

The case-based decision support system in the field of IT-consulting

T V Avdeenko, E S Makarova

Novosibirsk State Technical University, 20, Karl Marx Ave., Novosibirsk, 630073, Russia

E-mail: avdeenko@corp.nstu.ru

Abstract. In the present paper, we propose an approach to facilitation of decision-making in the field of IT-consulting. The approach is based on knowledge representation in the form of cases containing past history about decision making. We also proposed an ontology to store the cases as instances of classes. Classification of cases in the ontology is executed by means of fuzzy inference. The method of classification is based on the original algorithm of transformation of cases (precedents) sample to the set of linguistic rules allowing one to make relevant decisions. Research of the method has shown good accuracy of the decisions classification according to test data.

1. Introduction

The emergence of Decision Support Systems (DSS) in the mid-1960s was associated with a model-driven approach [1]. The first DSS were limited by the types of these models, mostly deterministic, which hardly ever used operational information about the circumstances under which one needs to make decisions. The situation has changed dramatically since the 1990s, when the DSS began to integrate first with operational databases, and then with the specialized data warehouses built on OLAP technology.

In the modern era of the Internet, when the environment in which the decisions are made, is very dynamic, the concept of DSS is undergoing dramatic changes. We have so much information (even there a stable phrase "big data" appeared) that the data on their own is no longer of great value. What is really valuable is the knowledge that can be extracted from the data to solve the urgent problem in a particular application domain. The problem of preparing the knowledge of such quality and organizing it into a specially-designed knowledge base, becomes now, in the era of big data, more relevant than ever.

Methods of knowledge representation and organization are traditionally developed within the discipline of artificial intelligence [2]. In the development of this scientific area, three alternative methods of knowledge representation have been formed: model-based, rule-based and case-based approaches. However, neither approach in itself is not able to provide a sustainable basis for the modern dynamic intelligence system based on knowledge, capable of self-organization and self-development based on extracting information from the environment. And, though there were attempts to create hybrid knowledge models, based on different variants of integration, the problem of constructing an integrated model of knowledge representation, allowing the transformation of knowledge from one form to another and on the basis of machine learning, is still far from final decision, in spite of its relevance.



We believe that such a self-developing integrated knowledge representation system can be constructed on the basis of a combination of basic approaches to knowledge representation. In [3], we have obtained encouraging results of improving the efficiency of cases classification by integrating case-based and a rule-based models with the use of fuzzy logic. In this paper, we propose a hybrid model of knowledge representation in the field of IT consulting.

2. The problems of knowledge management in the software maintenance

One of the stages of the software engineering process is a software support subprocess. Therefore, the IT-departments (or IT-companies) consisting of analysts, programmers, consultants began to appear in the organizations. The important part of their work is to organize the support of IT-users. A typical IT-company is engaged in introduction, development and maintenance of information systems and other software.

Consult-analysts plays the main role in providing the high-quality IT-support. Tasks that the consultant receives from the users can be divided into the following groups:

1. Simple questions - questions that consultant can respond immediately if there is a certain experience with the system (about a year). The time spent on the task is from 15 to 25 minutes.
2. Average complexity tasks - tasks that require analyzing the situation, recovering the action sequences. The time spent on the average complexity task is from 25 to 60 minutes.
3. Difficult tasks - tasks that require detailed study, careful analysis of the situation. It is not excluded that the task will require reworking the system by the programmers. The time spent on the difficult tasks is from 60 minutes to 120 minutes.
4. Specifications (simple) to develop the system. Due to legislative changes there is a need to refine certain modules from time to time. In this case, the time required depends on the complexity of the system improvement. This may be both 24 hours and a few days, or even a month.

During the consulting process a consultant needs to identify the users question, then to analyze the accumulated knowledge base proceeding from the experience, and, finally, to offer a certain variant of the solution to the user.

First, when the knowledge of the base is not enough, the consultants used to record the cases on paper. Then, after accumulating sufficiently large volumes of cases, their use has been a difficult task because of search difficulties. So the next step was to use the software Microsoft Word products to store the cases that sufficiently decreased the time required to find the relevant case. Table 1 shows the time required from the consultants to process various types of questions.

Table 1. The time spent on the solution to the problem

№	Category	Experienced consultant	Inexperienced consultant		
			Self-learning	Using cases written on paper without search function	Using cases with search function (MSWord)
1	Simple questions	7-10 min	15-25 min	12-20 min	10-17 min
2	Average complexity tasks	10-25 min	25-60 min	20-50 min	17-45 min
3	Difficult tasks	25-45 min	60-120 min	50-110 min	45-100 min
4	Simple specifications	>45 min	>120 min	>110 min	>100 min

With the increase of branch offices, which are supported by the IT-department, the load on the consultants significantly increases. The time to solve the tasks is greatly reduced. In this connection, we can highlight the problem of resources lack for the processing of all available information. Another problem is adaptation of new consultants and analysts. In any IT-company,

training new staff is a long process (from 5 months to one year). The company hires specialists taking into account that they do not have much experience, and it will take a lot of time and money before the effective work of the new employee begins.

To solve the above-mentioned problems, it seems appropriate to create a knowledge management system based on the use of artificial intelligence methods. Since the tasks faced by the consultants are largely repeated, it is reasonable to create a knowledge base of cases (problems with appropriate solutions) containing the past consulting experience. Using the knowledge base for solving the next task, will consist in extracting the relevant case and adapting it to the current situation.

Case-based reasoning (CBR) solves new problems by adapting previously successful solutions to similar problems. The work of Shank [4] in 1977 is widely known to be the origin of CBR. In this work the author has proposed generalizing the knowledge about previous situations and store it in the form of scripts that can be used to make conclusions in the similar situations. Later, Shank continued to explore the role that memory about previous situations (cases), represented as a knowledge container, plays both in the process of decision making and in the process of learning [5].

There are different ways of formulating the cases [6], from simple (linear) to complex hierarchical ones. A case could generally include a description of the problem and a solution to the problem. If the cases are used to solve practical problems, the additional component may be the result of case application (positive or negative). It should be emphasized that already in work [7], one can find references that natural domain concepts often cannot be classified according to a simple set of properties (features), and could be described by a more complex structure. This work was subsequently referred to as the philosophical basis of the case-based approach.

The case-based reasoning approach allowed one to overcome a number of restrictions inherent to the systems based on rules [8]. First, CBR does not require the explicit model of the knowledge domain, so the extraction of knowledge is transformed into a simple task of collecting stories (cases). Second, implementation of CBR-systems is reduced to the identification of essential features, describing the case, which is much an easier task than building an explicit knowledge model of the application domain. Third, it is possible to use database technology for storing large volumes of cases. Forth, CBR-systems can be self-learning from the case base, thus, it is possible to obtain new cases.

At the same time, there are some shortcomings of CBR. First, it should be noted that the description of the cases does not usually take into account the deeper knowledge of the application domain. Second, the large case base results in the reduced system performance. Moreover, it is difficult to determine good criteria for indexing and comparison of cases. Very often the search dictionaries and algorithms for determining similarity are needed to debug manually. It can neutralize the benefits of the case approach. In the next sections, we will consider how to organize cases in the case base into classes of the ontology and propose an algorithm of cases classification based on fuzzy linguistic rules.

3. The method of case-based support for the software maintenance with fuzzy linguistic rules for cases classification

Let us consider an IT-support system based on knowledge. An important component of such system is a capability of accumulating knowledge and using it subsequently in decision-making.

At the initial stages of creating a knowledge base, when there is no insight into the application domain, it is reasonable to use a case-based representation in which knowledge is formulated in the form of precedents (cases) of decision-making. The simplest form of case representation is linear in which it is necessary to determine the final set of features that uniquely determine the situation and a specific decision made in this situation. Let us consider a simple linear formulation of the case to be used for decision making as $(n + 1)$ - dimensional tuple

$$CASE = (x_1, x_2, x_3, \dots, x_n, d), \quad (1)$$

where $x_1 \in X_1, x_2 \in X_2, x_3 \in X_3, \dots, x_n \in X_n$ are the values of features identifying the situation, $X_i, i = \overline{1, n}$ are some sets of feature definition, $d \in D$ is a solution to the problem from the discrete set of solutions $D = \{d_1, d_2, \dots, d_K\}$ (classes of solutions in the classification problem).

Cases of form (1) can be accumulated in the case base for the future use in decision making. Thus, already at the earliest stages of the development of DSS, it is possible to extract similar cases, adapt and reuse them to solve the current problems of decision-making.

Cases of form (1) can be considered as a data table. And we can apply, for example, a decision tree method [9] to this sample of cases, in which the intermediate nodes of the tree are the features of cases $x_1, x_2, x_3, \dots, x_n$, and terminal nodes are possible solutions d_1, d_2, \dots, d_K of case (1), which in this aspect is considered as a classifying attribute. Another method of forming a set of rules from cases uses the transformation of the cases sample into the system of fuzzy rules, which is considered in the present paper.

Here, we consider a method of cases classification based on obtaining fuzzy rules from cases. The idea of the method is as follows. Let us consider an arbitrary case of form (1). We shall define linguistic variables V_i corresponding to features $x_i, i = \overline{1, n}$. Let each linguistic variable V_i has J term-values $A_i^{(j)}$ that are fuzzy sets determined by membership functions $\mu_{A_i^{(j)}}$, on universal set $X_i, j = \overline{1, J}$. Then rule *Rule*, corresponding to case (1), could be formulated as follows:

$$\text{Rule: IF } V_1 = F(x_1) \& V_2 = F(x_2) \& \dots \& V_n = F(x_n) \text{ THEN } R = d, \quad (2)$$

where R is a variable which nominal values are the names of decision classes in the decision making problem, $F(x_i)$ is a fuzzy term-value which accepts linguistic variable V_i as a result of fuzzification of feature value x_i :

$$F(x_i) = A_i^{(j^*)}, \text{ where } j^* = \arg \max_{j=\overline{1, J}} \mu_{A_i^{(j)}}(x_i). \quad (3)$$

Thus, the knowledge base consisting of a set of rules from the initial case base is formed in accordance with the following algorithm:

An algorithm of obtaining fuzzy rules from the case base for cases classification

Step 1. Define the number and the type of membership functions and universal sets X_i for each linguistic variable V_i corresponding to feature x_i .

Step 2. Determine fuzzy rule *Rule*, of form (2) for each case of form (1).

Step 3. Assign truth degree $TD(\text{Rule})$ to each rule. The simplest way to do this is to compute minimum of all $\mu_{A_i^{(j^*)}}(x_i)$ for every x_i and fuzzy sets $A_i^{(j^*)}$ defined by formula (3) as a result of fuzzification for each rule *Rule* of form (2). The result is obtained in the following way:

$$TD(\text{Rule}) = \min_{i=\overline{1, n}} \mu_{A_i^{(j^*)}}(x_i).$$

Step 4. Resolve conflicts between the rules. After stage 2, we have the set of rules uniquely corresponding to the set of cases. But this set can contain subsets of the rules with the same premises. These rules can have the same or different conclusions (solutions $d \in D$). If for all rules in such subset we have the same conclusion, we simply remove all duplicate rules from the set except one. For the rules with conflicting conclusions, we propose two different strategies for the classification problem. In each case, we leave only one rule in the subset with conflicted conclusions in the rule base. Let us have rules $\text{Rule}_1, \text{Rule}_2, \dots, \text{Rule}_m$ with the same premise. We will divide a set of indexes

$I = \{1, 2, \dots, m\}$ into K subsets I_1, I_2, \dots, I_K corresponding to the classes of solutions d_1, d_2, \dots, d_K in the conclusion of rules (2), $\dim I_k = m_k, \sum_{k=1}^K m_k = m$.

The first strategy of resolving conflicts is to obtain the optimal rule:

$$l^* = \arg \max_{l \in I} TD(Rule_l),$$

then to choose solution d corresponding to the conclusion part of this optimal rule.

The second strategy of resolving conflicts is more complex. Optimal solution d_{k^*} that is assigned to the conflicting set of rules could be obtained as follows:

$$k^* = \arg \max_k \left(\sum_{i \in I_k} TD(Rule_i) \right).$$

The experiments on test data have shown high accuracy of cases classification with the use of the above-mentioned algorithm. The second strategy of resolving conflicts between the rules showed better results.

4. The ontology for case base organization

The variety of software modules, for which we create a support system, as well as possible problems with the software, necessitated indexing in the knowledge base. An effective way to organize data is an ontological approach [10]. To organize a case base for the IT-consulting sphere, the ontology that includes a hierarchy of classes in the application domain has been created. In fig. 1, we presented a part of Protege ontology with classes "Staff" and "Accounting". In fig. 2, we see the ontological graph for the same classes.

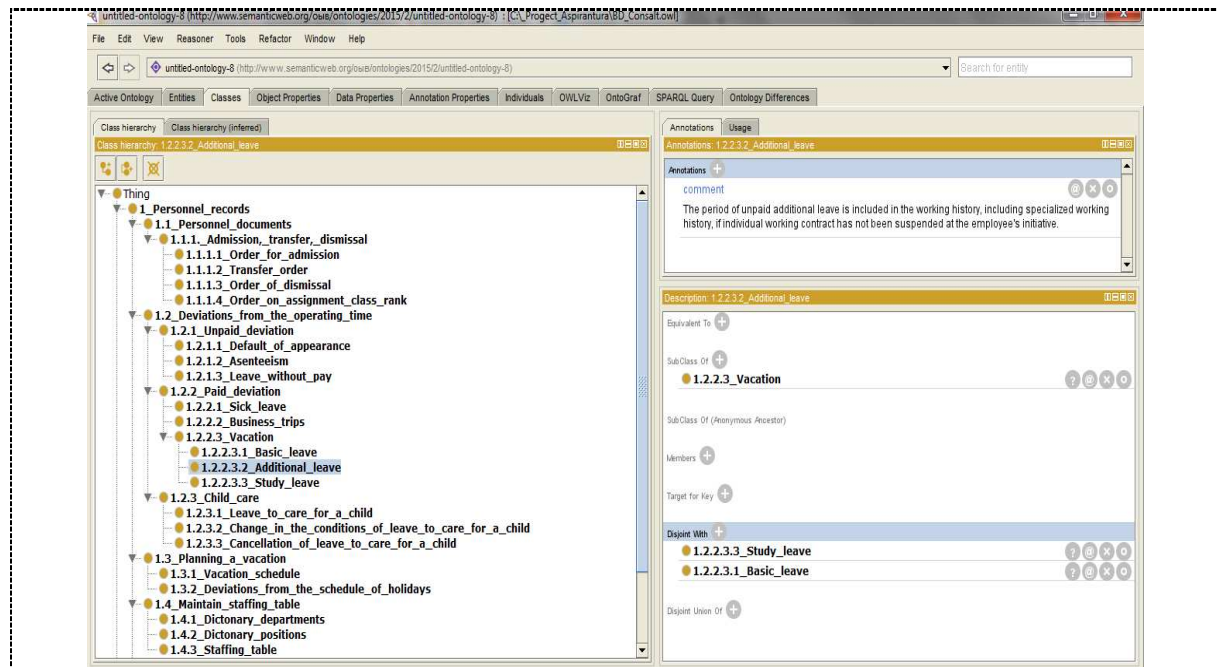


Figure 1. The part of the ontology with classes "Staff" and "Accounting" in Protege

Cases are the instances of the ontology classes. To retrieve a relevant case at the first stage, we determine its class in the ontology from the keywords as the features, provided that there is the degree of conformity of each keyword with the current problem. At the second stage, we browse all the instances of the ontology class and choose the one, similar to the current problem.

