Yugoslav Journal of Operations Research
6 (1996), Number 2, 193-203

INCORPORATING FUZZY LOGIC INTO REUSABLE OBJECT MANAGEMENT SYSTEM

Saša BOŠNJAK, Zita BOŠNJAK

University of Novi Sad, Faculty of Economics, Yugoslavia

Abstract: Reusability transfers software implementation into the selection problem: we search across the available reusable components, described by standard attributes that capture their functional characteristic, to depict the one that is most appropriate or fulfills our software development needs to the greatest extent. One of the problems in the process of selecting from the objects' repository lies in the fact that reusable components usually do not match our requirements perfectly. The central issue, therefore is, how to measure the degree of adequacy of the chosen reusable components ? In our article we propose one approach that incorporates the fuzzy sets theory and fuzzy logic into reusable objects (R-objects) management systems. The main idea is to establish one "template" R-object, based on expressed software implementation demands, and to compute its belongings to different R-objects' sets formed according to adopted R-objects classification. The measured values are in direct proportion to the correctness of selected elements from each discrimination set. Incorporating this method into the search algorithm enables quicker access to those R-objects that are grouped together in a particular set as being very similar, and at the same time are most adequate for embedding into software product under development. Both the creation of "template" R-object, based on object-oriented design, and the division of the R-object repository into calssification sets are described in this article.

Keywords: Reusable objects, fuzzy logic, membership functions, R-object retrieval,

1. INTRODUCTION

Requirements concerning software production, which are nowadays respected, are mainly related to the enhancement of software quality and also to the time-saving production of software. The use of software development methodology that deals with the reuse approach can make a significant contribution to higher quality and productivity. The reuse approach implies the reuse of existing solutions during the development of new software systems. In general, existing solutions are called *reusable objects*.

We may say that the reusability of products, processes and other knowledge is one of the key aspect to achieving a rise in productivity and quality in software production. Productivity should be increased especially in the implementation phase using software components from the reusable repository rather than creating entire software systems from the beginning. Because the promise of software reusability is great, there are several directions to research in this field. Some of the topics include a systematic approach to the reuse process, organization and implementation of reusable components and software libraries, the principle of composition and generation technologies, language-based systems, application generators and transformation systems. For the purposes of our research described in this paper, we restricted ourselves to the organization of reusable software libraries and development of the reusable object management tool. Our basic assumption is that a small percentage of software is unique or innovative. The remaining software can be reused from existing software products. Some reuse models enable the building of new software from existing ones by applying automated tools. We adopted this model as a basic viewpoint of our research, but included an expert system that facilitates the process of reusable components' retrieval and selection, which is, in our opinion, the best solution for this kind of problem. The chosen reuse component, the output from an expert system (ES), represents the input to another module of an automated tool that does not include artificial intelligence and can be considered conventional. We call this module the reusable object retrieval tool. This tool tailors the chosen reusable component and builds it into a new software solution.

One of the crucial problems in reuse-based methodologies is the selection and creation of reuse candidates and also how to incorporate them into the new environment. In this paper, a knowledge-based model for the selection of the reuse candidates and an automated tool for tailoring and integration are described. In the context of these solutions, we included fuzzy logic in some aspects of the reusable process. The reusable approach to the development and maintenance of software systems includes several very important attributes and criteria that do not have precise values. This is the main reason to use fuzzy concepts in our considerations.

2. FRAMEWORK FOR REUSABILITY

The reuse of existing experience and knowledge is a akey aspect to progress in any discipline, especially in software engineering. From our viewpoint reuse ia a process of employing knowledge that has been compiled through previous experience. We consider three different types of reuse:

- reuse of knowledge that exists solely in people (informal knowledge)
- reuse of plans or procedures; e.g. how to perform certain activities or how to structure and document certain products (schematized knowledge)
- reuse of tools and products (productized knowledge)

Reuse-oriented software development assumes that, given a project's specific requirements X' for an object X, we consider reusing an already existing object X_k instead of creating X from the beginning. Reuse involves identifying a set of reuse candidates $X_1,...,X_n$ from an experience base, evaluating their potential for satisfying X', selecting the best-suited candidate X_k and, if required, modifying the selected candidate

X_k into X.

There has been a long-standing desire in computer science to find a way to collect and use libraries of standard software components. Unfortunately, there has been only limited success in actually doing this. The lack of success stems not from any resistance to the idea, or from any lack of trying, but rather from the difficulty of choosing an appropriate formalism to represent components.

The biggest problem one faces when developing a formalization for components is that there are many different kinds of things that it would be beneficial to express as components. Essentially, any kind of knowledge shared between distinct programs is a candidate to becoming a reusable component. Many properties are required of a formalization in order for it to be an effective representation for reusable components:

- **expressiveness** - the formalism must be capable of expressing as many different kinds of components as possible;
- **convenient combinability** - the methods of combining components must be easy to implement and the properties of combining should be evident from the properties of the parts;
- **semantic soundness** - the formalism must be based on a mathematical foundation that allows correctness conditions to be stated for the library of components;
- **machine manipulability** - it must be possible to manipulate the formalism effectively using computer tools;
- **programming language independence** - the formalism should not be dependent on the syntax of any particular programming language.

It is not sufficient for a formalism to be merely capable of representing a given component, but a formalization must be able to represent the component in a straightforward way. A firm semantic basis is needed for a good formalization so that it is possible to be certain what a given component represents and certain that the combination process preserves the key properties of the components.

In order to be able to deal effectively with a large reusable component library, we developed a tool to support the automatic selection and modification of components.

3. THE REUSABLE OBJECT MAMAGEMENT SYSTEM MODEL

The reusable object mamagement system described in this article is made up of the following modules:

- expert system (with user-system interface, a knowledge base that consists of a frame base and a rule base, forward chaining inference mechanism)
- reusable object library management tool
- reusable object repository

Interaction and relationships between these three modules are shown on Fig. 1.

The task of the reusable object management system is the selection, tailoring and integration of reusable components into new software products. The selection process is performed by an expert system that based on user requirements and needed reusable object specifications searches the knowledge base for the most appropriate reusable candidates. The result of the expert system run is the recommendation of a reusable object(s) in the form of keyword(s) that is passed over to the automated reusable object retrieval tool. This tool then searches the reusable object repository and finds the appropriate reusable object and browses the related information. Furthermore, this tool enables the access to relevant data files that contain additional information about selected reusable components. This information allows for correct further implementation of the reused object into the new environment. The foundation of the reusable object repository is a special kind of database system which stores the attributes of every reusable software component. All information in the repository (library) is maintained by the library management system. In general, several types of software components can be entered into the repository, including functions, procedures, packages and programs. These components may be written in any language. We emphasize functions written in C language as reusable components.

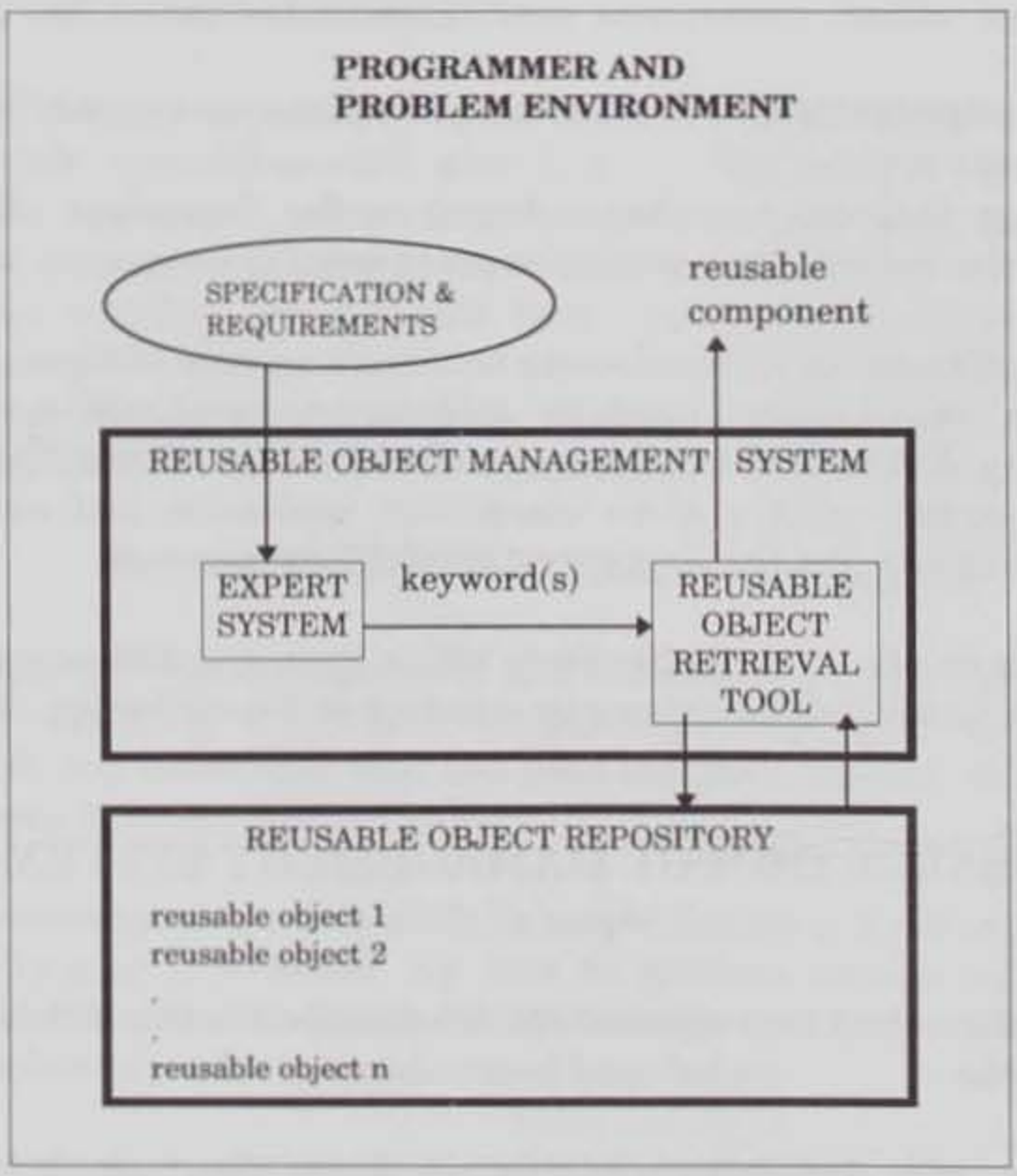

Figure 1. System architecture of the automated tool

The reusable object retrieval tool is based on the hash algorithm as a useful technique to manage the insertion, deletion and access of reusable elements in the repository. It is an efficient data maintenance methodology implemented by building a table of pointers (called the hash table) that points into the repository. A mathematical function (called the hashing function) translates each element to be stored in the database into a number. This number is then used as an index into the hash table which contains at this address a pointer to the element's location in the repository.

All the information about reusable objects is stored in the dictionary table, including:

- the name of the person who created the reusable object;
- the name of the person who modified the reusable object definition lastly;
- where the source code of the reusable object is stored;
- where is the reusable object declaration (header file) stored;
- an example of usage;
- ANSI prototype (with formal parameters);
- the description of each of the parameters;
- type of the returned value;
- maximum five keywords of related reusable objects.

The definition table contains variable-length reusable object definitions. Each record in the dictionary table points to a location in the definition table corresponding to the starting location of the definition.

4. KNOWLEDGE-BASED APPROACH AND FUZZY LOGIC IN REUSABILITY

Effective reuse of software designs requires a collection of good design components and knowledge about how to locate and combine appropriate components into a particular software design. Indeed, experienced knowledge engineers generally benefit from this kind of expertise. Other studies indicate large differences in the abilities of different software developers and a number of techniques have been proposed to take the best advantage of the more experienced software engineers. The application of expert system technology to software engineering is one mechanism through which the reusability of software components might be achieved. By encoding and abstracting the expertise of experienced software engineers into knowledge bases, their expertise can be made readily available to non-expert system developers.

Using expert systems technology, we can recommend several reusable objects as a suitable solution for users' demands. Recommended R-objects constitute a set of all adequate solutions. This set is a subset of our repository of several hundred R-objects, and by itself may have quite a lot of acceptable R-objects as elements. Consequently, further evaluation and selection among already chosen R-objects is necessary. It is based on three assumptions:

- different users of our system have different experience;

- several R-objects have similar attributes in one subset of the repository;
- the repository of R-objects is very large.

According to the above assumptions, we have selected those characteristic of R-objects that are important for deciding which is the most adequate R-object. Being "adequate" is obviously a fuzzy concept and for its description we have chosen three attributes: R-object's size, R-object's complexity and quality of the R-object's documentation. These characteristics establish the R-object template describing the ideal solution: the smallest and simplest one, with the best documentation. Unfortunately, in most cases, reusers have their own criteria that are based on their experience of the above-mentioned three characteristics, and the consequence of this fact is that reusers' experience level can be viewed as a fuzzy modifier of the chosen attributes. The basic problem is that "adequate" as a main characteristic of the chosen R-object is determined based on criteria that are without precise values. We solved this problem using the fuzzy set theory as a general way to evaluate R-objects' adequacy.

The first characteristic, the size of the R-object, is measured with numbers of lines of code, as a metric. Considering that all of our R-objects are implemented in C language, and using the general principles of the software metrics described in [7], we received three different membership distributions of the fuzzy set "small code size", shown in Table 1.

The fuzziness of the concept "small" determines an S-shaped curve. In our case, we consider reuse experience as a fuzzy modifier that determines a transition region of fuzzy functions, as illustrated on Fig. 2.

R-object complexity as a second characteristic depends on the number of modules in the concrete R-object and the number of links used to get to the final solution in the software development process. R-object documentation quality is of great importance to novice reusers, because good documentation decreases the reuse effort of novice programmer.

In the fuzzy theory, when more than one fuzzy set is in use (as in our case), the degree of membership is obtained by one of three possible ways:

- taking the minimum/maximum (and/or operator);
- combination of evidence (comb operator);
- pooling by taking the average (poll operator).

Table 1: Membership distributions of the fuzzy set "small code size"

LEVEL OF REUSER EXPERIENCE - **EXPERT**

Numbers of lines	150	135	120	105	90	75	60	45	30
degree of membership	0	0	0.1	0.2	0.4	0.5	0.7	0.9	1.0

LEVEL OF REUSER EXPERIENCE - **AVERAGE**

Numbers of lines	120	105	90	75	60	45	30	15
degree of membership	0	0	0.1	0.2	0.4	0.6	0.9	1.0

LEVEL OF REUSER EXPERIENCE - **NOVICE**

Numbers of lines	90	75	60	45	30	15
degree of membership	0	0	0.1	0.2	0.6	1.0

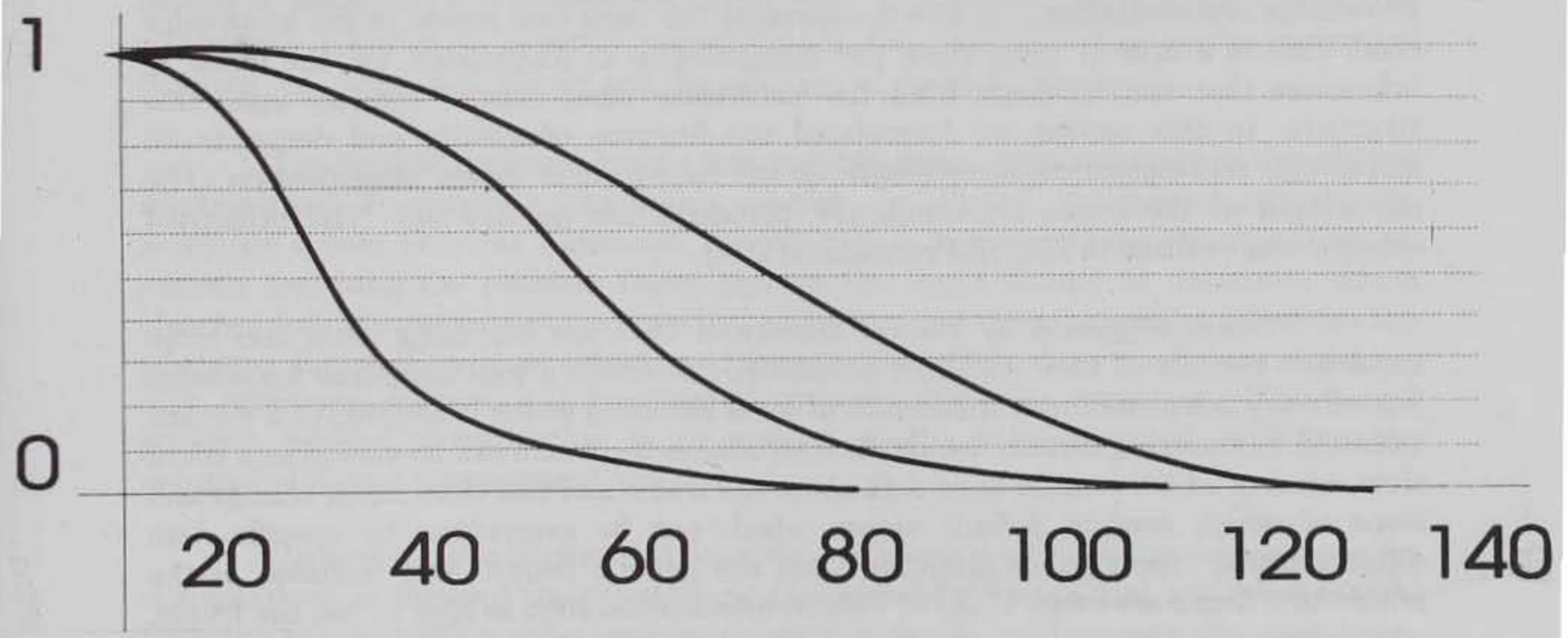

Figure 2: Membership function of the fuzzy concept "small number of code lines"

For the fuzzy approach in reusability we used the "and " operator. If the operator is "and ", the operation is to take the minimum degree of membership:

$$f_n(a_1, a_2, \ldots, a_n) = \min(a_1, a_2, \ldots, a_n) \quad \text{for all } n.$$

The degrees of membership for three different levels of reuser experience are summarized in the following table:

Table 2: Different values of membership function

experience / attribute	EXPERT	AVERAGE	NOVICE
SIZE	0.9	0.6	0.2
COMPLEXITY	0.7	0.6	0.2
DOCUMENTATION	0.8	0.5	0.2

As can be seen, the ranking of the same R-object gives different results. Each experience category has its own threshold, and as a consequence, the same R-object may or may not be accepted as appropriate.

5. PARADIGM FOR REPRESENTING REUSABLE OBJECTS

The major question pertaining to knowledge-based systems is that of knowledge representation, i.e. how to represent the facts that reside in the knowledge base. This is a critical issue since the manipulation of knowledge, i.e. the kinds of inferences that can be made from the knowledge base, depend directly upon this structure. In this section we formulated the features of quality and demands on knowledge representation components in ES for reusable object classification. The advantages of the frame representation paradigm are pointed out. The presented solution was realized in EXSYS Professional shell.

Frames suggested by Marvin Minsky in 1975 are becoming more and more important because of their excellent possibilities to create a well organized knowledge base. Frame is a generalized framework of some situation which is adapted to a current situation by changing certain details. As a structure, the frame can be viewed as a set of slots, some of which contain fixed data about the frame and therefore never change and some of which contain default values which can be overridden by specific data differentiating the current situation from the generic frame. Also included in the notion of a frame are other kinds of related information such as how to use the frame, what action can take place in the frame and what effects these actions can have. The main function of a frame system is to provide a retrieval capability for matching frames with "reality" or some partial description thereof. Frames are a good way of representing descriptive information. The organization of frames allows for efficient search because there is immediate access to relevant information. Frames also provide a natural method for representing hierarchies of information and allowing the inheritance of values through ISA and AKO slots.

We came to the conclusion that it would be useful to create a hybrid knowledge representation scheme combining frames with production rules. In such a scheme slots represent the knowledge of facts respectively pointing subframes whereas the expert knowledge for solving the selection task is integrated into the frames by production rules.

The frame system has some appealing characteristic with regard to the reusable object management problem. First it supplies a mechanism for linking user problem descriptions with actual model and data instances assuming a suitable frame reusable object representation. Secondly, frames may be appropriate as a representation vehicle for the reusability environment: each frame consists of a set of slots characteristically representing features of the object, pointing to subframes or containing comments.

5.1. Expert system inference process

The reusable object management system discussed in the previous section goes through two different stages. In the first stage, the diagnostic phase, the current programming environment state is determined and a corresponding reusable object is selected. In the second stage, the control phase, the reusable object library management tool is activated that incorporates the selected object into a programming environment.

The central function of the expert system is to construct an abstraction model, "template", that corresponds to the user specification and will support the retrieval of a reusable object from the reusable object repository. The knowledge about reusable objects is organized into abstraction hierarchies. This permits the sharing of common component features in abstract parent nodes and facilitates the user's navigation through the component library and the selection of desired components.

The vehicle proposed herein for reusable object representation is the **object abstraction**. It resembles the frame concept. In general, the system operates by accepting a user problem statement, constructing a frame of the problem statement, pattern matching the problem frame against the frame library to determine which objects / data are candidates for invoking/retrieving, and then the system calculates the adequacy degree of each candidate as a fuzzy variable. Only those candidates that have high adequacy degrees will be invoked/retrieved form the repository. The actual problem solving is generating a job stream which, when executed, will run the automated library management tool.

Concerning the diagnostic phase, the inference mechanism in principle can be characterized as forward classification on the basis of a breath first top down search. Utilizing the derivation rules we tried to determine the subframe on the next lower level that fulfills the implemented hypothesis with the highest preference. The values of preference called certainty factors (CF) values are calculated using the functions V and Y:

$$V_{F,i^n}(m_1,\ldots,m_k) \rightarrow (z_1,\ldots,z_k)$$

where: F - name of the frame
i - index of the function, $i=1,\ldots,I$
n - number of features in the given function, $N \ni n$
mj - features of the type sj, characterizing the programming state and stored in a dynamic reusable object repository
z - CF values, from -100 to 100
k - number of subframes directly linked with F

$$Y_{F,i} = Y_{F,i-1} + V_{F,i^n} \quad i = 1,2,\ldots,I$$
$$Y_{F,i} = \{a_1,\ldots,a_k\}$$
$$Y_{F,0} = \{0,\ldots,0\}$$

Each production rule combines a function V representing a certain reusable object with a preference vector than can be used for testing the membership of a subframe. In this way every subframe on a given level is assigned a preference value from each production rule. Performing the recursive Y all CF values are added step by step for each subframe. Those R-objects that are represented by frames with CF greater then the predefined threshold are processed in order to determine their adequacy for final reuse. If no subframe can be verified (because all CF values are too low) the inference process is interrupted and a message is given to the decision maker. The production rules used in inferencing the possible candidates for reusability are of the following form:

IF $F.slot = value$
THEN CF(F) = CF + increment

where instead of *slot* we have the exact slot name and instead of *value* there are concrete values (for example, F.AKO=password routines or F.FUNCTION=read, etc.). Before calling the procedure that activates the reusable component retrieval tool with selected keywords as parameters, some rules that determine the adequacy of each R-object for a particular user are invoked. Because of that, each user is asked about his experience level (novice, regular user, expert). The rules triggered in this process have the following form:

IF EXPERIENCE LEVEL = novice
AND SIZE(F) =small
AND COMPLEXITY(F) = simple
AND DOCUM_QUALITY(F) = excellent
THEN ADEQUACY(F) = high

IF CF(F) > THRESHOLD
AND ADEQUACY(F) = high
THEN APPLY p(F.NAME)

After terminating the diagnosis phase the inference procedure switches over to the control phase.

6. CONCLUSION

The reusable object management system described in this article uses a unique approach to reusability, based on the integration of benefits from the expert system's classification of reusable objects that include fuzzy concepts with automated retrieval and integration supported by the reusable object repository management tool. This system is an integral part of the software development life cycle and we expect a significant improvement in the software development process. The result of applying

our system is a more automated, reliable and productive method for developing software products, especially due to the incorporation of fuzzy logic in reusable candidate adequacy assessment. Reuse components may be accessed during the coding phase, and a result of this is a reduction in the time and cost associated with software development. Further research efforts in this area cover the inclusion of R-objects written in different programming languages and of different types (not only functions) into the repository, the evaluation of overall system performance and investigation of the usefulness and possibilities for fuzzification of the candidate R-object identification phase.

REFERENCES

[1] Basili, V.R., and Rombach, H.D., "Support for comprehensive reuse", *Software Engineering Journal*, September (1991) 303-317.

[2] Bošnjak, S., "Software development methodology based on reusable objects", PhD Dissertation, Dept. of Computer Science, Mathematics and Statistics, Faculty of Economics, University of Novi Sad, 1994.

[3] Caldiera, G., and Basili, V.R., "Identifying and qualifying reusable software components", *Computer*, 24 (1991) 61-70.

[4] Ege, R.K., *Object Oriented Programming with C++* , AP, 1994.

[5] Leung, K.S., Wong, M.H., and Lam, W., "A fuzzy expert database system", *D&K Engineering*, 4 (1989) 287-304.

[6] Kosko, B., *Neural Networks and Fuzzy Systems*, Prentice Hall, 1994.

[7] Musa, J.D. , Iannino, A., and Okumoto, K., *Software Reliability*, McGraw-Hill, 1992.