COMPUTATIONAL INTELLIGENCE

www.ieee-ssci.org

CIDUE 2011

2011 IEEE Symposium on
Computational Intelligence in Dynamic and Uncertain Environments

Table of Contents
Technical Sessions
Author Index

ISBN: 978-1-4244-9929-8
IEEE Catalog Number: CFP1106N-CDR

Technical Support:
Chris Dyer
Conference Catalysts, LLC
Phone: +1 785 341 3583
cdyer@conferencecatalysts.com

IEEE
IEEE Computational Intelligence Society

Organized and sponsored by the IEEE Computational Intelligence Society

2011 IEEE Symposium on Computational Intelligence in Dynamic and Uncertain Environments

(CIDUE 2011) Proceedings

Additional copies may be ordered from:
IEEE Service Center
445 Hoes Lane
Piscataway, NJ 08855-1331 USA

+1 800 678 IEEE (+1 800 678 4333)
+1 732 981 1393
+1 732 981 9667 (FAX)
email: customer-service@ieee.org

IEEE Catalog Number: CFP1106N-CDR
ISBN: 978-1-4244-9929-8

TABLE OF CONTENTS

16:00 - 16:30

PS8: CIDUE - 2011
Poster Session

16:30 - 17:30

S64: CIDUE - Keynote
Chair: Robi Polikar (Rowan University, USA)

From Heuristics to Statistics: an Overview of the ADEPT Project
Gavin Brown (University of Manchester, United Kingdom

IEEE CIDUE 2011 Committee

Symposium on Computational Intelligence in Dynamic and Uncertain Environments (IEEE CIDUE 2011)

Computational Intelligence (CI) methodologies, including evolutionary algorithms, neural networks and fuzzy systems have shown to be sell suited to deal with significant uncertainties that may be encountered in solving real-world problems. The purpose of this symposium is to bring together scientists, engineers, and graduate students to present and discuss recent advances in employing CI for solving scientific and engineering problems in the presence of uncertainties.

Symposium Co-Chairs
Yaochu Jin, University of Surrey, UK
Shengxiang Yang, Brunel University, UK
Robi Polikar, Rowan University, USA

Program Committee
Cesare Alippi, Politecnico di Milano, Italy
Gavin Brown, University of Manchester, UK
Chaochang Chiu, Yuan Ze University, ROC
Ernesto Costa, University of Coimbra, Portugal
Moufid Harb, Larus Technologies Corp., Canada
Haibo He, University of Rhode Island , USA
Yan Meng, Stevens Institute of Technology, USA
Ferrante Neri, University of Jyvaskyla, Finland
Yew-Soon Ong, Nanyang Technological University, Singapore
David Pelta, University of Granada, Spain
Hendrik Richter, University of Leipzig, Germany
Chuan-Kang Ting, National Chung Cheng University, ROC
Mauro Tucci, University of Pisa, Italy
Sima Uyar, Istanbul Technical University, Turkey

Fuzzy Systems for Scenario Modelling and Analysis

Qiang Shen
Aberystwyth University, United Kingdom

Solving complex real-world problems usually requires timely and intelligent decision-making. This requires the analysis of a large volume of vague and indiscernible information. For example, intelligence experts have commented that the failure in detection of terrorist activity is often due to the difficulty in relating and interpreting the available intelligence on time, rather than a lack of data. Whilst experienced analysts can suggest plausible scenarios, the sheer amount of possibly relevant data may not be humanly interpretable in a short time-frame. The hypothetical (re-)construction of the activities that may have generated the intelligence data, therefore, presents an interesting and significant research topic. This talk will present a knowledge-based framework for the development of intelligent decision support systems under uncertain environments. This helps to assist (but not to replace) intelligence analysts by: a) identifying plausible scenarios of criminal or terrorist activity, and b) assessing the reliability, risk and urgency of generated hypotheses. In particular, the talk will introduce an integrated use of some recent advances in fuzzy systems for the monitoring and interpretation of intelligence data. Employing such advanced approaches offers an effective means for the generation and assessment of plausible scenarios. The work has the potential to facilitate rapid response in devising and deploying preventive measures. This talk will conclude with a discussion about the expansion of the techniques to encompass novel problem domains as well as some important challenges which lie ahead.

From Heuristics to Statistics: An Overview of the ADEPT project

Gavin Brown
University of Manchester, United Kingdom

ADEPT (Adaptive Dynamic Ensemble Prediction Techniques) is a multi-site UK research council project, aiming to capitalize on the synergistic interface between three fields: evolutionary computation, ensemble learning, and probabilistic modelling. I will present our experiences with a paradigm from the evolutionary computation literature - Learning Classifier Systems - and their subsequent translation into an ensemble-based probabilistic model. The probabilistic model can precisely reproduce the capabilities of the LCS - an online supervised learning system, continuously adaptive, maintaining a parsimonious set of human-interpretable rules. However, the new model stands apart from the parameter-laden heuristic nature of LCS, having the advantages of a statistical underpinning: flexibility and a solid probabilistic foundation. This talk will give a whistlestop tour of the project, from the first experiences with voting systems and Adaboost, through to our current emphasis on non-stationary rule learning.

Diagnosis of Software Erosion through Fuzzy Logic

Ricardo Pérez-Castillo, Ignacio García Rodríguez de Guzmán and Mario Piattini
Alarcos Research Group, University of Castilla-La Mancha
Paseo de la Universidad, 4 13071,
Ciudad Real, Spain
{ricardo.pdelcastillo, ignacio.grodriguez, mario.piattini}@uclm.es

Abstract— **Companies have a vast number of existing software systems, which are not immune to software erosion and ageing as a consequence of uncontrolled maintenance over time. Currently, there are several metrics to measure and quantify software erosion, which also recommends some maintenance actions to deal with software erosion. Unfortunately, there are many symptoms at the same time and several possible maintenance actions that could be carried out. As a consequence, this uncertain environment implies that the best set of actions is unknown and cannot be certainly linked to specific detected erosion symptoms. This paper provides a fuzzy rule-based system to address that challenge. The system is divided into two levels: the first one recognizes precise software erosion metrics and provides fuzzy software erosion symptoms; and the second one takes the fuzzy symptoms and finally obtains fuzzy maintenance actions. This system is therefore a decision-making mechanism to select the best set of actions depending on the specific software erosion symptoms. This system has been implemented using the Matlab Fuzzy Logic Toolbox and it was simulated using Simulink.**

Keywords. Fuzzy Rule-Based System; Software Erosion; Maintenance; Decision-Macking.

I. INTRODUCTION

According to the Lehman's first law, a software system must continually evolve or it will become progressively less suitable in real-world environments [8]. Indeed, companies count on a vast number of existing software systems which are not immune to software erosion and software ageing, i.e., existing software systems that become progressively less maintainable [14].

The successive changes in a software system degrade its quality, and thus, a new and improved system should replace the previous one. However, the wholesale replacement of these systems from scratch is risky since it has a great impact in technological, human and economic terms [7, 16]. The technological and human point of view is affected since replacement would involve retraining all the users in order to understand the new system and the new technology, or the new system may lack specific functionalities that are missing due to the technological changes. Moreover, the economic point of view is also affected since the replacement of an entire legacy system implies a low Return of Investment (ROI) in that system. In addition, the development or purchase of a new system could exceed a company's budget.

In order to address the phenomenon of software erosion, the evolutionary maintenance is a better solution to obtain improved systems, without discarding the existing systems, thus minimizing the software erosion effects. Evolutionary maintenance makes it possible to manage controllable costs and preserves the valuable business knowledge embedded in the legacy system, since 78% of maintenance changes are corrective or behavior-preserving [3].

When companies are faced with the phenomenon of software erosion, they have two main challenges. Firstly, they should know how their systems' erosion levels are, i.e., the erosion symptoms. Secondly, companies should know the best set of maintenance actions to carry out in order to solve, or at least mitigate, those detected symptoms. In addition, the selected actions should be carried out carefully without committing more erosion problems. There are some works in the literature addressing software erosion symptoms detection. A common and widely-used diagnosis framework was proposed by *Vissagio* [19]. That framework recognizes a set of symptoms for the diagnosis of software erosion, as well as a set of formal metrics to measure those symptoms. In addition, this framework provides some maintenance actions to address each symptom. TABLE I summarizes symptoms, metrics and maintenance actions proposed by *Vissagio* [19].

The metrics proposed in that framework can be accurately scored by observing software systems and counting the specific elements (e.g., the number of clone programs, number of dead lines of source code, and so forth). However, the recommended maintenance actions usually are carried out in an uncertainly environment, since there are similar actions that are the same for different erosion symptoms (e.g. refactor, reverse engineering, etc). In addition, some actions while solve some symptoms could make another symptoms worse. Establishing certainly relationships between symptoms and actions is therefore a hard task.

This paper proposes a fuzzy rule-based system to address the uncertainty in the decision-making process related to select the most appropriate set of maintenance actions to eradicate software erosion symptoms. The objective of this paper is to propose a fuzzy diagnosis system to detect software erosion symptoms, which makes it possible to select the most appropriate set of maintenance actions. That is, a set of actions that reduce the maintenance effort (and therefore the maintenance cost), and minimize the erosion symptoms in a higher level.

978-1-4244-9929-8/11/$26.00 ©2011 IEEE

TABLE I. FRAMEWORK TO MEASURE EROSION SYMPTOMS AND RECOMMENDED MAINTENANCE ACTIONS (ADAPTED FROM [19])

Sint.	Metric	Description	Action
Pollution	Clone Programs	There are duplicate programs from a functional viewpoint	Indentify most update version and remove the remaining clones
	Obsolete Programs	There are source code files without its corresponding executable file	Remove obsolete programs
	Sourceless Programs	There are executables files without its corresponding source file to be maintained	Rewrite source code by means of reverse engineering
	Dead Data	There are created data that are not used by the programs	Remove pieces of source code that create dead data
	Dead Code	There are pieces of source code that cannot be reached by the control flow	Remove dead code
Missing Knowledge	Missing Documentation	There are pieces of source code without documentation	Re-document through reverse engineering
	Missing Functionalities	There are some functionalities that are not met for any program	Create, split or modify programs to support those functionalities
	Poor Lexicon	There are data and programs with inconsistent names	Rename and refactor
Coupling	Pathological Files	There are files that can be created or modified by several programs	Refactor by means of reverse engineering
	Control Data	There are data that control the execution flow of several programs	Refactor by removing control data
Anomalous Data	Useless Data	There are external data (e.g., databases) that are not used for any program	Remove programs that create obsolete data
	Obsolete Data	There are external data files created by a program that are not updated by any program	Remove programs that create obsolete data
	Semantic Redundant Data	There are external data semantically equal or contained in other one	Remove synonym data
	Computational Redundant Data	There are derived data that are calculated with the same value in database	Remove equivalent external data

The design of the proposed fuzzy rule-based system follows the *Mandami* fuzzy controller configuration [9, 10], which is also knows as Fuzzy Inference System (FIS). The proposed FIS is justified by the fact that it can deal with the uncertainty [20], which mainly appears in three key parts of the FIS.

- The software erosion symptoms (e.g., pollution, missing knowledge, coupling and anomalous data) are a combination of some metrics and they cannot be accurately established for a particular software system. Each software erosion symptom can be defined by a fuzzy set establishing the level (between 0 and 1) which the symptoms appear in a software system.
- Metrics represent input, certain variables that decide each erosion symptoms. These precise values are taken from the measurement of certain aspects of the software (e.g., number of clone programs, number of dead data, etc). Nevertheless, each precise variable can become a fuzzy set in order to achieve fuzzy values of each erosion symptoms.
- Output variables of the proposed FIS system are a set of maintenance actions. These actions are carried out in a fuzzy way. This is due to the fact that there is an uncertainty derived by the search of an agreement between cost and benefit (in terms of software erosion reduction) of the maintenance actions. As a consequence, the fuzzy definition of the output variables can establish fuzzy rules between inputs and outputs in the proposed FIS system.

The remaining of this paper is organized as follows. Section II briefly show related work with this paper. Section III presents in detail the proposed FIS system. Section IV provides an implementation of the systems and Section V simulates the FIS system with a real-life software system. Finally, Section VI discusses conclusions and future work.

II. RELATED WORK

Software maintenance is a time-consuming and hard task, which requires most effort than software development throughout the software lifecycle [6]. The detection of software erosion in the maintenance activity is a key task to know if new maintenance actions are (or are not) necessary. For this reason, maintenance levels measurement has been widely studied in literature for many years.

Hall et al. [4] provided a set of relations between some metrics and specific demands in different maintenance areas. *Basili et al.* [2] presented a study to deal with the prediction of maintenance process, although that work does not focus on the software erosion symptoms. *Lehman et al.* [8] also take into account the evolution of some metrics related to the maintenance activity. *Hayes et al.* [5] provide a recent model to estimate the human maintenance effort related to some maintenance metrics. However, that work does not consider the software erosion metrics and its relation with specific maintenance actions. *Vissagio* [19] provides a framework focusing on the relationship between software erosion metrics and the needed maintenance.

All this work does not take the uncertain maintenance environments into account. For this reason, some works try to solve this problem through the fuzzy logic. For instance, *Ning et al.* [13] provide a learning system to predict software erosion. However, that work ignores the recommended maintenance actions, and in addition it focuses on application server. *Mittal et al.* [11, 12] provide a fuzzy logic technique to measure the maintainability level of software systems, but they do not find out the best set of maintenance actions either.

In contrast, this paper proposes a fuzzy rule-based system to detect the level of a set of software erosion symptoms, as well as to recommend the most appropriate set of actions depending on the recognized symptoms. The main advantage of our proposal is that it not only employs fuzzy logic in the input (erosion symptoms), but also the output (maintenance actions) are treated through fuzzy logic. The uncertainty level of maintenance environments is therefore reduced by means of our proposal.

III. FUZZY INFERENCE SYSTEM

The proposed FIS system consists of a fuzzy rule-based system. This system considers metrics related to software erosion symptoms as inputs and provides a set of recommended actions as outputs. The architecture of the proposed FIS system (see Figure 1) consists of five sub-systems organized in three levels:

- *Metric level* offers the input of the system, and is defined by the precise values measured from software systems according to the metrics presented in TABLE I. This level does not organize any subsystem.
- *Symptom level* adapts each precise metric value to a specific fuzzy set. These fuzzy variables are the inputs of four fuzzy rule-based subsystems, i.e., one subsystem for each software erosion symptom (see TABLE I). Subsystems establish fuzzy rules to obtain the four fuzzy values for each symptom, i.e., pollution, missing knowledge, coupling and anomalous data (see Figure 1).
- *Diagnosis level* defines a finite set of six maintenance actions according to the *Vissagio* framework (see Figure 1). This level contains the last subsystem, which takes the symptoms variables (obtained in the previous level) as input and generates fuzzy values concerning the maintenance actions as output (see Figure 1).

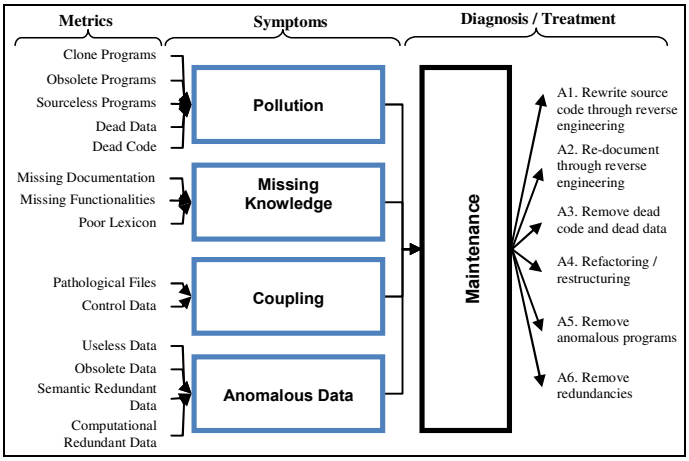


Figure 1. The proposed FIS architecture

The FIS system allows maintainers to evaluate a software system. The FIS system provides specific membership degrees for each maintenance actions (established as fuzzy sets). A maintenance expert takes that information to modify the software system to mitigate its erosion symptoms. The proposed FIS system makes it possible the iterative and incremental maintenance according to the following process.

Firstly, maintenance experts prioritize the set of maintenance actions by arranging the membership degrees of each action. Maintainers carry out the action with the highest membership value, then the second one, and so on. However, after the software system has been modified, some erosion symptoms might be reduced, but other symptoms could be worse. As a consequence, maintainers could carry out solely the first action, and then reevaluate the software system through the FIS system in order to know how the erosion symptoms have changed.

Secondly, maintainers observe what symptoms were eradicated after the first iteration. When the software system is again evaluated, new membership degrees are obtained for each action. Maintainers start here a new iteration following the same steps. This incremental and iterative process finish when the membership degrees of actions represent negligible values.

Moreover, maintenance experts who know the budget of the company and the available resources must establish a threshold value to stop the iterative process. For instance, if a company has a small budget for software maintenance, the maintainers can establish a higher threshold value around 0.5. This means that the maintenance actions with a lower membership degree than 0.5 (the threshold) will be ignored. As a consequence, the proposed FIS system aids the decision-making process when companies want to maintain their software systems.

The following subsections presents in detail the five subsystems grouped into the symptom level and diagnosis level (see Figure 1). Firstly, Subsection III.A provides the fuzzy set inputs for the four first levels related to erosion symptoms. Secondly, Subsection III.B presents the fuzzy set definition for the maintenance actions, the output of the last subsystem. Finally, Subsection 0 specifies all the rules concerning the five subsystems.

A. Software Erosion Symptoms and Input Variables

The software erosion symptoms level (see Figure 1) consists of four FIS subsystems. Each subsystem is in charge of evaluation of a symptom. Our proposal is based in the framework proposed by *Vissagio* [19], which characterizes the erosion of software systems by four symptoms: pollution, missing knowledge, coupling and anomalous data.

Each subsystem takes certain values since according to a set of metrics, which are directly quantifiable by observing software systems. Both specific metrics (as input) and erosion symptoms (as outputs) are defined as a fuzzy set. TABLE II shows the fuzzy set definition of all the metrics. Each fuzzy sets defines its domain between 0 and 1, provides a set of linguistic labels, and specifies the specific trapezoids in numerical and graphical way for each label. TABLE III specifies the fuzzy sets for the subsystem outputs related to each software erosion symptom.

The identified fuzzy sets were defined in a heuristic manner by taking into account the opinion of several maintenance experts. After collect information from different experts, that information was merged using the Delphi technique [15], which allows achieving a workable consensus within time limits. Despite this proposal, different fuzzy sets and linguistic labels could be proposed for the FIS system. Indeed, the mechanism employed in this paper to obtain the fuzzy sets is not the most appropriate. For instance, a better definition could imply a massive collection of information about real-life maintenance and development projects. Thereby, the fuzzy set could be obtained from this information through statistical analyses, or by means of learning systems based on fuzzy logic [1]. However, the definition of the optimal fuzzy sets is outside of the scope of this paper.

Moreover, domains of all the fuzzy sets are defined between 0 and 1 considering the density ratio of each metric, since software systems can have different sizes. For instance, the cloned program metric is defined as the number of duplicate programs divided into the total number of programs

of the software system (i.e., it represents the percentage of cloned programs). The remaining of metric values are accurately calculated in a similar way.

TABLE II. FUZZY SETS FOR THE INPUT METRICS

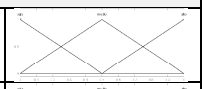
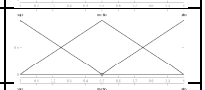
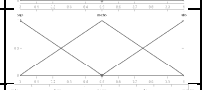
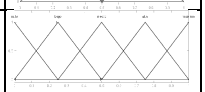
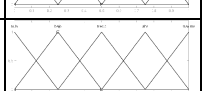
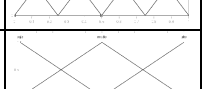
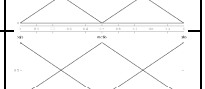
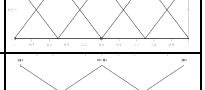
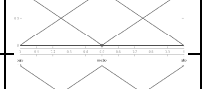
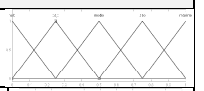
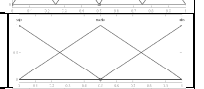
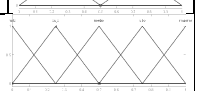
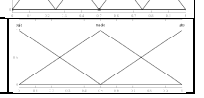
St	Metric	Linguistic Label	Trapezoids	
Pollution	Clone Programs	{Low, Medium, High}	{[-0.5 0 0.5], [0 0.5 1], [0.5 1 1.5]}	
	Obsolete Programs	{Low, Medium, High}	{[-0.5 0 0.5], [0 0.5 1], [0.5 1 1.5]}	
	Sourceless Programs	{Low, Medium, High}	{[-0.5 0 0.5], [0 0.5 1], [0.5 1 1.5]}	
	Dead Data	{Null, Low, Medium, High, Maximum}	{[-0.25 0 0.25], [0 0.25 0.5], [0.25 0.5 0.75], [0.5 0.75 1], [0.75 1 1.25]}	
	Dead Code	{Null, Low, Medium, High, Maximum}	{[-0.25 0 0.25], [0 0.25 0.5], [0.25 0.5 0.75], [0.5 0.75 1], [0.75 1 1.25]}	
Missing Knowledge	Missing Documentation	{Low, Medium, High}	{[-0.5 0 0.5], [0 0.5 1], [0.5 1 1.5]}	
	Missing Functionalities	{Low, Medium, High}	{[-0.5 0 0.5], [0 0.5 1], [0.5 1 1.5]}	
	Poor Lexicon	{Low, Medium, High}	{[-0.5 0 0.5], [0 0.5 1], [0.5 1 1.5]}	
Coupling	Pathological Files	{Null, Low, Medium, High, Maximum}	{[-0.25 0 0.25], [0 0.25 0.5], [0.25 0.5 0.75], [0.5 0.75 1], [0.75 1 1.25]}	
	Control Data	{Null, Low, Medium, High, Maximum}	{[-0.25 0 0.25], [0 0.25 0.5], [0.25 0.5 0.75], [0.5 0.75 1], [0.75 1 1.25]}	
Anomalous Data	Useless Data	{Low, Medium, High}	{[-0.5 0 0.5], [0 0.5 1], [0.5 1 1.5]}	
	Obsolete Data	{Low, Medium, High}	{[-0.5 0 0.5], [0 0.5 1], [0.5 1 1.5]}	
	Semantic Redundant Data	{Low, Medium, High}	{[-0.5 0 0.5], [0 0.5 1], [0.5 1 1.5]}	
	Computational Redundant Data	{Low, Medium, High}	{[-0.5 0 0.5], [0 0.5 1], [0.5 1 1.5]}	

TABLE III. FUZZY SETS FOR THE EROSION SYMPTOMS

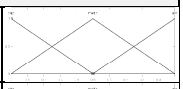
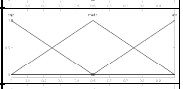
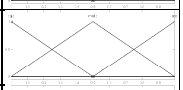
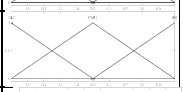
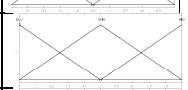
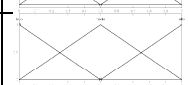
Erosion Symptom	Linguistic Label	Trapezoids	
Pollution	{Null, Low, Medium, High, Maximum}	{[-0.25 0 0.25], [0 0.25 0.5], [0.25 0.5 0.75], [0.5 0.75 1], [0.75 1 1.25]}	
Missing Knowledge	{Low, Medium, High}	{[-0.5 0 0.5], [0 0.5 1], [0.5 1 1.5]}	
Coupling	{Null, Low, Medium, High, Maximum}	{[-0.25 0 0.25], [0 0.25 0.5], [0.25 0.5 0.75], [0.5 0.75 1], [0.75 1 1.25]}	
Anomalous Data	{Low, Medium, High}	{[-0.5 0 0.5], [0 0.5 1], [0.5 1 1.5]}	

B. Maintenance Actions and Output Variables

The diagnosis level depicted in Figure 1 contains a sole FIS subsystem. This subsystem takes as input the fuzzy outputs of symptoms subsystems of the previous level, and it generates as final output the fuzzy values for each maintenance action. For this purpose, six predefined actions (identified from *A1* to *A6*) have been selected according to the *Vissagio* framework (see TABLE I).

- *A1. Rewrite source code through reverse engineering*. Reverse engineering techniques are used to discover embedded or missing knowledge. After that, new executable source code is generated. This action mainly addresses the pollution and missing knowledge symptom.
- *A2. Re-document through reverse engineering*. This action is very similar than previous one, however the objective of reverse engineering is to extract meaningful information about the software system instead source code generation. This action deals with the missing knowledge symptom.
- *A3. Remove dead code and dead data*. This action analyses and removes lines of dead code as well as data that are not reached. This action mainly deals with the pollution symptom.
- *A4. Refactoring / restructuring*. This action reorganizes and restructures the software system in order to deal with any erosion symptom.
- *A5. Remove anomalous programs*. This action remove programs in order to eradicate erroneous programs, i.e., duplicated programs, obsolete programs, programs that generate pathological files, and so on. It mainly addresses the pollution and anomalous data symptom.
- *A6. Remove redundancies*. This action removes those redundant data from the semantic and computational viewpoint, thus it deals with the anomalous data symptom.

TABLE IV. FUZZY SETS FOR THE MAINTENANCE ACTIONS

Maintenance Action	Linguistic Label	Trapezoids	
A1. Rewrite source code through reverse engineering.	{Low, Medium, High}	{[-0.5 0 0.5], [0 0.5 1], [0.5 1 1.5]}	
A2. Re-document through reverse engineering.	{Low, Medium, High}	{[-0.5 0 0.5], [0 0.5 1], [0.5 1 1.5]}	
A3. Remove dead code and dead data	{Low, Medium, High}	{[-0.5 0 0.5], [0 0.5 1], [0.5 1 1.5]}	
A4. Refactoring / restructuring	{Low, Medium, High}	{[-0.5 0 0.5], [0 0.5 1], [0.5 1 1.5]}	
A5. Remove anomalous programs	{Low, Medium, High}	{[-0.5 0 0.5], [0 0.5 1], [0.5 1 1.5]}	
A6. Remove redundancies	{Low, Medium, High}	{[-0.5 0 0.5], [0 0.5 1], [0.5 1 1.5]}	

These actions can fully or partially address one or more erosion symptoms. The output of the last subsystem provides a membership value for each action. In order to establish fuzzy rules between symptoms and the set of actions, these actions must be defined as fuzzy sets. TABLE IV shows the fuzzy sets defined in the proposed FIS system. These fuzzy sets were also established using the *Delphi* technique by involving information provided by maintenance experts.

C. Fuzzy Rules

Finally, to complete the definition of the FIS system, a set of fuzzy rules must be established to able the inference of an entire diagnosis of a software system. Fuzzy rules are *if-then* rules with a three-part process: Firstly, the precise metric values are fuzzified with the metric fuzzy sets (see TABLE II). Thus, all fuzzy inputs in the antecedent are resolved as a membership degree between 0 and 1. If the antecedent involves only one fuzzy set, then this is the degree of support for the rule. Secondly, fuzzy logic operators (e.g. and, or) are applied to multiple part antecedents. These operators resolve the antecedent to a single number between 0 and 1, which is the degree of support for the rule. Thirdly, the degree of support for the entire rule is used to shape the output fuzzy set, i.e., the consequent of the fuzzy rule. If the antecedent is partially true (i.e., its membership value is lower than 1), then the output fuzzy set is truncated according to the implication method.

On the one hand, a set of fuzzy rules are established to define the outputs related to the four symptoms subsystems: pollution, missing knowledge, Coupling and Anomalous Data.

Pollution fuzzy rules:

1. If (*CloneProg* is low) and (*SourcelessProg* is low) then (*Pollution* is null)
2. If (*CloneProg* is medium) and (*SourcelessProg* is low) then (*Pollution* is low)
3. If (*CloneProg* is high) and (*SourcelessProg* is low) then (*Pollution* is high)
4. If (*CloneProg* is medium) and (*SourcelessProg* is medium) then (*Pollution* is high)
5. If (*CloneProg* is medium) and (*SourcelessProg* is high) then (*Pollution* is maximum)
6. If (*CloneProg* is high) and (*SourcelessProg* is high) then (*Pollution* is maximum)
7. If (*ObsoleteProg* is low) and (*DeadData* is null) and (*DeadCode* is null) then (*Pollution* is null)
8. If (*ObsoleteProg* is medium) and (*DeadData* is low) and (*DeadCode* is low) then (*Pollution* is low)
9. If (*ObsoleteProg* is medium) and (*DeadData* is medium) and (*DeadCode* is medium) then (*Pollution* is low)
10. If (*ObsoleteProg* is medium) and (*DeadData* is high) and (*DeadCode* is high) then (*Pollution* is medium)
11. If (*ObsoleteProg* is medium) and (*DeadData* is maximum) and (*DeadCode* is maximum) then (*Pollution* is high)
12. If (*ObsoleteProg* is high) and (*DeadData* is maximum) and (*DeadCode* is maximum) then (*Pollution* is maximum)
13. If (*ObsoleteProg* is high) and (*DeadData* is high) and (*DeadCode* is high) then (*Pollution* is high)
14. If (*ObsoleteProg* is high) and (*DeadData* is medium) and (*DeadCode* is medium) then (*Pollution* is medium)
15. If (*CloneProg* is high) and (*ObsoleteProg* is high) and (*SourcelessProg* is high) then (*Pollution* is maximum)
16. If (*CloneProg* is low) and (*ObsoleteProg* is low) and (*SourcelessProg* is low) and (*DeadData* is null) and (*DeadCode* is null) then (*Pollution* is high)

Missing Knowledge fuzzy rules:

1. If (*MissingDocument* is low) and (*MissingFunct* is low) and (*PoorLexicon* is low) then (*MissingKnowledge* is low)
2. If (*MissingDocument* is low) and (*MissingFunct* is low) and (*PoorLexicon* is medium) then (*MissingKnowledge* is low)
3. If (*MissingDocument* is low) and (*MissingFunct* is medium) and (*PoorLexicon* is low) then (*MissingKnowledge* is medium)
4. If (*MissingDocument* is medium) and (*MissingFunct* is medium) and (*PoorLexicon* is low) then (*MissingKnowledge* is medium)
5. If (*MissingDocument* is medium) and (*MissingFunct* is medium) and (*PoorLexicon* is medium) then (*MissingKnowledge* is medium)
6. If (*MissingDocument* is high) and (*MissingFunct* is medium) and (*PoorLexicon* is low) then (*MissingKnowledge* is medium)
7. If (*MissingDocument* is not low) and (*MissingFunct* is high) and (*PoorLexicon* is not low) then (*MissingKnowledge* is high)
8. If (*MissingDocument* is low) and (*MissingFunct* is high) and (*PoorLexicon* is low) then (*MissingKnowledge* is medium)

Coupling fuzzy rules:

1. If (*PathologicalFiles* is not maximum) and (*ControlData* is maximum) then (*Coupling* is maximum)
2. If (*PathologicalFiles* is high) and (*ControlData* is maximum) then (*Coupling* is maximum)
3. If (*PathologicalFiles* is high) and (*ControlData* is high) then (*Coupling* is maximum)
4. If (*PathologicalFiles* is medium) and (*ControlData* is high) then (*Coupling* is high)
5. If (*PathologicalFiles* is medium) and (*ControlData* is medium) then (*Coupling* is medium)
6. If (*PathologicalFiles* is low) and (*ControlData* is medium) then (*Coupling* is medium)
7. If (*PathologicalFiles* is low) and (*ControlData* is low) then (*Coupling* is low)
8. If (*PathologicalFiles* is null) and (*ControlData* is low) then (*Coupling* is low)
9. If (*PathologicalFiles* is null) and (*ControlData* is null) then (*Coupling* is null)

Anomalous Data fuzzy rules:

1. If (*SemRedundant* is not high) and (*CompRedundant* is not high) then (*AnomalousData* is not high)
2. If (*SemRedundant* is medium) and (*CompRedundant* is not low) then (*AnomalousData* is medium)
3. If (*SemRedundant* is medium) and (*CompRedundant* is not medium) then (*AnomalousData* is medium)
4. If (*SemRedundant* is high) and (*CompRedundant* is not high) then (*AnomalousData* is high)
5. If (*UselessData* is not low) and (*ObsoleteData* is not low) and (*SemRedundant* is not low) and (*CompRedundant* is not low) then (*AnomalousData* is not low)
6. If (*UselessData* is high) and (*ObsoleteData* is high) and (*SemRedundant* is not low) and (*CompRedundant* is not low) then (*AnomalousData* is high)
7. If (*UselessData* is medium) and (*SemRedundant* is high) then (*AnomalousData* is high)
8. If (*ObsoleteData* is medium) and (*SemRedundant* is high) then (*AnomalousData* is high)
9. If (*ObsoleteData* is medium) and (*CompRedundant* is high) then (*AnomalousData* is high)

On the other hand, after establishing fuzzy rules concerning the FIS subsystems of symptom level, the set of rules of the action subsystem must be also defined. These rules were established in the similar way than the previous sets.

Maintenance Actions fuzzy rules:

1. If (*Pollution* is not low) then (*A1* is high)(*A3* is medium)(*A5* is medium)
2. If (*MissingKnowledge* is high) then (*A1* is medium)(*A2* is medium)(*A4* is low)
3. If (*MissingKnowledge* is not low) and (*AnomalousData* is not low) then (*A4* is medium)
4. If (*AnomalousData* is high) then (*A5* is medium)(*A6* is high)
5. If (*AnomalousData* is not high) then (*A5* is low)(*A6* is medium)
6. If (*Pollution* is medium) and (*AnomalousData* is not low) then (*A4* is low)(*A6* is medium)
7. If (*Coupling* is not low) then (*A4* is alto)(*A5* is medium)
8. If (*Coupling* is high) then (*A4* is alto)(*A5* is low)
9. If (*Pollution* is low) and (*MissingKnowledge* is low) then (*A1* is low)(*A2* is low)
10. If (*Pollution* is low) then (*A3* is low)

IV. IMPLEMENTATION

This paper also provides an implementation of the proposed FIS system in order to demonstrate its feasibility and facilitate its adoption. The FIS system has been implemented using the

Fuzzy Logic Toolbox of *Matlab* [17]. In addition, the FIS's real performance has been also simulated by means of *Simulink* [18], another *Matlab* module.

The implementation of the FIS system using the Fuzzy Logic Toolbox was carried out following a set of steps for each aforementioned FIS subsystem. Due to space limitations, it focuses in the *Coupling* subsystem to illustrate the entire implementation process.

1. The inputs and output of the FIS system are firstly selected. Relationships between the input fuzzy variables and output fuzzy variables are established. This task can be easily carried out by means of the FIS Editor of *Matlab* (see Figure 2). The type of the FIS subsystem can be also selected by means of that editor. All subsystem are based in the *Mandami* controller configuration [9, 10].
2. In the second step, input and output fuzzy variables are defined by means of fuzzy sets. Each fuzzy variable has several linguistic labels and each label is defined as a fuzzy set. This task is also performed through the *Matlab* FIS editor according to the fuzzy variables presented in Section III. For instance, Figure 3 shows the definition of the fuzzy set for the *'null'* linguistic label within *'PathologicalFiles'* input variable in the *Coupling* subsystem.
3. The third step involves the fuzzy rules definition. This task establishes specific relationship through the linguistic labels of both input and output fuzzy variables. *Matlab* also provides a graphical editor to easily generate fuzzy rules.
4. Finally, the logic operators as well as the *implication* and *defuzzification* operations must be established through the *Matlab* FIS editor (see Figure 2). The 'and' and 'or' logic operators are used in antecedents of fuzzy rules to combine several fuzzy sets. In this case we respectively select the *minimum* and *maximum* operators to combine several fuzzy sets. These operators are well-known and commonly used to represent the intersection (and) and union (or) of fuzzy sets. Moreover the implication function is selected in this step. The *implication function* is used to obtain a membership value in the consequent fuzzy set from the membership value of the antecedent. This FIS system uses the minimum operation. Another operation that must be established is the *aggregation function*. This function merges all membership values obtained in a particular fuzzy set that appears in consequents of several rules. The FIS system uses the maximum operation since it guarantees to obtain the higher value obtained for a particular fuzzy set. Finally, a *deffuzzification function* must be selected to obtain a real value between 0 and 1from the aggregated area obtained in a particular consequent fuzzy set. The proposed FIS system uses the *centroid* function, which use the weighted average of a few data points in the aggregated area.

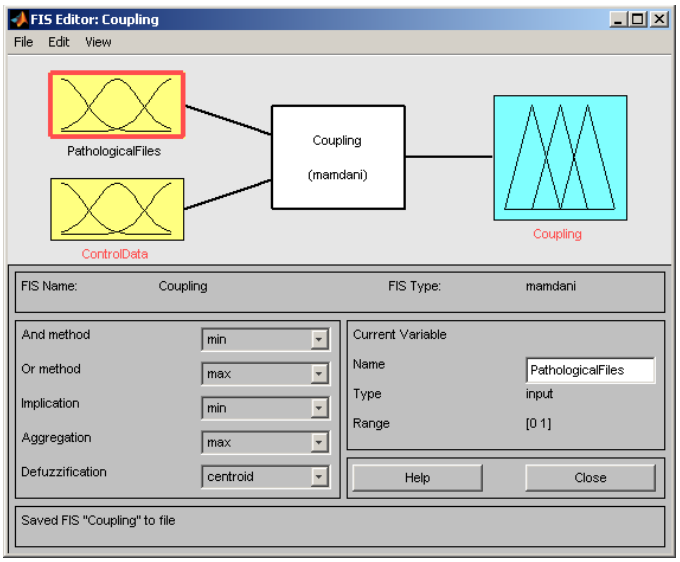


Figure 2. The FIS subsystem configuration for the coupling symptom

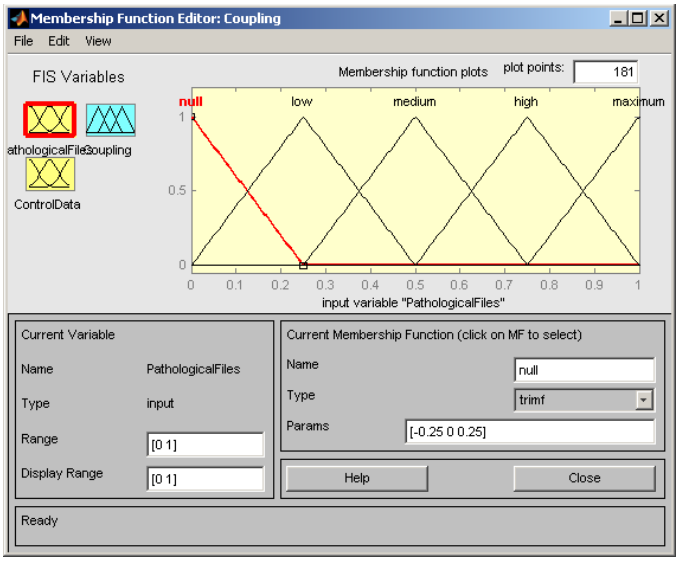


Figure 3. Fuzzy set definition for the coupling subsystem

V. SIMULATION

To simulate the FIS system implemented though the *Matlab Fuzzy Logic Toolbox*, we use *Simulink* [18]. *Simulink* is an environment for multi-domain simulation and model-based design for dynamic and embedded systems. It makes it possible to simulate, implement, and test a variety of time-varying systems.

Figure 4 shows the simulation model of the proposed FIS system, which is built in *Simulink* as a fuzzy logic controller following the architecture presented in Figure 1. Firstly, the fuzzy logic controller will receive the precise values of the all maintenance metrics, and it then will trigger the respective fuzzy rules of the four subsystems in the symptom level. Outputs of the symptom subsystems are used as the input for the maintenance subsystem, which generates the membership degree (as a value between 0 and 1) to each of the six possible maintenance actions. In addition, the controller filters out

actions under the aforementioned maintenance threshold, which can be established by maintenance experts in each company. As a consequence, the controller indicates by means a light what actions should be carried out.

In order to simulate the performance of the FIS system, a real-life software system was evaluated. The software system named *VillasanteLab* manages the operation of a Spanish company in the water and waste industry. *VillasanteLab* system manages information related to chemical laboratories, customers and products such as chemical analysis, dilutions, and chemical calibrations. The analyses supported by the application examined different parameters, including a large number of physical, chemical and microbiological parameters according to current regulations and laws for controlling water quality. From a technological point of view, *VillasanteLab* system is a traditional web application developed using Java platform and consists of 28.800 lines of source code. This software system was released for four years ago, and it has had three major modifications with seven medium modifications in total. Therefore, the *VillasanteLab* system probably has been eroded over time.

Firstly, all the maintenance metrics used in our FIS system were accurately evaluated. TABLE V shows the fourteen metric values obtained after evaluating the *VillasanteLab* system. These values were introduced in the simulation model and the model was then executed in *Simulink* (see Figure 4). The obtained results for the erosion symptoms were respectively 0.2, 0.5, 0.5 and 0.4; and the final results concerning maintenance actions were 0.80, 0.50, 0.43, 0.55, 0.41 and 0.5 (see TABLE V). In addition, the fuzzy rules executed as well as the fuzzy sets values were obtained during simulation (see Figure 5).

The configuration of simulation considers a maintenance threshold of 0.5. Therefore, the recommended maintenance actions for the *VillasanteLab* system were *A1. Rewrite source code through reverse engineering* as well as *A4. Refactoring / restructuring*.

VI. CONCLUSIONS

This paper presented a fuzzy rule-based system (also known as Fuzzy Inference System) to find out the software erosion symptoms of a concrete software system. The proposed FIS system does not only diagnose the kind of software erosion, but also recommend the best treatment for the eroded system. We refer to "the best treatment" as a set of maintenance actions that must be carried out to reduce or mitigate the software those erosion symptoms without incurring in more erosion itself.

The design of the FIS system was mainly divided into two levels. The first level evaluates the erosion symptoms, which consists of four subsystems (one for each symptom). Each subsystem accepts as input metrics accurately measured from the software system. These precise values are then fuzzified and a set of fuzzy rules establish as output the membership degrees for each symptom subsystem. The second level has only one subsystem, which takes the fuzzy values of each symptom subsystem and it obtains the membership degrees for each candidate maintenance action.

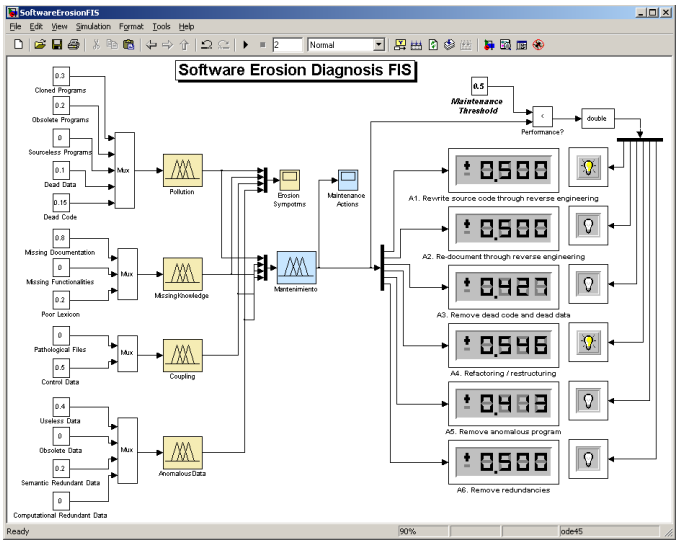


Figure 4. Simulation model for the proposed FIS system

TABLE V. SIMULATION DATA FOR AN EXISTING SOFTWARE SYSTEM

Metric	Precise Value	Symptom	Value	Action	Value
Clone Programs	0.30	Pollution	0.229	A1	0.804
Obsolete Programs	0.20				
Sourceless Programs	0			A2	0.500
Dead Data	0.10				
Dead Code	0.15			A3	0.427
Missing Documentation	0.80	Missing Knowledge	0.500		
Missing Functionalities	0			A4	0.546
Poor Lexicon	0.20				
Pathological Files	0	Coupling	0.500	A5	0.413
Control Data	0.50				
Useless Data	0.40	Anomalous Data	0.386	A6	0.500
Obsolete Data	0				
Semantic Redundant Data	0.20				
Computational Redundant Data	0				

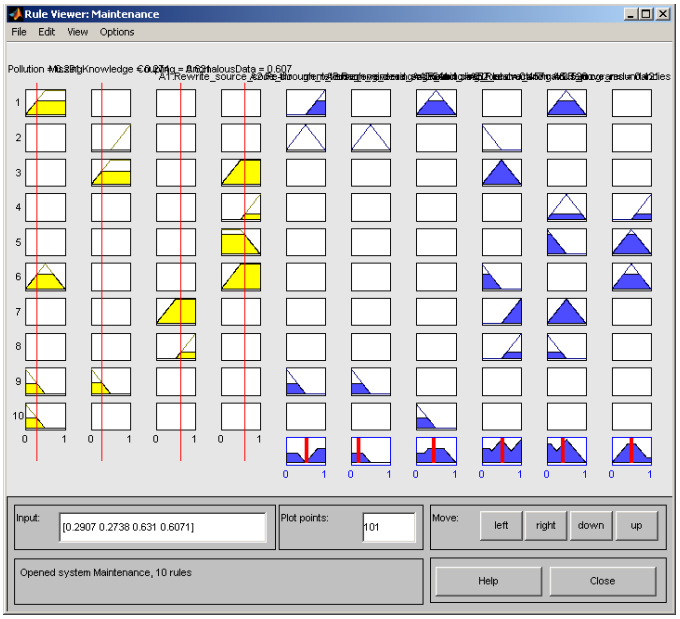


Figure 5. Obtained results during simulation

In order to validate the feasibility of our proposal and its adoption, the proposed FIS system was also implemented in *Matlab* using the *fuzzy logic toolbox*. Furthermore, the implemented system was also simulated through *Simulink*. The advantage of the implemented simulation system is that maintainers can know the best set of maintenance actions to be carried out. The set of actions is prioritized, which helps maintainers to distribute maintenance resources and efforts. In addition, maintainers can estimate the reduction of erosion metrics in the hypothetical case to carry out the recommended actions. As a consequence, they might simulate a set of iterations of incremental maintenance and predict by means of our system the expected reduction of erosion symptoms. If a simulation shows that the software system is much eroded and a lot of maintenance effort will be necessary, the system could be discarder without starting the maintenance process. In conclusion, the proposed FIS system supports the decision-making process in the maintenance stage, and it can save maintenance cost by means of the proposed simulation model.

Moreover, the simulation of our system allowed us to detect potential improvements for the FIS system. Those improvements were progressively applied to adjust the definition of fuzzy sets. In addition, the rules were established by experts according to the *Delphi* technique and they were progressively improved through the simulation. However, the *Delphi* technique adopted to establish the rules is not the best approach, since it has a heuristic nature. As a consequence, we propose as future work the improvement of the fuzzy rules by using statistical techniques from real-life maintenance data or learning systems to find out the optimum fuzzy rules.

ACKNOWLEDGMENT

This work was supported by the FPU Spanish Program; by the R+D projects funded by JCCM: ALTAMIRA (PII2I09-0106-2463) and PRALIN (PAC08-0121-1374); and the PEGASO/MAGO project (TIN2009-13718-C02-01) funded by MICINN and FEDER.

REFERENCES

[1] Albusac, J., J.J. Castro-Schez, and D. Vallejo-Fernandez. "Learning Maximal Structure Rules with pruning based on distances between fuzzy sets". in 12th International Conference on Information Processing and Management of Uncertainty in Knowledge-Based Systems (IPMU'08). 2008. Malaga, Spain p. 441-447.

[2] Basili, V., L. Briand, S. Condon, Y.-M. Kim, Walc\, \#233, l.L. Melo, and J.D. Valett. "Understanding and predicting the process of software maintenance release". in Proceedings of the 18th international conference on Software engineering. 1996. Berlin, Germany: IEEE Computer Society p. 464-474.

[3] Ghazarian, A. "A Case Study of Source Code Evolution". in 13th European Conference on Software Maintenance and Reengineering (CSMR'09). 2009. Fraunhofer IESE, Kaiserslautern, Germany: IEEE Computer Society p. 159-168.

[4] Hall, R. and S. Lineham, "Using metrics to improve software maintenance". BT Technology Journal, 1997. 15(3): p. 123-129.

[5] Hayes, J.H., S.C. Patel, and L. Zhao. "A Metrics-Based Software Maintenance Effort Model". in Proceedings of the Eighth Euromicro Working Conference on Software Maintenance and Reengineering (CSMR'04). 2004: IEEE Computer Society p. 254.

[6] ISO/IEC, ISO/IEC 14764:2006. Software Engineering -- Software Life Cycle Processes -- Maintenance. http://www.iso.org/iso/catalogue detail.htm?csnumber=39064. 2006, ISO/IEC.

[7] Koskinen, J., J. Ahonen, H. Lintinen, H. Sivula, and T. Tilus, Estimation of the Business Value of Software Modernizations. 2004, Information Technology Research Institute. University of Jyväskylä.

[8] Lehman, M.M., D.E. Perry, and J.F. Ramil. "Implications of Evolution Metrics on Software Maintenance". in Proceedings of the International Conference on Software Maintenance. 1998: IEEE Computer Society p. 208-217.

[9] Mamdani, E.H., "Application of fuzzy algorithms for control of simple dynamic plant". Proceeding of IEEE, 1974. 121(12).

[10] Mamdani, E.H. and S. Assilian, "An Experiment in Linguistic Synthesis with a Fuzzy Logic Controller". International journal of human-computer studies, 1999. 51(2): p. 135-147.

[11] Mittal, H. and P. Bhatia, "Software maintainability assessment based on fuzzy logic technique". SIGSOFT Softw. Eng. Notes, 2009. 34(3): p. 1-5.

[12] Mittal, J.P., P. Bhatia, and H. Mittal, "Software maintenance productivity assessment using fuzzy logic". SIGSOFT Softw. Eng. Notes, 2009. 34(5): p. 1-4.

[13] Ning, M.H., Q. Yong, H. Di, C. Ying, and Z.J. Zhong. "Software Aging Prediction Model Based on Fuzzy Wavelet Network with Adaptive Genetic Algorithm". in Proceedings of the 18th IEEE International Conference on Tools with Artificial Intelligence. 2006: IEEE Computer Society p. 659-666.

[14] Polo, M., M. Piattini, and F. Ruiz, Advances in software maintenance management: technologies and solutions. 2003: Idea Group Publishing.

[15] Rowe, G. and G. Wright, "The Delphi technique as a forecasting tool: issues and analysis". International journal of forecasting, 1999. 15(4): p. 353-375.

[16] Sneed, H.M., Estimating the Costs of a Reengineering Project. Proceedings of the 12th Working Conference on Reverse Engineering. 2005: IEEE Computer Society.

[17] The Mathworks, I., Fuzzy Logic Toolbox of Matlab http://www.mathworks.com/products/fuzzylogic/. 2009.

[18] The Mathworks, I., Simulink. Simulation and Model-Based Design. http://www.mathworks.com/products/simulink/. 2009.

[19] Visaggio, G., "Ageing of a data-intensive legacy system: symptoms and remedies". Journal of Software Maintenance, 2001. 13(5): p. 281-308.

[20] Zadeh, L.A., "Fuzzy Sets*". Information and Control, 1965. 8: p. 338-353.