Information Technology for Decision-Making Based on Integration of Case Base and the Domain Ontology

Tatiana V. Avdeenko and Ekaterina A. Makarova

Novosibirsk State Technical University, Novosibirsk, Russia,
`avdeenko@corp.nstu.ru, e.s.makarova@corp.nstu.ru`

Abstract. The paper considers an approach to creation of new information technology to support decision-making in the field of IT consulting. The proposed approach is based on the storage and subsequent use of decision-making cases from the case base integrated with the domain ontology. This integration is realized on the basis of establishing weighted associative links of cases with ontology concepts, and the subsequent identifying classes of semantically close precedents based on the proposed algorithm for calculating the matrix of closeness of cases to the terminal ontology concepts. Thus, the proposed approach makes it possible to increase the relevance of retrieved cases to the current decision-making problem.

Keywords: Information technology, case-based reasoning, ontology, IT consulting

1 Introduction

Currently, information technology is a principal instrument of any company that plays a significant role in its functioning. Due to frequent changes in legislation and rapid emergence of new interfaces and services, employees of organizations frequently have problems with application of the software products. To help in solving these problems, consultants-analysts are traditionally used being the workers of IT-departments engaged specifically in advising on the problems appearing while using the software. Consulting activity of an analyst includes the identification of an issue from the user, a careful analysis and search of the solution, and subsequent formulation of possible answers to the question. Requests that come from the users to the analyst can be divided into the following groups by their difficulty: simple requests that a consultant can respond immediately; requests of medium difficulty that require analysis of the situation; complex requests that require detailed study of the situation; and simple specifications to refine the system.

The average time spent by a consultant for a single request depends on the experience of his work and on the complexity of the problem. At the same time, it was noted that if an unexperienced consultant uses his or other people's knowledge about previous cases and solutions, he spends in average less time to solve the problem. This could be explained by the fact that different users often meet the same, or very similar, problems in their work.

Thus, one can conclude that the use (of even very simple means of recording and retrieval of precedents that took place in the past), brings the overall performance of an unexperienced consultant to the effectiveness of an experienced one. Therefore, in the field of IT support, it is promising to build knowledge-based system capable not only to accumulation of the previous experience in the case base, but, also to the effective search of cases that are semantically close to the current problem.

An important initial step of building an efficient knowledge-based system is the choice of the knowledge representation model for the knowledge base. Such systems allow one to apply the logical inference algorithms to known rules and facts to extract new (unknown) facts from the knowledge base. Rules are convenient and expressive constructions for representing knowledge in various subject areas. It is with this method of knowledge representation that the "success stories" in the Artificial Intelligence (AI) is connected, for the AI transfer from the field of the simplest solutions to the sphere of real applications.

Since 1980s, an alternative reasoning paradigm has increasingly attracted more and more attention. The case-based reasoning (CBR) solves new problems by adapting previously successful solutions to similar problems, just as a human-being does it. This approach has been successfully used in medical diagnostics, and in legal counseling. In our opinion, this method is suitable for creating a system of consulting the users using the experience of solving similar problems in the past.

The CBR is based on the fundamental idea that this is often the way the decision-maker arrives, when finding in his memory information most closely related to the problem being solved and adapting the past conditions to new realities. The foundations of the CBR approach were set forth in the works of Shank [1,2], who first proposed to generalize knowledge about past situations in the form of the so-called scripts (knowledge containers that can be used both in the learning process and in the decision-making). The model of dynamic memory later became the basis for creation of a number of other systems: MEDIATOR [3], CHEF [4], and JULIA [5].

The CBR allowed one to overcome a number of limitations of the rule-based model [6]. It does not require the construction of an explicit domain model. The complex process of knowledge acquiring is reduced to the task of accumulating cases of making decisions in the past (precedents), as well as analyzing the results of decision making. Implementation of an intellectual system based on the CBR in the simplest case is reduced to identifying the main variables (features) describing the case and accumulating the data described by this set of features. However, with the CBR development, there appeared the main drawbacks. One of them begins to appear when a very large database of precedents is accumulated. This raises the problem of retrieving relevant cases from a huge amount of big data.

It seems to us that the shortcomings of the CBR could be significantly reduced on the basis of its integration with the knowledge representation models in the form of an ontology. For example, in paper [7], the representation of the IT application domain in the form of ontology was used to improve the semantic search for documents based on the indexing of documents by the ontology concepts in comparison with the usual indexing by keywords.

In the present paper, we propose an original method of integrating the cases of IT-consulting with the domain ontology on the basis of establishing the relationships of cases with the concepts of ontology. Each case can be related with several concepts of the ontology, which allows one to describe more adequately the semantics of a precedent of an IT consultation. The proposed method can be used for the subsequent effective extraction of cases relevant to the current situation using various Data Mining methods, for example, the method of constructing the fuzzy rules proposed in [8,9].

The paper is organized as follows. In Section 2, we describe the ontology of the application area, in which the task of IT-consulting is being solved, as well as its implementation in the Protege editor. Section 3 describes the structure of the class, the instances of which are specific cases (stories) of user counseling. In Section 4, we describe the proposed mechanism for the integration of cases with the ontology concepts.

2 Domain ontology for IT consultation

Ontology is a formal explicit description of the notions (concepts) of the application domain and the relations between these concepts [10]. The ontology could be represented by the following tuple:

$$O = \langle C, R, S, G, T, E \rangle, \tag{1}$$

where $C = \{c_i | i = \overline{1, n}\}$ is a finite non-empty set of classes (concepts) describing the basic notions of the application domain; $R = \{r_i | i = \overline{1, m}\}$ is a finite set of binary relations between the classes, $R \subseteq C \times C$, $R = \{R_{ISA}\} \cup \{R_{ASS}\}$, R_{ISA} is an antisymmetric, transitive and non-reflexive hierarchy relation "class-subclass" defining a partial order on the set of classes; R_{ASS} is an associative relationship used to establish a link between the case base and the ontology concepts; $S = \{s_i | i = \overline{1, k}\}$ is a finite set of slots (class attributes); $G = \{gs_i | i = \overline{1, l}\}$ is a finite set of facets (slot attributes); $E = \{e_i | i = \overline{1, u}\}$ is a finite set of class instances; T is a finite non-empty set, which determines the controlled vocabulary of the domain terms built on a set of basic terms $B = \{b_i | i = \overline{1, n}\}$ being a set of names of the ontology classes

$$T = \bigcup_{i=1}^{n} T_i, T_i = \{b_i\}, \bigcap_{i=1}^{n} T_i = \oslash. \tag{2}$$

The structure of the class is defined as follows:

$$c = \langle Name, (\mathrm{is-a} \quad \mathrm{c_{parent}}), (\mathrm{s_1}, ..., \mathrm{s_{n(c)}}) \rangle, \tag{3}$$

where $c, c_{\text{parent}} \in C$ are the ontology classes connected by the hierarchy relation R_{ISA}, $s_i \in S$ are the class slots, $Name_c \in B$ is the class name being the base term of the vocabulary T. Hierarchy of classes (taxonomy) is formed by means of indicating the relation "is-a" and the name of the class-parent c_{parent} in the descendant class. We call classes that have no descendants the terminal concepts of the ontology (terminals). Terminals will play the role of keywords in the semantic annotation of cases.

The structure of the slot is defined as follows:

$$s_c = \langle Name_{S,C}, (gs_1, ..., gs_{k(S,C)}) \rangle, \tag{4}$$

where $s_c \in S$ is a slot of the class C, $gs_i \in G$ is a slot facets (slot properties), $Name_{S,C}$ is the slot s_c name.

A fragment of the taxonomy (hierarchy) of the upper level concepts, which are direct descendants of the general *Thing* class, is shown in Fig. 1. The ontology was created in the Protege 4.2 Editor that is free software for building ontologies.

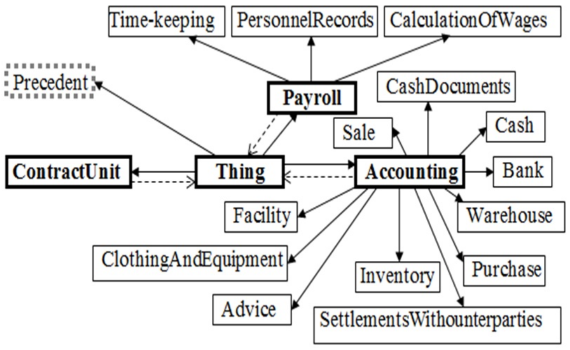

Fig. 1. Ontological graph of the upper level concepts

The main concepts of the upper level ontology are *Accounting*, *Payroll*, and *Contractunit*. The concept *Accounting* describes the main subsections of accounting. The concept *Accounting* has twelve subordinate concepts forming the taxonomy. The concept *Payroll* describes the main subsections of the taxonomy "Calculations with the staff". In this taxonomy, the tasks of automating the activities of both managers (who make decisions on the salary of staff) and accountants of salaries are solved. It also ensures the maintenance of mutual settlements with the employees of the enterprise, records payroll costs as part of the cost of production and services, starting with the input of documents on actual production and payment of sick leaves up to the formation of documents for payment of wages and reporting to Governing supervisory bodies. The concept *Contractunit* describes the main subsections of the subject area "Contractual Block". The contractual block is intended for automating the work of users in the sphere of registration and conducting contracts of counterparts.

The hierarchy of concepts created in Protege 4.2 contains 71 concepts of the taxonomy *Payroll*, 82 concepts of the taxonomy *Accounting*, and 11 concepts of the taxonomy *Contractunit*.

Thus, the ontology of the application domain described above is a combination of three taxonomies of concepts connected by hierarchical relations. In principle, if the descendants of a certain parent have unequal influence on the parent concept, then it is possible to introduce weight coefficients into the taxonomy. To do this, each concept with a parent is added a slot, containing the weight of the concept. In present version of the ontology it is assumed that all the children of the same parent have identical weight equal to $\frac{1}{G}$, where G is the number of children of the given parent.

3 Structure of Precedent class

Case based reasoning (CBR) is an approach that allows one to solve the new problem by using or adapting a solution previously taken in a similar situation. By the case, we mean a description of the problem or situation in conjunction with a detailed sequence of actions taken in this situation to solve this problem. When a new situation is considered, the system finds a similar case in the knowledge base as an analog to the task being solved and tries to apply the solution of the found precedent. If necessary, a close precedent is adjusted to the current situation. After applying the analogy-based solution to the current problem, the analysis of the results is processed, after which a new case is added to the case base for future use. So, the complete case description should include the following elements: description of the situation with the features; the decision that was made in this situation; interpretation (result) of applying the solution.

The case can be represented in various ways, for example, with the help of tree structures, rows in databases, frames, etc. It is important to understand that the choice of a precedent representation is necessary that is based on the overall objectives of the system. The main problems when presenting a precedent are: the choice of information that should be included into the description of the case, the search for a convenient precedent structure, and the organization of a knowledge base for optimal and efficient search.

It is important to understand that the choice of a case representation should be based on the overall objectives of the system. The major problems in the cases representation are: the choice of information that should be included into the description of the case, the search for a convenient case structure, the efficient organization of the knowledge base for optimal, and efficient retrieval of cases. The initial case representation could be simple (linear)

$$CASE = (x_1, x_2, x_3, ..., x_n, s), \tag{5}$$

where $x_1, x_2, x_3, ..., x_n$ are the values of attributes (features) identifying the situation, s is solution to the problem defined in the case. Subsequently, with the deepening into the problem domain, possible complication of case structure is possible through, for example, the introduction of hierarchy and other relationships between the attributes.

To integrate the ontology of the application domain with a description of cases, a class *Precedent* was created. This class does not have a branched hierarchical structure like the domain ontology classes (concepts), but it does not have child classes at all. The purpose of the class *Precedent* is to create the complete structure to input the information about the decision-making cases and, also, to establish a semantic link of the case with the domain ontology. The class *Precedent* includes three groups of properties (slots) dividing structurally and substantively the information included into the case description.

The slot *Main* has the following child slots:

– *Decision* contains a complete description of the sequence of the user actions (technology) to solve the problem;
– *DescriptionUser* contains information about the problem that the user transfers to the consultant when formulating the request;
– *Error* is filled with the information about technical error (if it is), which could be solved only by reprogramming;
– group of slots *Keyword* 1 ... 3 contains one or several slots to create semantic link with the ontology keywords;
– *SoftwareProduct* contains information about the software where the error occurs (1C, Axapta, etc.);
– *UserRole* determines the user, which can be an employee of the personnel department, an accountant, a timekeeper, an auditor, etc. (the functionality that can be used to solve the problem depends on the user's role);
– *VersionProgram* determines release or version of the software product. Software products are constantly updated developers correct errors, therefore, before deciding the user's problem it is necessary to understand which release the user is working on.

The slot *Changes* of the class *Precedent* slot is useful when several consultants work with the same database. With this information, one can always understand who and when changed the case sample. This slot has the child slots (*Period* is the date and time when the case was created or the changes on the case were made and *User* the name of the user who made the changes over the case). The slot *File* has the child slots *FileDescription* containing a brief description of the file and *FileName* determines the path to the file attached to the case. This can be a file with an error that occurs in the request or a file with a troubleshooting guide.

The structure of the *Precedent* class described above has the necessary completeness and non-redundancy. We determine where the problem arose (software and its version), who meets the problem (user role), how the user sees the problem (user description, error). The consultant gives professional definition of the problem characteristics and determines the place of the problem in the domain ontology through associative links with the ontology concepts. The case contains the information about modification of the case by the consultant. Finally, a file that contain instructions for solving the problem can be attached to the case.

4 Integration of cases with the ontology

It seems promising to make a comparison between the current situation and cases assessing the degree of their connection with the concepts of ontology. Thus, closeness of cases to each other is estimated by degree of the semantic closeness of the concepts associated with these cases. To achieve that, it is necessary to determine the semantic links of newly introduced cases with the ontology concepts at the stage of creating the knowledge base.

The link of the instances of the *Precedent* class with the ontology concepts is established by setting the associative relation R_{ASS} for the slots of the *Keywords* group for *Precedent* class (slot *Main*). The link is determined by the explicitly specifying the associated ontology concept name as the slot value. To implement this associative link, the type D_{class} is used as the type for the group of slots *Keywords*. If, for example, the i-th slot of the group *Keywords* has the type D_{class} with the associated class C_i, then as the slot values (when creating the class *Precedent* instances) we can use the classes of a set $Tr(C_i)$ of the transitive closure of the concept C_i including the class $C_i = C_i^0$ and all its subclasses that are below in the hierarchy

$$Tr(C_i) = \{C_i = C_i^0\} \bigcup ISA(C_i^0), \tag{6}$$

where $ISA(C^0) = \bigcup_{l=1}^{L}\{C^l \in C | \exists R_{ISA}(C^{(l-1)}, C^l)\}$. L being the maximum depth of the descendants of the class. Here, the classes *Precedent* and C_i are connected by the associative relation $R_{ASS}(Precedent, C_i)$.

Establishing the connection of a specific case with the ontology, the analyst chooses concepts that are closest by the meaning to the case. It can be either terminal (having no descendants), the most specified concepts, and non-terminal (intermediate) concepts that have a more general meaning. Necessity in the links with non-terminal concepts arises if the current problem cannot be unequivocally referred to the terminal concept or the analyst does not have sufficient experience and it is easier for him to classify the case to a more general concept.

It should be emphasized that we allow establishing of not unique but several different links for the case with the ontology concepts. This expands the expressive possibilities of our approach and can be used when the problem arises at the junction of several concepts and its adequate description requires consideration of this interdisciplinary character.

Let in addition to the concept name C_i, the weight value v_i, $0 \le v_i \le 1$, $\sum\limits_{i=1}^{I} v_i = 1$, is given as the facet (property of a slot) for the i-th slot of the group *Keywords*, which establishes the strength of the relationship between the case and the corresponding ontology concept. The more is the weight, the closer by the meaning the case is to the corresponding concept of the application domain.

Now let us consider how to organize the procedure of classifying and retrieving semantically close cases using the integrated model described above. Further, we distinguish between terminal and nonterminal concepts of the ontology. Let particular shall terminal concepts be the keywords for indexing the cases. Let we have J keywords and each keyword kw_j,$j = \overline{1, J}$, corresponds to the weight w_j,$j = \overline{1, J}$, $\sum\limits_{j=1}^{J} w_j = 1$, that can be computed from the weights v_i for the cases and the weights of the hierarchy relations in the ontology.

The procedure for calculating the weights w_j, $j = \overline{1, J}$, can be organized as follows. Without loss of generality we assume that the ontology concepts with which a certain case is connected do not enter into the transitive closure of each other (that is, they should not be located on the same hierarchical branch). This assumption is quite natural, since if we can link the case instance to a more specific concept-descendant, then there is no need in its connection with a more general concept-ancestor. In this case, the procedure for forming a vector of weight coefficients w_j, $j = \overline{1, J}$, for the keywords kw_j, $j = \overline{1, J}$, can be represented as follows. Suppose that considered case is related to concepts C_1, C_2, ... , C_I.

First, we assign w_j, $\forall j = \overline{1, J}$. Second, introduce the cycle for all concepts C_i, $i = \overline{1, I}$, connected with the precedent

- if C_i is a terminal concept ($kw_j = C_i = C_i^0$), then $w_j = v_i$;
- if C_i is not a terminal concept, *i.e.*, terminal concept kw_j is the L-level descendant of the intermediate concept C_i, ($kw_j = C_i^L$), then $w_j = v_i * \prod\limits_{l=1}^{L} v_i^l$, where v_i^l is the weight of the hierarchical relation from the parent concept $C_i^{(l-1)}$ to the child concept C_i^l on the way from the concept C_i connected with the case instance to the terminal concept kw_j. The weights of concepts being descendants to the one parent in the ontology are considered to be the same, as discussed in Section 2.

Let us illustrate calculation of the keyword weights for a specific example. Suppose that in the knowledge base we have a specific case associated only with the concepts belonging to the transitive closure of the class *Payroll* with respect to the relation R_{ISA}, a fragment of an ontology with a fully expanded class (up to terminal concepts), is shown in Fig. 2. The expert-analyst has established three associative relationships from the case to the ontology, and two links are established to the terminal concepts $TransferOrder$ and *OtherOrders*. The weights of the terminals $TransferOrder$ and *OtherOrders*, accordingly, are assumed to be equal to the corresponding weights established by the analyst $v_1 = 0.3$ and $v_2 = 0.1$.The third link is established to the non-terminal concept $Staffing$, which is the parent of the three terminal concepts *DepartmentDirectory*, *PostDirectory*, and $StaffingSpecification$. Accordingly, the weight $v_3 = 0.6$, introduced by the analyst to link this case with the concept $Staffing$ is divided into these three keywords equally. The weights corresponding to the keywords are equal to $w_j = 0.6 * \frac{1}{3} = 0.2$. The other keywords remain unrelated to the case sample, and accordingly, have zero weights. Thus, for a subset of the eight terminal concepts *OrderOnAdmission*, $OrderOfAssigmentOfClassRank$,*OrderOfDismissal*,*OtherOrders*, *Trans-ferOrder*,*DepartmentDirectory*, $StaffingSpecification$, and *PostDirectory* represented in Fig.2, we have the following sub-vector of weights

$\widetilde{w} = (0; 0; 0; 0.1; 0.3; 0.2; 0.2; 0.2)^{\mathrm{T}}$ corresponding to the current case. Since this case has the relations only with the concepts of this fragment of the ontology, all remaining keywords of the ontology have zero weights.

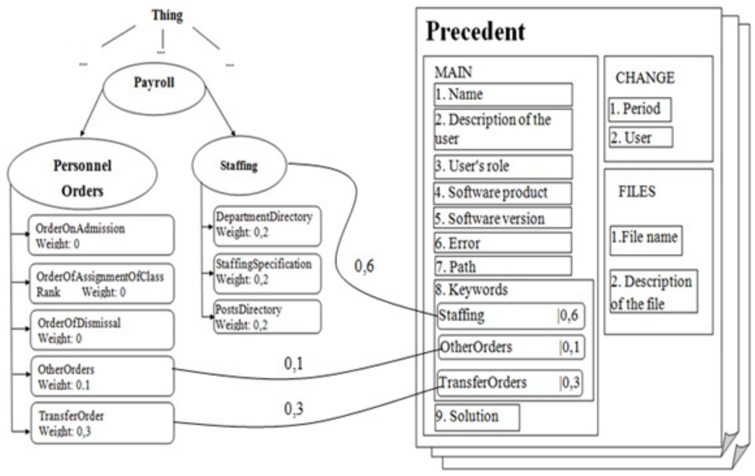

Fig. 2. Calculating the relationships of the case with the ontology terminals

Thus, all the cases stored in the knowledge base are indexed using the ontology concepts and corresponding keywords. Each keyword (terminal concept of the ontology) is included into the case representation with a weight calculated on the basis of the associative relationships between the case and the ontology concepts. As a result, we obtain the data table with the values of weights w_j, $J = \overline{1, J}$, for each case in the case base being the instances of the *Precedent* class. The number of rows of the data table is equal to the number of cases and the number of columns is equal to the number J of the ontology terminal concepts. One can further apply data mining methods to the obtained data table extracting knowledge from data. For example, it is possible to conduct cluster analysis to divide the set of cases into the classes of semantically close cases.

After obtaining semantically close classes of cases as a result of clustering the cases, the procedure for retrieving cases relevant to the current problem consists in the following. Describing the current situation, the analyst should relate the current case with the ontology concepts, as it was done with the cases in the knowledge base. As a result, we will have a vector of keyword weights corresponding to the problem being solved. In this case, the task of extracting cases relevant to the current problem can be reduced to classification of the current problem. Here, the most suitable method is the classification of cases based on building the system of fuzzy linguistic rules proposed in [8,9]. In these papers, the high accuracy of classification of cases have been confirmed on the test data. In addition, this method permits obtaining rules being the knowledge of a higher organizational level than cases in the form of data table. The rules can be analyzed by experts and added to the knowledge base for the direct use in the explicit form.

5 Conclusion

The paper proposed an approach to creation of information technology for decision-making in the field of IT consulting based on integration of case base with the domain ontology. As a result of application of the approach we obtain data matrix characterizing semantic closeness of cases through their closeness to the domain ontology concepts. Further one can apply machine learning methods to the data matrix, first, to derive classes of semantically close cases, second, to retrieve relevant decision-making cases using classification algorithms.

Acknowledgments

The reported study was funded by Russian Ministry of Education and Science, according to the research project No. 2.2327.2017/4.6.

References

1. Schank, R., Abelson, R.: Scripts, Plans, Goals and Understanding, Erlbau (1977)
2. Schank, R.: Dynamic Memory: A theory of reminding and learning in computers and people. Cambridge University Press (1982)
3. Simpson, R.: A Computer Model of Case-Based Reasoning in Problem Solving: An Investigation in the Domain of Dispute Mediation. Technical Report GIT-ICS-85/18, Georgia Institute of Technology, School of Information and Computer Science (1985)
4. Hammond, K.: CHEF: A model of case-based planning. In: Proc. American Association for ArtificialIntelligence, AAAI-86, Philadelphia, PA (1986)
5. Hinrichs, T.: Problem Solving in Open Worlds. Lawrence Erlbaum (1992)
6. Watson, I., Marir, F.: Case-based reasoning: A review. The Knowledge Engineering Review. 9(4), 327–354 (1994)
7. Shanavas, N., Asokan, S.: Ontology-Based Document Mining System for IT Support Service. Procedia Computer Science. International Conference on Information and Communication Technologies (ICICT 2014), 329–336 (2015)
8. Avdeenko, T., Makarova, E.: Integration of case-based and rule-based reasoning through fuzzy inference in decision support systems // Procedia Computer Science. 103, 447–453 (2017)
9. Avdeenko, T., Makarova, E.: The case-based decision support system in the field of IT-consulting // Journal of Physics: Conference Series. 803, 6 (2017)
10. Gruber, T.: Ontolingua: A Mechanism to Support Portable Ontologies, Technical Report KSL 91-66, Knowledge Systems Laboratory, Stanford University (1992)