QUESTION-ANSWER MODELS OF DECISION-MAKING TASKS IN AUTOMATED DESIGNING

Petr I. Sosnin
Department of Computer Science
Ulianovsk State Technical University
Z-432027, Ulianovsk, Russia
E-mail: sosnin@ulstu.ru

KEYWORDS
Automated designing, Collaborative work, Decision-making, Modeling, Software engineering.

ABSTRACT

The key problem of the successful development of a software intensive system (SIS) is adequate conceptual interactions of stakeholders at the early stages of designing. Success of development can be increased with using of Artificial Intelligence means including models of reasoning. In this paper a number of question-answer means (QA-means) for decision-making (DM) is suggested. The base of such means is a set of the DM tasks for each of which the QA-model with the typical architecture structure is being built and used. Suggested and implemented means are embedded to the QA-processor adjusted for automated designing the SIS.

INRODUCTION

Designing of the Software Intensive Systems too often gives the results which are not corresponding to the planned expectations. The significant number of the SIS developments either are being stopped, or are being exceeded planned time and/or finance, or reach the end in the poorer version (Charette 2005). The degree of success (expressed in percentages of the number of projects, coming to the end according to the initial plans) is extremely low (about 35 %)
The named kind of works is impossible without using the strict technological discipline in the DM acts which are being usually fulfilled collaboratively in the corporate network. It is important to notice that the collaborative decision-making process is a danger source of mistakes caused by misunderstanding and other problems of using the joint intellectual activities. All of these problems decrease the degree of the successfulness of designing the SIS.
Therefore special means are needed to use for rational integrating the intellectual activities of designers in the corporate net during solving the project task including the DM tasks.
It is reasonable to choose useful means for integration from a set of relevant AI means and includes them to the instrument of designing the SIS. First of all the relevance must be estimated in the context of intellectual capabilities such as “consciousness” and “understanding” used for control the consciousness processes.
When we choose or build the relevant AI means we need to remember that the base form of consciousness work revealing is a reasoning and the dialogue (question-answer process in the brain structure) is a nature of reasoning.
One way for modeling collective (collaborative) “consciousness” is to create the question-answer model (QA-model) of collaborative reasoning which is reflecting consciousness process. But for feedback the joint QA-model in any its current state must be open for the interactive “pressure” on the brain of any member of collective “consciousness” if it is useful for solving tasks. In this case individual consciousness will be combined with the interactive “pressure” of the QA-model of collaborative reasoning (collaborative consciousness).
The QA-model of collective consciousness can be used for creating the model of collective understanding which must be useful for checking intellectual activity applied to solving tasks.
This paper presents the QA-model of the DM tasks in automated designing the SIS in the corporate network. Developed question-answer means are included to the technology of designing. They allow the real time integrating of intellectual activities, which can increase the degree of success in the DM and not only in developing the SIS.

RELATIVE WORKS

The problem of rational reasoning into the development process of the SIS is well known. More then 10 years this problem is investigated in the Software Engineering Institute (SEI) of Carnegie Mellon University (Bass and Merson 2005). But the question-answer approach is not used and the problem of “a real time integration of intellectual efforts” is not indicated in interests of the SEI to the schemes of reasoning and their formalizing.
Artificial intelligence means don’t use for supporting reasoning of developers in such well-known technology as Rational Unified Process (Kroll and Kruchten 2003) and in other similar technologies, for example, in Microsoft Solution Framework and Eclipse.
It is very interesting because there are many types of reasoning which are investigated and modeled in AI.

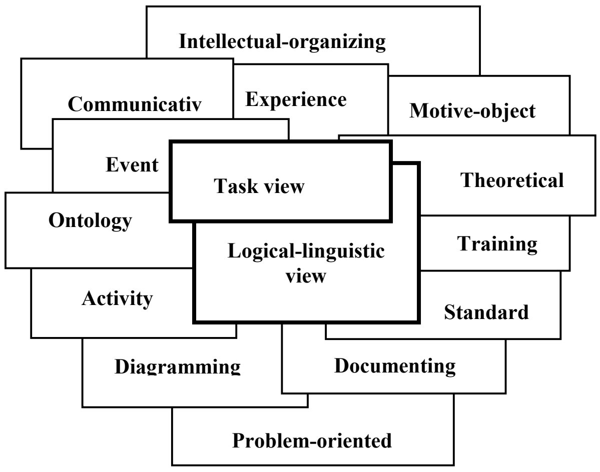
Proceedings 22nd European Conference on Modelling and Simulation ©ECMS Loucas S. Louca, Yiorgos Chrysanthou, Zuzana Oplatková, Khalid Al-Begain (Editors)
ISBN: 978-0-9553018-5-8 / ISBN: 978-0-9553018-6-5 (CD)

For example, the Programs of the European Conferences on AI (ECAI) include about 20 topics connected with modeling reasoning (analogical reasoning, case-based reasoning, common-sense reasoning, reasoning about actions and others).
We have answer to the question "Why AI means don't use in technologies for developing the SIS?"
Adequate AI means which can increase the successfulness of designing the SIS are absent till now because problem-solving and decision-making based on the real time integration of intellectual resources are investigated in AI only partially (different kinds of models for reasoning which are useful in definite classes of design situations, first of all case-based reasoning models).
We are convinced, that investigation of question-answer reasoning is a perspective way for finding the AI means which can give the positive results helping to solve the successfulness problem of designing the SIS.
In the number of relative works using "questions and answers" (or QA), for example, we can mention reasoning in the "inquiry cycle" (Potts et al. 1994) for working with requirements, "inquiry wheel" (Reiff et al. 2002) for scientific decisions and "inquiry map" (Rosen 2008) for education. Similar ideas are used in the special question-answer system which supports development of the SIS (Henninger 2003). The typical schemes of reasoning for the SIS development are presented in (Yang et al. 2003), in paper (Rich and Feldman 1992) reasoning is presented on seven levels of its application together with used knowledge and in (Lee 2000) model-based reasoning is presented as useful means for software engineering.
But in all publications referenced above, issue (Burger et al. 2001) and special report (Hirschman and Gaizauskas 2001) the task of real time integration of the intellectual resources in processes of problem-solving and decision-making is not mentioned.

QUESTION-ANSWER MODEL OF THE TASK

In developing the SIS for each problem of any degree of complexity the concrete developer is appointed for its decision. His colleagues are entered to the solving process in that case when the intellectual help is required for the developer. Externally the general intellectual activity of the developer and its colleagues is observed in the form of their reasoning which can be registered by the useful way.
We suggest use the QA-model (named above) for simulating collaborative reasoning in any state of the decision process for operative including the such kind of models into the intellectual activity of the developer appointed to the task. The QA-model which reflects collaborative reasoning in the frame of the definite task Z(t) we shall name as the QA-model of this task. The QA-model of the task is a model of collaborative reasoning (and integrated consciousness) in the real time process of solving the task.

The QA-model is a systematized representation of reasoning used during the decision of the task Z(t) and kept in the special QA-database. Any QA-model is a set of interactive objects such as "question", "answer" and "task" with the certain attributes and operations.
Therefore specifications of the QA-models will be presented from the interactive system viewpoint or another words as specification of a specialized software intensive system SIS^{QA}. Such position gives the possibility to use the experience of the SIS to the SIS^{QA} first of all the experience of the architecture description.
We defined and investigated the QA-model of the task which is architecturally presented on the Figure 1.



Figutre 1. Architectural Description of QA-Model

The structure and content of the QA-model are being described as a system of architectural views which includes following main views:
1. Logical-linguistic view presents QA(Z(t)) within the frames of logic and linguistics of questions and answers. The visual representation of the view (Figure 2) includes a system of QA-protocols corresponding to the task tree of Z(t). In general case any task Z can include subordinate tasks.

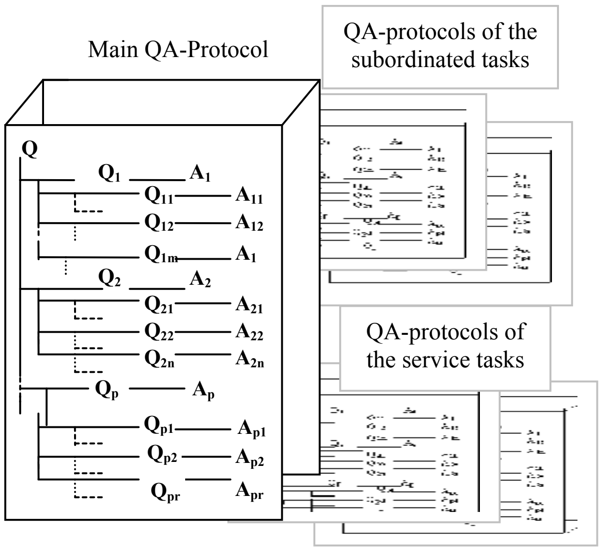


Figure 2. System of QA-Protocols

The logical part of the view describes the hierarchical tree of "questions" and "answers". Each "question" or "answer" is an interactive object with the unique name and rich visual presentation. The QA-tree as a whole consists of the main QA-tree (corresponding to task Z(t)) and a set of subordinated QA-trees for subordinated tasks.

The linguistic part of the view includes a system of texts presented the information content of questions and answers. The text of each unit of the view is a result of QA-formalization which is applied to the definite volume of reasoning at the natural language. Such result can be interpreted as a translation of reasoning from the natural language to the QA-language.

Dynamics of the view reflects the history of reasoning registered step-by-step as the history of each unit of the view. Therefore the logical-linguistic view was named as the QA-protocol.

2. Task view opens QA(Z(t)) as an interactive task tree (Figure 3) including the interactive model of Z(t) with models of all subordinated tasks.

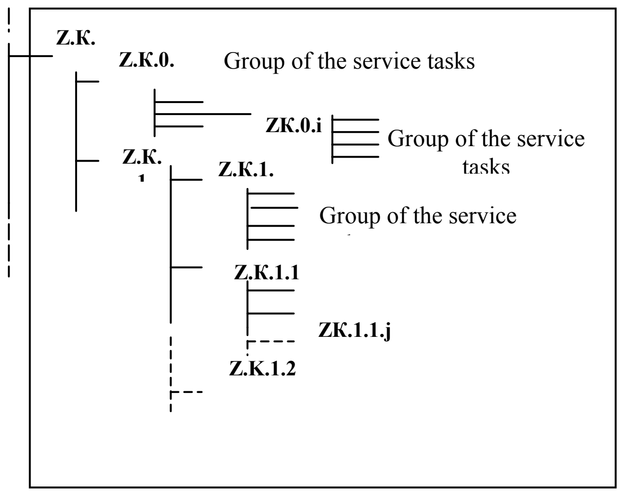


Figure 3. Fragment of the Task Tree for Z=Z.K: K-index name, symbol "0" indicates service tasks, other names indicate the project tasks

Other views are based on the logic-linguistic and task views describing them with different and important point of views. Some of these views will be opened below.

In a basis of integration of views presented above, we shall put the usage of means for modeling of QA-reasoning which reflects the processes of integrated consciousness and integrated understanding at the solution of any tasks.

In integration of views we shall distinguish two interconnected components:

- declarative-visual integration, within the frame of which all artifacts of the QA-model are presented and interactively visualized ;
- procedural integration as the techniques system for QA-modeling.

Declarative-visual integration is responsible for coding, structuring, ordering, storage and delivery by inquiries of those artifacts which are parts of the QA-model. The function of integration of such type we shall assign to the question-answer database of the SIS project. Such solution allows save all QA-models at the file-server accessible for any workplace of the corporate network. It opens the possibility to include in the architecture of the SIS^{QA} two styles – repository and client-server styles.

Visualization of artifacts is another problem for decision of which it is rational to use architectural style MVC (Model-View-Controller) widespread in practice of development the SIS. Architectural style MVC is the style oriented specially on the human-computer interaction.

QUESTION-ANSWER MEANS

The QA-model of the task Z(t) and its system of views are defined as the problem-oriented base of the software intensive system SIS^{QA} designated for supporting the design of the SIS. In general case the task Z(t) is a task Z*(t) solving of which is conceptual designing the SIS. Therefore the SIS^{QA} must be built as a complex of QA-means designated for conceptual designing any SIS with the help of real time creating and using the QA-models of the task Z*(t) and all its subordinated tasks.

The system of QA-means named as QA-processor WIQA (Working In Question-Answers) has been implemented in several versions. Developments of the two last versions were based on architectural views of the QA-model and usage of repository, MVC, client-server and interpreter architectural styles. Moreover in developing the versions have been used object-oriented, component-oriented and service-oriented architectural paradigms. One of the last versions named as NetWIQA has been programmed on Delphi 6.0 and the second version (named as WIQA.Net) has been created on C# at the platform of Microsoft.NET 2.0.

The system WIQA.Net was developed for its usage as a kernel of the product line each unit of which is an application based on this kernel. Product line includes applications "Conceptual design", "System of documenting", "Decision-making system" and "Training system" adjusted for their usage in the corporate network. The NetWIQA was used as a reach source of assets (Sosnin 2004) for developing the kernel.

Server possibilities of the WIQA.Net are opened for thin clients in the corporate network and for Web-clients via the Web-shell programmed with using means of ASP.NET. The component structure of the WIQA.Net is implemented as the open sytem of plug-ins supported the implementation of all architectural views of the QA-model. It is generally (without means reflected and supported Web-access to the QA-means) presented in the Figure 4.

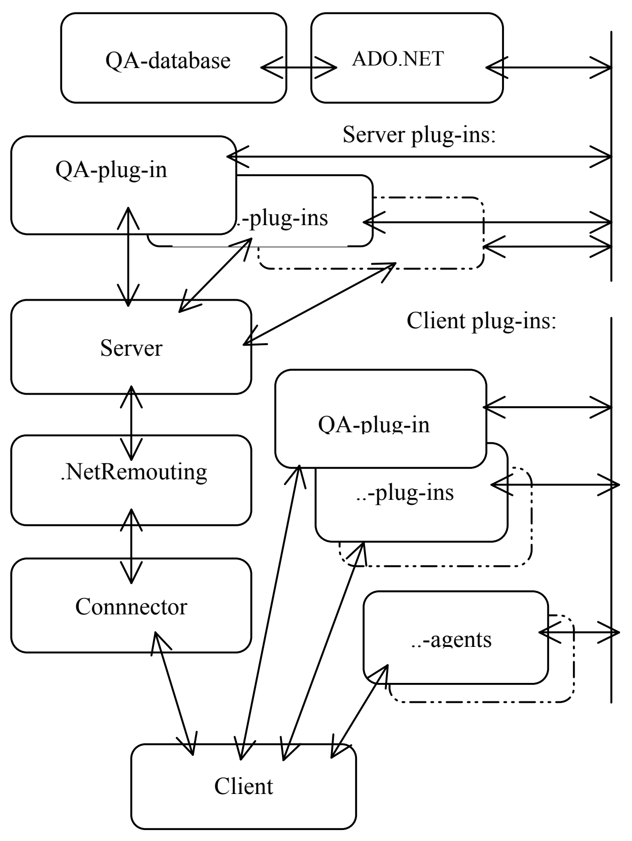


Figure 4. Component Structure of the WIQA.Net

All components are allocated on following tiers:
1. **Database tier** is implemented with using the technology ADO.NET. It is used as a repository of all data.
2. **Server plug-ins tier** encapsulates all function for interaction with the database. The tier is formed dynamically during starting the server with using the technology of a dynamic reflection of types which is supported by a platform .NET. Interfaces of the tier (server plug-ins) contain the functions intended for direct adding, updating and removing of data. These functions are called only by a client tier (client plug-ins).
3. **Server tier** is implemented with using the technology .NET Remouting, that allows address to server from the remote client workplace. When any client workplace is starting the server carries out registration of the client, giving out to it a unique key which is used for checking the access rights to various functions of the system. The server registers also starting of any component of a client plug-ins tier. The server is conducting also logging the most important events of the system (i.e. formation of magazine of events).
4. **Connector tier** (an intermediate tier). Usage of the technology .NET Remouting demands the certainty for all functions of the removed object for access to it. The connector hides the implementation of the server from the client. Therefore the interface of the server is being stored in the connector.
5. **Client tier** is an application with which begins the work a user on the client workplace. This tier is responsible for the program encapsulation of the work with the client plug-ins tier and also for encapsulation of the removed interaction with the server. In case of an allocation of the server and the client on one computer, through a configuration file it is possible to disconnect the usage of the technology .NET Remouting. It will raise the speed of work of the program. The client is also responsible for the reception of the name and the password of the user.
6. **Client plug-ins tier** consists of two parts. One part supports the active interaction with the user and other part is adjusted on the processing of the certain events in the system.
The first part gives the access for the user to the functionalities of the implemented architectural views. Each plug-in of this tier contains a unique key which gives the possibility to distinguish plug-ins of the tier for controlling the access of the user to the appointed plug-ins only. The QA-plug-in of this tier provides the access to commands for the work with the logic-linguistic view (with QA-protocols). Each plug-in of this part of the tier supports the interaction with objects of the corresponding architectural view. The second part of the tier supports the automatic work with a definite set of events during QA-modeling. It provides by the units programmed as agents.
7. **The tier of continuous developing** is activated as a real time access for the developer of additional plug-in (or agent) from the client workplace to the .Net 2.0 means in the context of the current state of the QA-processor. Such additional units are shown in Figure 4 by chain lines.
The Web-access in WIQA.Net is arranged as the one-page site (asp.net) with dynamic additional loading the data, executed on the technology AJAX. At initial loading the user registers the URL address of a resource on which site is placed.

QUESTION-ANSWER MODELING

Question-answer models, as well as any other models, are created “for extraction of answers to the questions enclosed in model”. Moreover, the model is a very important form of representation of questions, answers on which are generated during interaction with the model. Questions are fixed in QA-models in the form of “objects-questions” obviously and implicitly in the forms of ambiguities used in textual QA-units.
The essence of QA-modeling is an interaction of stakeholders with artifacts included to the QA-model in their current states. For such interaction the developer can use the special set of commands (QA-commands), their sequences and a set of plug-ins combining with QA-commands. Such work is similar to programming (QA-programming) on the base of means of the special processor (QA-processor) the role of which fulfils the SIS^{QA}.

In order to get the definite positive effect from the concrete QA-model of the definite task the developer will need to program the definite volume of his interaction with this model and executes the created QA-program.

There are a number of expected positive effects for each of which the needed QA-program (or a set of alternative QA-programs) is rationally built. It has given the possibility to create a library of typical QA-programs accessible to the developer in the real time.

The main subset of positive effects of QA-modeling is connected with the real time integrating of intellectual resources and this subset includes:

- controlling and testing current reasoning of the developer with the help of "integrated reasoning" and "integrated understanding" included to QA-models;
- correcting individual understanding with the help of comparing it with "integrated understanding";
- combining models of collective experience with individual experience for increasing the intellectual potential of the designer on the definite working place;
- including individual experience of the developer in accordance with request and through evolving of QA-models in the designing on the other working places in the corporate network..

Any developer can get any programmed positive effect with the help of QA-modeling as "answer" on question actually or potentially included in the QA-model (figuratively shown on Figure 5).

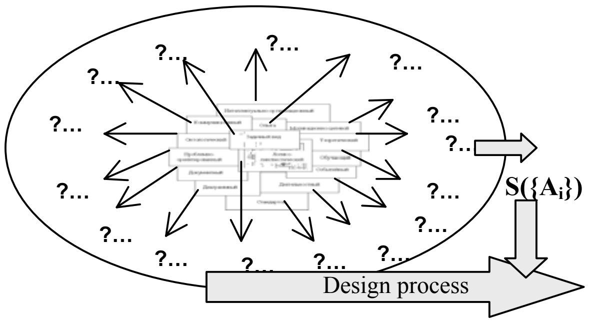


Figure 5. QA-Model as a Source of Answers

As it shown in this scheme any view is a source of answers accessible for the developer as results of his interactions with the QA-model. There are different forms for building answers with the help of QA-modeling, not only linguistic forms.

The developer fulfils the positive interaction in the frames of project, service and technological tasks of the designing process.

All project tasks $Z^P=\{Z^P_r\}$ are derived from the process described above. Any task Z^P_r is a question qualified by stakeholders as a task-question answer of which can be found only through the decision process.

Any service task Z^C_m has its QA-pattern kept in the special library. Such pattern helps to build the model $QA(Z^C_s)$ for the definite conceptual artifact. Service tasks are defined and implemented for creating documents and visual diagrams, for supporting the typical schemes of communicating and training.

The work with questions, answers and other conceptual artifacts are executed with the help of technological tasks $Z^T= \{Z^T_n\}$ solving of each of which uses a scenario form. The solving of the definite technological task is an executing of the corresponding technique programmed for the QA-processor.

The main application of the QA-processor is a "Conceptual design" the process of which is based on workflows "Interactions with Experience". Each workflow is implemented on the base of the corresponding technological tasks.

To demonstrate the functions of technological tasks let us name a few of them, e.g. "To solve a task", "To answer a question", "To construct the answer", "To prove the answer", "To discuss the answer", "To fill-in the pattern", "To set the requirement", "To determine the front of work", "To plan", "To generate the educational unit", "To control knowledge". The given utilities show that some applications of the QA-processor support also the operative training for the members of a design team.

QUESTION-ANSWER PROGRAMMING

As told above means of the QA-processor gives the possibilities for special kinds of QA-programming and executing the QA-programs.

The first kind is similar to event-driven programming when the state of the QA-model is interpreted as the description of situation used for the next steps of designing. If any question or answer of the QA-model is under constructing then such unit is the object of the potential work. Analyzing the state of the QA-model (more truly "event view" of the model) we can choose its more priority question or answer for continuing or finishing the work with such interactive object ("activity view" of the QA-model).

In any state of any question or answer the developer can activate useful interaction with the chosen object using accessible QA-commands, plug-ins or techniques. Such actions are similar to the work of the QA-processor as "an interpreter".

The second kind is used for simulating the technique steps in the QA-form where "questions" are used for coding steps of the technique and "answers" are used for registering facts of executing steps of the technique. Any step of the technique as a definite volume of the work is being presented as "question" for which is needed to build "the answer".

There are many different types of the QA-commands, plug-ins and techniques used in designing with the help of QA-modeling. Any designer has the possibility for using a set of base techniques for designing, decision-making, communicating, documenting and training.

A set of especially important actions (operations, commands, plug-ins mechanisms, techniques) of the QA processor includes:

1. For questions: detection of obvious questions (on their indicators), predication (through translation on Prolog-like language), identification (on patterns), concrete definition (for types), assignment of meanings to attributes (as to the phenomenon of event type), argumentation of question.

2. For answers: creation, assignment of a type, change of a type, registration of a condition, editing of the contents, assignment of meanings to attributes (as to the phenomenon of event type), argumentation.

3. For QA groups: transformation to the node, expansion into the QA-structure, transformation to the event net, visualization of a network, analysis of a condition, choice of a direction of development, scrolling of dynamics (on inquiries).

4. For text: creation, transformation, grammar analysis, semantic analysis, transformation to the semantic graph, visual analysis, supporting of a phenomenon of attention.

Programs are created for their executing. For QA-programs three version of executing are used. The first version is happen during creation of any part of any QA-program. The first executing of each part of the QA-program is combined with the creating of this part.
The second version is using the "line" of the QA-program for controlling the design process by the declarative or procedural way.
The third version is the visual "pressure" chosen parts of QA-programs on the intellectual activity of the designer. The developer can include several parts from the different QA-program into the request for visualizing. There is a special subsystem in the QA-processor which supports the preliminary preparation of the visualizing information from the QA-database. Any prepared block of questions and answers can be visualized in the suitable temp by switching special windows on the screen or using effects of the 25-th card.

QA-PROGRAMMING IN DECISION-MAKING

There are many different types of DM methods. Choosing methods suitable for designing the SIS we need to take into account the specificity of such kind of activity. Therefore for certainty in QA-programing we are being limited the interactive decision-making, decision-making in group, decision-making by experts and also analysis models of situation and alternatives.
In order to use the possibilities of the QA-processor the typical task is defined for each DM method suitable for designing the SIS. Moreover each typical task is presented as the corresponding technique of the DM coded in the form of the QA-program. The QA-programming of the chosen DM tasks showed that decision-making by the technique of the expert system has specicfity in implementing and executing.
At first we present the general version of QA-programming and then QA-programming for the expert DM task. For example, the QA-code of the technique for the Plus-Minus-Interesting analysis or PMI-analysis (Harris 1998) is presented on the block (Figure 6):

Question-Answer Template File
"The scheme of PMI management"
Q1. Specifying a problem of decision-making?
Q2. Creating a subtask of decision-making (PMI)?
Q3. Creating a subtask of documenting of decision-making?
Q4. Loading a pattern of the PMI method?
Q5. Identifying of problems and the purposes in a pattern?
Q6. Forming the list and an estimation of pluses?
Q7. Forming the list and an estimation of minuses?
Q8. Forming the list and an estimation of interests?
Q9. Loading of the module of gathering and calculation of estimations?
Q10. Calculation of the general estimation?
Q11. Analysing result and decision-making?
Q12. Creating documents

Figure 6. QA-Model of the Technique

The answers in the template are absent till its using. After loading this template in the task tree it will receive the status of "the QA-model" for the new subordinated PMI task and for each question of this model the answer will be opened. When the technique of the PMI task is executed the symbol "*" is added to the answers for questions which have been fulfilled as points of technique.
Usage of value "*" in answers gives the possibility for breaking the technique in any position and returning to the next point of the technique in suitable time. Moreover it gives the possibility for tracing the way of executing the technique with any executable structure.
Definite documents are used in the DM tasks and in presented PMI task also. For working with documents the QA-processor allows to use the special class of service tasks. Each task of such class also uses the QA-template and the corresponding QA-model. Such possibility is reflected in Figure 1 as "Documenting view".
For the PMI task its base document has the following table structure:

Table 1: Table Structure

Positive		Negative		Interesting	
Plus	Value	Minus	Value	Interest	Value

The document table structure is being transformed to the corresponding QA-protocol the potential fragment of which is presented on Figure 7.

…………………………………………
QJ.1. What is the first plus from the decision?
AJ.1.Xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx.
QJ.1.1. What importance of the first plus (+1 till +10)?
AJ.1.1. +6.
QJ.2. What is the second plus from the decision?
AJ.2.Yyyyyyyyyyyyyyyyyyyyyyyyyyyyy.
QJ.2.1. What importance of the second plus ?
AJ.2.1. +8.
……………………………………………
QK.1. What is the first minus of decision (-1 till -10)?
AK.1. -5.
……………………………………………..

Figure 7. Fragment of the Estimation Results

Externally simple structure of the both QA-models hides the rich possibilities of their transforming, processing, and visualizing which are caused by the system of their architectural views.
Moreover the PMI task and the corresponding task of its documenting are included to the task tree of designing the SIS and all tasks of this tree have similar QA-models and are being controlled by unified rules.
Different kinds of relations between tasks are used in the task tree of designing the SIS. Relations been possible for the DM tasks are presented on the Figure 8 where tasks of the definite type are marked by labels.

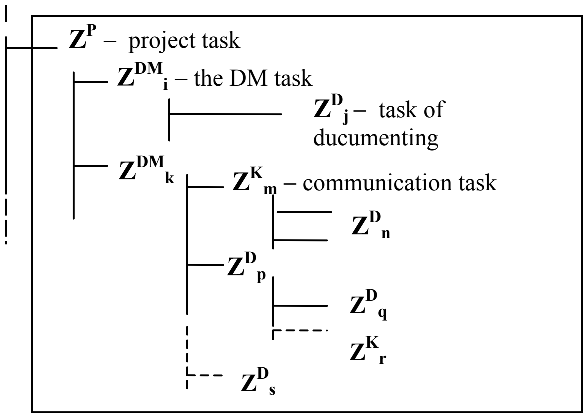


Figure 8. Examples of Relations between Tasks

In accordance with circumstances the developer can open for solving:

- the DM task Z^{DM}_i subordinated to the project task Z^P;
- the task of documenting Z^D_j subordinated to the DM task Z^{DM}_i;
- the communication task Z^K_m subordinated to the DM task Z^{DM}_k;
- the task of documenting Z^D_q subordinated other task of documenting Z^D_p;
- the communication task Z^K_r subordinated to the task of documenting Z^{DM}_p.

The developer can include to the any point of the task tree any task of any type if it is needed for him. The necessity of decision-making can be requested in any time of designing the SIS. If such event the appropriate method of decision-making as a special task will be loaded in the task tree of designing the SIS.
If it will be useful for the developer he can include to the needed point of the task tree the suitable communication task supported by the QA-processor (special e-mail, the task of collective estimating, the task of argumentation or the brainstorming task). Estimations into the DM tasks are being fulfilled with the help of the communication tasks. For all tasks of communication the QA-templates are created and loaded in the common library of QA-templates. A set of communication tasks supports "the communicative view" of the QA-model.

EXPERT SUBSYSTEM FOR DECISION-MAKING

Into the complex of QA-means for decision-making the expert subsystem is included. Such subsystem is implemented as plug-ins of the QA-processor for supporting "the experience view" of any QA-model.
"The knowledge base" of the expert subsystem gives the possibility to keep precedents of two types – declarative and procedural precedents each of which includes the production rule combined with the QA-model of this rule. The fact of storage together with precedent of its QA-model opens additional (to known) means for checking adequacy of precedent and specification of its version for using the precedent in a situation of a choice. Such QA-model can be interpreted as a DM task positive solution of which confirms the relevance of the corresponding precedent. Moreover a set of keys can be appointed to the result of solving the DM task for potential working with precedents composing other precedents.
The relations between the expert DM task Z^E_j and subordinated expert DM tasks { Z^E_{jk} }in the task tree of designing the SIS are demonstrated on the Figure 9.

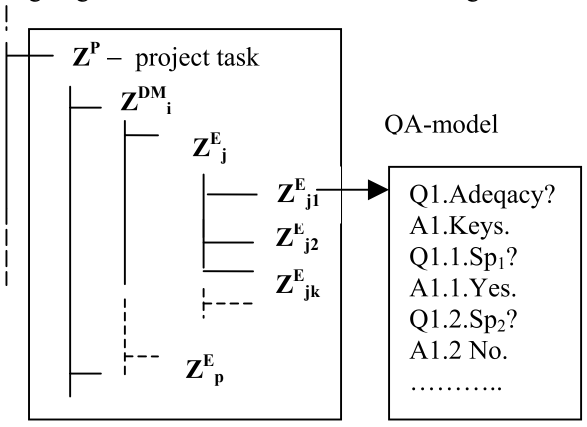


Figure 9. Examples of Relations between Expert Tasks

For working with the expert DM tasks in the WIQA.net following sequence of action can be used:

1. The statement of the project task (for example task Z^P on Figure 9) is transformed to the list of keys for accessing to the knowledge base.

2. The list of keys is used for interacting with the expert subsystem and forming the list of potentially relevant precedents.

3. The expert DM task (for example task Z^E_j) is being created for each potentially relevant precedent and this task is being loaded in the task tree.

4. If the task Z^E_j has a compound type its subordinated task are being loaded in the task tree also.

5. Decision-making the loaded task is being started with subordinated task if they are.

6. If the solution of the subordinated task (for example task Z^E_{j1}) confirms relevancy then keys of this task (answer A1) are being included to the common list of keys.

7. Changing the list of keys is a cause for the next access to the knowledge base.

8. The process of the expert decision-making is being finished when all loaded DM tasks are solved.

The general rules of working with the expert DM task after its loading to the task tree of designing the SIS are fulfilled as working with the others tasks in this tree. The expert DM task can be interrupted in any time with returning to the solution process in the suitable time. The useful service or technjlogical task (any the DM task even) can be subordinated to the expert DM task.

CONCLUSION

This paper presents system of QA-means for decision-making in designing the SIS with the help of QA-modeling. The base unit for QA-modeling is a task of any type which is solving during designing. Each type of the tasks has a specifity which is need to take into account in QA-modeling. For any DM task its technique is known usually. Such specifity gives the possibility to create a set of the QA-programms for the typical DM tasks and loads it to the special library of QA-templates. The designer can use the needed DM-template in any time when it will be useful for designing.
QA-modeling of the DM tasks opens a numer of positive effects such as: real time integrating the intellectual activities of the DM task solvers; effective monitoring of a DM-process; analysis of opportunities of parallel coordination of work in a design team; demonstration (at a suitable speed) of the former steps of the DM; personification of events for subsequent definition of authorship and contribution of the members of a design team.
When the QA-processor is used the DM tasks are included to the task tree of designing the SIS in accordancwe with general rules. Proposed means have confirmed the practical usefulness in development of a number of the SIS, including "The automated control system for the sea ship" (the expert DM-task on the base of the collision avoided rules).

REFERENCES

Bass L.; Ivers J.; Klein M. and P. Merson. 2005. "Reasoning Frameworks," Software Engineering Institute, Carnegie Mellon University, Pittsburgh, PA, Tech. Rep. CMU/SEI-2005-TR-007.
Charette R.N. 2005. "Why software falls", IEEE Spectrum, vol. 42, #9, 36-43.
Burger J. et al. 2001. "Issues, Tasks and Program Structures to Roadmap Research in Question & Answering (Q&A)," Tech. Rep. NIST.
Harris R. 1998 "Introduction to Decision Making" available at http://www.virtualsalt.com/crebook5.htm
Henninger S. 2003. "Tool Support for Experience-Based Software Development Methodologies," Advances in Computers, vol. 59, 29-82.
Hirschman L.and R. Gaizauskas. 2001. "Natural Language Question Answering: The View from Here".Natural Language Engineering, vol. 7, 67-87.
Kroll P. and Ph. Kruchten. 2003. "The Rational Unified Process Made Easy: A Practitioners Guide to the RUP," Addison-Wesley.
Lee M.H. 2000. "Model-Based Reasoning: A Principled Approach for Software Engineering", Software - Concepts and Tools,vol.19, #4, 179-189.
Potts C.; Takahashi K. and A. Anton. 1994. "Inquiry-based Requirements Analysis, " IEEE Software, vol.11, #2, 21-32.
Reiff R.; Harwood W. and T. Phillipson. 2002. "A Scientific Method Based Upon Research Scientists' Conceptions of Scientific Inquiry," In Proc.2002 Annual International Conference of the Association for the Education of Teachers in Science, 546–556.
Rich C. and Y. Feldman. 1992. "Seven Layers of Knowledge Representation and Reasoning in Support of Software Development," IEEE Transactions on Software Engineering, vol, 8, # 6,.451-469.
Rosen D. J. 2008. "How to Make Inquiry Maps". Available : http://alri.org/pubs/im3.html
Sosnin P. 2004. "Question-Answer Processor for Cooperative Work in Human-Computer Environment", In Proc. the 2-d International IEEE conference Intelligent System, Bulgaria, 452-456.
Yang F.; Shen R. and P. Han. , 2003. "Adaptive Question and Answering Engine Base on Case Based and Reasoning Technology," Journal of Computer Engineering, vol.29, #11, 27-28.

PETER SOSNIN was born in Ulianovsk in the USSR, on July 12, 1945. He graduated from the Ulianovsk Polytechnic Institute (1968).
His employment experience included the Ulianovsk Polytechnic Institute and Ulianovsk State Technical University. His special fields of interest includes AI applications for computer aided design. P. Sosnin defended doctor degree in Moscow Aviation Institute (1994). He is an author of eight books and more one hundred articles.