Polibits

ISSN: 1870-9044

polibits@nlp.cic.ipn.mx

Instituto Politécnico Nacional

México

Sosnin, Petr

Creation and Usage of Project Ontology in Development of Software Intensive Systems

Polibits, vol. 42, 2010

Instituto Politécnico Nacional

Distrito Federal, México

Creation and Usage of Project Ontology in Development of Software Intensive Systems

Petr Sosnin

***Abstract*—The key problem of successful developing of the software intensive system (SIS) is adequate conceptual interactions of designers during the early stages of development. The success of the development can be increased by using of a project ontology, the creation of which is being embedded into the processes of conceptual solving the project tasks and specifying the project solutions. The essence of the conceptual design is a specification of conceptualization. The main suggestion of this paper is a creation of the project ontology in the form of a specialized SIS that supports the conceptual activity of designers. For creation of the project ontology of such type, the instrumental shell was developed. For creation of the project ontology the designers should fill this shell with the adequate information. The basic reasons for evolving the content of the ontology are negative results of testing of the used text units according to the conformity to the ontology. Such shell (in any state of its using) includes the created ontology and its working version (working dictionary) which helps to manage the informational flows, to register the life cycles of the conceptual units and to provide the representativity of their usages.**

***Index terms*—Project ontology, system development, software engineering, task solving.**

I. Introduction

NOWADAYS one of the most challenging area of computer applications is "Development of Software Intensive Systems", within the frame of which the collaborative works of developers and other stakeholders are being carried out in corporate networks. The success of such activity in this area, which is being estimated regularly by corporation Standish group [18] for last 16 years, is extremely low (a little more than 30%). Failures can occur in development of the SIS related to any part of the SIS's definition [15]: "A software intensive system is a system where software represents a significant segment in any of the following points: system functionality, system cost, system development risk, development time."

A very important cause of the failures is semantic mistakes in the collective intellectual activity of developers and other persons involved to the development of the SIS. The necessary condition of the developers success is their mutual understanding in collaborative actions based on reasoning over textual information including the statements of task and definitions of project solutions. Developers of the SIS should be supplied with useful and effective techniques for the prevention and correction of semantic mistakes.

At the beginning stage of the SIS development the necessary understanding usually is absent. The adequate understanding is formed only gradually and step by step during the interaction in working groups. Evolution of understanding follows step by step the design of the SIS in the collaborative development environment (CDE) and the current state of understanding includes its positive influences on the management of the development process.

The important role of understanding (personal and mutual) in the development of the SISs is well known. For exploiting of this phenomenon the special techniques for "interactions" with understanding are being created and are used. One type of such technique is a glossary. The specialized version of the glossary is applied, for example, in widely used methodology (and technology) Rational Unified Process (RUP) [14]. Let us notice that in the RUP such artifact is normatively defined, though it does not have collaborative techniques for its informational filling in real time of design. The problems of dynamically extracting, defining, modeling, registering, keeping and visualizing the units of understanding in designing the SISs do not have satisfactory solution.

In this paper for the explicit work with understanding of SIS designers, a specialized system of the project ontology, which is creating as a subsystem embedded into the developing SIS, is proposed. Moreover, it is suggested to create the project ontology as an interactive system of the SIS type. Such system which will be denoted below as SIS^{ONT}, is implemented on the base of an ontology shell which supports the collaborative extracting and checking of ontology units from statements of project tasks and definitions of project solutions.

The implemented ontology shell is included into the instrumental system WIQA [16] which is aimed to designing the complex system of the SIS type. The WIQA is based on question-answer reasoning and the models of the units' flow, of which the informational source of concepts' usages embedded into the project ontology is extracted.

II. Related Works

A set of typical kinds of ontologies, according to their level of dependence on a particular task or a point of view, includes the top-level ontologies, domain ontologies, tasks ontologies and applied ontologies. All these types of ontologies are defined in [9] and [10] as techniques that are used in different systems.

Manuscript received May 9, 2010. Manuscript accepted for publication August 3, 2010.

Petr Sosnin is with the Computer Department, Ulyanovsk State Technical University, Ulianovsk, 432028, Russia (sosnin@ulstu.ru).

For the SISs, the more adequate type of ontologies is applied – the type that must be expanded usually by means of the other ontologies' types. In accordance with the publication [11], the theory and practice of applied ontologies "will require many more experiences yet to be made".

It is necessary to notice that the project ontology as a subtype of applied ontologies is essentially important for SISs. Project ontologies mainly are aimed at the process of design but after refining they can be embedded into implemented SISs.

The specificity of project ontologies is indicated in a number of publications. In the technical report [5] the main attention is concentrated on "people, process and product" and collaborative understanding in interactions. Investigation of the possibility of the ontology-based project management is discussed in the paper [1].

The usage of the ontology potential in developing the program system and ontological problems of program products are investigated in the paper [4]. This article describes the experience of development of the task ontologies taking into account first of all the role of different kinds of knowledge. The introduction of knowledge into the task ontologies is reflected and discussed in the work [2]. The role of knowledge connected with problem-solving models is presented in the paper [12].

In all mentioned publications there are many useful ideas but the approach to the ontology as to the specialized SIS^{ONT} – for extracting, defining and assembling concepts into the ontology in the process of designing the SIS – is not considered. The Internet search of publications with key words which include such phrases as "project ontology" and "software intensive system", has remained without competitive results coinciding with results suggested in this paper.

Let us remind that the main goal in using the project ontology is to provide the necessary understanding in collaborative design which is impossible without human-computer interaction. Therefore the theory and experience of human-computer interaction as presented in [13] were taken into account in this paper.

III. Specificity of Suggested Ontology

Attempts to view the project ontology from the side of creating the specialized SIS^{ONT} leads to the questions about its architecture, life cycle and used models which must be coordinated with the evolution of the project ontology. Below we answer these questions.

The architecture of any SIS^{ONT} for the definite SIS has a problem-oriented type the materialization which begins its life cycle from the ontology shell with architectural solutions, inherited and kept by the SIS^{ONT} without changing. The principle architecture of the shell (and any SIS^{ONT} also) is presented in Fig. 1.

For any dictionary entry of the ontology there is a corresponding analog in the working dictionary. Such analog is used firstly as a representative set of samples registering the variants of the concept usages extracted from statements of project tasks and definitions of project solutions (or shortly from text units). Samples are being gathered naturally in interactions of designers who are testing (implicitly or explicitly on different working places) the used concepts according to their conformity to the ontology.

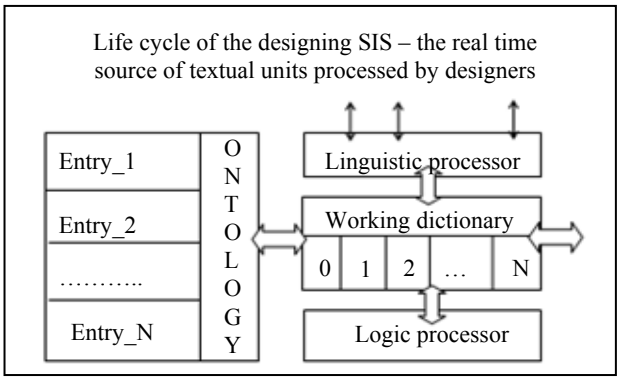

Fig. 1. Architecture of the project ontology.

Filling the ontology by the content is connected with a specialized project task appointed to an administrator of the ontology. The work of the administrator is managed:

- By events each of which is generated when the result of comparison of the used concept with the ontology is not correct;
- In accordance with a sequence of actions supporting the normative state of the project ontology (current levels of adequacy and systematization).

The necessary informational material for the administrator of the ontology is supplied by designers with the help of the predicative analysis. Designers must test and confirm the authenticity of concepts which are used in statements of tasks and definitions of project solutions. For achieving such aim they have to extract firstly the usage of concepts (from the text units) and then to compare them with the ontology. The differences of comparisons (new concepts or additional parts of existing concepts, additional questions which require answers) are used as the informational material for evolving the ontology. Let us notice that any extracted concept usage includes its expression as a simple predicate but not only this (the full expression will be presented below).

Used concepts are the main part of the project ontology which should be expanded by systematizations and axiomatic relations. Techniques of systematizations are embedded into the ontology component while axiomatic relations are being created with the help of the logic processor.

The logic processor is intended to build the axiomatic relations as formulas of the logic of predicates. Such work is being implemented in the frame of the appropriate article (entry) of the working dictionary where the necessary simple predicates are being accumulated. Ontology axioms express materialized units of the SIS and first of all those of them which corresponds to UML-diagrams. Any built axiom is registered in the definite entry (article) of the ontology.

The main architectural view presents the project ontology from the side of its components and informational content which defines the dynamics of the life cycle for the SIS^{ONT}. In a typical case such life cycle is being implemented in the form of the real time work of several dozens of designers who have solved and are solving several thousands of tasks. Models which are used in the ontology life cycle will be presented below.

IV. Linguistic Processor

The life cycle of the SIS^{ONT} is embedded into the life cycle of the designing SIS from which all (named above) text units are being introduced into the linguistic processor. Another possibility is to apply some term-extraction technique, for example, as described in [8].

For testing any text unit, it is transformed into a set of simple sentences and in such transformation the pseudo-physics model of the compound sentence or complex sentence of the other type is applied. In the pseudo-physics model of the sentence all used words are interpreted as objects which take part in the "force interaction" which is visualized on the monitor screen. Formal expressions of pseudo-physics laws are similar to the appropriate laws of the classic physics.

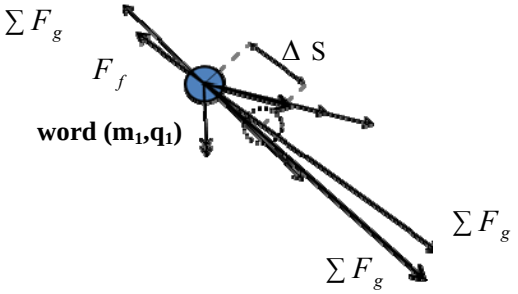

Fig. 2. Interaction of forces.

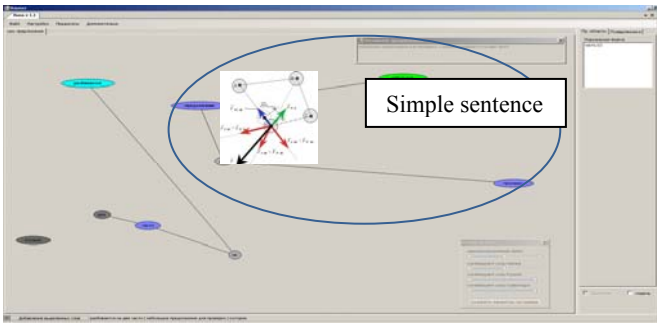

Fig. 3. Extraction of the simple sentence.

In accordance with acting forces (forces of "gravitation" F_g, "electricity" F_q, "elasticity" F_e and "friction" F_f) and attributes appointed to the "word-objects" such objects after moving are being grouped in definite places of the interaction area. The possible picture of the forces interaction for one word of the investigated sentence is shown in Fig. 2.

In the stable state (Fig.3), each group of words-objects will present the extracted simple sentence after finishing dynamic process on the screen.

The screenshot in Fig.3 and other screenshots of this paper are used with labels for the generalized demonstrations of the visual forms and objects with which the designers are working. The language of these screenshots is Russian.

Let us notice that in the assignment of attributes (values of m_i, q_i and others values and parameters) two mechanisms are applied – the automatic morphological analysis and the automated tuning of object parameters. Values are assigned in accordance with the type of the part of speech. The suitable normative values were chosen experimentally. For description of morphological analysis see works [6], [7].

After extraction of simple sentences the designer begins their semantic analysis aimed to testing the correctness of each simple sentence (SS_i). In such work the designer uses the model of SS_i and its relations with surrounding, as presented in Fig. 4. This picture shows the type of SS_i which is used for registering the appointment of the property for object. The other type of model for registering the appointment of relation between two objects has the similar scheme.

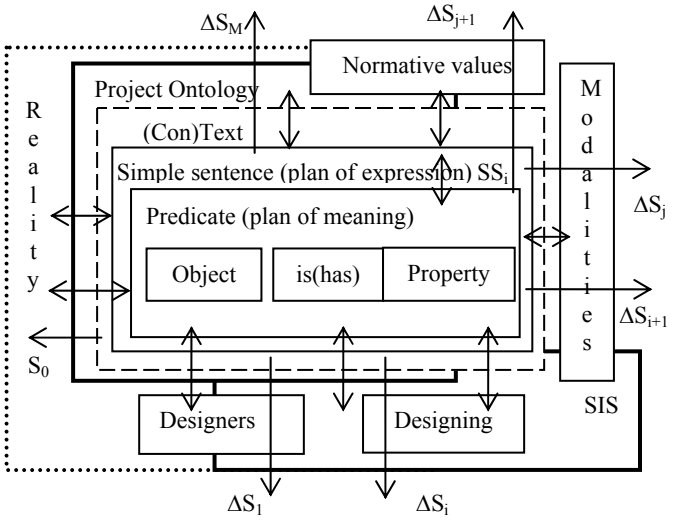

Fig. 4. Model of the simple sentence.

The scheme of relations was used for defining and implementing the techniques for their semantic testing. First of all the expression of semantics for SS was chosen. The structure of the semantics value as a set of semantic components ($S_0 \cup (\cup \Delta S_n)$) is presented generally in Fig. 4 where the component S_0 indicates for the sentence SS its conformity to the reality.

Definition and testing of any other semantic component ΔS_i helps to precise the semantic value of the SS if that can be useful for the design of the SIS. Additionally, the work with any semantic component increases the belief in the correctness of the testable simple sentence (and embedded simple predicate) and can lead to useful questions. In the work with additional semantic components the conditional access to appropriate precedents is used.

Elements of the typical set of semantic components are estimated, applied and tested in the definite sequence. Such work begins from the component S_0 which is compared with elements of the ontology. The result of comparing can be positive or can lead to questions which should be registered.

The positive result does not exclude the subsequent work with additional semantic components.

Semantics of subjectivity and understanding (part ΔS1) are estimated and tested for the relation with designers. The fact of the non-understanding leads to questioning or even to interruption of the work with the testable sentence.

Actual or future material existence of the sentence semantics is a cause for testing the semantic relation of the SS with designing (part ΔSi). Such type of relations is used in the ontology for its systematization.

The greater part of semantic relations of the modality type (parts ΔSi+1 − ΔSj) is aimed to defining and testing of the uncertainties of measurable and/or probable and/or fuzzy types. The semantic relations with normative values (parts ΔSj+1 − ΔSM) suppose the potential inclusion of the SS or its parts into the useful informational sources, for example, into the ontology.

V. Sources of Text Units

As it is shown in Fig. 1 the primary information for filling the project ontology is being extracted by designers from the life cycle of designing the SIS in the real time.

For the designers interaction with the life cycle of the SIS the specialized instrumental system **WIQA** (**W**orking **I**n **Q**uestions and **A**nswers) was created. The main interface of the WIQA is presented in Fig. 5 (with commentary labels).

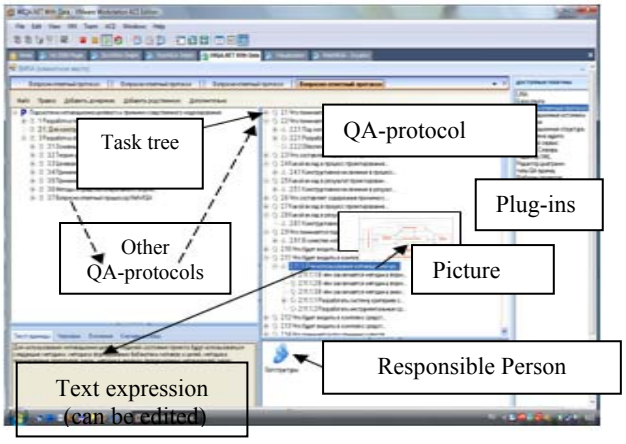

Fig. 5. The main interface of the WIQA.

The WIQA is intended for registering the current state of designing in the form of a dynamic set of project tasks combined into an interactive tasks tree. Each task of such tree is defined with the help of the question-answer protocol of its solving. Any QA-protocol opens the access to the question-answer model (QA-model) of the corresponding task.

The screenshot shows that for the chosen task Z_i from the task tree its QA-model is opened through the QA-protocol of the registered question-answer reasoning (QA-reasoning). Let us notice that any unit of reasoning (question Q_{ij} or answer A_{ij}) has a textual expression with necessary pictures (for example, with UML-diagrams and/or "block and line schemes"). Any task with its statement and any unit of QA-reasoning has the unique name Z.I or Q.J or A.J where I or J is a compound index expressing the subordinations of the corresponding unit. So any text unit is visualized and has a unique index which can be used as its address.

More specifically, any unit of the Z-, Q- or A-type is the interactive object the properties of which are being opened when the special plug-ins are used. One of such plug-ins registers and indicates the responsibility (the assignment of the tasks) in the designer group.

The WIQA is created on the base of the QA-model and the usage of following architectural styles − repository, MVC, client-server and interpreter. So for the current state of design of the definite SIS the WIQA can open to designers the statement of any task from the tasks tree and the definition of any project solution accessible as the definite answer in the corresponding QA-protocol.

Let us notice that the usage of the WIQA as the source of text units is a solution proposed by the author but the suggested ideas are possible to use for creating the project ontology with other instrumental systems which can supply designers by statements of project tasks and definitions of project solutions.

VI. Working Dictionary

The role of the working dictionary is very important in creating the project ontology. This component as the preliminary version of the ontology accumulates all necessary information and distributes informational units between dictionary articles. Carrying out functions of transportation of information, the working dictionary registers relate the text units with their sources. The index name of unit, the number of its sentence and the number of the corresponding simple sentence are used for such referencing.

After extracting the simple sentence with the help of the linguistic processor the predicate model of this sentence is being included into the virtual article of the working dictionary (the article with zero index). Zero article is a temporal memory in the working dictionary which keeps predicates till finishing their testing on the ontology conformity. Zero article, the interface of which is presented in Fig. 6, can be interpreted as a queue of predicates in their mass service.

After extracting any simple sentence and transforming it to the simple predicate, the designer has to start the test of the predicate (as the definite usage of the definite concept). The test begins usually without knowing the "normative usage of the concept" for this predicate in the ontology. Moreover, such usage of the concept in the ontology can be absent or the result of comparing with the appropriate concept will be negative. That is why any tested sentence and corresponding

predicate start their life cycles in the working dictionary from zero article.

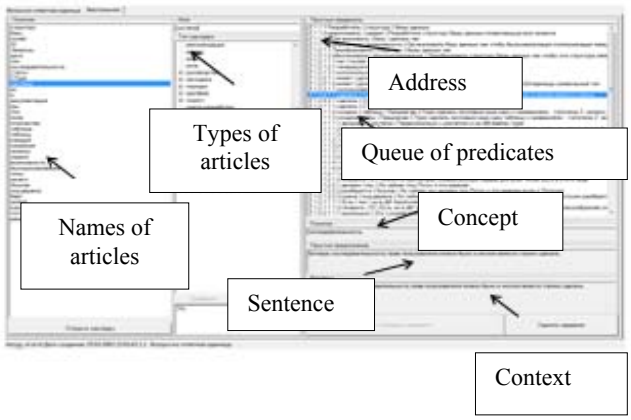

Fig. 6. Virtual article of the working dictionary.

The "normative usage of the concept" for any tested predicate is localized into the corresponding ontology article. If the result of comparing is negative but the designer is convinced that "predicate is truth" then the new ontology article is to be created or the new variant of the concept usage is to be built into the existed ontology article. The first of such results requires to create the new article in the working dictionary also and to transport the tested predicate from the virtual article into this new article (Fig. 7).

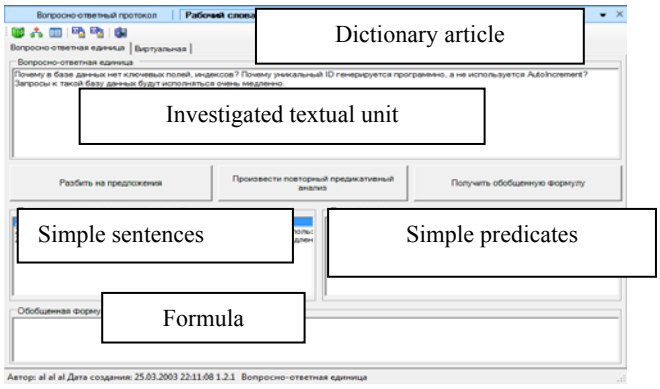

Fig. 7. Typical article of the working dictionary.

The type of the new article in the working dictionary is being chosen by designers in accordance with the type of the ontological unit of the designing SIS for representation of which the transported predicate will be used.

Processing the second result includes the transportation of the tested predicate but into the existed article (Fig. 7) of the working dictionary. In general case such predicate is transported into several articles of the working dictionary each of which materializes the tested predicate in the definite form.

If the test of the predicate on the conformity to the ontology is positive then this predicate should be transported in the article of the working dictionary, but only in the article of the definite concept for achieving its representativity. So (step-by-step) predicates (and their parent sentences) are being accumulated into corresponding articles of the working dictionary.

There is a set of types of materialized SIS units which are reflected in the project ontology. The set includes concepts about "parts" of the reality embedded in the SIS and materialized in its software (in the form of variables, classes, functions, procedures, modules, components and program constructions of the other types) and axioms which combine concepts. Each of such unit is found as its initial textual expression in statements of project tasks or in definitions of project solutions. But when this unit is included into the ontology article it is usually rewritten, redefined and reformulated. All informational material for the execution of the similar work is accumulated in the corresponding article of the working dictionary. After creating the adequate textual expressions and formulas they are rewritten from the working dictionary to the corresponding articles of the project ontology.

VII. Logic Processor

The logic processor is intended to build the formal description of the text unit from simple predicates accumulated in the definite article of the working dictionary. Such work is being fulfilled by designer in the operational space presented in Fig. 8 where designer assembles simple predicates in the formula watching them in the graphical window. Necessary predicates are being chosen by designer from the processed article of the working dictionary.

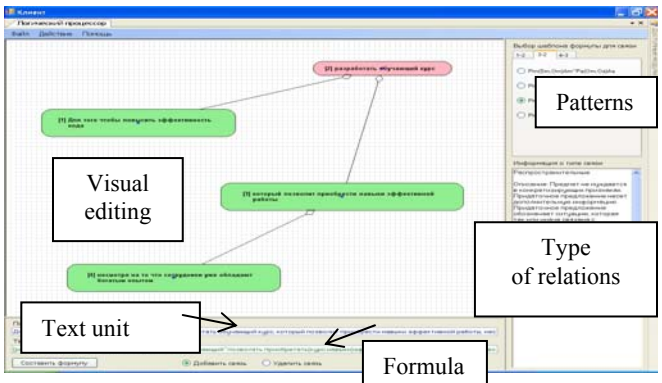

Fig. 8. Assembling the formula for a text unit.

To assemble the predicates the designer has possibilities to use the patterns of two bound predicates and setting of the typical relations between predicates by editing the "picture" (using the drag and drop and lexical information) and registering the final result as the formula of the first predicate logic.

Patterns for two bound predicates has been extracted by author from the grammars of Russian (46 patterns) and English (32 patterns). Such patterns are formalized as typical formulas of the predicates logic.

Mechanisms of assembling the formulas were evolved with experimental aims as the complex of instrumental procedures that provides (for statements of tasks) the creation of prolog-like descriptions. The transformation of the formalized

statement of task to the prolog-like description is being implemented as an automated translating of the formula registered in the appropriate article of the working dictionary. Now the method of translating exists in the preliminary version which will be rationalized by the author.

VIII. Systematization of Ontology

The most important feature of any ontology and the project ontology in particular is its systematization. In suggested case the project ontology is defined initially as the Software Intensive System, the integrity of which is provided by the system of architectural views. Some of these views are reflected implicitly by screenshots used in this paper. But such version of the systematization is only one possibility.

Let us present the other way of the systematization. First of all it is the classification of concepts in accordance with structures of the SIS and process of its design. Such system features of the ontology are formed implicitly through definitions of concepts and corresponding axioms.

The next classification level of the ontology is bound with classifying the variants of concept usages. In this case for any concept its article in the project ontology is being formed, which includes the ordered group of concept usage variants and the textual definition of the concept.

The group of usage variants is a list of sub-lists each of which includes main word (or phrase) as a name of the concept (C_i) and subordinated words (or phrases) as names of characteristics (w_{i1}, w_{i2}, …, w_{iN}) of this concept. The definite sub-list w_{i1}, w_{i2}, …, w_{iN}, C_i is an example of the "normative usage of the concept" which can be used in testing of the investigated predicate on the conformity to the ontology.

The basic operation of testing is a comparison of the normative (ontological) sub-list of words with words extracted from the investigated predicate. Two similar sub-lists of words can be extracted from the simple predicate when it indicates the feature and three sub-lists when the predicate registers the relation.

After testing the chosen sub-list of words, which expresses the definite variant of the concept usage, the following results of comparison are possible:

- positive result when the chosen sub-list (w'_{i1}, w'_{i2}, …, w'_{iN}, C_i) is included into the normative sub-list;
- interrogative result when chosen sub-list crosses the normative sub-list or the tested sub-list is outside of all norms (the role of questions was explained above).

The next direction of the systematization is related to binding concepts. For uniting the ontology concepts into the system the following relations are used: basic relations (the part and the whole, the hereditary, the type of the materialization), associative relations (in accordance with the similarity, the sequence, common time and common space) and causality relations.

This type of the view onto the ontology (onto the system of concepts) is formed by administrator of the ontology at the screen shown in Fig. 9. Any unit of any such form is opened for interactive action of designer.

To use the concept relation the designer chooses the necessary concept by its names in the area "keys of entry" and then designer can switch among groups (nodes of the relations system) up to the necessary relation. For the group of relations presented in Fig. 9 the designer may navigate in these directions – "part of", "whole for", "has attribute", "attribute of", "descendant of", "parent for", "has type" and "materialized as". Similar schemes of navigation are used for the other classes of relations also.

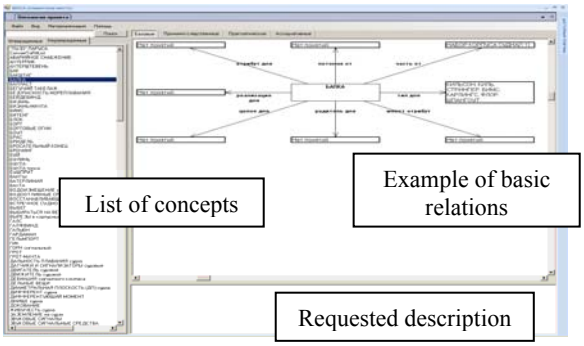

Fig. 9. Systematization of ontology concepts.

In any state of the navigation the description of any visualized unit can be opened. Let us notice that all forms of the ontology systematization are inherited by the working dictionary where it opens the possibility for useful switching between its articles.

IX. Collision Avoidance of Sea Vessels

The proposed version of the project ontology was created and used in the development process of the "Expert system for the collision avoidance of the sea vessels" which is implemented with using the WIQA capabilities [17].

One of the important components of this expert system is a knowledge base which includes the normative rules for the vessel movement. Any unit of such rules was formalized as a precedent with conditional and behavioral parts. Such precedents were extracted from the textual descriptions of normative rules in accordance with their formalizing and coding in expert system by the WIQA capabilities.

At the first stage of the expert system development about 150 textual expressions describing precedents were extracted from 37 rules of The International Regulations for Preventing Collisions at Sea 1972 (COLREGS-72) presented in [3].

Each textual expression was processed with the usage of techniques described above. As a result about 300 concepts with their variants of usages and about 500 precedents were extracted from the textual information. One possibility of the access to the extracted concepts is presented in Fig. 9. Each typical usage of any concept was embedded to the project ontology with its declaration in C#. After developing the expert system the project ontology was refined and included into the created system as its ontology.

As told above all necessary and useful axioms are included into the project ontology also. Any formal expression of any precedent is an axiom binding the definite group of variables indicating the definite concepts.

Each precedent into the project ontology has five variants of these expressions: the textual expression, the predicate formula, the question-answer form, the source code in C# and the executing code. The chosen version of precedent materializations is suitable not only for the automated access by the sailor on duty but for the automatic access of program agents modeling the vessels in the current situation on the sea.

One of these precedents which correspond to the fifteenth rule of MPPSS-72, has the following predicate expression:

if Condition = (Velocity V_1, "keep out of the way")
&& (| Bear_1 - Bear_2 | > 11, 5°)
&& (CPA-DDA- ΔD1 ≤ 0) then
Reaction = Maneuver_Mi.

The precedent (where CPA is a "Closest Point of Approach", DDA is a normative distance between vessels and ΔD is an error of the distance measuring) is included into the article with as demonstration without full explaining the variables and expressions. The expression of this precedent (as the axiom) is included into the ontology of the expert system for the collision avoidance of the sea vessels.

Let us notice that the set of articles of the project ontology (in development process of the expert system) includes not only units for named variables and precedents. The common quantity of project ontology articles (still under refining) was about two thousand.

X. Conclusion

This paper presents the system of techniques for the creation and usage of the projects ontology in the development of the SIS when enormous quantity of project tasks is being solved by the team of designers in the corporate instrumental network. The success of such activity essentially depends of mutual understanding of designers in their specification of conceptualization for solving project tasks and making project decisions. Therefore any project ontology is to be being created as the dynamic subsystem included into the life cycle of the created SIS.

The main suggestion of the paper is the creation of the project ontology as the problem-oriented SIS^{ONT} which is intended for supporting the evolution of understanding and mutual understanding of designers in their step-by-step conceptual activity.

The other important specificity of suggested techniques is the usage of the working dictionary as the preliminary version of the ontology which helps to manage the informational flows and to register the life ways of the informational units and their representativity.

Special attention is given to basic informational units the roles of which are being fulfilled by simple sentences and simple predicates extracted from them. For working with basic informational units the linguistic and logic processors are developed and used. The linguistic processor supports the testing of the statements of project tasks and specifications of project solutions (including requirements and restrictions) on their conformity to the ontology and reality. Arising questions are used for evolving the project ontology.

The logic processor helps to build ontology axioms as predicate formulas. Its experimental research shows that this processor can be (and will be) evolved till the automated creation of the prolog-like description of project tasks.

All interfaces of suggested techniques are adjusted to Russian but only the morphological analyzer and the library of the patterns for two bound predicates are dependent from the specific natural language. The library of patterns for English is created also.

Various and useful techniques of the systematization are embedded into the project ontology for the real time work of designers. Such techniques are accessible both in the ontology component and in the working dictionary.

As the source of the primary information for the creation of the ontology the specialized instrumental system WIQA which supports the usage of question-answer reasoning in the work with project tasks and project solutions is used. Still suggested and developed techniques can be adjusted to the other sources supplying the created ontology by the primary information.

References

[1] H.-J. Bullinger1, J. Warschat, O. Schumacher, A. Slama, and P. Ohlhausen, "Ontology-Based Project Management for Acceleration of Innovation Project," *Lecture Notes in Computer Science,* Vol. 3379, pp. 280-288, 2005.

[2] B. Chandrasekaran, J. R. Josephson1 and V. R.Benjamins, "Ontology of Tasks and Methods," in *Proc. of the Workshop on Applications of Ontologies and Problem-Solving Methods*, held in conjunction with ECAI'98, Brighton, UK, 1998, pp. 31-43.

[3] A.N. Cockcroft, *Guide to the Collision Avoidance Rules: International Regulations for Preventing Collisions at Sea.* Butterworth-Heineman, 2003.

[4] A. H. Eden and R. Turner, "Problems in the Ontology of Computer Programs," *Applied Ontology,* Vol. 2, No. 1, Amsterdam, IOS Press, pp. 13–36, 2007.

[5] A. C. B. Garcia, J. Kunz, M. Ekstrom and A. Kiviniemi, "Building a Project Ontology with Extreme Collaboration and Virtual Design & Construction," *CIFE Technical Report # 152*, Stanford university, 2003.

[6] A. Gelbukh and G. Sidorov, "Approach to construction of automatic morphological analysis systems for inflective languages with little effort," *Lecture Notes in Computer Science*, N 2588, Springer-Verlag, pp. 215–220, 2003.

[7] A. Gelbukh and G. Sidorov, "*Morphological Analysis of Inflective Languages Through Generation*," J. Procesamiento de Lenguaje Natural, No 29, Sociedad Española para el Procesamiento de Lenguaje Natural (SEPLN), Spain, pp. 105-112, September 2002.

[8] A. Gelbukh, G. Sidorov, E. Lavin-Villa and L. Chanona-Hernandez, "Automatic Term Extraction using Log-likelihood based Comparison with General Reference Corpus," *Lecture Notes in Computer Science* 6177, pp. 248-255, 2010.

[9] N. Guarino, "Formal Ontology and Information Systems" in *Proc. of FOIS'98*, Trento, Italy, 1998, Amsterdam, IOS Press, pp. 3-15.

[10] N. Guarino, "Understanding, Building, And Using Ontologies," *Human-Computer Studies*, Volume 46 , Issue 2-3, pp. 293-310, 1997.

[11] N. Guarino, D. Oberle and S. Staab, "What is an Ontology?" in S. Staab and R. Studer (eds.), *Handbook on Ontologies*, Second Edition. International handbooks on information systems. Springer Verlag, pp. 1-17, 2009.

[12] M. Ikeda, K. Seta, O. Kakusho and R. Mizoguchi, "Task ontology: Ontology for building conceptual problem solving models," in *Proc. of*

ECAI98 Workshop on applications of ontologies and problem-solving model, 1998, pp. 126-133.

[13] F. Karray, M. Alemzadeh, J. A. Saleh and M. N. Arab, "Human-Computer Interaction: Overview on State of the Art," *Smart sensing and intelligent systems*, vol. 1, No. 1, pp. 138-159, 2008.

[14] P. Kroll and Ph. Kruchten, *The Rational Unified Process Made Easy: A Practitioners Guide to the RUP*. Addison-Wesley, 2003.

[15] Software Intensive systems in the future. *Final peport ITEA 2 Symposium*, 2006, 68 p. Available: http://symposium.itea2.org/ symposium2006/ main/publications/ TNO_IDATE_study_ ITEA_SIS_in_the_future_ Final_Report.pdf.

[16] P. Sosnin, "Question-Answer Means for Collaborative Development of Software Intensive Systems," *Complex Systems Concurrent Engineering. Part 3*, Springer London, pp. 151-158, 2007.

[17] P. Sosnin, "Question-Answer Expert System for Ship Collision Avoidance," in *Proc. 51th International Symposium ELMAR,* Zadar, 2009, pp. 185-188.

[18] The Standish Group. Available: http://www.standishgroup.com.