A Fuzzy Expert System for Cost-Effective Regression Testing Strategies

Amanda Schwartz, Hyunsook Do
North Dakota State U.
{amanda.j.schwartz, hyunsook.do}@ndsu.edu

***Abstract*—Different testing environments and software change characteristics can affect the choice of regression testing techniques. In our prior work, we developed adaptive regression testing (ART) strategies to investigate this problem. While the ART strategies showed promising results, we also found that the multiple criteria decision making processes required for the ART strategies are time-consuming, often inaccurate and inconsistent, and limited in their scalability. To address these issues, in this research, we develop and empirically study a fuzzy expert system (FESART) to aid decision makers in choosing the most cost-effective technique for a particular software version. The results of our study show that FESART is consistently more cost-effective than the previously proposed ART strategies. One of the biggest contributors to FESART being more cost-effective is the reduced time required to apply the strategy. This contribution has significant impact because a strategy that is less time-consuming will be easier for researchers and practitioners to adopt, and will provide even greater cost-savings for regression testing sessions.**

***Index Terms*—Regression testing, test case prioritization, adaptive regression testing strategy, AHP, fuzzy AHP, empirical studies**

I. Introduction

Software maintenance is a large part of the software development life-cycle. Maintaining a software system includes many different tasks, such as fixing defects, adding new features, or modifying the software to accommodate different environments. After the software system has been modified, it needs to be tested to ensure that the changes did not have any adverse effects on the previously validated code. Regression testing is the process of checking modified software systems to ensure continued quality. Regression testing is often performed by re-running existing tests from previous versions along with new tests which test new features.

However, as software systems grow, the size of the test suite can become too large, making it too time-consuming and costly to run all the tests. For example, in [1], one company mentioned has a software product with a regression test suite containing over 30,000 test cases that requires over 1,000 machine hours to execute. In all situations, including that one, regression testing is still necessary to ensure the continued quality of the software system. However, requiring 1,000 hours to run all test cases is not a feasible option, so reducing the cost and time required for regression testing sessions has considerable importance.

Many regression testing techniques and maintenance approaches have been proposed to reduce the costs of regression testing, such as test case prioritization, test case selection, and test case minimization. Also, empirical studies have been performed to evaluate the cost-effectiveness of the different regression testing techniques. Some of these studies have shown that various environmental and testing factors affect the cost-effectiveness of the techniques [1], [2], [3]. Therefore, the technique which is most cost-effective for one version may not be the most cost-effective for every version of a software system. We can say that there is no single regression testing technique that is the most cost-effective for every version of a software system.

Because there is no single technique which is most cost-effective for every version of a software system, there is potential for large cost-savings by choosing the most cost-effective technique for each software version considering various factors that affect the overall costs and benefits. However, very little research has been done on the problem of helping practitioners choose appropriate techniques [4], [5] under particular testing environments. To address this issue, in our prior studies [6], [7], we investigated *adaptive regression testing* (ART) strategies that try to choose the most cost-effective regression testing techniques for each regression testing session considering various evaluation factors. In the first study [6], we utilized the analytical hierarchy process (AHP) [8] to choose the best test case prioritization techniques across the system lifetime. In the second study [7], we conducted additional research using fuzzy AHP to address the problem of imprecision by decision makers in the pairwise comparisons that we observed in the first study. The results of both studies indicate that the techniques chosen by ART strategies are consistently more cost-effective than those used by approaches that do not consider system lifetime and testing processes.

Although the prior studies showed promising results, there are still several limitations which remain with the proposed ART strategies. First, comparisons made by the decision maker during the pairwise comparison process are often inconsistent [9], [10]. Judgements made for one comparison often contradict judgements made for another comparison. Second, the pairwise comparisons are very time-consuming for the decision maker [11], [12]. Third, the use of pairwise comparisons is not scalable. Because of the work required by pairwise comparisons, there is a limit to the number of criteria and alternatives that can be considered [13]. To address these problems, other decision making methods need to be considered.

One method which has frequently been used to solve problems which normally require human experts is a fuzzy expert system (e.g., fuzzy expert systems have been used in the medical field to diagnose heart disease [14] and back pain [15]). Fuzzy expert systems provide a mechanism to simulate the judgement and reasoning of experts in the particular field. Fuzzy expert systems have two major components which simulate expert judgement: a knowledge base and an inference engine. The knowledge base contains knowledge about the particular domain which is used by human experts to solve the problem, and the inference engine contains a set of rules which utilizes the knowledge to determine appropriate output.

We believe that fuzzy expert systems are able to address the limitations of the previously proposed ART strategies for the following reasons. First, a fuzzy expert system does not require pairwise comparisons, so the issue of inconsistencies with the comparisons is eliminated. Second, without pairwise comparisons, there is less input needed from the decision maker, making it less time-consuming for the decision maker. A method which requires pairwise comparisons with n alternatives requires $\frac{n^2-n}{2}$ comparisons for each criteria, where a fuzzy expert system would only require n number of inputs for each criteria. In addition to a fuzzy expert system being less time-consuming because it requires fewer input per criteria and alternative, a fuzzy expert system is more scalable because the input required per criteria and alternative does not grow as quickly as a method with pairwise comparisons.

In this research, we develop and empirically study a fuzzy expert system for ART. In the next section, we describe the background information and related work relevant to prioritization techniques and decision making strategies. Section III describes fuzzy expert systems, including how we developed a fuzzy expert system for ART. Section IV presents our experiment design and discusses threats to validity. Section V presents the results of the study and data analysis. Section VI discusses our results, and Section VII presents our conclusions and future work.

II. Background and Related Work

To date, many regression testing techniques have been proposed and empirically evaluated, but here, we limit our discussion to test case prioritization techniques which are most closely related to our work. Further, we provide related work relevant to decision making processes by focusing on AHP and fuzzy expert systems.

A. Test Case Prioritization

Test case prioritization techniques reorder test cases according to some goal (e.g., increasing the rate of fault detection) so that maximum benefit can be achieved even if testing is halted early. To date, many other prioritization techniques have been proposed, and a recent survey by Yoo and Harman [16] provides an overview of many different techniques.

With many different techniques available, recent research has begun to include empirical studies to evaluate the cost-benefit trade-offs among techniques by considering various factors and testing contexts [1], [2], [3]. These studies show that various techniques have strong potential for reducing the cost of regression testing, but the studies also reveal wide variances in performance. The varying performance is attributed to the different factors involving the program under test, the test suites used to test them, the types of program modifications, and the testing processes. More recent studies [6], [7] introduced adaptive regression testing (ART) strategies to identify the most cost-effective regression testing techniques for each regression testing session.

B. Multiple Criteria Decision Making (MCDM) Methods

Choosing a prioritization technique involves many different factors which have trade-offs. These trade-offs are considered to be conflicting criteria. A problem which has multiple conflicting criteria is known as a multiple criteria decision making (MCDM) problem.

Analytic Hierarchy Process (AHP) is one of the widely used MCDM methods. It has been used in many different areas. For instance, Aull-Hyde and Davis [17] discuss how AHP is used as an important tool for decision making in the U.S. military, and Subramanian and Ramanathan [18] provide a review of how AHP is used in operations management. Further, AHP has recently been used in software engineering areas, such as aiding early effort estimations [19] and prioritizing software requirements [20]. Although AHP is a widely used method for decision making problems, it has been noted throughout the literature that there are several limitations to this method, such as its subjectiveness of decision maker's judgements [21], [22], inconsistency in pairwise comparisons [10], time-consuming comparison process [11], [23], and scalability problem (a limit of 7 ± 2 alternatives is suggested in [13]).

Fuzzy AHP has been used by many researchers to address the imprecision in judgments made by the decision maker [24], [21], and was used as an ART strategy in [7]. However, fuzzy AHP still requires pairwise comparisons and, therefore, has the drawbacks of inconsistent comparisons, being very time-consuming, and not being very scalable.

Fuzzy expert systems have been used in many different domains to aid in complex decision making problems. For example, fuzzy expert systems have been developed in the medical field to diagnose heart disease [14] and back pain [15], and in economics for choosing stock in the stock exchange [25]. Fuzzy expert systems have also been developed in the area of software engineering. They have been used frequently for software cost estimation [26], [27]. There has been very little use in the area of software testing, however. Xu et. al developed a fuzzy expert system to build a new test selection technique [28]. Our work develops a fuzzy expert system that helps choose the most cost-effective regression testing technique for regression testing sessions. In the next section, we discuss fuzzy expert systems in more detail as well as how a fuzzy expert system can be developed for ART.

III. Fuzzy Expert Systems

In this section, we describe fuzzy expert systems, and how a fuzzy expert system can be utilized for creating a new Adaptive Regression Testing (ART) strategy.

A. Fuzzy Expert Systems

A fuzzy expert system is an expert system comprised of fuzzy membership functions and rules. It contains three main parts: fuzzification, fuzzy inference, and defuzzification. A fuzzy expert system is represented in Figure 1. This figure shows how crisp input is given by the decision maker to the fuzzification process, which determines a fuzzy input set. That fuzzy input set is used in the inference process. The fuzzy inference process uses fuzzy rules built from a knowledge base to determine the fuzzy output set. The fuzzy output set is then defuzzified in the defuzzification process to determine crisp output. Crisp output is then used by the decision maker in the decision making process. Each of these parts are explained in more detail in this section.

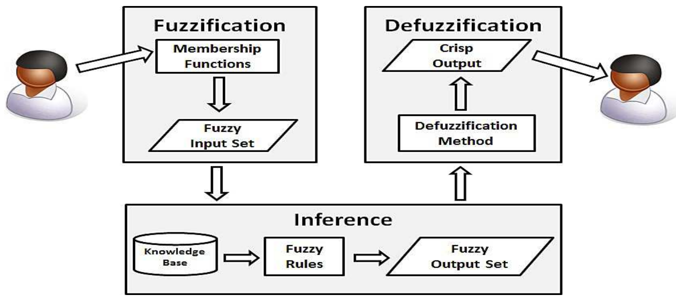

Fig. 1. Fuzzy Expert System.

Fuzzy Set Theory

To understand the fuzzification process, some knowledge of fuzzy set theory is necessary. Fuzzy set theory, which was introduced by Zadeh in 1965 [29], defines fuzzy sets as an extension of conventional sets. In conventional sets, elements are considered to either be a part of a set or not be a part of a set. The membership, $\mu_A(x)$, of an element, x, of a classical set, A, is defined by the equation below:

$$\mu_A(x) = \begin{cases} 1, & if \;\; x \in A \\ 0, & if \;\; x \notin A \end{cases} \tag{1}$$

Fuzzy sets allow partial membership. The degree of membership is calculated using a membership function which generates the degree of membership on the interval [0, 1].

Fuzzy sets can be formally defined by:

$$A = (x, \mu_A(x))|x \in X, \mu_A(x) : X \to [0, 1]) \tag{2}$$

where A is the fuzzy set, μ_A is the membership function, and X is the universe of discourse.

Fuzzy sets can be used effectively to represent linguistic values in a fuzzy expert system. For example, if you are going to describe the service and food at a restaurant, you may describe the service as *poor*, *average*, or *excellent* and the food as *poor*, *good*, or *delicious*. Because fuzzy sets allow partial membership, you would be able to specify a degree of membership for each fuzzy set, so the food could be classified as *somewhat poor* and *somewhat good*. By being able to allow partial membership, fuzzy set theory can handle the imprecision of input from the decision makers.

Fuzzification

The fuzzification process takes input from the decision maker and determines its degree of membership to the fuzzy sets using membership functions defined in the fuzzy expert system. Fuzzification is necessary to convert the input data into fuzzy sets for the inference engine to process. Three of the most commonly used membership functions are triangular, trapezoidal, and gaussian. The triangular membership function is described using three values (a, b, c) where b is the modal value, a is the minimum boundary, and c is the maximum boundary. The trapezoidal membership function is described using four values (a, b, c, d) a is the minimum value, b is the minimum support value, c is the maximum support value, and d is the maximum value. The gaussian membership function transforms the values into a normal distribution with the midpoint defining the ideal definition for the set. The midpoint is assigned a degree of membership of 1.

Fuzzy Inference

The fuzzy inference system takes the fuzzified input from the fuzzification process and determines fuzzy output. The fuzzy inference process maps all inputs, $x = [x_1, x_2, \ldots, x_n]$ to an output, $f(x)$. The mapping is done using fuzzy rules. The antecedent of the fuzzy rule defines the fuzzy region of the input space, and the consequent defines the fuzzy region of the output space. The fuzzy inference process is modeled in Figure 2. In this figure, the fuzzy inference process is shown in the area outlined by the dotted line. This particular inference system has three rules that are used to map the input, x, to an appropriate output set. $A1$, $A2$, and $A3$ are linguistic variables that categorize the input. Based on the categorized input, the rule determines the output ($B1$, $B2$, or $B3$). For example, using Rule 1, if x is categorized as linguistic variable $A1$, then the output set is $B1$.

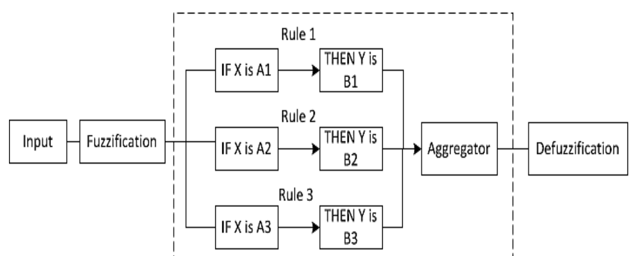

Fig. 2. Fuzzy Inference Process.

There are two popular inference systems: the Mamdani inference system [30] and the Takagi-Sugeno inference system [31]. In this research, we will be using the Mamdani inference system which is a more commonly used system, so we will limit our discussion to that inference system.

The first step in a Mamdani fuzzy inference system is to match the input to the fuzzy rules which have some degree of truth in the antecedent forming the fuzzy conclusion set. Then, the fuzzy rules in the fuzzy conclusion set are evaluated. The next step of the fuzzy inference system is the aggregation of the rule output. All the *then*-parts of the rules are combined into a final output set. The final output set is a fuzzy set which requires defuzzification for the final output.

Fuzzy Rules

A fuzzy rule is a conditional statement that uses linguistic variables. Fuzzy rules are used to determine output from fuzzy input. The knowledge needed to construct fuzzy rules in a fuzzy expert system comes from a combination of several different sources. The most widely used sources are human knowledge and expertise, historical data analysis of a system, and engineering knowledge from existing literature. Fuzzy rules express knowledge about the relationship between input and output variables. A generic fuzzy rule assumes the following form:

If x is A then y is B

where A and B are linguistic values defined by fuzzy sets. The first part of the rule, the *if*-part, is called the antecedent, and the *then*-part is called the consequent. Any rule that has some truth in the antecedent will be included in the fuzzy conclusion set. In the fuzzy conclusion set, if the antecedent is true to some degree of membership, then the consequent is also true to that same degree of membership. Some rules may contain more than one input in the antecedent, and the input variables may be combined using fuzzy set operators such as *AND* or *OR*. A generic fuzzy rule with two inputs, one using *AND* and one using *OR* is:

If x is A AND y is B then z is C

If x is B OR y is B then z is C

where $A, B,$ and C are linguistic values defined in the fuzzy set, x and y are the input variables, and z is the output variable. One of the most common ways for evaluating fuzzy rules with fuzzy operators is the Zadeh technique [29], which is also referred to as the min-max technique. The Zadeh technique for the fuzzy intersection takes the minimum degree of membership for the membership values of the antecedent. The technique is defined by:

$$\mu_{A \cap B}(x) = min[\mu_A(x), \mu_B(x)] \tag{3}$$

The Zadeh technique for fuzzy union takes the maximum degree of membership for the membership values of the antecedent. The technique is defined by:

$$\mu_{A \cup B}(x) = max[\mu_A(x), \mu_B(x)] \tag{4}$$

Defuzzification

Defuzzification is the way the fuzzy output from the inference process is converted to a crisp value. Many different defuzzification techniques have been proposed, but the center of gravity is the most widely accepted and regarded as being accurate [32], [33]. The definition for the center of gravity is:

$$y^* = \frac{\int \mu_B(y) y dy}{\int \mu_B(y) dy} \tag{5}$$

where y^* is the defuzzified output, $\mu_B(y)$ is the aggregated membership function, and y is the output variable.

B. A Fuzzy Expert System for ART (FESART)

In this section, we outline the fuzzy expert system we developed for ART, FESART. We describe each of the main parts of a fuzzy expert system: fuzzification, fuzzy inference using fuzzy rules, and defuzzification.

Fuzzification

The fuzzification process takes input from the decision maker and determines its degree of membership to the fuzzy sets using the membership functions defined in the fuzzy expert system. The input provided by the decision maker contains information which would aid in the decision making process. For ART, we consider the following criteria:

- Cost of applying the test case prioritization technique: the time required to run a test case prioritization algorithm
- Cost of software artifact analysis: the costs of instrumenting programs and collecting test execution traces
- Cost of delayed fault detection: the waiting time for each fault to be exposed while executing test cases under a test case prioritization technique
- Cost of missed fault: the time required to correct missed faults

The decision maker will evaluate each criterion on a scale from 1 to 9, with 9 being a high cost. This input will then be fuzzified according to its degree of membership to the membership functions provided in the fuzzy expert system. The fuzzy expert system contains three triangular membership functions for each criterion being considered. Triangular membership functions are defined by three values (a, b, c) where b is the modal value, a is the minimum boundary, and c is the maximum boundary. These membership functions are shown in Table I. The resulting fuzzy input set from the fuzzification process is used as input for the fuzzy inference process.

TABLE I
MEMBERSHIP FUNCTION FOR INPUT VARIABLES

Linguistic Value	Triangular Fuzzy Numbers(*a, b, c*)
Low	(-3, 1, 5)
Average	(1, 5, 9)
High	(5, 9, 13)

Fuzzy Inference

The fuzzy inference process takes the fuzzified input from the fuzzification process and determines the fuzzy output set. The fuzzy output set for FESART contains eight triangular membership functions. The output is rated on a scale from 1 to 9, with the membership functions being evenly distributed across these values. The membership functions are shown in Table II. The output set was built to categorize the overall cost for the regression testing technique and are categorized from low to high. $L1$, $L2$, and $L3$ are considered low costs, with $L1$ being the lowest. Then, $A1$ and $A2$ are categorized as average cost, with $A1$ being lower than $A2$. $H1$, $H2$, and $H3$ are all high costs, with $H3$ being the highest cost.

The fuzzy output set is determined by using fuzzy rules. More detail about how the fuzzy rules for ART were developed is provided in the next section.

TABLE II
MEMBERSHIP FUNCTION FOR OUTPUT VARIABLE

Linguistic Value	Triangular Fuzzy Numbers (*a, b, c*)
L1	(-.14, 1, 2.14)
L2	(1, 2.14, 3.29)
L3	(2.14, 3.29, 4.43)
A1	(3.29, 4.43, 5.57)
A2	(4.43, 5.57, 6.71)
H1	(5.57, 6.71, 7.86)
H2	(6.71, 7.86, 9)
H3	(7.86, 9, 10.14)

Fuzzy Rules

In a fuzzy expert system, the fuzzy rules bring expert knowledge into the system to aid in the decision making process. The knowledge needed to construct fuzzy rules in a fuzzy expert system comes from a combination of several different sources. The most widely used sources are human knowledge and expertise, historical data analysis of a system, and engineering knowledge from existing literature. To develop rules for a fuzzy expert system in ART, knowledge about the factors that influence cost-benefits for regression testing techniques is needed. To gain this knowledge, each of the previously mentioned methods were used. For example, from the literature [34], [35], it can be said the costs of delayed fault detection are greater than the costs related to setting up and running the test cases, so the fuzzy rules involving these two items would give more importance to the cost of delayed fault detection.

Each criterion was considered and evaluated through information gained from the methods listed above. Using this knowledge, the criteria were ordered by their impact on cost-benefit trade-offs. The order was determined to be the cost of missed faults (CF), cost of delayed fault detection (CD), cost of applying the prioritization techniques (CR), and costs of software artifact analysis (CA), with the cost of missed faults having the strongest impact and the cost of software artifact analysis having the least impact.

For each of the criterion possible combinations of membership functions were considered. There are four input variables, and three membership functions for each one, so there are 81 unique combinations. Each combination was evaluated and assigned an appropriate output set. Then, the rules were studied to see if any of them could be combined or eliminated. We were able to reduce the rule set to 67. The following example demonstrates how we were able to reduce the rule set. In the original rule set, the following three rules existed:

- *IF CF is H and CD is H and CR is H and CA is H then Cost is H3.*
- *IF CF is H and CD is H and CR is H and CA is A then Cost is H3.*
- *IF CF is H and CD is H and CR is H and CA is L then Cost is H3.*

For each possible value for CA, the output set was still H3. The value of CA did not have any impact on these particular rules, so we can reduce the three rules into the following rule:

IF CF is H and CD is H and CR is H then Cost is H3

A subset of the fuzzy rules is shown in Table III. Because CF and CD were determined to have the most impact on the cost-benefit calculations and are classified as a high cost in these rules, the final cost is categorized in the different high cost output sets ($H1$, $H2$, and $H3$). When CR and CA have a higher cost, then the final cost is determined on the higher end of the high output sets ($H2$ or $H3$). When CR and CA have a low cost, the final cost is still considered high because CF and CD are high and have more impact, but the cost on the lower end of the high cost ($H1$).

Once the fuzzy set is determined through the use of the fuzzy rules, the defuzzification process is used to determine a crisp value. The defuzzification process is described in the next section.

TABLE III
FUZZY RULES FOR ART

2. IF CF is H and CD is H and CR is A and CA is H then Cost is H3
5. IF CF is H and CD is H and CR is A and CA is A then Cost is H2
3. IF CF is H and CD is H and CR is A and CA is L then Cost is H2
6. IF CF is H and CD is H and CR is L and CA is H then Cost is H2
4. IF CF is H and CD is H and CR is L and CA is A then Cost is H1
7. IF CF is H and CD is H and CR is L and CA is L then Cost is H1

Defuzzification

Defuzzification is the way that the fuzzy output from the inference process is converted to a crisp value. Many different defuzzification techniques have been proposed. We chose to use the center of gravity method because it is the most widely accepted and is regarded as being accurate [32], [33]. The output provided from the defuzzification process gives a crisp value to use in the decision making process. In the case of ART, the goal is to identify the most cost-effective regression testing technique for a specific software version. The number provided by the defuzzification process for one technique can be used to compare with numbers from the defuzzification process for other techniques. For ART, the lower the number, the more cost-effective the regression testing strategy is. The technique with the lowest number is the technique which is expected to be the most cost-effective for that particular software version.

IV. EMPIRICAL STUDY

In this study, we address the following research question:

RQ: Is a fuzzy expert system that considers testing environments and contexts more cost-effective than the previously proposed ART strategies at choosing the most cost-effective regression testing techniques across a system lifetime?

To investigate this research question, a controlled experiment was conducted. The experiment setup was similar to that of our earlier studies [6], [7], with the test case prioritization technique application mapping strategies considered being: AHP, fuzzy AHP, and FESART. The same object programs and versions from the previous study [6] were used. The following subsections present, for this experiment, our objects of analysis, independent variables, dependent variables and measures, experimental setup and design, and threats to validity.

A. Objects of Analysis

The objects of analysis for this study included five Java programs from the SIR infrastructure [36]. Table IV lists the objects of analysis, as well as, for each object of analysis, data on its associated "Versions" (the number of versions of the object program), "Classes" (the number of class files in the latest version of that program), "Size (KLOCs)" (the number of lines of code in the latest version of the program), and "Test Cases" (the number of test cases available for the latest version of the program). To study the research question, we needed fault data, so we utilized mutation faults provided with the programs [37]. The rightmost column, "Mutation Faults," is the total number of mutation faults for the program (summed across all versions).

TABLE IV
EXPERIMENT OBJECTS AND ASSOCIATED DATA

Objects	Versions	Classes	Size (KLOCs)	Test Cases	Mutation Faults
ant	9	914	61.7	877	412
jmeter	6	434	42.2	78	386
xml-sec.	4	145	15.9	83	246
nanoxml	6	64	3.1	216	204
galileo	16	68	14.5	912	2494

B. Variables and Measures

1) Independent Variable: Our experiment consists of one independent variable and one dependent variable. The independent variable is the *test case prioritization technique application mapping strategy* which assigns, to a specific sequence of versions, $S_i, S_{i+1}, \dots S_j$, for system *S*, specific test case prioritization techniques. There are three strategies used in this paper. Each strategy chooses one of four prioritization techniques (total block coverage, additional block coverage, random order, and original order). Total block coverage sorts test cases by the order of the number of blocks they cover. Additional block coverage selects a test case that yields the greatest block coverage, adjusts the coverage information for the remaining test cases to indicate their coverage for the blocks not yet covered, and then repeated this process until all blocks are covered by at least one test case. Random order is the average of a number of runs (in our experiment 30 runs) with random ordering of test cases. Original order executes the test cases in the order given in the test script provided with the object programs. The three strategies used are as follows:

- AHP: Uses the ART strategy utilizing the AHP method across all versions. This technique is used as the baseline strategy.
- Fuzzy AHP: Uses the ART strategy utilizing the fuzzy AHP method across all versions.
- FESART: A new ART strategy that utilizes a fuzzy expert system to select the best technique across all versions.

2) Dependent Variable and Measures: The dependent variable in the study is the *relative cost-benefit value*. To calculate this value, we used the EVOMO economic model [34]. EVOMO[1] involves two equations: one that captures the costs related to the salaries of the engineers who perform regression testing (to translate time spent into monetary values) and one that captures the revenue gains or losses related to changes in the system release time (to translate time-to-release into monetary values). Significantly, the model accounts for costs and benefits across entire system lifetimes, rather than snapshots (i.e. single releases) of those systems, through equations that calculate the costs and benefits *across entire sequences of system releases*. The major cost components that EVOMO captures are as follows: costs for applying regression testing techniques, costs associated with missed faults, costs for artifact analysis, costs of delayed fault detection feedback, and costs associated with obsolete tests.

[1]Due to space limitations, here we just summarize each equation briefly. See [38], [34] for detailed descriptions.

In addition to the costs already considered in the EVOMO model, we modified the EVOMO model to consider one additional cost: the cost of applying the ART strategy (C_{ART}). C_{ART} is a cost related to human effort, so it is applied in the equations in the same way as other costs related to human effort (by capturing the cost related to the salary of the engineer who performed the activity).

The cost and benefit calculations for the EVOMO model are measured in dollars. To determine the *relative cost-benefit* of the ART strategy, *S*, with respect to baseline strategy, *base* (the strategy utilizing the AHP method), we use the following equation:

$$(\mathrm{Benefit}_S - \mathrm{Cost}_S) - (\mathrm{Benefit}_{base} - \mathrm{Cost}_{base}) \qquad (6)$$

When this equation is applied, positive values indicate that S is beneficial compared to the $base$, and negative values indicate otherwise.

C. Experiment Setup and Procedure

In order to measure costs such as delayed fault detection, the object programs needed to contain some faults. We used mutation faults and mutant groups created by the *ByteME* (Bytecode Mutation Engine) tool from the SIR Repository [37]. Each mutant group contained, at most, 10 mutants that were randomly selected per version.

To evaluate the fuzzy expert system, two decision makers, each having seven years of industry experience in software development, rated the input criteria described in the previous section for each version of every object program. The decision makers rated each criterion for each prioritization technique in terms of their cost on a scale from 1 to 9, with 9 being considered a very high cost. This input was used in the fuzzy expert system. Then, based on the output provided by the fuzzy expert system, the decision maker determined which technique should be used for every version of each program. Techniques with lower numbers represent a lower cost for using that technique on that particular software version.

Regression testing sessions are often faced with strict deadlines, budgets, or both. Therefore, software companies often need to cut their testing activities short. How much the company has to cut the testing by can change due to the particular software version or a company's circumstances (e.g., different amount or complexity for a feature update, technical personnel loss, etc.). For this reason, we consider varying the time constraints for each version when we apply regression testing strategies in this experiment. We randomly assigned the level of time constraints (25%, 50%, or 75%) for each version. These time constraint levels represent situations where time constraints shorten the testing process by 25%, 50%, and 75%.

To implement time constraint levels, we shortened the test execution process for each version by the assigned time constraint level. Further, we ran four sets of random assignments across all versions for each program as shown in Figure 3. For instance, for run 1, we randomly assigned time constraints for each version: 50% for V1, 25% for V2, 75% for V3, and 50% for V4. We repeated this random assignment four times and defined each random assignment across versions as "Run n" (n = 1, 2, 3, and 4). Finally, each decision maker rated the criteria considering these time constraints, and these ratings were used in the fuzzy expert system. Cost-benefit calculations were collected for all strategies and used to determine which strategy was most cost-effective for that particular software version.

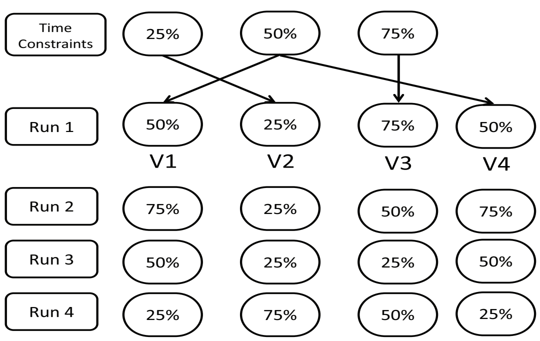

Fig. 3. Random Assignment of Time Constraint Level

D. Threats to Validity

This section discusses the construct, internal, and external threats to the validity of our study.

Construct Validity: The construct validity could be threatened by the number of criteria considered in this experiment. We considered four criteria, but additional criteria could be considered which could change the results. Also, we developed 67 rules for FESART. A fuzzy expert system with fewer or more rules could be developed and potentially change the results.

Internal Validity: The ratings from the decision maker were entered into the fuzzy expert system built in MATLAB. Each of the produced outputs from the expert system were double-checked, but the possibility of small marginal human errors still exists due to the ratings being hand entered into MATLAB.

External Validity: The external validity of this experiment could be limited in a couple ways. First, we chose to use three triangular membership functions for the input set and eight triangular membership functions for the output set. Many different numbers of membership functions could be considered, as well as different types (e.g. gaussian, trapezoidal, etc.). We cannot generalize our results because the type and number of membership functions we used are not representative of those functions. Also, we used two decision makers in this study. The backgrounds and experience levels for the decision makers could differ from those of professional programmers, so we cannot generalize our findings. We tried to reduce this risk by selecting decision makers who have several years of industry experience.

V. Data and Analysis

In this section, we present the results of our study. Each version for every program is assigned a random time constraint (25%, 50%, or 75%). This procedure is performed four times giving four runs of random time constraint levels for every version for all programs. The cost-benefit results for the four runs are shown in Table V. The AHP-based ART strategy is used as the baseline strategy in the relative cost-benefit calculation, so it is not displayed in the table.

The cost-benefit values for three programs are displayed in a subtable of Table V. The data in the table show the cost-benefit values, in dollars, with respect to the AHP-based ART strategy (baseline) defined in Section IV-B2. Positive cost-benefit values indicate greater cost-benefits than the baseline strategy, and negative values indicate fewer cost-benefits than the baseline strategy. Two decision makers were used in this study and the previous study [7] evaluating the AHP and fuzzy AHP strategies. Results for the first decision maker for the ART strategy utilizing fuzzy AHP are labeled Fuzzy AHP-1, and Fuzzy AHP-2 is used for the second decision maker. Similarly, the results for the first decision maker for FESART are labeled FESART-1, and FESART-2 is used for the second decision maker.

When examining the total cost-benefit values for each version of the program, we found that FESART was more cost-effective than the other ART strategies for all four runs of all five programs. One of the biggest reasons FESART was consistently more cost-effective than the other two strategies was because the cost of applying the ART strategy was much lower. If the strategies picked the same prioritization technique, FESART would be more cost-effective because the cost of applying the strategy was lower than that of the AHP and fuzzy AHP methods. For example, for version 2 of run 1 for *jmeter*, all strategies chose the additional block coverage technique as the most cost-effective technique. However, the costs of applying the AHP and fuzzy AHP strategies were higher, so FESART produced a better result. The AHP strategy took, on average between the two decision makers, 34 minutes, and the fuzzy AHP, averaged between the two decision makers, was 36 minutes. FESART took, on average, half the time of the other two strategies at 13 minutes. When looking at the total cost-benefit values between AHP and fuzzy AHP, fuzzy AHP was frequently more cost-effective than AHP.

To summarize our results and show them visually, we present them in bar graphs by averaging the total cost-benefit values for the four runs in Figure 4. The figure

TABLE V
EXPERIMENT RESULTS: RELATIVE COST-BENEFIT IN DOLLARS

ant

	Run 1				Run 2				Run 3				Run 4			
	Fuzzy AHP-1	Fuzzy AHP-2	FES-ART-1	FES-ART-2	Fuzzy AHP-1	Fuzzy AHP-2	FES-ART-1	FES-ART-2	Fuzzy AHP-1	Fuzzy AHP-2	FES-ART-1	FES-ART-2	Fuzzy AHP-1	Fuzzy AHP-2	FES-ART-1	FES-ART-2
v1	52	-5	88	37	-7	-3	30	173	-6	-5	81	36	52	-5	88	37
v2	-4	-2	29	39	-4	-2	29	39	-6	-2	31	39	-4	-2	34	40
v3	104	108	140	149	-47	-2	-13	38	104	108	140	149	-5	-47	29	38
v4	-3	-15	31	25	11	-2	46	39	26	28	60	67	-3	-15	31	25
v5	-3	-5	30	39	-3	-5	30	38	-3	-5	30	39	-3	-5	29	38
v6	-3	0	29	36	-3	0	29	36	-3	0	30	37	-3	0	30	38
v7	-3	-3	30	35	86	-3	71	36	-3	-3	29	34	-3	-3	30	35
v8	-3	2	29	36	-3	2	29	36	173	176	206	211	97	100	130	136
Total	**137**	**80**	**406**	**396**	**30**	**-15**	**251**	**435**	**282**	**297**	**607**	**612**	**128**	**23**	**401**	**387**

jmeter

	Run 1				Run 2				Run 3				Run 4			
	Fuzzy AHP-1	Fuzzy AHP-2	FES-ART-1	FES-ART-2	Fuzzy AHP-1	Fuzzy AHP-2	FES-ART-1	FES-ART-2	Fuzzy AHP-1	Fuzzy AHP-2	FES-ART-1	FES-ART-2	Fuzzy AHP-1	Fuzzy AHP-2	FES-ART-1	FES-ART-2
v1	29	-3	65	32	-7	42	76	78	29	-3	65	32	-7	-3	31	34
v2	-3	-2	34	36	-3	-2	33	36	-3	-2	34	36	-3	-2	34	36
v3	6	-2	44	34	6	-2	44	34	-5	-2	32	32	-5	-2	32	32
v4	-3	-2	36	130	-3	-2	35	39	-3	-2	36	40	-3	-2	36	40
v5	58	59	93	95	5	7	32	43	5	7	32	43	58	60	93	95
Total	**87**	**50**	**272**	**327**	**-2**	**43**	**220**	**230**	**23**	**-2**	**199**	**183**	**40**	**51**	**226**	**237**

xml-security

	Run 1				Run 2				Run 3				Run 4			
	Fuzzy AHP-1	Fuzzy AHP-2	FES-ART-1	FES-ART-2	Fuzzy AHP-1	Fuzzy AHP-2	FES-ART-1	FES-ART-2	Fuzzy AHP-1	Fuzzy AHP-2	FES-ART-1	FES-ART-2	Fuzzy AHP-1	Fuzzy AHP-2	FES-ART-1	FES-ART-2
v1	-5	-3	33	36	-5	-3	31	34	-5	-3	33	35	-5	-3	31	34
v2	-3	-2	36	40	-3	-2	36	35	-3	-2	37	41	-3	-2	36	40
v3	-2	0	34	36	-2	0	35	37	-2	0	34	36	-2	0	35	37
Total	**-10**	**-5**	**103**	**112**	**-10**	**-5**	**102**	**106**	**-10**	**-5**	**104**	**112**	**-10**	**-5**	**102**	**111**

nanoxml

	Run 1				Run 2				Run 3				Run 4			
	Fuzzy AHP-1	Fuzzy AHP-2	FES-ART-1	FES-ART-2	Fuzzy AHP-1	Fuzzy AHP-2	FES-ART-1	FES-ART-2	Fuzzy AHP-1	Fuzzy AHP-2	FES-ART-1	FES-ART-2	Fuzzy AHP-1	Fuzzy AHP-2	FES-ART-1	FES-ART-2
v1	-7	4	42	33	-7	-39	33	34	-6	3	40	40	-7	4	42	33
v2	-3	-2	35	35	-3	-2	34	35	-3	-2	34	35	-3	-2	35	35
v3	-5	-2	32	34	-5	-2	32	34	-5	-2	33	35	-5	-2	33	35
v4	47	49	87	91	47	49	87	91	47	49	87	91	47	49	87	91
v5	-3	-3	33	34	-3	-3	34	35	-3	-3	33	34	-3	-3	34	35
Total	**29**	**46**	**229**	**227**	**29**	**3**	**220**	**229**	**30**	**45**	**227**	**235**	**29**	**46**	**230**	**229**

galileo

	Run 1				Run 2				Run 3				Run 4			
	Fuzzy AHP-1	Fuzzy AHP-2	FES-ART-1	FES-ART-2	Fuzzy AHP-1	Fuzzy AHP-2	FES-ART-1	FES-ART-2	Fuzzy AHP-1	Fuzzy AHP-2	FES-ART-1	FES-ART-2	Fuzzy AHP-1	Fuzzy AHP-2	FES-ART-1	FES-ART-2
v1	-7	-5	30	37	-6	-5	-31	36	-7	-5	30	38	-6	-5	-31	36
v2	-4	-2	32	39	-4	-2	33	39	-4	-2	33	39	-4	-2	-35	40
v3	-5	-2	30	38	-5	-2	29	38	-5	-2	30	39	-5	-2	29	38
v4	-3	69	103	110	-3	69	103	110	-3	-2	31	38	-3	-2	-12	38
v5	-3	-3	29	-9	-3	-3	30	38	-3	-3	30	38	-3	-3	-17	39
v6	-3	0	30	38	-3	0	30	38	-3	0	47	54	-3	0	19	37
v7	-3	-3	30	35	-3	-3	29	34	-3	-3	30	36	-3	-3	29	34
v8	-3	-2	29	36	-3	-2	30	37	-3	-2	30	37	-3	-2	30	38
v9	-4	-2	33	39	-4	-2	32	39	-4	-2	34	-55	-4	-2	32	39
v10	-5	-2	30	39	-5	-2	30	39	-5	-2	29	38	-5	-2	30	38
v11	-50	-49	31	38	-120	115	31	38	-120	115	31	38	-120	115	32	155
v12	3	3	36	44	38	-3	71	38	1	1	30	43	38	-3	71	38
v13	135	139	169	176	135	139	169	176	174	177	206	213	174	177	207	214
v14	-3	-1	193	200	-3	90	70	127	-3	-89	116	36	-3	90	70	127
v15	-3	-2	30	38	-3	-2	30	38	-3	-2	29	36	-3	-2	30	37
Total	**42**	**138**	**835**	**898**	**8**	**387**	**686**	**865**	**9**	**179**	**736**	**668**	**47**	**354**	**484**	**948**

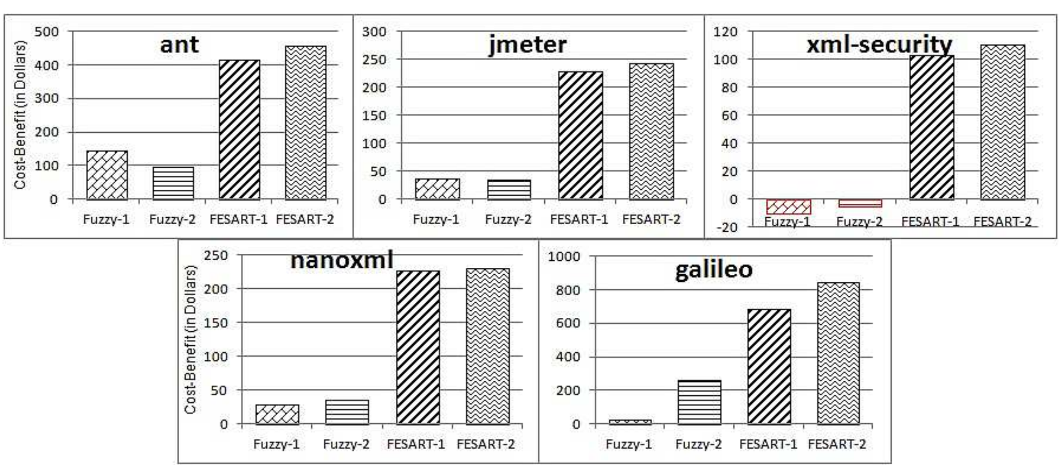

Fig. 4. Average Percentage of Cost-Effective Versions

shows the average totals for FESART being largely more cost-effective than the other strategies. In particular, in the case of *galileo*, the differences between FESART and others are more outstanding than other programs. Also, the totals show fuzzy AHP being more cost-effective than AHP for four of the five programs. For *xml-security*, the average results are negative, which means the strategy is less cost-effective than the baseline. The results for fuzzy AHP are negative for *xml-security* because the techniques chosen were often the same as the techniques chosen by the AHP strategy, and the cost of applying the fuzzy AHP strategy was slightly higher, making it less cost-effective for those cases.

The total cost-benefit values provide a general trend about the data, but one version's cost-benefit value can skew the results. Thus, we further examine the results for individual versions. Examining the results of each version provides more insight about the ART strategies. First, examining it this way shows that FESART is consistently more cost-effective than the other two strategies across all versions for all programs except for a few cases (e.g., version 7 of run 2 for the first decision maker for *ant*). Second, comparing AHP and fuzzy AHP for individual versions, we find that the overall trend is different from what we observed with the total value comparison. Although the fuzzy AHP is frequently more cost-effective than AHP in regards to the total cost-benefit values for programs, there are many versions which are less cost-effective than AHP because the cost of applying the fuzzy AHP strategy is slightly higher than the AHP strategy. We used a tool for the AHP calculations, but no such tool was available for fuzzy AHP. Instead, we wrote code in MATLAB to do the calculations and then manually entered the pairwise comparisons into MATLAB to calculate the results. This process is slightly more time consuming than the tool used for AHP, so when the two strategies choose the same technique, the fuzzy AHP strategy is slightly less cost-effective.

VI. Discussion and Implications

We developed FESART to address the limitations of the previously proposed ART strategies utilizing the AHP and fuzzy AHP methods. In this section, we will discuss how FESART effectively addresses these limitations as well as the implications of our results for researchers and practitioners.

FESART Strategy Results: Developing a strategy that does not require pairwise comparisons eliminates some of the problems with the previous ART strategies. First, the issue of inconsistent comparisons is eliminated. By not requiring the decision maker to rank the alternatives compared to other alternatives, the risk of inconsistency in the rankings is eliminated. Second, FESART is less time consuming than a strategy requiring pairwise comparisons. The number of rankings needed in FESART is reduced from the number of rankings required by pairwise comparisons, making it less time-consuming for the decision maker. Third, the decreased number of rankings required by the decision maker helps address the issue of scalability. Fewer rankings makes FESART more scalable than the other strategies.

By addressing these limitations, our results indicate that FESART is more cost-effective than the previously proposed ART strategies. One of the biggest contributors to FESART being more cost-effective is the reduction in the amount of time it takes to apply the strategy. Because the time required by the FESART strategy was less than the other two strategies, if the strategies chose the same technique, FESART was more cost-effective. In addition, in some situations, FESART chose a more cost-effective technique than the other strategies, making the total cost-savings even greater. One possible explanation for FESART choosing a more cost-effective technique than the other strategies is that some of the expert knowledge is placed in the rule base in the fuzzy expert system, so the amount of knowledge required by the decision maker is not as high as it is with the previous strategies.

Further Understanding About the Implications of the Results: The findings of our study provide practical implications for practitioners and researchers in software engineering. Our results show that FESART, that considers cost criteria related to testing environments and contexts, improves the cost-effectiveness of that regression testing session.

Savings of hundreds of dollars presented in this study may be unimportant. In practice, however, regression testing could take days or even weeks, so if results such as those presented in this study scale up, savings of the dollar amount may be substantial. For instance, in this study, we used small/medium sized programs, but typical industrial applications have millions of lines of code (e.g., a popular accounting software, Quickbooks, has over 80,000 files and ten million lines of code). Thus, if they were to apply the FESART strategy, the savings would be far greater than those presented in our study.

Further, the costs associated with the defects escaped into the released system could impact the results greatly. This study considered ordinary defects (not severe defects). A survey by Shull et al. [39] suggests that the effort to find and fix severe defects is far more expensive than non-severe defects. Thus, if we take into account different types of defects, our approach could have an even greater impact on cost savings related to early fault detection.

VII. Conclusions and Future Work

In this paper, we investigated a new adaptive regression testing (ART) strategy by building a fuzzy expert system for ART (FESART). FESART addresses the limitations of other proposed ART strategies. We conducted an empirical study to examine the cost-benefits for three ART strategies (*AHP*, *fuzzy AHP*, and *FESART*). Our empirical study included five Java programs with multiple versions. The results showed that FESART was consistently more cost-effective than the other ART strategies. These results were true when looking at the data by the total cost-benefit values for all versions and the number of versions that were most cost-effective.

In this study, we addressed many limitations of the previous ART strategies. There are still some limitations, mentioned in Section IV-D, which could be addressed. For example, our study considered three triangular membership functions to determine the fuzzy input set. Therefore, for future work, we intend to use other types and number of membership functions (e.g. gaussian or trapezoidal) for FESART. Further, with a different number of membership functions, we plan to update the rule set to investigate how these changes affect the outcome.

Through the results found in this paper and any additional work addressing the limitations mentioned above, we hope that useful insight is provided to help researchers and practitioners consider testing and environment factors to choose a regression testing technique for a particular software version.

Acknowledgements

This work was supported in part by NSF CAREER Award CCF-1149389 to North Dakota State University.

References

[1] H. Do, S. Mirarab, L. Tahvildari, and G. Rothermel, "The effects of time constraints on test case prioritization: A series of controlled experiments," *IEEE Transactions on Software Engineering*, vol. 26, no. 5, pp. 593–617, Sep. 2010.

[2] S. Elbaum, A. G. Malishevsky, and G. Rothermel, "Test case prioritization: A family of empirical studies," *IEEE Transactions on Software Engineering*, vol. 28, no. 2, pp. 159–182, Feb. 2002.

[3] A. Malishevsky, G. Rothermel, and S. Elbaum, "Modeling the cost-benefits tradeoffs for regression testing techniques," in *Proceedings of the International Conference on Software Maintenance*, Oct. 2002, pp. 204–213.

[4] S. Elbaum, G. Rothermel, S. Kanduri, and A. G. Malishevsky, "Selecting a cost-effective test case prioritization technique," *Software Quality Journal*, vol. 12, no. 3, 2004.

[5] M. J. Harrold, D. Rosenblum, G. Rothermel, and E. Weyuker, "Empirical studies of a prediction model for regression test selection," *IEEE Transactions on Software Engineering*, vol. 27, no. 3, pp. 248–263, Mar. 2001.

[6] M. Arafeen and H. Do, "Adaptive regression testing strategy: An empirical study," in *Proceedings of the International Symposium on Software Reliability Engineering*, 2011, pp. 130–139.

[7] A. Schwartz and H. Do, "A Fuzzy AHP Approach to Improve Adaptive Regression Testing Strategies," North Dakota State University, Tech. Rep. NDSU-CS-TR-13-001, 2013.

[8] T. L. Saaty, *The Analytic Hierarchy Process*. McGraw-Hill, 1980.

[9] J. Benítez, X. Delgado-Galván, J. Gutiérrez, and J. Izquierdo, "Balancing consistency and expert judgment in ahp," *Mathematical and Computer Modelling*, vol. 54, no. 7, pp. 1785–1790, 2011.

[10] L. Lin and C. Wang, "On consistency and ranking of alternatives in uncertain ahp," *Natural Science*, vol. 4, no. 5, pp. 340–348, 2012.

[11] S. Hatton, "Early prioritisation of goals book series," *Advances in Conceptual Modeling Foundations and Applications*, vol. 4802, no. 235–244, pp. 517–526, 2007.

[12] O. López-Ortega and M.-A. Rosales, "An agent-oriented decision support system combining fuzzy clustering and the ahp," *Expert Systems with Applications*, vol. 38, no. 7, pp. 8275–8284, 2011.

[13] T. L. Saaty and M. S. Ozdemir, "Why the magic number seven plus or minus two," *Mathematical and Computer Modelling*, vol. 38, no. 3, pp. 233–244, 2003.

[14] A. Adeli and M. Neshat, "A fuzzy expert system for heart disease diagnosis," in *Proceedings of International Multi Conference of Engineers and Computer Scientists, Hong Kong*, vol. 1, 2010.

[15] M. A. Kadhim, M. A. Alam, and H. Kaur, "Design and Implementation of Fuzzy Expert System for Back pain Diagnosis," *International Journal of Innovative Technology & Creative Engineering (IJITCE)*, no. 9, pp. 16–22, 2011.

[16] S. Yoo and M. Harman, " Regression testing minimisation, selection and prioritisation: A survey," *Software Testing, Verification, and Reliability*, pp. 67–120, Mar. 2010.

[17] R. Aull-Hyde and K. A. Davis, "Military applications of the analytic hierarchy process," *International Journal of Multicriteria Decision Making*, vol. 2, no. 3, pp. 267–281, 2012.

[18] N. Subramanian and R. Ramanathan, "A review of applications of analytic hierarchy process in operations management," *International Journal of Production Economics*, vol. 138, no. 2, 2012.

[19] Y. Zhang, X. Zhang, X. Zhao, and T. Zhang, "Early effort estimation by ahp: A case study of project metrics in small organizations," in *2012 IEEE International Conference on Computer Science and Automation Engineering (CSAE)*, vol. 1. IEEE, 2012, pp. 452–456.

[20] A. Perini, F. Ricca, and A. Susi, "Tool-supported requirements prioritization: Comparing the AHP and CBRank methods," *Information and Software Technology*, vol. 51, pp. 1021–1032, Jun. 2009.

[21] C. Shen, M. Cheng, C. Chen, F. Tsai, and Y. Cheng, "A fuzzy AHP-based fault diagnosis for semiconductor lithography process," *International Journal of Innovative Computing, Information and Control*, vol. 7, no. 2, pp. 805–815, Feb. 2011.

[22] C.-C. Sun, "A performance evaluation model by integrating fuzzy ahp and fuzzy topsis methods," *Expert systems with applications*, vol. 37, no. 12, pp. 7745–7754, 2010.

[23] V. Ahl, "An experimental comparison of five prioritization methods," *Master's Thesis, School of Engineering, Blekinge Institute of Technology, Ronneby, Sweden*, 2005.

[24] S. Lee, "Using fuzzy AHP to develop intellectual capital evaluation model for assessing their performance contribution in a university," *Expert Systems with Applications*, vol. 37, no. 7, pp. 4941–4947, 2010.

[25] M. Fasanghari and G. A. Montazer, "Design and implementation of fuzzy expert system for tehran stock exchange portfolio recommendation," *Expert Systems with Applications*, vol. 37, no. 9, pp. 6138–6147, 2010.

[26] S. Kad and V. Chopra, "Fuzzy logic based framework for software development effort estimation," *Research Cell: An International Journal of Engineering Sciences*, vol. 1, pp. 330–342, 2011.

[27] M. Kazemifard, A. Zaeri, N. Ghasem-Aghaee, M. Nematbakhsh, and F. Mardukhi, "Fuzzy emotional cocomo ii software cost estimation (fecsce) using multi-agent systems," *Applied Soft Computing*, vol. 11, no. 2, pp. 2260–2270, 2011.

[28] Z. Xu, K. Gao, and T. M. Khoshgoftaar, "Application of fuzzy expert system in test case selection for system regression test," in *Proceedings of International Conference on Information Reuse and Integration*. IEEE, 2005, pp. 120–125.

[29] L. Zadeh, "Fuzzy sets," *Information and Control*, vol. 8, no. 3, pp. 338–353, 1965.

[30] E. H. Mamdani and S. Assilian, "An experiment in linguistic synthesis with a fuzzy logic controller," *International journal of man-machine studies*, vol. 7, no. 1, pp. 1–13, 1975.

[31] T. Takagi and M. Sugeno, "Fuzzy identification of systems and its applications to modeling and control," *IEEE Transactions on Systems, Man and Cybernetics*, no. 1, pp. 116–132, 1985.

[32] P. C. Shill, K. K. Pal, M. F. Amin, and K. Murase, "Genetic algorithm based fully automated and adaptive fuzzy logic controller," in *IEEE International Conference on Fuzzy Systems*. IEEE, 2011, pp. 1572–1579.

[33] P. R. Srivastava, S. Kumar, A. Singh, and G. Raghurama, "Software testing effort: An assessment through fuzzy criteria approach," *Journal of Uncertain Systems*, vol. 5, no. 3, pp. 183–201, 2011.

[34] H. Do and G. Rothermel, "Using sensitivity analysis to create simplified economic models for regression testing," in *Proceedings of the International Conference on Software Testing and Analysis*, Jul. 2008, pp. 51–62.

[35] S. Elbaum, A. Malishevsky, and G. Rothermel, "Incorporating varying test costs and fault severities into test case prioritization," in *Proceedings of the 23rd International Conference on Software Engineering*. IEEE Computer Society, 2001, pp. 329–338.

[36] H. Do, S. Elbaum, and G. Rothermel, "Supporting controlled experimentation with testing techniques: An infrastructure and its potential impact," *International Journal on Empirical Software Engineering*, vol. 10, no. 4, pp. 405–435, 2005.

[37] H. Do and G. Rothermel, "On the use of mutation faults in empirical assessments of test case prioritization techniques," *IEEE Transactions on Software Engineering*, vol. 32, no. 9, pp. 733–752, Sep. 2006.

[38] ——, "An empirical study of regression testing techniques incorporating context and lifecycle factors and improved cost-benefit models," in *Proceedings of the ACM SIGSOFT Symposium on Foundations of Software Engineering*, Nov. 2006, pp. 141–151.

[39] Shull, F. et al., "What we have learned about fighting defects," in *Int'l. Softw. Metrics Symp.*, 2002, pp. 249–258.