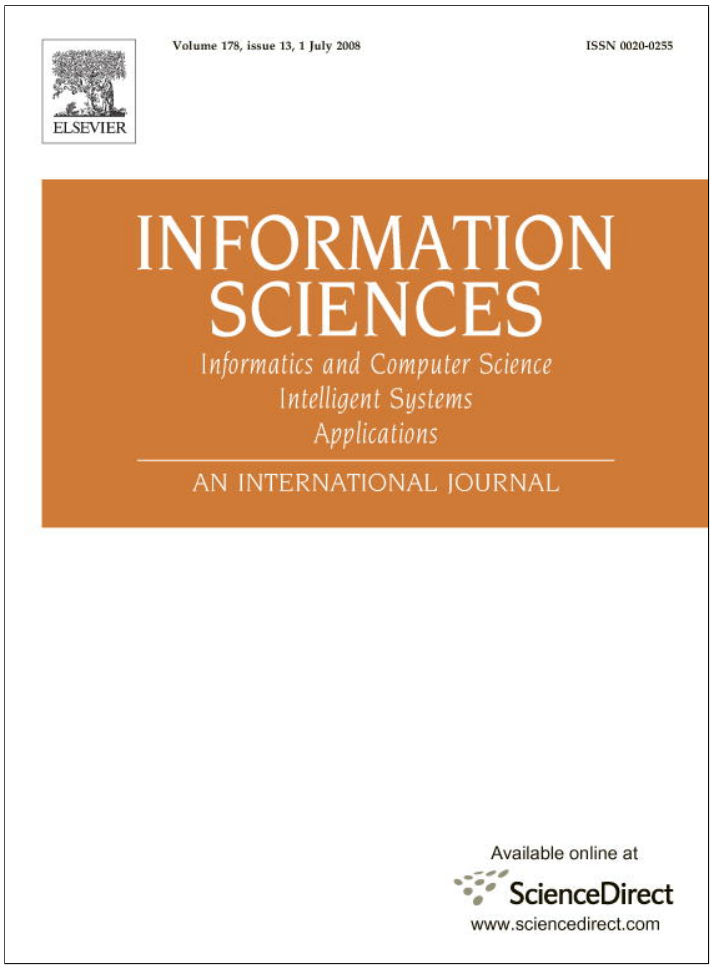

Available online at www.sciencedirect.com
ScienceDirect
Information Sciences 178 (2008) 2780–2793
INFORMATION SCIENCES
AN INTERNATIONAL JOURNAL
www.elsevier.com/locate/ins

Fuzzy inference system for software product family process evaluation

F. Ahmed [a,*], L.F. Capretz [b], J. Samarabandu [b]

[a] College of Information Technology, PO Box 17551, United Arab Emirates University, Al Ain, UAE
[b] Department of Electrical and Computer Engineering, University of Western Ontario, London, Ontario, Canada N6A 5B9

Received 30 April 2005; received in revised form 25 February 2008; accepted 3 March 2008

Abstract

When developing multiple products within a common application domain, systematic use of a software product family process can yield increased productivity in cost, quality, effort and schedule. Such a process provides the means for the reuse of software assets which can considerably reduce the development time and the cost of software products. A comprehensive strategy for the evaluating the maturity of a software product family process is needed due to growing popularity of this concept in the software industry. In this paper, we propose a five-level maturity scale for software product family process. We also present a fuzzy inference system for evaluating maturity of software product family process using the proposed maturity scale. This research is aimed at establishing a comprehensive and unified strategy for process evaluation of a software product family. Such a process evaluation strategy will enable an organization to discover and monitor the strengths and weaknesses of the various activities performed during development of multiple products within a common application domain.

Keywords: Software product family; Adaptive neuro-fuzzy inference system; Fuzzy logic; Process maturity; Software process assessment; Software engineering

1. Introduction

Effective utilization of software assets is one of the major concerns of software development organizations. Such utilization has the potential for reducing the development time, product defects and cost of software products considerably. In recent years software development organizations have shown a growing interest in the concept of a software product line because it deals with effective utilization of software assets. The software product line is a comprehensive model for an organization that is building applications which are based on common architecture and software assets [29]. Clements defines a software product line as a set of software-intensive systems sharing a common, managed set of features that satisfy the specific needs of a particular

* Corresponding author. Tel.: +1 971 050 9357086; fax: +1 971 03 762 6309.
E-mail addresses: f.ahmed@uaeu.ac.ae (F. Ahmed), lcapretz@eng.uwo.ca (L.F. Capretz), jagath@uwo.ca (J. Samarabandu).

0020-0255/$ - see front matter © 2008 Elsevier Inc. All rights reserved.
doi:10.1016/j.ins.2008.03.002

market segment or mission and that are developed from a similar set of core assets in a prescribed way [7]. Synonyms of the term "software product line" have been widely used in Europe. Some of these include "product family", "product population", and "system family". The European project, Engineering Software Architecture, Processes and Platforms for System-Families (ESAPS) defines, "system family" as a group of systems sharing a common, managed set of features that satisfy the basic needs of a scoped domain [11]. Ommering further elaborated the term "product population" to refer to a collection of related systems based on similar technology but having many differences among them [20]. Clements et al. report that software product line engineering is a growing software engineering sub-discipline and many organizations such as Philips®, Hewlett–Packard®, Nokia®, Raytheon® and Cummins® are using it to achieve extraordinary gains in productivity, time to market and product quality. The economic potential of software product family has long been recognized in the software industry [3,28].

Software process maturity evaluation has been a key research area in the software research community because of its impact on the productivity of the development process. An organization dealing with a software product family requires a methodology to evaluate the process maturity of the software product family. Such a strategy includes the definition of process maturity levels as well as a process assessment approach. In this paper, we propose such a definition and an assessment approach based on a fuzzy inference system. Fig. 1 illustrates the conceptual layout of the proposed software product family process evaluation approach. The first stage of the process assessment approach identifies four variables, which constitute the overall maturity of a software product family process. The acronym Business-Architecture-Process- Organization (BAPO) is used to indicate the four variables used in the evaluation of the maturity of a software product family process [27]. The second stage is to develop an assessment framework for each variable. In case of software product lines, this results in a numeric value on a scale of 1–5 for each of the four variables. The last stage is to establish a relationship among these four variables in order to assign an overall process maturity level to an organization. The software product family process maturity level reflects the current process maturity in the organization. The main contribution of this research is in establishing a strategy for process evaluation of software product family. A five-level scale is proposed in this paper to reflect the maturity of software product

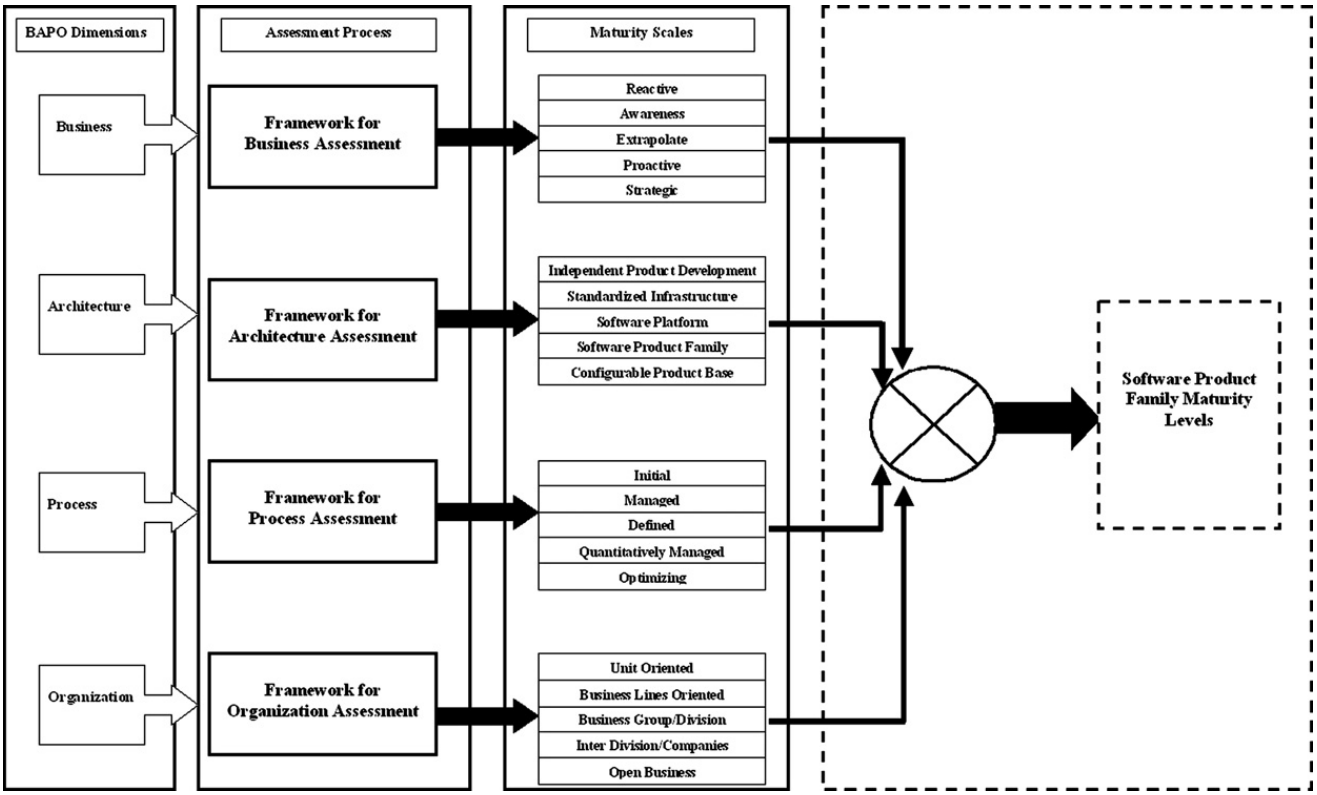

Fig. 1. Software product family process evaluation.

family process. A fuzzy inference system is presented that establishes a relationship among four variables of software product family process using the proposed maturity scale.

1.1. Related work in software product family process evaluation

Software product family process evaluation is relatively a young area of research. Currently, researchers from academe and industry are attempting to come up with a prescribed and systematic way of measuring the maturity of a software product family process. Jones and Soule discuss the relationships between the software product line process and the Capability Maturity Model Integration (CMMI) and observe that the software engineering process discipline as specified in CMMI provides an essential foundation for the software product line process [18]. They conclude that besides the key process areas of the CMMI-model, the software product line also requires the mastery of many other essential practice areas. Although Jones and Soule compare the key process areas of the software product line with the CMMI-model and find some similarities, they do not discuss any procedure to evaluate the maturity of software product family process. They conclude that there is a growing need to establish a comprehensive strategy for process assessment of software product lines. The Software Engineering Institute (SEI) proposed the Product Line Technical Probe (PLTP) which is aimed at discovering the ability of an organization to adapt and succeed with the software product line approach [8,9]. It identifies those potential areas of concern that require attention while managing software product line process. In the PLTP framework there are 29 practice areas. But this framework also does not discuss any methodology to evaluate the maturity of a software product family.

In a previous study, Ahmed and Capretz proposed a set of rules for developing and managing a software product line within an organization [1]. On the basis of those rules, a fuzzy logic based Software Product Line Process Assessment Tool (SPLPAT) was designed and implemented. This tool preprocesses the software product line process data and evaluates the maturity of the software product line process within a company. A number of case studies were conducted on the process data from reputable software development organizations. Outcome of this approach was compared with the existing CMMI-level of the organization in order to compare the assessment produced by two different approaches. This study also suggested that there is a need to establish a unified and comprehensive strategy for process assessment of a software product line.

Linden et al. proposed a software product family evaluation model based on the BAPO concept of operations, which provides a foundation for systematic and comprehensive strategy for the evaluation of a software product family process [27]. The solid rectangles in Fig. 1 illustrate the conceptual layout of the proposed evaluation model. The four variables of the model are business, architecture, process and organization. Each variable identifies an evaluation scale of up to five-levels in ascending order as shown in the rectangle of "maturity scales" in Fig. 1. In this study, the overall process maturity evaluation of an organization is proposed as a set of four values.

1.2. Research objectives

The purpose of this research is to contribute in establishing the BAPO-based process evaluation methodology of software product family by addressing some of the areas that have not been investigated. These areas are represented by dashed rectangles in Fig. 1 and highlight the scope of the work presented in this paper. These include:

- Defining a maturity scale for software product family process.
- A methodology to evaluate the overall maturity level of an organization once the assessment results of individual variables, such as business, architecture, process, and organization have been obtained.

The maturity levels proposed in this paper highlight the capability of an organization to implement, execute and control the software product family process. We also propose a fuzzy inference system (FIS) for evaluating a software product family process. In order to assign a maturity profile to an organization the proposed FIS establishes a relationship among the four variables of BAPO. The rest of this paper is organized as follows: we begin with a brief introduction of the maturity levels of software product family in Section 2.

In Section 3 we present the details of the proposed FIS for the software product family process evaluation. Finally, we discuss the conclusion of this study in Section 4.

2. Software product family maturity levels

This paper proposes five-levels maturity scale of software product family process. The five maturity levels of the software product family are "initial", "infrastructure", "launched", "institutionalized" and "optimized". The first level (initial) and the fifth level (optimized) are similar to the lowest and highest respective levels being used by most of the existing software process assessment approaches such as CMMI and BOOTSTRAP. We intentionally define our lowest and highest levels on the previously existing approaches in order to keep software product family maturity levels close to the existing popular scales already used in the software industry. The three intermediate levels have been specifically defined to reflect the maturity of a software product line process within an organization. Each maturity level indicates the competence of an organization for handling and implementing the concept of the software product family. The profile of an organization in terms of these maturity levels depicts its ability to organize, establish, maintain and control the product family process.

At level one, the "initial" level process maturity for a software product family indicates that the organization has not yet introduced a standardized and organized environment for establishing a product family. Above this, the "infrastructure" level represents an organized process with defined strategies and polices to establish the infrastructure of a software product family in order to produce multiple products within an application domain. At level three, the "launched" level highlights the success of an organization in developing and maintaining a software product family. The established product line processes and the organization-wide plans predict that businesses will be successful in capturing a major portion of the market segment. On the fourth level, the "institutionalized" level indicates the ability of an organization to manage the software product family process effectively from both technical and organizational perspectives. Finally, the companies at the "optimized" level tend to improve the performance of the software product family process by examining each and every step in order to increase the productivity of the process. In the following sub-sections, we discuss each maturity level in detail.

2.1. Initial (level-1)

The initial level of the software product family process maturity indicates that the organization has not yet introduced an organized environment for establishing a software product family. They are at an early stage of their commitment to establishing the software product family process. The policies regarding the organizational structure, the development of core assets and the business decisions have not been clearly defined. Most of the activities related to the software product family are conducted on an informal basis and there is a lack of understanding of software product family process. The organization has not yet introduced software product family as a part of their strategic planning. The product development activities are carried out independently and there is no established link present for the commonality and variability of features among successive products. Software product family architecture is completely missing.

2.2. Infrastructure (level-2)

The infrastructure level requires an organized process with defined strategies and policies to set up the software product family platform. An organization at this level concentrates on establishing the foundation for the software product family either actively or proactively. This results in an initial repository of core assets, architecture definition and recovery as well as the identification of commonality and variability among business cases. Software product family requirements are clearly defined and documented. A comprehensive domain analysis exercise results in the scope definition of the product line. At this level, the organization is able to produce multiple test products resulting from the software product family. Strategic plans of such a company consider the software product family as an asset. Due to initial cost of operations and long-term payback period, the potential economic benefits of the software product family are not very evident at this level.

2.3. Launched (level-3)

The launched level highlights the success of an organization in developing and maintaining a software product family. The consistencies in the process definitions and execution plans predict the ability of an organization to achieve success in software product family. Repositories of core assets are well maintained and updated regularly. A well-established communication pattern among various groups of the organizational structure allows using the core assets repositories in a more effective way. Business cases are generated after a comprehensive market orientation. The organization establishes a comprehensive software architectural platform upon which multiple products may be developed. The resulting products share this common architecture platform and have variable features in their functionalities. The company is able to produce multiple products based on market demands and is able to start enjoying the economic impacts of establishing a software product family. The organizational culture in such a company supports the concept of software product family. Their strategic plans consider software product family as an important asset in achieving organizational goals.

2.4. Institutionalized (level-4)

The institutionalized level indicates the ability of an organization to effectively control the software product family process from technical and organizational perspectives. The organizational behavior supports the software product family process in perception, attitude, and responsibilities. This level depicts the richness of the organizational culture and organizational commitment in the software product family. Employees in the company actively participate in achieving the benefits from the software product family. Such an organization perceives software product family as a strategic asset for business growth. The product development decisions of the organization are influenced by the software product family infrastructure. Successful business cases from the product line capture major portions of the market segments for the organization. The entire software product family process is precisely planned, agreeably executed and competently controlled at all levels of the organization. Organizations at this level are able to establish and successfully maintain more than one product family in order to achieve its business goals. Finally, the overall engineering process is mature enough to achieve the required goals of the organization.

2.5. Optimized (level-5)

The organization at the optimized level tends to improve the performance of the software product family process by examining each and every step in order to increase the productivity of the process and to improve the quality of products. The company learns from previous projects and prepares strategies to overcome potential failures. The decreased defect ratio of successive products shows a tremendous growth in established quality controlled environment. At the optimized level, an organization shows continuous growth in the software product family process. It prepares long-term plans to improve current technology and demonstrates its motivation towards researching new ideas and concepts in order to increase the effectiveness of the software product family process. The business explores new venues and ventures to support the current software product family infrastructure. Finally, the organization adopts a policy of open business, and it collaborates with other companies to establish joint ventures that produce better options of developing new products and that achieves most of the benefits offered by the product family concept.

3. Fuzzy inference system for software product family process evaluation

Fuzzy logic has been a key area of research and its application has been reported in various application domains such as control systems, data mining, medicine, software engineering, robotics and business [4,12,13,22,24,26]. The FIS is a popular computing framework based on the concepts of fuzzy set theory, fuzzy "if-then" rules, and fuzzy reasoning [16]. In the process of evaluating the maturity of a software product family, the set of four variables, business, architecture, process and organization, are used to estimate the maturity of a software product family. Each variable indicates the maturity level of an organization in its respective category on a scale of 1–5.The FIS for evaluation of the maturity of software product family process proposed

in this work establishes a relationship among the four variables of business, architecture, process and organization in order to assign an overall maturity level to an organization. In our implementation we used adaptive neuro-fuzzy inference system (ANFIS) [15]. The objective of using ANFIS is to optimize the parameters of the equivalent FIS by applying a learning algorithm using input–output data sets. The resulting FIS is a Takagi–Sugeno (TS) fuzzy model [25]. There are two major reasons for using the TS-fuzzy model in comparison to the Zadeh [31–34] or Mamdani [19] models in our proposed approach. First, it is relatively easy to implement TS-fuzzy model in ANFIS. Secondly the output generated by FIS does not require further defuzzification.

Fig. 2 illustrates the conceptual layout of ANFIS-based approach of a FIS for software product family maturity evaluation. The use of ANFIS approach enables us to establish a relationship among the four variables of business, architecture, process and organization. This leads to a single metric that characterize the overall process maturity level of an organization. The process of constructing FIS for software product family maturity evaluation using the approach of ANFIS involves the following steps:

1. Defining an initial set of membership functions for the four variables: business, architecture, process and organization.
2. Defining the overall software product family maturity levels to the output membership functions.

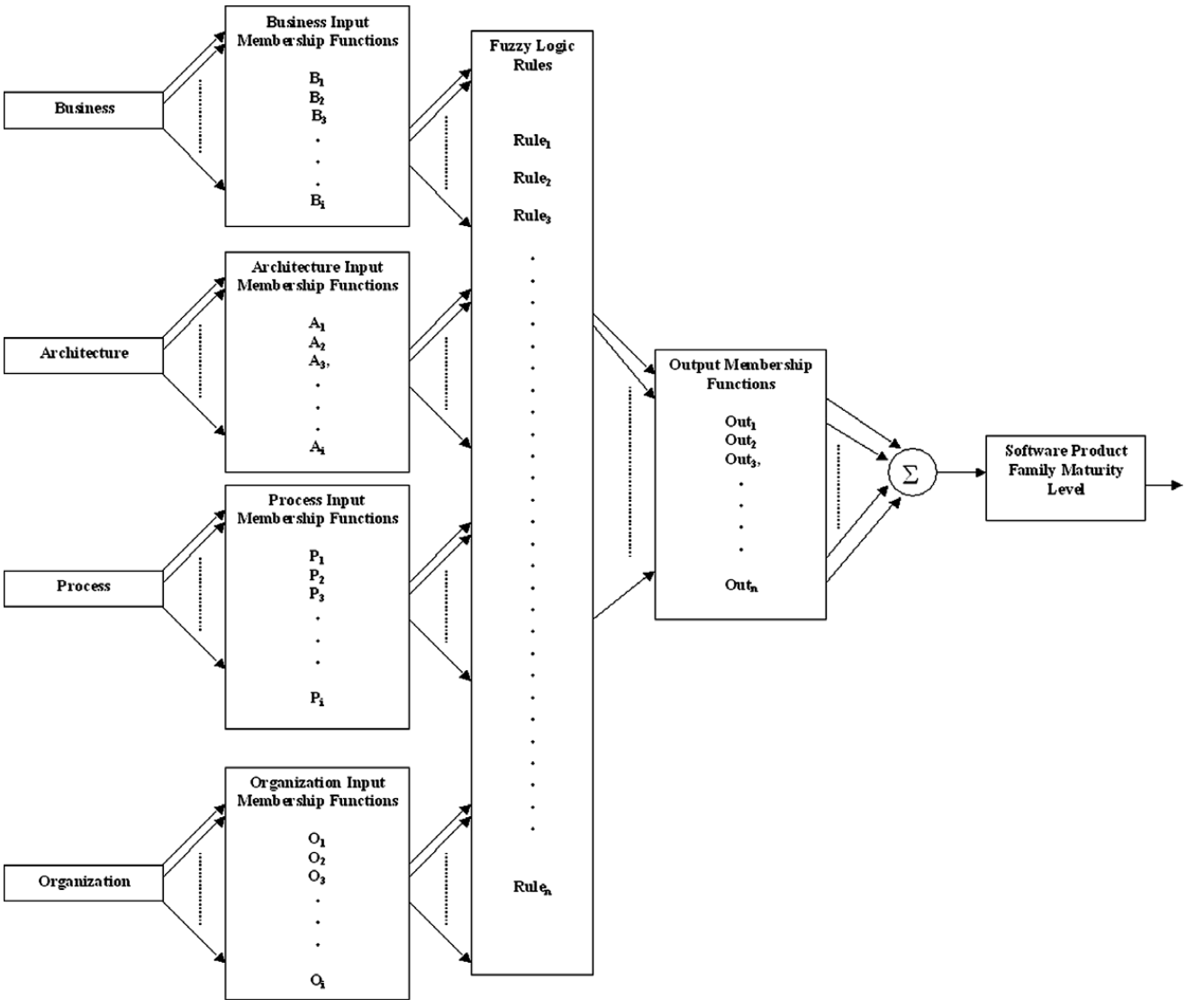

Fig. 2. ANFIS-based conceptual model of a FIS for software product family process evaluation.

3. Tuning the membership functions of business, architecture, process and organization by a learning process using an artificial neural network.
4. Constructing fuzzy inference rules.
5. Evaluating the output membership function as a single-valued output.

3.1. Linguistic variables

There are four input variables of the software product family maturity evaluation process in BAPO model. Therefore, input is divided into four linguistic variables of business, architecture, process, and organization. Each input variable is divided into categories of five maturity levels in ascending order [2]. We propose five-level maturity scales in this work (Section 2) for the output linguistic variable of software product family maturity. Thus, input and output to the system falls in the range of 0–5 for each linguistic variable.

A Gaussian function, as shown in Eq. (1), is used to represent the mapping between fuzzy membership in the range of 0 to 1, and input values in the range of 0–5. Besides smoothness and concise notations there are other advantages in using a Gaussian function. First, the characteristics of the linguistic labels and their parameters such as the center and width values denoted as C_i and σ_i in Eq. (1) can be easily adjusted. Secondly, the output classification does not introduce irregular variations due to small changes in the input pattern.

$$\mu_{A^i}(x) = \exp\left(-\frac{(c_i - x)^2}{2\sigma_i^2}\right). \tag{1}$$

Fig. 3 illustrates the shape of initial membership functions for the input and output linguistic variables of software product family evaluation.

The levels 1–5 for each input dimension of business, architecture, process and organization and the output maturity of software product family process are presented in Table 1, together with the maturity levels and the output linguistic variables.

3.2. Data preparation

A fuzzy model is based on set of "if-then" rules that describe the relationships between variables. Babuska defined the term "fuzzy structure identification" as techniques and algorithms used for constructing fuzzy models from data set [2]. The commonly used approaches for constructing fuzzy models from data set are expert knowledge that is translated into a set of "if-then" rules or fuzzy model constructed from data based on certain algorithms. This work utilizes the first approach where expert knowledge is translated into a set of if-then-rules. Pedrycz reported that fuzzy rule-based systems could be used as models constructed from the

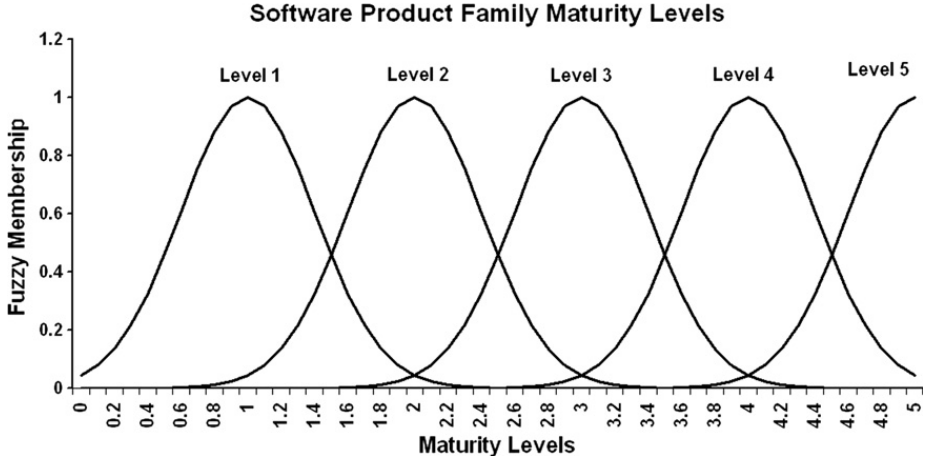

Fig. 3. Initial membership functions for software product family process evaluation.

Table 1
Input–output linguistic variables

Maturity levels	Mapping of input linguistic variables				Output linguistic variable
	Business	Architecture	Process	Organization	Software product family maturity levels
Level 1	Reactive	Independent product development	Initial	Unit oriented	Initial
Level 2	Awareness	Standardized infrastructure	Managed	Business lines oriented	Infrastructure
Level 3	Extrapolate	Software platform	Defined	Business group/ division	Launched
Level 4	Proactive	Software product family	Quantitatively managed	Inter division/ companies	Institutionalized
Level 5	Strategic	Configurable product base	Optimizing	Open business	Optimized

knowledge of experts in a particular field [21]. In order to collect the knowledge from experts we prepared a questionnaire which contains various combinations of values for the variables of business, architecture, process and organization. These combinations of values in the questionnaire were generated randomly. We requested experts in the software industry to provide us with their knowledge and judgment of possible corresponding software product family maturity level. After receiving the data from experts, we divided it into two sets. One set is used for training the FIS and the other is used for validating the generated model.

We processed the knowledge received from experts by using the clustering approach. The purpose of clustering was to separate data set into a number of similar groups. This measure of similarity reflects the consensus among experts as well. Commonly used clustering algorithms are "*k*-means" and "fuzzy *c*-means". We did not use these two algorithms in this study because the number of clusters to be determined is required in order to run these algorithms. Yager and Filev proposed mountain clustering, which was further improved by Chiu and called "subtractive clustering" [30,6]. Subtractive clustering eliminates the restriction imposed by *k*-means and fuzzy *c*-means algorithms. In this work we used subtractive clustering approach. We observed 15 clusters showing the various combinations of business, architecture, process and organization values with respective output value. Appendix I illustrates the results of the subtractive clustering. The range of influence, squash factor, acceptance ratio, and rejection ratio were set at 0.5, 1.25, 0.5 and 0.15, respectively during the process of subtractive clustering.

3.3. Fuzzy inference rules

The fuzzy inference rules in TS-fuzzy model have antecedent and consequent parts. The antecedent part is comprised of the input membership function which is assigned to each linguistic variable. The consequent part consists of parameters of the output function. ANFIS uses back propagation and the method of least square in order to learn antecedent and consequent parameters. ANFIS uses the following two steps during learning process.

- Step-I: Input patterns (antecedent parameters) are propagated and consequent parameters are estimated by using iterative least square method. In this learning process, input parameters are considered to be constant for the current cycle.
- Step-II: Back propagation is used to train the antecedent parameters (membership function) while keeping consequent parameters fixed. The whole process is repeated until convergence is obtained.

The use of ANFIS approach in this study for software product family maturity evaluation resulted in 21 fuzzy inference rules. The antecedent parameters of the resulting fuzzy inference rules represent the modified membership functions of four input variables of business, architecture, process, and organization generated after the completion of training cycle. In this study we used four inputs and one output variable. The proposed FIS generates a six dimensional transfer function, which shows the relationship among the four inputs and the

output variable. In order to provide an interpretation we show a pair of three-dimensional cross sections of the decision surface generated by FIS in Figs. 4 and 5. These cross sections illustrate the three-dimensional plot between two inputs and the output while keeping other two inputs fixed. Fig. 4 shows the decision surface with respect to the parameters "business" and "architecture", while keeping the parameters "process" and "organization" fixed at a maturity level of 3. Fig. 5 shows the decision surface with respect to the parameters "process" and "architecture", while keeping parameters "business" and "organization" fixed at maturity values of 2.5 and 3, respectively.

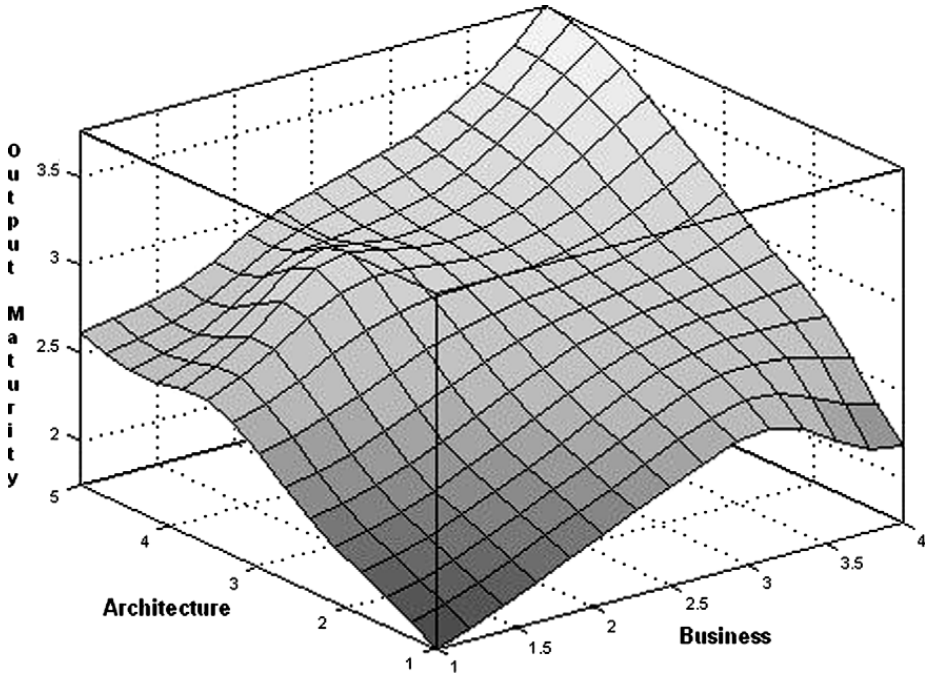

Fig. 4. Decision surface of business and architecture.

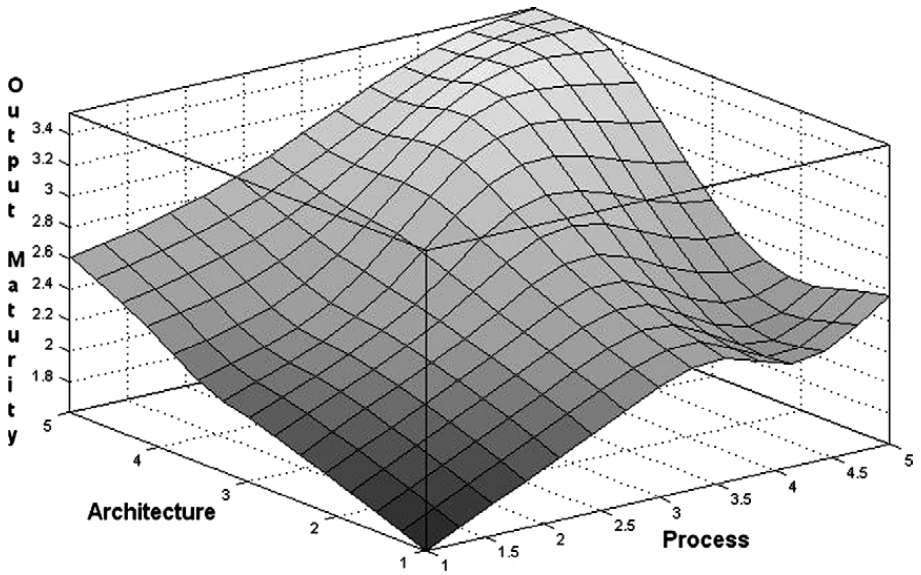

Fig. 5. Decision surface of process and architecture.

3.4. Interpretability and degree of overlap of fuzzy sets

The work presented in this paper used ANFIS approach to generate TS-fuzzy model. Subtractive clustering technique is used in this work in order to generate fuzzy sets based on knowledge received from experts. Clusters generated by subtractive clustering technique can overlap each other. This leads to the issue of linguistic interpretability of the fuzzy sets because a higher degree of overlap makes them quite indistinguishable. We observe this phenomenon in our study where the ANFIS generates a large number of fuzzy sets with a high degree of overlap. For example, in case of business membership function, the ANFIS generated 21 fuzzy sets and the similarity among the fuzzy sets was more than 0.8. Similar issues are present in other membership functions of architecture, process and organization. In order to improve the interpretability of fuzzy sets, we merged the fuzzy sets that are similar [5,17,23]. Fig. 6 depicts the flowchart of the approach used in this study.

The approach illustrated in Fig. 6 is an iterative procedure. First, it finds the most similar fuzzy sets in the rule base. Then the fuzzy sets that are most similar are merged and the rule base is updated accordingly. This process continues until there are no more pairs of fuzzy sets with a similarity greater than the given threshold. Once all the fuzzy sets above the given similarity threshold are merged, then it iteratively checks the similarity in the premise part of each rule. The rules are reduced wherever it is applicable. Main components of this interpretability improvement process are similarity measurement (Step-I), merging (Step-II) and rule reduction (Step-III). These steps are elaborated as follows:

Step-I: Similarity/overlapping measurement:Geometric approaches or set-theoretic approaches can be used in order to measure the similarity among fuzzy sets [23]. The geometric approach calculates the similarity as a function of distance among the fuzzy sets, as illustrated by Eq. (2). In contrast, the set-theoretic approach uses intersection and union operations to determine the similarity between fuzzy sets as shown in Eq. (3) [10]. Two fuzzy sets are similar if the similarity measure $S(A,B)$ is greater then a threshold value. Previous studies have shown that set-theoretic approach is more suitable for capturing similarity among overlapping fuzzy sets [35]. Hence, in this paper we used this set-theoretic similarity measure.

$$S(A,B) = \frac{1}{1 + D(A,B)}, \tag{2}$$

$$S(A,B) = \frac{|A \cap B|}{|A \cup B|}. \tag{3}$$

In case of Gaussian function the distance between fuzzy sets A and B is expressed by Eq. (4):

$$D(A,B) = \sqrt{(C_a - C_b)^2 + (\sigma_a - \sigma_b)^2}. \tag{4}$$

Step-II: Merging of fuzzy setsWhen two fuzzy sets are similar they are merged together to produce an updated fuzzy set. In case of Gaussian function, the merged fuzzy set can be constructed by calculating C and σ as given by Eqs. (5) and (6), as follows [14]:

$$C_{\text{Merged}} = \frac{c_a\sigma_a + c_b\sigma_b}{\sigma_a + \sigma_b}, \tag{5}$$

$$\sigma_{\text{Merged}} = \frac{\sigma_a + \sigma_b}{2}. \tag{6}$$

Step-III: Rule reductionIn the rule reduction step, we check the equality in the premise parts of all the rules. If premise parts of "n" rules are equal, then we remove "$n-1$" rules and re-estimate the consequent parameters. Different approaches such as weighting and averaging as well as using the training data set have been proposed to re-estimate the consequent parts [23].

3.4.1. Results of interpretability improvement

After simplifying fuzzy sets and rules, the membership functions of business and organization are reduced from 21 to 4 fuzzy sets. In the case of architecture and process, they are reduced to 5 fuzzy sets. We experi-

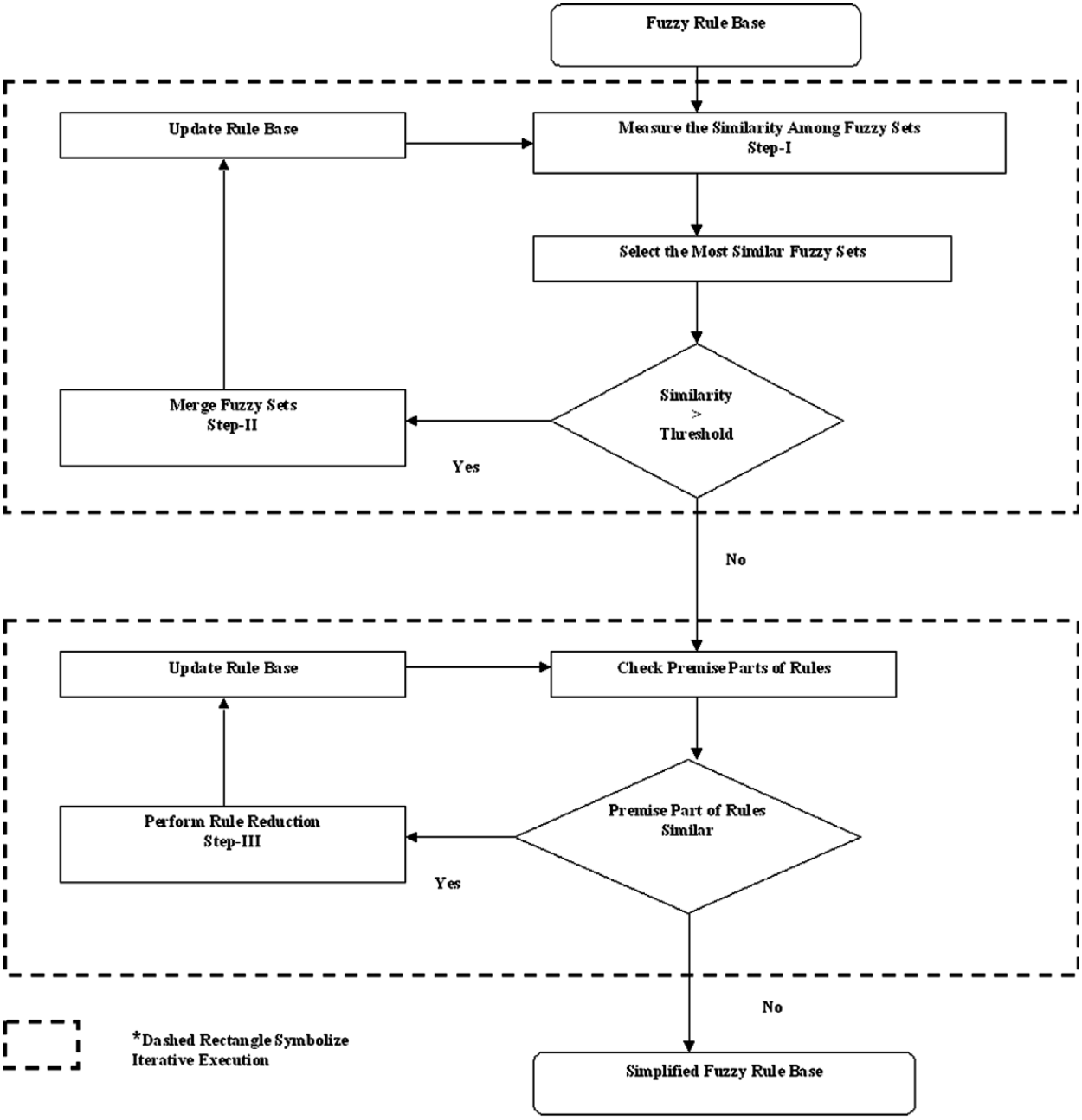

Fig. 6. Interpretability improvement process flow chart.

mented with different similarity threshold values and found the optimal to be 0.65. Table 2 depicts the values for "C" and "σ" of the Gaussian functions for simplified business, architecture, process and organization. Rule reduction (Step-III) was not applicable to current fuzzy rule set because we did not find equalities in the premise parts of the rules. Hence, the number of fuzzy rules remained at 21. Full list of the simplified rule base is given in Appendix II.

Additionally, the "business" and "organization" metrics have four fuzzy sets each whereas the metrics "architecture" and "process" each have five fuzzy sets. We used a questionnaire that was constructed by randomly generating the combinations of business, architecture, process, and organization metrics in order to get expert opinion. We observed that business and organization dimensions do not have a value of level 5 present in any rows of the questionnaire. In order to confirm this observation we generated three new rows in the

Table 2
Antecedent simplified membership functions: Gaussian parameters

Business (B_i)	C	σ	Organization (O_i)	C	σ
B_1	1.002	0.531	O_1	1.000	0.704
B_2	1.998	0.528	O_2	2.002	0.706
B_3	2.999	0.526	O_3	2.995	0.704
B_4	4.000	0.530	O_4	4.001	0.706
Architecture (A_i)	C	σ	Process (P_i)	C	σ
A_1	1.000	0.707	P_1	0.999	0.706
A_2	2.000	0.707	P_2	2.001	0.708
A_3	3.000	0.705	P_3	3.001	0.707
A_4	4.000	0.706	P_4	3.999	0.709
A_5	5.000	0.706	P_5	4.999	0.707

training data set with a level 5 as an input to business and organization each. When we ran the simulation again with additional data, we observed that all four variables of business, architecture, process and organization have five fuzzy sets each.

3.5. Discussion on validation and comparison

When we received the data from experts about their opinion of corresponding maturity level to the set of business, architecture, process and organization values, we divided the received responses into two data sets. We used one data set for the training cycle of ANFIS and other for the validation purpose. In order to validate the accuracy of the proposed FIS, we compared the predicted maturity level with the maturity level given by the experts for a given set of inputs, which have not been used during the training cycle of ANFIS. We used mean error and mean magnitude of relative error as criteria sets for validating the approach. The mean error between expert opinions and the output of the proposed FIS was 0.0035 with a standard deviation of 0.058. Mean magnitude of relative error (MMRE) of 0.0116 was observed. It is important to note there that the current validation of the model is also based on knowledge received from experts because the data about software product family maturity is currently not available.

Software product family process assessment is relatively a new area of research, where so far very little work has been done. Currently, researchers from both academia and industry are attempting to develop a systematic way of measuring the maturity of a software product family process. The fuzzy inference system proposed in this paper aims at establishing the relationship among the four input variables of software product family process. We proposed a possible solution to one of the research problem identified in BAPO-based model of software product family maturity evaluation. The main limitation to the external validity of the FIS proposed in this paper is due to absence of any study directly related to this work for comparison. The fuzzy logic approach itself has been successfully used and validated in number of application domains where experts' opinions have been used to construct a fuzzy inference system. Therefore, we believe that this fact supports the external validity of the proposed fuzzy inference system.

4. Final remarks

The work presented in this paper aimed at achieving two major objectives. The first objective was to define the maturity levels for software product family process evaluation. This categorizes organizations based on their maturity levels. The five-levels of the software product family maturity are based on the ability of an organization to adopt and understand the software product family process. Our second goal was to identify the structure of a FIS, which can capture the relationship among four variables of business, architecture, process and organization. The FIS for the software product family maturity evaluation provides an approach to assign a maturity level to an organization when the values of business, architecture, process, and organization are evident. In summary, this paper established a comprehensive and unified strategy for a process evaluation of the software product family. We have continued working on a comprehensive process maturity model for

software product family maturity evaluation, and the process assessment presented in this paper is a part of that research.

Appendix I

Result of subtractive clustering

Cluster number	Potential	BAPO-combination of input values				Output
		Business	Architecture	Process	Organization	Maturity level
1	1.00	2	2	2	3	2
2	0.83	2	2	2	1	2
3	0.74	2	2	1	1	1
4	0.69	3	3	5	1	3
5	0.68	2	2	2	4	2
6	0.68	2	1	1	3	1
7	0.66	2	3	4	4	3
8	0.63	2	2	1	1	2
9	0.63	2	1	1	3	2
10	0.62	1	1	1	3	1
11	0.59	4	5	3	3	4
12	0.59	1	1	2	3	1
13	0.55	4	4	4	2	4
14	0.53	2	2	2	1	1
15	0.50	1	3	4	4	3

Appendix II

Appendix-II shows the fuzzy rule base. Table 2 (Section 3.4.1) depicts the antecedent membership parameters of B_i, A_i, P_i, and O_i. There are 21 fuzzy rules:

If B is B_2 and A is A_2 and P is P_2 and O is O_3 then output $= 0.3678B + 0.2221A + 0.3539P - 0.0301O + 0.1905$
If B is B_2 and A is A_2 and P is P_2 and O is O_1 then output $= 0.2257B + 0.2229A + 0.3623P + 0.3633O + 0.1893$
If B is B_3 and A is A_3 and P is P_5 and O is O_1 then output $= 0.2990B + 0.3909A + 0.1632P + 0.1555O + 0.0281$
If B is B_2 and A is A_3 and P is P_4 and O is O_4 then output $= 0.0842B + 0.8087A - 0.2027P + 0.3164O - 0.0653$
If B is B_4 and A is A_4 and P is P_4 and O is O_3 then output $= 0.3867B + 0.3874A + 0.1871P - 0.0280O + 0.1070$
If B is B_1 and A is A_1 and P is P_1 and O is O_3 then output $= 0.4367B + 0.2569A + 0.2578P - 0.0702O + 0.2565$
If B is B_2 and A is A_2 and P is P_1 and O is O_1 then output $= 0.1636B + 0.1565A + 0.1638P + 0.1769O + 0.1072$
If B is B_4 and A is A_4 and P is P_3 and O is O_1 then output $= 0.3591B + 0.3563A - 0.0067P - 0.1047O + 0.0897$
If B is B_1 and A is A_2 and P is P_4 and O is O_4 then output $= 0.1766B + 0.1931A + 0.3095P + 0.1358O + 0.0790$
If B is B_3 and A is A_3 and P is P_3 and O is O_4 then output $= 0.1136B + 0.1434A + 0.7725P - 0.0636O - 0.0038$
If B is B_3 and A is A_5 and P is P_2 and O is O_2 then output $= 0.2238B + 0.2735A + 0.1516P + 0.1553O + 0.0617$
If B is B_4 and A is A_1 and P is P_1 and O is O_4 then output $= 0.2449B + 0.0611A + 0.0611P + 0.2449O + 0.0613$
If B is B_4 and A is A_2 and P is P_5 and O is O_2 then output $= 0.2655B + 0.4175A + 0.2114P + 0.0755O + 0.0378$
If B is B_4 and A is A_3 and P is P_2 and O is O_1 then output $= 0.3012B + 0.2763A + 0.1936P + 0.0894O + 0.0750$
If B is B_1 and A is A_5 and P is P_5 and O is O_1 then output $= 0.0748B + 0.2212A + 0.3665P + 0.0749O + 0.0736$
If B is B_3 and A is A_5 and P is P_3 and O is O_4 then output $= 0.2128B + 0.2000A + 0.2956P + 0.2089O + 0.0718$
If B is B_1 and A is A_5 and P is P_4 and O is O_3 then output $= 0.1376B + 0.1953A + 0.2323P + 0.2452O + 0.0560$
If B is B_4 and A is A_5 and P is P_1 and O is O_2 then output $= 0.2437B + 0.2947A + 0.0857P + 0.1325O + 0.0612$
If B is B_1 and A is A_3 and P is P_2 and O is O_4 then output $= 0.0096B + 0.3204A + 0.1640P + 0.3046O + 0.0755$
If B is B_1 and A is A_2 and P is P_5 and O is O_3 then output $= 0.3109B + 0.1530A + 0.2217P + 0.2518O + 0.0942$
If B is B_1 and A is A_5 and P is P_5 and O is O_2 then output $= 0.0584B + 0.2817A + 0.2877P + 0.1115O + 0.0568$

References

[1] F. Ahmed, L.F. Capretz, A framework for software product line process assessment, Journal of Information Technology Theory and Application 7 (1) (2005) 135–157.
[2] R. Babuska, Fuzzy Modeling for Control, Kluwer Academic Publishers, 1999.
[3] G. Buckle, P.C. Clements, J.D. McGregor, D. Muthig, K. Schmid, Calculating ROI for software product lines, IEEE Software 21 (3) (2004) 23–31.
[4] D. Chakraborty, Fuzzy Logic and its Application to Technology and Management, Narosa Pub House (2006).
[5] M.Y. Chen, D.A. Linkens, Rule-base self-generation and simplification for data-driven fuzzy models, Fuzzy Sets and Systems 142 (2) (2004) 243–265.
[6] S. Chiu, Fuzzy model identification based on cluster estimation, Journal of Intelligent and Fuzzy Systems 2 (3) (1994) 267–278.
[7] P.C. Clements, On the importance of product line scope, in: Proceedings of the 4th International Workshop on Software Product Family Engineering, 2001, pp. 69–77.
[8] P.C. Clements, L.G. Jones, L.M. Northrop, J.D. McGregor, Project management in a software product line organization, IEEE Software 22 (5) (2005) 54–62.
[9] P.C. Clements, L.M. Northrop, Software Product Lines Practices and Pattern, Addison-Wesley, 2002.
[10] D. Dubois, H. Prade, Fuzzy Sets and Systems: Theory and Application, Academic, New York, 1980.
[11] ESAPS Project, 1996. Available from: <http://www.esi.es/en/Projects/esaps/overview.html>.
[12] J. Harris, Fuzzy Logic Applications in Engineering Science (Intelligent Systems, Control and Automation: Science and Engineering), Springer, 2005.
[13] A. Ibrahim, Fuzzy Logic for Embedded Systems Applications, Newnes, 2003.
[14] M. Jamei, M. Mahfouf, D.A. Linkens, Elicitation and fine-tunning of fuzzy control rules using symbiotic evolution, Fuzzy Sets and Systems 147 (1) (2004) 57–74.
[15] J.S.R. Jang, ANFIS: Adaptive-network-based fuzzy inference systems, IEEE Transactions on Systems Man and Cybernetics 23 (3) (1993) 665–685.
[16] J.S.R. Jang, C.T. Sun, E. Mizutani, Neuro-fuzzy and Soft Computing: A Computational Approach to Learning and Machine Intelligence, Prentice-Hall, NJ, 1997.
[17] Y. Jin, Fuzzy modeling of high-dimensional systems: complexity reduction and interpretability improvement, IEEE Transactions on Fuzzy Systems 8 (2) (2000) 212–221.
[18] L.G. Jones, A.L. Soule, Software process improvement and product line practice: CMMI and the framework for software product line practice, 2002. Available from: <http://www.sei.cmu.edu/pub/documents/02.reports/pdf/02tn012.pdf>.
[19] E.H. Mamdani, Application of fuzzy logic to approximate reasoning using linguistic systems, Fuzzy Sets and Systems 26 (1977) 1182–1191.
[20] R.V. Ommering, Beyond product families: building a product population, in: Proceedings of the Conference on Software Architectures for Product Families, 2000, pp. 187–198.
[21] W. Pedrycz, Relevancy of fuzzy models, Information Sciences 52 (1990) 285–302.
[22] W. Pedrycz, Neurofuzzy systems, in: P.S. Szczepaniak, P.J.G. Lisboa, J. Kacprzyk (Eds.), Fuzzy Systems in Medicine, Physica Verlag, 2000, pp. 174–203.
[23] M. Setnes, R. Babuška, U. Kaymak, H.R.V. Nauta Lemke, Similarity measures in fuzzy rule base simplification, IEEE Transactions on Systems Man Cybernetics 28 (3) (1998) 376–386.
[24] I.S. Shaw, Fuzzy Control of Industrial Systems: Theory and Applications, Springer, 1998.
[25] T. Takagi, M. Sugeno, Fuzzy identification of systems and its application to modeling and control, IEEE Transactions on Systems Man and Cybernetics 15 (1) (1985) 116–132.
[26] H.N. Teodorescu, A. Kandel, L.C. Jain, Fuzzy and neuro-fuzzy systems in medicine, CRC (1998).
[27] F. van der Linden, J. Bosch, E. Kamsties, K. Känsälä, H. Obbink, Software product family evaluation, in: Proceedings of the 3rd International Conference on Software Product Lines, 2004, pp. 110–129.
[28] F. van der Linden, Software product families in Europe: the Esaps & Café projects, IEEE Software 19 (4) (2002) 41–49.
[29] T. Wappler, Remember the Basics: key success factors for launching and institutionalizing a software product line, in: Proceedings of the 1st Software Product Line Conference, 2000, pp. 73–84.
[30] R.R. Yager, D.P. Filev, Approximate clustering via the mountain method, IEEE Transactions on Systems Man and Cybernetics 24 (8) (1994) 1279–1284.
[31] L.A. Zadeh, Outline of a new approach to the analysis of complex systems and decision processes, IEEE Transactions on Systems Man and Cybernetics 1 (1973) 28–44.
[32] L.A. Zadeh, The concept of a linguistic variable and its applications to approximate reasoning-I, Information Sciences 8 (1975) 199–249.
[33] L.A. Zadeh, The concept of a linguistic variable and its applications to approximate reasoning-II, Information Sciences 8 (1975) 301–357.
[34] L.A. Zadeh, Towards a generalized theory of uncertainty (GTU) – an outline, Information Sciences 172 (2005) 1–40.
[35] R. Zwick, E. Carlstein, D.V. Budesu, Measure of similarity among fuzzy concepts: a comparative analysis, International Journal of Approximate Reasoning 1 (1987) 221–242.