information

MDPI

Article

# Substantially Evolutionary Theorizing in Designing Software-Intensive Systems

**Petr Sosnin**

Computer Department, Ulyanovsk State Technical University, Severny Venetc, 32, 432027 Ulyanovsk, Russia; sosnin@ulstu.ru; Tel.: +7-951-095-4466

Received: 6 February 2018; Accepted: 5 April 2018; Published: 13 April 2018


**Abstract:** Useful inheritances from scientific experience open perspective ways for increasing the degree of success in designing of systems with software. One such way is a search and build applied theory that takes into account the nature of design and the specificity of software engineering. This paper presents a substantially evolutionary approach to creating the project theories, the application of which leads to positive effects that are traditionally expected from theorizing. Any implementation of the approach is based on a reflection by designers of an operational space of designing onto a semantic memory of a question-answer type. One of the results of such reflection is a system of question-answer nets, the nodes of which register facts of interactions of designers with accessible experience. A set of such facts is used by designers for creating and using the theory that belongs to the new subclass of Grounded Theories. This sub-class is oriented on organizationally behavioral features of a project's work based on design thinking, automated mental imagination, and thought experimenting that facilitate increasing the degree of controlled intellectualization in the design process and, correspondingly, increasing the degree of success in the development of software-intensive systems.

**Keywords:** conceptual designing; interaction with experience; model of precedent; question-answering; semantic memory; software engineering; theory of project

## 1. Introduction

The extremely low degree of success in the development of software-intensive systems (SISs) is an essential reason for searching for new approaches to designing in this subject area [1]. One set of such approaches can be connected with taking into account both software intensity and the intensive use of knowledge and experience in conditions of human-intensive systems. Additional intensities are directly related to the designing of systems because, in such a work, the team of designers has to creatively use the workflows with complicated toolkits that also belong to the class of SISs. The integral result of these intensities is a high complexity of the environment with which the designer is forced to interact.

The system or its any component is complex if the designer (interacting with the system) does not have sufficient resources for the achievement of the necessary level of understanding or achieving the other planned aims. If some of the necessary resources are not available, designers should create them in the design process, and in this kind of work, it would be useful for them to use appropriate scientific approaches based on experiments and theorizing, especially at the conceptual stage of activity.

The following reasons cause such waiting:

1. At this stage, designers are active in an operational space (OS), including all concerned components of involved reality, and in a conceptual space (CS), components of which are generated with intensive use of mental actions in the real time of designing. Switching between

these spaces, designers must reflect regularities of OS onto their conceptual descriptions in an adequate manner;

2. In the interactions between designers and the complicated surrounding, facts of understanding are embodied in conceptual forms taking into account their coordination. In similar conditions, scientists use relevant theories.

However, in software engineering, the theoretical base is very poor. Among the last attempts to find the General theory of this subject area, the initiative SEMAT (Software Engineering Methods and Theory, 2009 year) occupies the central place [2]. The primary intention of this initiative is bound with the re-shaping of software engineering, and references on the "Theory" in the denotation of the initiative indicates that such a theory was not available at that time. As will be shown below, the General Theory of software engineering is absent until now.

The investigated versions of theorizing have shown that there are many different points of view on the construction of theories for applications with software, and the ongoing attempts of such constructions can lead to the General Theory. One such effort is presented in this paper.

This paper presents a substantially evolutionary approach to creating the applied theories of SIS-projects. In the foundation of this approach lies a specialized framework, the structure, and semantics of which are defined by a reflection of an operational space of designing onto a semantic memory of a question-answer type [3]. Such reflection is based on question-answer reasoning (QA-reasoning) and its protocoling, which is a valuable source of facts disclosing the interactions of designers with experience and its models in realizing certain projects.

Having such a source of facts registered in an organizationally behavioral environment, we have mastered the use of question-answering in necessary actions of designers at the conceptual stage of designing. On this basis, we were ready to semantic processing the textual content of facts and their compositions. Moreover, we have built a specialized toolkit Working In Questions and Answers (WIQA) reflecting the operational space on a conceptual space and the use of its components including their processing. Below, we will describe how got experience and the toolkit help to create and use a theory of the corresponding project in parallel with designing the system with software.

## 2. Materials and Methods

### *2.1. Preliminary Bases*

Over the last 15 years, our research has focused on the use of question-answering in the conceptual design of software intensive systems. This choice was due to the following reasons:

1. Lack of sustained progress in enhancing the success in designing of SISs despite numerous attempts to change the painful state of affairs.
2. Our understanding that:

   - A very important source of reasons affecting the design process is the human factor that manifests itself in the designer's interactions with the accessible experience and its computerized models.
   - Interactions with experience are based on what people call "questions" and "answers".

At the first stage of this research, we oriented on

1. The availability of regular and long-term statistical reports containing information on the positive and negative factors that influence the success in designing the SISs (reports of the Corporation Standish Group in the first place [1]).
2. Workflows of conceptual activity applied in well-known technologies providing the development of systems with software (workflows of the technology "Rational Unified Process" [4]).

Particularly valuable in indicated reports of Standish Group are ranked lists of reasons contributing to the success of projects or adversely affecting its achievement. The components of such lists are useful tips for trying to contribute to the solution of the problem of success, and such hints were taken into account in our studies, the history of which is shown in Figure 1.

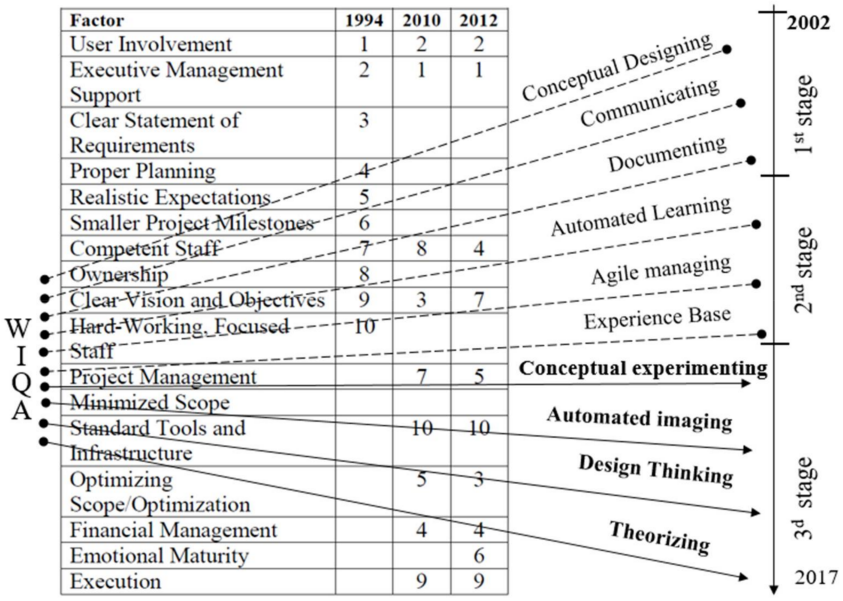


**Figure 1.** Retrospection of mastering the question-answer (QA)-nets.

The greater part of work in our research and development was fulfilled for project organization. More than 1000 employers created the automated systems (AS) including hardware, software, and peopleware. These systems are a kind of SISs that inherit all problems of developing the software systems. Moreover, in developing the AS, designers meet additional problems caused be coordinated combining components of indicated types of "ware". However, in the conceptual space, designers can use similar models and actions.

Therefore, a common feature of all research directions presented in this figure is the use of reflecting an operational space OS of designing onto a semantic memory of a question-answer type (QA-type). Such reflection (QA-reflection) leads to creating the conceptual space CS, in which the designers can work with conceptual models of chosen components of the operational space—for example, to work with a structure of project tasks in their current state or with a model of a team of designers. The generalized scheme of the reflection is shown in Figure 2, where the main components of the OS are called and symbolically designated for reuse in this article.

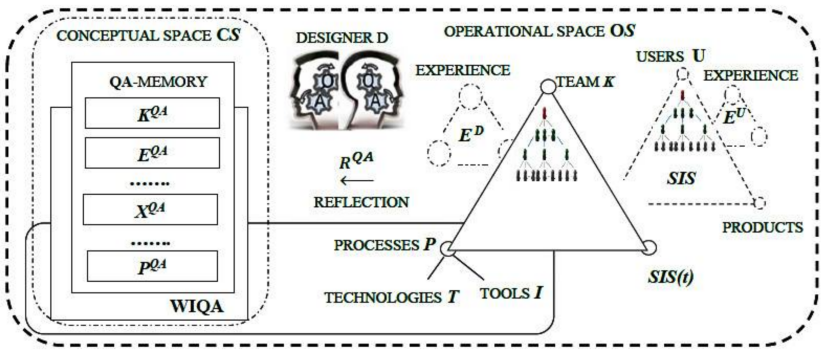


**Figure 2.** The common view on reflection.

The scheme also underlines the reflection accompanying the process of designing the SIS(t) in its life cycle, the dynamics of which can be written by the following expression:

$$OS(P,\ K,\ E^{D},\ E^{U},\ SIS_i(t),\ \ldots,\ X,\ t) \overset{R^{QA}}{\rightarrow} CS(K^{QA}(t),\ E^{QA}(t),\ P^{QA}(t),\ SIS^{QA}(t),\ \ldots,\ X^{QA}(t)),$$

where $R^{QA}$ is a reflection process that provides generating the space CS in its current state, including models for components designating in the space OS and models for other necessary components $X^{QA}$. This process is based on real-time interactions of designers with accessible experience when they use workflows "Interactions with Experience" embedded into the toolkit WIQA [5]. Such works leave their evolutionary traces on current states of experience and models of its items.

The development of the toolkit WIQA, including its workflows, was based and continuously improved in the context of our understanding a "question" as an artificial phenomenon that appears in the mind of a certain person who implicitly or implicitly tries to apply natural experience. To control the access to the experience, the person expresses this phenomenon by the use of appropriate signs of the naturally professional language. Thus, verbal expression of the "question" is no more than its sign model that includes "keys" for searching and/or creating an appropriate answer. A transition to the verbal models discloses the possibility for the use of question-answer reasoning (QA-reasoning) in interactions of designers with the accessible experience. In this sense, "task" is a type of "question", any usage of which finds its material embodiment outside the brain in the form of models, among which a special place is occupied by a model such as "a statement of the task".

This position is partially detailed in Figure 3, which implicitly indicates an organizationally behavioral nature of the activity of designers when they should solve the tasks in the conceptual space that are located in QA-memory.

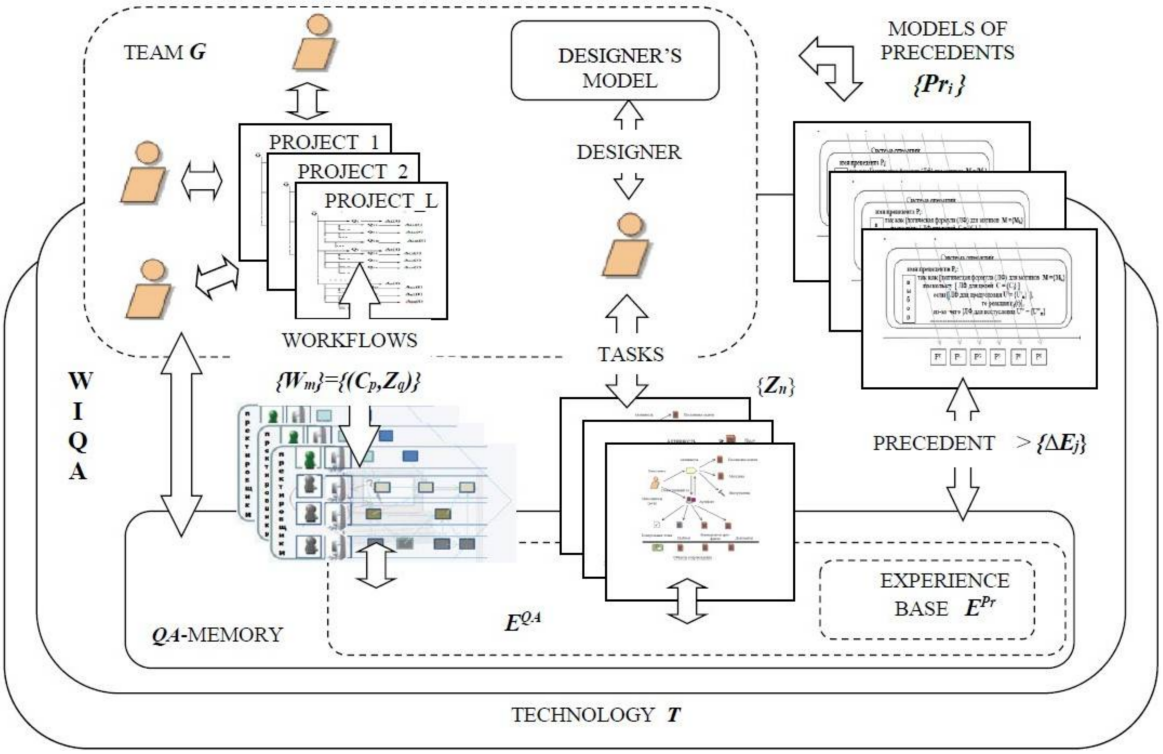


**Figure 3.** Some details of operational space.

Let us note that any component of the space CS is a semantic model built on the base of QA-reasoning registered in the semantic memory of the toolkit WIQA in forms of the question-answer net (QA-net) or a coordinated composition of such nets. In the course of our research and the

development of the toolkit WIQA, QA-nets were continuously improved, and their potential was enriched.

### *2.2. Question-Answer Approach*

At the first stage of our study, a question-answer approach (QA-approach) occupied the central place. Our understanding and intentions caused its invention and embodiment.

1. If in solving any task the designer will register interactions with experience in forms of visual protocols of QA-reasoning, then such protocols (QA-protocols) can be used for activating the similar process of interactions with experience when it will be useful for the designer (for example, for correction of semantic errors or results of understanding obtained on the basis of registered interactions with experience).
2. QA-protocols registered during the solution of corresponding tasks are very useful models of these tasks (QA-models of tasks) as in the design process, so in its results.
3. In designing the SISs, QA-protocols should be combined and visualized in forms that are similar forms of combining and visualizing files in File Manager Systems such as FAR or Total Commander.

In the toolkit WIQA, its kernel is a complex of methods and means that provide creating and using QA-models for project tasks solved at the conceptual stage of designing the SISs. Among methods used in the embodiment of QA-approach, stepwise refinement fulfills the important role as in work with project tasks so in describing their statements.

In the general case, the life cycle of the projects begins with an innovative idea that should be evolved until the initial statement of the root project task $Z(t_0)$, from which the designers can start to apply the stepwise refinement that (step by step and level by level) leads to a hierarchical system of the root task and subordinated tasks. In the CS-space, designers register the generated results of stepwise refinement with the use of artifacts that are shown in Figure 4.

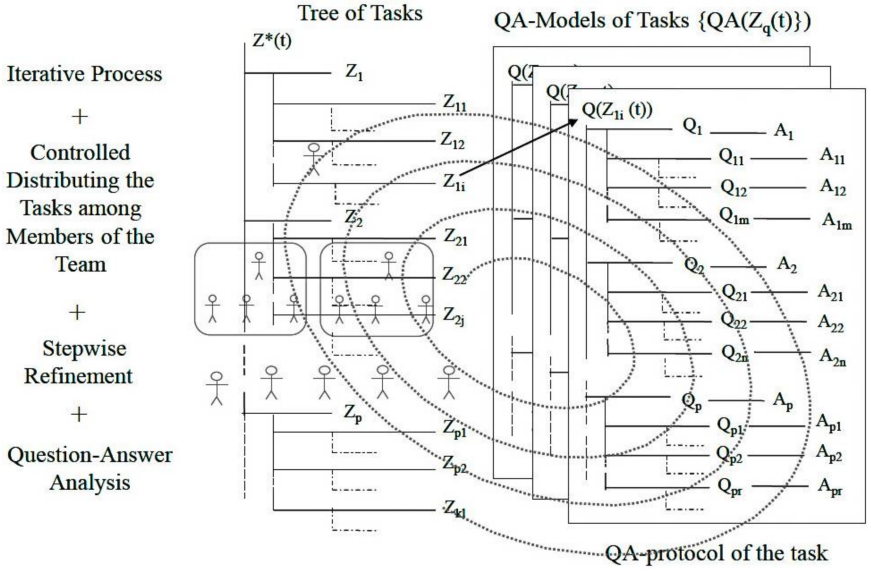


**Figure 4.** Creating the tree of tasks for the project.

The scheme in this figure indicates the essential features of QA-approach, and it includes the tree of project tasks with QA-models for all of them. Any of these artifacts is an example of QA-nets. The primary applications of the toolkit WIQA are iterative creating of shown QA-nets, control distributing the tasks of the tree in its current state among members of the team, and solving the tasks using the stepwise refinement based on the question-answer analysis. It developed the semantic

memory with its cells to provide these applications. For access to QA-nets uploaded onto this memory, the toolkit WIQA includes the specialized subsystem, the functions of which are similar to the functions of Management Database Systems.

Thus, each of the constructs Z*(t) or $\{QA(Z_q(t))\}$ is visualized as a tree of nodes whose representation in the semantic memory of the WIQA environment can be interpreted as a set of coordinated QA-nets. Each QA-net is a system of its nodes, any of which is accessible as an interactive object (QA-object) uploaded in QA-memory. The typical structure of the cell and the uploaded QA-object are shown in Figure 5.

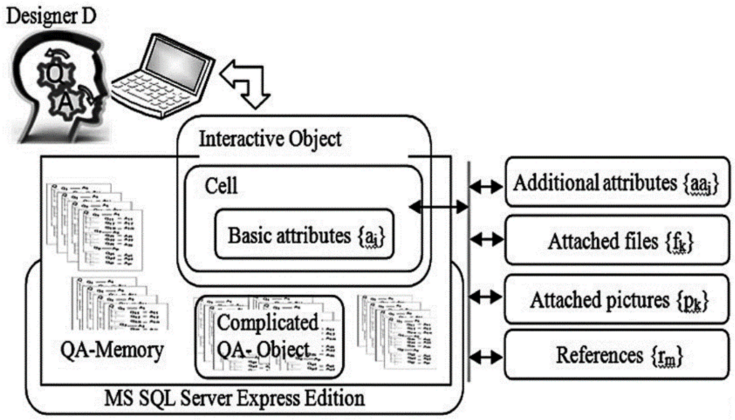


**Figure 5.** The typical structure of a QA-object.

In the most general case, the memory cell must provide an information store that conceptually specifies the task located in the corresponding tree. Therefore, in specifying the cell, this requirement has led us to a set of basic attributes, one of which is intended for registering a textual part of a description of the task, while the necessary tables and graphics components of the task description are attached to the cell as files. Some of the other basic attributes will be clarified below. Here, we note that, except for the basic attributes, designers can define additional attributes and references if it is useful for the object uploaded to the cell.

In any case, cells of QA-memory are intended to upload QA-objects. Such objects can be simple if each of them is uploaded in one cell, for example, representations of "questions" of different types (Q-nodes), including questions like "task" (Z-nodes), and "answers" (A-nodes) to "questions". When an interactive object is located in a set of the bound cell, it is a complicated QA-object. Each QA-object is associated with its representation in the semantic memory and to the system of operations, including operations for visualizing the objects.

Cells of QA-memory can be used for uploading units of data that have appropriate structure content interpreted in the question-answer sense—for example, "name of objects—the question" and "some of its attributes—answer". Such simulating was used including the plug-ins "Organizational structure" that is intended for reflecting the features of the designers' team onto QA-memory in the form of the corresponding QA-net. The main function of this plug-ins is to provide the control distributing the tasks among members of the team in designing the definite SIS.

The similar way was used for sets of designer practices, behavioral operations of which were simulated using the following interpretations: "name of behavioral operation—the question" and "execution of this operation by the designer—answer". Such simulating led us to behavioral programs that can be included in the work of designers in the WIQA environment. Such work is shown figuratively in Figure 6, where one can see some QA-nets.

The figure shows

1. QA-tree of the organizational structure (team K), which relates organizational units (the symbolic designation of groups G) onto any depth of their relations with the members of the collective

(symbolic designation D). Due to additional attributes, the net nodes can obtain extended semantics, as well as connectivity, which makes it possible to present an organizational structure of any type.

2. QA-nets simulating workflows:

   - Technologies used in the process of conceptual design of the SIS being developed;
   - Business processes that will be implemented in the SIS after its commissioning.

3. Question-answer forms for encoding systems of methods (practices) S(WIQA, $\{M_n\}$) used in the technological workflows "Interaction with Experience", serving the application of the toolkit WIQA.

It should be noted that in the encoding of techniques, QA-objects were used to represent their $\{C_q\}$ commands, which were executed by designers (reactions $\{R_q\}$), who interact with the method steps visually. In this interaction, the designers acted as a "processor", which also responds to the GOTO $C_{iu}$ transition commands.

Taking into account QA-nets shown in Figure 6, as well as the structure of the memory cell, we will clarify the specificity of the author's approach (QA-approach) to the reflecting the operational space of designing the SIS onto the semantic memory (QA-memory).

1. The dynamics of QA-approach is represented by the current state of the tree of task Z*(t) that is built by the team K* of designers $\{D_{vs}\}$ with the use of the stepwise refinement applied to the initial statement $St(Z^*, t_0)$ of the root task Z*.
2. For each node $Z_i$, expressed in the tree $TT(Z^*, t)$ by the statement $St(Z_i, t)$ of the corresponding task Zi, the designer (responsible for this task) also performs question-answer analysis (QA-analysis) in the form of the stepwise refinement, and this analysis leads to the question-answer model (QA-model, $QA(Z_i, t)$) of this task.
3. For solving the task $Z_i$, the designer can use the technological tasks $\{M_n\}$ visualized in versions of behavioral QA-nets of commands.
4. The tree Z*(t) and the result of QA-analysis for each task of the tree are recorded in a special database (question-answer base, QA-base located in QA-memory) in hierarchical forms, any of which is visually accessible as a whole, and at the level of its nodes (any node or a group of nodes).
5. Tasks of the tree Z*(t) are distributed among designers of the team, the organizational structure of which is also registered in QA-base as QA-nets in conditions shown in Figure 6.

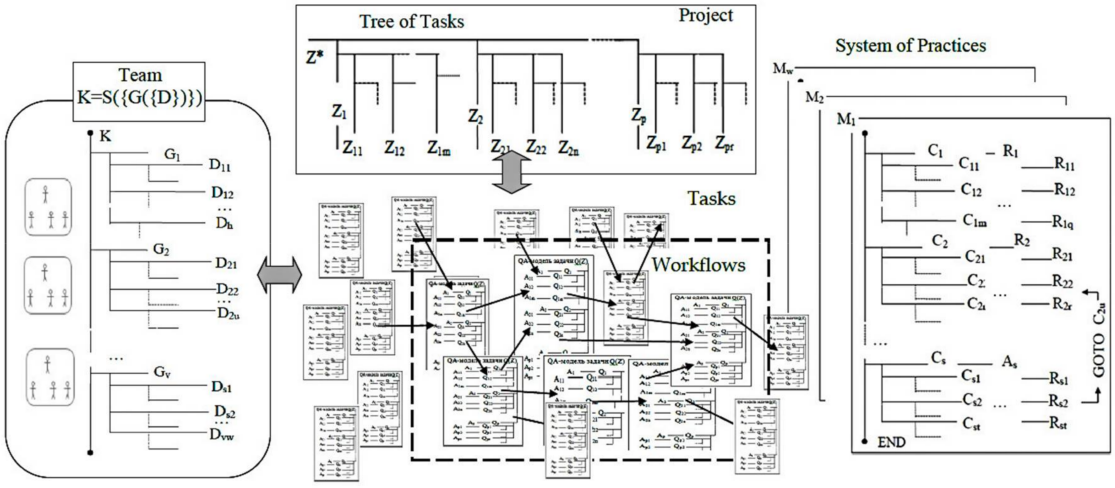


**Figure 6.** QA-nets of design process.

At the first stage of our research, the behavioral programming in the WIQA environment was applied for learning the operators of hardware complexes with software components. In this application of the toolkit, QA-nets were used for structuring the complex description, behavioral programming the operator's actions, and testing the results of learning.

One more extension of the toolkit WIQA in applications of QA-approach was bound with the project documentation. As mentioned above, a semantic representation of the cell, and hence of the corresponding QA-object, can be expanded by connecting the necessary set of additional attributes. At the first stage of our research, the very first version of such an extension was its use in solving the tasks of normative documentation. For example, in the practice of documenting, the designers should fill the sections of document templates with information, the certain blocks of which are identical in different documents (these blocks bind documents). In the WIQA-environment, the tasks of documenting are included in the tree of tasks by designers who use QA-nets for coding the templates of documents and define additional attributes for automatic support of the identical content in the necessary blocks of documents.

In concluding this subsection, Figure 7 demonstrates the basic interface form of disclosed inheritances from File Managers Systems.

Like the other interfaces of the WIQA toolkit, this form of the interface is programmed in Russian, and therefore, some form fields are marked with labels.

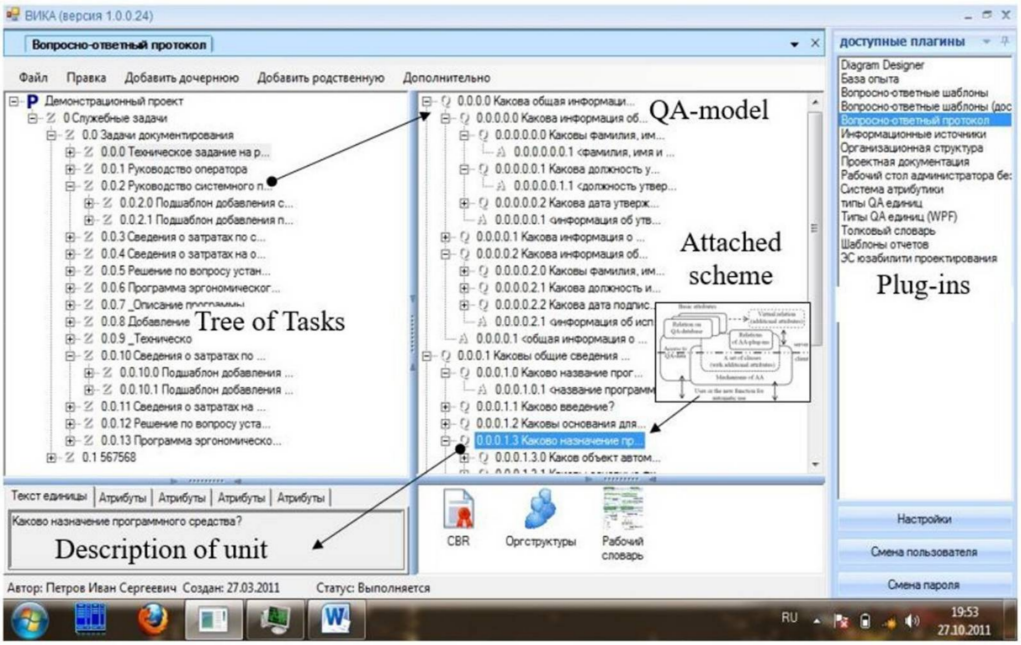


**Figure 7.** Basic interface form for visual interactions of the designer with QA-nets.

## 2.3. QA-Nets of Pseudo Codes

At the end of the first stage of our research and practice, we decided to evolve the potential of a mastered version of behavioral programming. This decision was aimed at creating the specialized pseudocode language with the following features:

- Orientation on automated behavioral programming in conditions described in the previous subsection;
- Support interactions of designers with accessible experience;
- Automatic execution of pseudocode programs when it is possible and useful;
- The possibility for the use of programmed components written in appropriate programming languages;
- Intertwining automated and automatic forms of coordinated execution for parts of programs;

- Embedding this language in the toolkit WIQA for the possibility of its evolving.

Conceived pseudocode language $L^{WIQA}$ has developed in the second stage, and it is evolved and used until now. Let us clarify its features.

In the applications mentioned above, we have already started using the elements of this language, such as using QA objects to represent data in organizational units or some commands in techniques executed by designers. The experience of such applications of QA cells has led to the expediency of using the following useful interpretations:

- "question" → "variable name for a simple data type used in traditional programming" and "answer" → "value of this variable";
- "certain composition of questions" → "data of a composite type (for example, array, record, set, array of records or table, stack, queue or other composite data types)" and "corresponding composition of answers" → "the current value of data of this type";
- "question" → "pseudo-code program operator" and "answer" → "execution of this operator", leading to a certain "result";
- "a certain composition of questions" → "connected set of operators" (for example, in a procedure or function) and "corresponding composition of answers" → "execution of this operators" with the corresponding result;
- "question" → "reason" and "answer "→ "effect".

Based on these interpretations and with orientation on tools supported the traditional programming, we developed and tested the $L^{WIQA}$ language and an environment [5] serving the use of this language in the conceptual design of the SISs. Potential of this language is sufficient for the creation and use of QA-sets for various purposes [5], including pseudo-code programming of nets, figuratively shown in Figure 8.

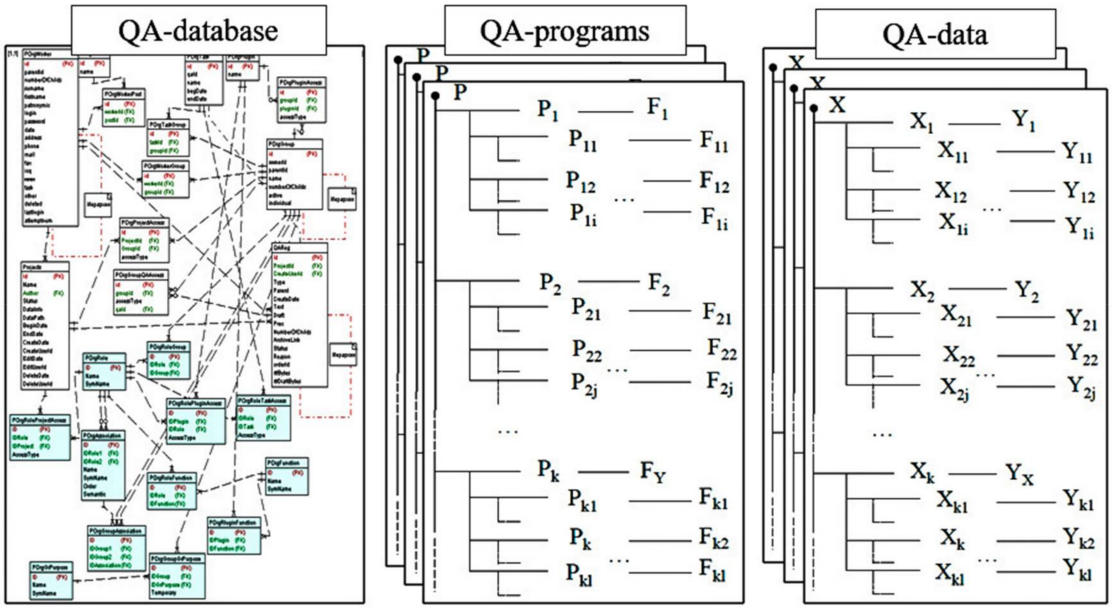


**Figure 8.** QA-nets of pseudo code programs (QA-programs).

The scheme underlines that QA-nets including their versions in forms of QA-programs and QA-data are uploaded in the semantic memory (QA-database), and any such net or any of its component are accessed for manual interactions through the shell of MDBS-type. For program access to QA-nets and their components, this shell was evolved so that any of its manual operations have the functional analog for the automated or automatic execution.

Let us note, in advance, one of the special features of the language $L^{WIQA}$: the execution programs written in this language are carried out (in the "interpretation" mode) by the designer who performs the role of an "intellectual processor" (I-processor) [6]. This role is active in workflows "Interaction with experience", providing the processes of conceptually solving the project tasks.

In work with a task, acting as I-processor, the designer is responsible for building its precedent model on the basis of automated mental experimenting with necessary units of experience [6]. Such work is fulfilled in parallel with solutions of tasks, and it is accompanied by QA-reasoning registered in QA-memory of the toolkit WIQA.

### *2.4. Interpretation of QA-Nets in the Frame of Conceptual Space 313*

As told above, in our research and practice of conceptual designing the SYSTEMs, we master the reflection of basic essences of the operational space onto their conceptual representations in forms of QA-nets embedded into CS(t). Interactions of the designer with nodes of such nets are shown in Figure 9, where pictured QA-nets fulfill the role of a background reflected the structure of CS(t)

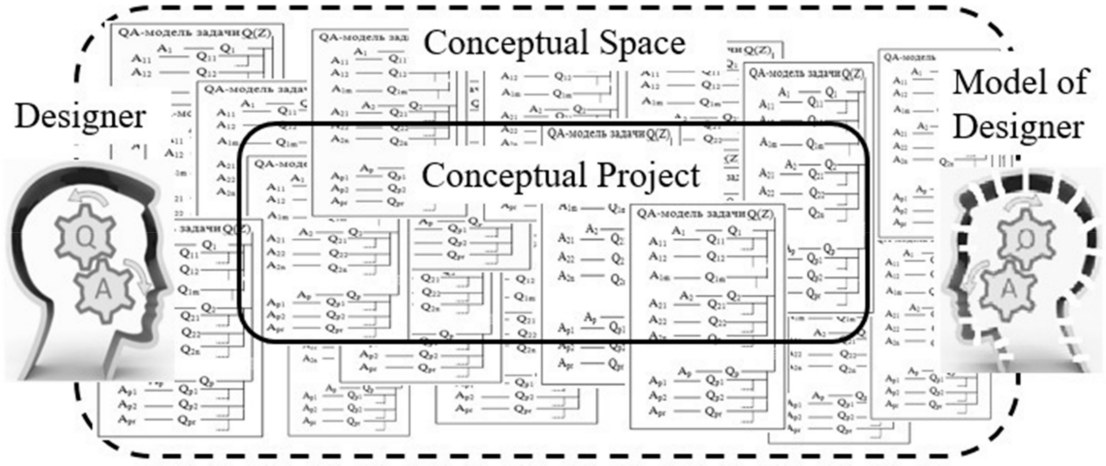


**Figure 9.** Interactions of the designer with nodes of QA-nets.

The conceptual project (CP) has a representation that is located in the CS(t), and components of this representation are also objectified in forms of QA-nets. In accordance with the system approach, the CP is a system located in the environment designated as the CS(t). Moreover, the CP in its current state consists of components that were extracted from the CP and processed by the necessary way.

Interacting with objects of the CP or CS(t), the designer interacts with the certain node or nodes of the corresponding QA-nets. Any chosen node has a verbal expression (corresponding basic attribute) that register "traces" of the designer's access to the natural experience, psychological actions with which are activated in the mental space. During interactions between the designer and the node, we interpret the activated part MS(t) of the mental space (MS) as an extension of the CS(t) in conditions when such an extension is bound with nets (shown in Figure 9) by indicated verbal "traces" of the corresponding node. Similar functions fulfill the images and verbal components of files attached to the node.

Thus, verbal expressions of questions and answers fulfill the roles of intermediaries between CS(t) and MS(t). Signs of questions activate processes in MS(t) among which the central place occupies "thinking" because "mental space—a space to think" [7]. There are different aims for activating MS(t), for example, to restore the certain state of MS(t) for repeatable activating the corresponding process in MS(t) or to activate the creative potential of the designer for the necessary 'tips' received from the intellect in solving the task.

For our study, MS(t) is a "black box" that manifests itself only through input and output verbal "traces" objectified in nodes of QA-nets. In the suggested approach, designers use MS(t) as "a suitable space for containing, ruminating and making use of experience" [7]. In the described case, the CS(t) consists of interrelated objects, any of which is built with the use of corresponding QA-nets. These objects are formed on the basis of facts registering the interactions of designers with the accessible experience during solving the project tasks. In the WIQA-environment, for the creation of QA-nets, designers fulfill ways, a system of which we designated as the QA-approach [5].

### *2.5. Precedent-Oriented Approach*

Applications of $L^{WIQA}$ language began with using the QA-approach to projects "Resource Testing of Parts and Components in the Aircraft Industry", "Expert System for Monitoring the Environment of a Marine Vessel", and "System for Multi-Agent Modeling of the Vessel Environment", all of which was developed at the beginning of the second stage of our research. With each of these projects, the potential of the language and its instrumental support expanded and was used for the managed enrichment of the potential of the toolkit WIQA.

However, at the second stage, our attention was concentrated on the following intentions:

1. To master the forms and mechanisms for building the tasks' models that simulate units of naturally professional experience;
2. To check the basic decisions oriented on the creation and use of Experience Base, kernel of which integrates precedents models of corresponding project tasks.

Such intentions have led us to the precedent-oriented approach and its embodiment, allowing the designers to build reusable models of the solved tasks in the forms of models of precedents [8]. Any precedent is the activity of a person or a group of persons associated with an action, decision or behavior carried out in the past that is useful as an example for repeated use and/or justification of repeated actions on such a pattern.

The main feature of any precedent as a unit of the human behavior is its repeatability. However, units of the human experience also define and manage repeatable behavior actions. This similarity prompts us to build models of precedents using the analogies of units of experience. Moreover, well-thought out and tested models of precedents can be used as models for units of the human experience.

The search of well-thought models must concern the essence of the human experience. Before the "invention" of such kind of the experience, nature had built the forms of the experience coded in the brain's structure on the basis of conditioned reflexes. The evolution of such a phenomenon by nature has to lead to the human experience, the units of which are processed by consciousness using natural language.

Therefore, in the offered approach, units of the human experience are understood as "intellectually processed conditioned reflexes" that underline the use of intellectual mechanisms in the formation and evolution of this phenomenon [8]. This understanding can be transferred to the logical scheme of the precedent with the following structure (Figure 10):

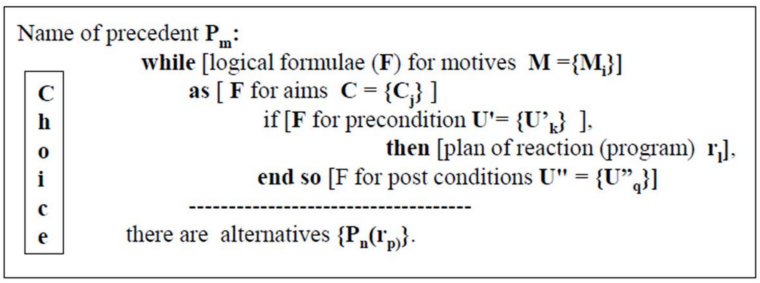


**Figure 10.** Logical model of precedent.

This model is a human-oriented scheme. The human interaction can activate the internal logical process on the level of the second signal system in human brains. So, the model opens the possibility of the managed checking of the fitness of the estimated precedent. The model presents the conditions of the precedent fitness. It also opens the access to "codes" of the corresponding precedent in the human brain. The conditioned part of the model is a source of questions the answers on which can be used for checking the fitness of the corresponding precedent. Providing access to units of the human experience is the basic kind of the activity of consciousness that supports by question-answer reasoning (or, shortly, QA-reasoning). Consciousness has a dialog nature.

Figure 11 shows the representation of the task tree, in which each question of the "task"-type is supplemented with an answer "Task Solution" (S) that binds the relevant model of the precedent (placed in the Base of Precedents) with the scheme of its iterative creating.

Placement of all components of the scheme in QA-memory in the conditions of using pseudo-code programming simplifies the construction of the precedent model by its forming in the memory area allocated only for the time of its construction. After construction, the model is registered in the Base of Precedents, where it becomes available for any repeated applications if it is necessary.

The model is tied to the life cycle of building a precedent and its development, in the course of which the following specialized models are created:

1. Text model $P^T(t)$, representing the formulation of the problem, as a result of which a precedent sample was created (as a result of the intellectual processing the solution of the corresponding task $Z_j$);
2. Logical model $P^L(t)$, which specifies a typical logical model in the form of a formula for the logic of predicates written in the language of the statement of the problem $P^T(t)$;
3. The graphical model of the $P^G(t)$ use case, which represents it in a generalized way using "block and line" tools (for example, activity diagrams in the UML language);
4. The question-answer model $P^{QA}(t)$, corresponding to the task $Z_j$;
5. The $P^I(t)$ model, representing the behavior introduced into the precedent in the form of the source code of its pseudocode program;
6. Model $P^E(t)$, referencing to the executable code of the program that implements the sample use case;
7. An integral model of the $MP_J$ precedent in the form of its scheme, integrating all the specialized precedent models into a single whole.

Let us clarify this feature of the model $MP_j$ in detail. As mentioned above, the task $Z_j$ and its model $MP_j$ begins their life with zero-states when even the initial statement of the task is absent. In our profound conviction, the initial statement must be formulated with the short text that most abstractly (but in the sufficient measure) expresses the essence of the task.

After creating the initial statement of the task, the designer turns to an analysis of its text and implementing the other normative actions of the used technology. During these actions, step by step, the statement $S(Z_j, t)$ will be enriched while its uncertainty will decrease.

The enrichment is caused by generating increments $\Delta S(Z_j, t_1)$, $\Delta S(Z_j, t_2)$, ..., $\Delta S(Z_j, t_{K-1})$ prioritization of which essentially determines the characteristics of the task being solved. This feature indicates the necessity of managing the development of the task statement.

For managing, the offered approach uses the following solutions:

- Using stepwise refinement for decreasing the uncertainty of the task statement $S(Z_j, t)$;
- Coordinating the statement development with the process of creating the corresponding model of the precedent.

The precedent-oriented approach in its combining with the QA-approach was used to develope the Experience Base [9] for one of the project organizations. The business interests of this organization with more than 1000 employers are bound with developing the specialized AS. Thus, both of indicated

approaches were tested in conditions of professional work. The features of the Experience Base and its creation are accessible in some publications—for example [10] (https://www.researchgate.net/profile/Petr_Sosnin/contributions).

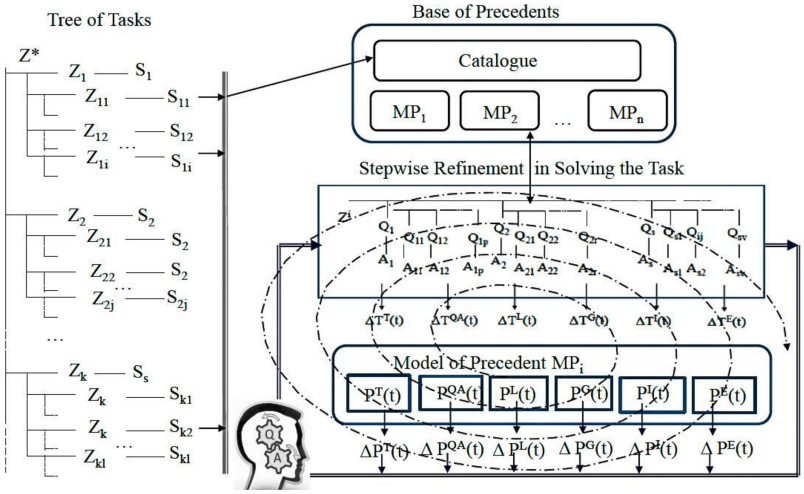


**Figure 11.** Creating the model of the precedent.

## 2.6. Substantially Evolutionary Approach to Creating and Using the Theories of Projects

At the end of the second stage of our study, developed methods and means of the WIQA allowed the designers to work in conditions that are shown in Figure 12 in a generalized manner.

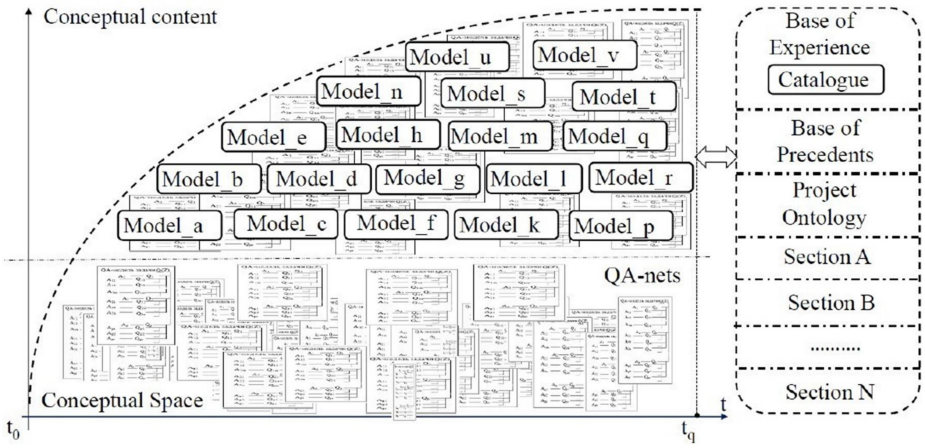


**Figure 12.** Beginning the results of conceptual solving the tasks.

The structure and content of the scheme demonstrate the life cycle of the CS(t) in conditions when the designers apply QA-approach and precedent-oriented approach in conceptual solving the project tasks. In applying the precedent-oriented approach to the certain project task, the designer creates the corresponding model of the precedent, preparing the task solution for the reuse. Such models, we interpret as models of experience that are generated in the design process and accumulated in the Base of Precedents. Any of such models is a regularity in the developed CS(t).

Conditions shown in Figure 12 have led to the following questions about l development of the CS(t):

1. What system of rules will facilitate the rational and consistent formation of the conceptual content indicated on the scheme?

2. How can such a system be implemented in the WIQA environment?

Answers to these questions were bound using the scientific approaches based on experiments and theorizing in realizing the project of SIS at the conceptual stage of designing. In the systematization of the conceptual content, the main role was laid on substantially evolutionary theories of the projects, each of which should be built as an applied Grounded Theory [11] for the project of the corresponding SIS.

From the theoretical point of view, the development of any project of SIS is a unique phenomenon, the essence of which is determined primarily by human-computer activities implemented by a team of designers interacting with the involved stakeholders in conditions of very high complexity. Each person acting in the process of designing has a unique experience that is applied unpredictably, and parts of this experience can find objectification in the designed SIS. Except all, the numerous situational factors can unpredictable influence on a course of the project in different points of its lifecycle. That is why the creation of the General Theory of software engineering is very problematic.

As will be shown below, attempts to build the General Theory are without results until now, and there exists a steady belief that the way to the General Theory is through mastering applied theories of disclosed software engineering from different viewpoints. In the described case, we choose the version of theorizing from the viewpoint of Grounded Theories.

A detailed justification of the reasons for choosing this type of grounded theories will be given in the following section. In this point of the text, we mark only the following but very important reasons:

1. Designing the SISs (and not only the SISs) has an organizationally behavioral nature (describing the social phenomena is typical for features of Grounded Theories);
2. Becoming of any Grounded Theory is based on a set of facts gathered and processed on the course of theorizing (we suggest gathering and processing the facts, any of which register definite act of interaction with experience);
3. Any of Grounded Theories of a constructive type begins its life cycle with a root question (in our case, any applied will start with a root task).

Thus, in the course of mastering the processes of formation, transformation, integration, and use of QA-nets, questions of checking their correctness and systematization were repeatedly raised. All this led to a complex of methods and means of theorizing, allowing for the creation of substantially evolutionary theories of the project (below SE-theories), the typical structure and content of which is shown in Figure 13 in the generalized form.

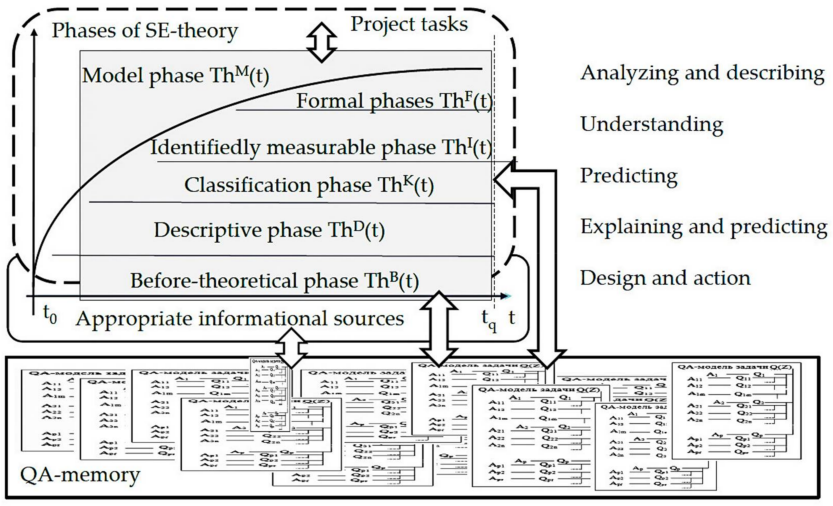


**Figure 13.** Phase structure of the theory of the project.

The scheme discloses that any applied SE-theory has a phase structure filling by the informational content in parallel with other activities of the design process. This structure includes following phases:

1. Before-theoretical phase $Th^{B}T(t)$ involves the collection of data D(t) as a set of facts $\{F_k\}$, relevant to the corresponding subject area, but rather to the properties of its components $\{B_p\}$ and the relations between them;
2. In the descriptive phase $Th^{D}(t)$, registered facts are used for constructing the texts of $T(t) = \{T_i\}$ linking the facts in the description $\{D_i = T_i\}$ of essences distinguished in the subject area, and these constructs already bring a certain ordering in the set of facts;
3. In building the theory, special attention focuses on vocabulary and especially on developing its part that presents a system of concepts $S^1(\{N_j\})$, semantic value and order of which defines a certain set of classifications $K = S^2(\{K_m\})$. Developing the system $S^1(\{N_p\})$ on the base of $S2(\{K_m\})$ is a very important phase of becoming the theory. In Figure 13, this part of the theory marked as the classification phase $Th^{C}(t)$. In the described version of SE-theory, this phase is built as a project ontology [10].
4. In becoming the theory and its use, the identifiedly measurable phase $Th^{I}(t)$ introduces the possibility of (empirical) interpreting the theoretical constructs, in particular, the ability to control the adequacy of any concept (notion) in its use in a chosen fragment of a description or fact. The typical approach to the identification is a pattern recognition with the use of appropriate means that lead to names of classes. The measurement helps to defines values of attributes for recognized essences of the theorized reality;
5. Formal phase is usually expressed by one or a number of formal theories $Th^{F}(t) = \{Th^{F}{}_{p}(t)\}$ systematizing material of prior phases and transferring the solution of the task from the level of manipulations with entities $\{U_r\}$ of the subject area SA to the level of manipulations with symbolic constructs of the theory. Such transition opens the possibility to use an inference for a prediction and apply a proof to verify the prediction;
6. It should be noted; any theory is created for its uses, a very important kind of which is models M. Models are intermediary between theories and reality when people interact with them. Therefore, models form a useful area of theory applications that indicated in Figure 13 as a model phase $Th^{M}(t)$.

Thus, in the current state of its becoming, SE-theory $Th^{P}(t)$ for the definite project P can be presented by the following expression:

$$Th^{P}(t) = [D_p(t) \cup Th^{T}{}_{p}(t) \cup Th^{K}{}_{p}(t) \cup Th^{I}{}_{p}(t) \cup \{Th^{F}{}_{pk}(t)\}] \leftrightarrow Th^{M}{}_{p}(t) = P(\{M_{pq}(t)\}),$$

where the symbol "↔" points out on the relation between the theory and models that the theory helped to build.

Theories of SE-type have the following features:

1. **The essence of facts**. In reflecting the components of the operational space OS on their models in the conceptual space, designers use interactions with experience and register such behavioral acts by textual models of questions and answers. In creating SE-theories, designers extract the facts $\{F_k\}$ from sets of such models. Thus, facts $\{F_k\}$ are traces of designers' interactions with the used experience;
2. **Gathering the facts**. Applied SE-theories form a sub-class that corresponds to the kind of Grounded theories of a constructive type specified by K Chermes [11]. One of the features of this kind is starting the creation of the theory from the root question. For SE-theories, such root question is an initial statement $S(t_0)$ of the root task $Z^*(t_0)$ of the corresponding project P. The principal role in gathering the raw facts fulfills an abductive search;

3. **The relevance of facts**. For any SE-theory, a role of facts play only such of them that are bound with solving the project tasks that are understudied as naturally artificial essences with the following features:

    - As told above, tasks are the type of questions, answer on which are constructed in forms of tasks' solutions;
    - They are oriented on achieving the definite goals that must be confirmed by obtained results and their checks;
    - In the general sense, a life cycle of a task can include creative actions that help to overcome definite gaps as problems if such gaps are revealed (but "task" has another sense then "problem");
    - Solved tasks are a type of values, that should be prepared for the future reuse.
    - The task description that provides the task reuse can be interpreted as the model of the corresponding precedent, that simulates the unit of experience.

4. **Location of theoretical constructs**. Figure 13 indicates that facts $\{F_k\}$ used for creating SE-theory are placed in QA-memory, from which they are extracted for the necessary processing. In the descriptive phase, typical results of processing are concept, attribute, property, attitude, postulate, statement, idea, axiom, principle, hypothesis, goal, motive, condition, cause, effect and other constructs, and from their compositions, for example such as rules of inference, statement of problems, developing Project Theory, and other compositions. It must be added, any other sub-theory with its components is also uploaded in QA-memory;
5. **Correlation of actions**. As told above, any SE-theory is created in parallel with other lines of designers' actions who are forced to switch from these lines to a set of lines of theorizing. Possible versions of switching among lines are shown in Figure 14.

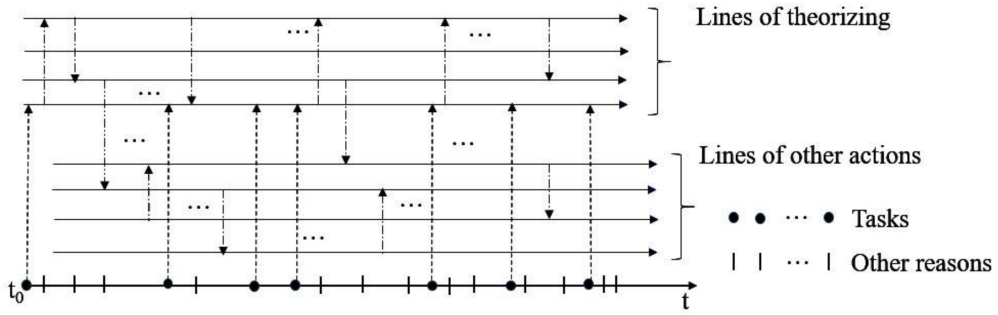


**Figure 14.** Switching among lines of actions.

The scheme indicates that there are two types of reasons that initiate switching between lines. One type corresponds starting the work with tasks while the second is connected with other reasons, a part of which is caused by applications of SE-theory in its current state.

6. **Applications of SE-theory**. If it is necessary or useful, designers can use the current state of SE-theory for analyzing, describing, understanding and explaining of situations or for predicting the results of possible actions or for creating the useful models for implementing the next steps of the design process. Any application is constructive because it is accompanied by creating the necessary QA-objects and their compositions;
7. **Referents of theory**. Any SE-theory reflects the experience that is objectified in the developed system in the current state of its lifecycle;
8. **Coding**. On the course of becoming any SE-theory, the basic way for coding facts and theoretical constructs is their coding as QA-objects;

9. **Memoing**. There is two version for registering memos, which provide a link with reality and are a source of prompts for the development of the theory. In the first version, the designer can attach the memo to the construct (uploaded in the cell of QA-memory) through a reference. The second way is the use of subordinated answer in the corresponding QA-net;
10. **Relations with Literature**. Founders of Grounded Theory indicated that creators of applied theories are better to compare theories with relevant publications at the final stages of their work. In constructivist grounded theories, such prescription is violated. In designing, relevant publications are a useful source of prompts both as for designing so for theorizing;
11. **Access to Facts and Constructs**. Any SE-theory exists in QA-memory where components of any theoretical phase are accessible for the use of pseudo-code programs, and such possibility helps in automating behavioral actions in the creation and use of the theory;
12. **Basic Principle of Evolution**. In SE-theories, tasks are reasons of evolution. Any applied SE-theory begins its life cycle from the initial statement $St(t_0)$ of the root task $Z^*(t_0)$, QA-analysis of which leads to subordinated tasks combined in the tree of tasks. As it will show below, in the described way of creating of SE-theories, it is used design thinking approach [12] for the work with any new project tasks. In evolving applied SE-theory, any implementation of such approach fulfills a role of a "soft" rule of inference. Any such rule of inference must satisfy the principle of "additivity", the essence of which is clarified by the following reasoning.

Any such rule of inference must satisfy the principle of "additivity", the essence of which is clarified the following reasoning. Any solved task $Z_i$ leaves textual traces in theory $Th^P(t)$ understood as the system $S(\{T_j\}, t)$ of textual units $\{T_j\}$. Let us assume, that any textual unit $T_j$ of traces included in $S(\{T_j\}, t_j)$ is its increment $\Delta S(T_j, t_j)$ in the moment of time $t_j$. Then, this system of texts can be expressed in a form

$$Th^P(t_j) = S(\{T_j\}, t_j) = S(T_0, t_0) \cup (\cup_j \Delta S(T_j, t_j)$$

where $T_0$ is a textual description with which designers start their work with the project P(t) and its theory $Th^P(t)$ in the moment of time $t_0$.

Because the expression corresponds the textual structure of the theory $Th^P(t)$, it can be presented in the form

$$Th^P(t_j) = Th^P(t_0) \cup (\cup_j \Delta Th^P(t_j)$$

where $\Delta ThP(t_j)$ is a next increment of the theory in its current state $Th^P(t_j)$.

The absence of contradictions is possible when any increment $\Delta Th^P(t_j)$ does not contradict the theory state $Th^P(t_{j-1})$. If it is so, then the increment $\Delta Th^P(t_j)$ corresponds to the principle additivity that strategically guides by evolving of SE-theory.

By additivity principle, any textual unit $T_j$ generated during the creation of SE-theory can embeds into $Th^P(t)$ only if it is sufficiently justified and does not lead to changing in the previous state of the theory. There are two versions of such embedding:

1. In the first version, the increment $T_j$ defines a new construct included in the theory (extension in width);
2. In the second version, the increment $T_j$ extends the specification of the construct that was already included in theory (extension in depth).

Thus, before embedding any increment $T_j$ into the current state of the theory $Th^P(t)$, the increment is needed to check on its correspondence to the principle of additivity. In the practice of designing the systems, there are cases when the checking of $T_j$ indicates the violation of this principle, but the solution of the correspondence task Z should be included in the project. Similar cases are usually processed by the specialized workflows "Management of Changing".

In creating of SE-theories, both indicated versions of embedding the increment are realized with the use of QA-analysis that is a kernel of the QA-approach. In this analysis, the important place occupies discovering the questions and cause-and-effects relations in textual units.

There is a way for discovering the questions with the use of a communicative view on an investigated textual unit $T_j$ described with an initial uncertainty $\nabla U(T_j, t_0)$. The essence of this way is shown in Figure 15.

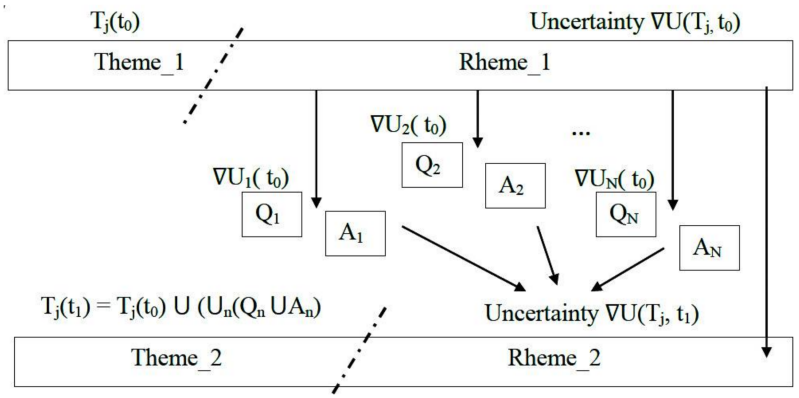


**Figure 15.** Reducing the uncertainty of textual unit.

This figure demonstrates controlled reducing the uncertainty $\nabla U(T_j, t_0)$ of the description $T_j$ by the use of question-answering in an actual division of its sentences. In interactions with the text, actual division as a communicative mechanism breaks up any sentence on two parts—"theme" and "rheme", where "theme" is the predictable starting point of the communicative unit, and "rheme" indicates on new information that should receive an additional value.

In any textual unit, the designer can separate rheme and extract questions $\{Q_n\}$ reducing the existed uncertainty with the help of the project ontology in its current state. Automated discovering the questions is one of the basic functions in applying the project ontology as the classification phase of SE-theory. Descriptions of questions and corresponding answers lead to additional textual units, the content of which reduce initial uncertainty and so on.

At the end of this subsection, let us present the initial statement of the root task for the project of subsystem intended for creating and using the SE-theories:

1. In order to promote increasing the successfulness in designing the SISs, it needs to develop a set of tools (?) that provide step-by-step creating (in parallel with other actions of conceptual designing) the applied SE-theories (?), any of which is represented by an interrelated set of textual, graphical and conceptually algorithmic constructs (?);
2. In the basis of the step-by-step creating the SE-theory, it needs to put the mechanisms of QA-approach, precedent-oriented approach and design thinking, implementation of which is based on conducting of automated mental experiments in the condition of graphical and ontological support;
3. The development of SUBSYSTEM should be carried out in the instrumentally modeling environment of WIQA so that the built-in set of tools is included in it as an extension.

Some points in the text include the sign "?" indicated on directions of question-answer analysis. This root task was used as for developing the specialized instrumental means for the methodological realization of the substantially evolutionary approach to theorizing. This methodology is based on combining the question-answer, precedent-oriented, and substantially evolutionary approaches in their applying to creating the applied SE-theories for corresponding projects of SISs.

Once again, in theorizing, designers formulate such constructs as concept, attribute, property, attitude, postulate, statement, idea, axiom, principle, hypothesis, goal, motive, condition, cause, effect and other constructs, and from their compositions, such as rules of inference, statement of problems, developing SE-theory, and compositions of the other types.

It should be noted that in creating any SE-Theory, automated design thinking fulfills the role of the typical rule of inference. Examples of this rule are realized in conceptual solutions of corresponding tasks of the project. Any SE-theory is the specialized set of QA-nets nodes of which are marked by the same additional attribute. Such way helps to extract theoretical constructs from QA-memory. QA-nets of the certain SE- theory is placed in the memory area that occupied by the experience Base. The Project Ontology is a very important part of the corresponding theory.

## 3. Discussion

From the theoretical point of view, the development of any project of SIS is a unique phenomenon, the essence of which is determined, first of all, by human-computer activities implemented by a team of designers interacting with the involved stakeholders in conditions of very high complexity. Each person acting in the process of designing has a unique experience that is applied unpredictably, and parts of this experience can find objectification in the designed SIS. Except all, the numerous situational factors can unpredictable influence on a course of the project in different points of its lifecycle. That is why the creation of the General Theory of software engineering is very problematic.

As told above, last years, the greater part of attempts in searching the new ways of theorizing and building the General Theory was caused by the initiative SEMAT. In its initial settings, and even in the title, the SEMAT-approach stresses the need for theorizing the foundations of software engineering. In the same time, the normative document "Kernel and Language for Software Engineering Methods, Version 1.1" does not include constructive answers about the creation and use of appropriate theories in the development of software systems.

The state of affairs with theorizing was evaluated at annual conferences and workshops held within the framework of SEMAT (for example, the General Theory of Software Engineering (GTSE 2011–2015) and the 5th International Workshop on Theory-Oriented Software Engineering (2016)). At the same time, it can be stated that the satisfactory general theory is absent till now.

Typically, after workshop its results were generalized, and such report for the second workshop marked that the General Theory must "explain and predict software engineering phenomena at multiple levels, including social processes and technical artifacts, should synthesize existing theories from software engineering and reference disciplines, should be developed iteratively, should avoid common misconceptions and untheoretical concepts, and should respect the complexity of software engineering phenomena" [13].

The similar report discussing results obtained before the fifth workshop [14] underlines the importance of the separability principle in studying and creating of the General Theory. In its conclusion, this report marked the expediency of "separating a general theory into multiple pieces (e.g., one for software artifacts and another for the process of developing software artifacts) and devising separability principles".

Such approach is sequentially analyzed in publication [15] where their author argued that the general theory of software development should have "two logical parts: design and evaluation, D and E, each of which should be qualified as a theory. Based on the analysis of this position, the author delineates the rich diversity of theories related to D and E and regards them as sub-theories critical for understanding the relationship between theories in D and E. Moreover, the author conducts division D and E on subordinated theories, for example, Theories of Behavior and Constraints, Theories of Model Structure, Theories of Interface Usability, Theories of Usefulness, Theories about Evaluating Models and Theories about Evaluating Theories". However, detailed descriptions and specifications of indicated theories in the articles are not given.

Among study reported on these meeting, we mark the following publications:

1. Ralph [16], who "distinguished the variance theory (which predicts a dependent variable concerning independent variables) and the theory of processes (which explain how the phenomenon occurs)".

2. Ng [17], who "offered an approach to software development, described by Essence. He argues that each software project is unique and sensitive to its context".

The search for new ways of theorizing occurs in other communities of theorists and practitioners (IEEE, ACM, ... ). From year to year, the state of affairs with theorizing in the domain of software engineering is enriched, but some important questions of theorization are still open, and the search for answers to them is topical.

Very often, researchers, who were participated in attempts to build the general theory have referred to the paper [18] that includes an analysis of assignments of theories and a way for creating the useful theories oriented on their applications in software engineering. This paper is a source of the following prompts:

- Classification of theories with the viewpoint of their responsibility (analysis. explanation. prediction. explanation and prediction. design and action).
- Typical steps of building the theory (defining the constructs, defining the propositions, providing explanations to justify the theory, determining the scope, testing through empirical research).
- Evaluated characteristics: testability, empirical support, explanatory power, parsimony, generality, utility

Early and especially last years some of the researchers have been conscious of expediency to build the applied Grounded Theories for definite organizationally behavioral phenomena in applications of software engineering. One of the reasons of such intentions is bound with achieving the positive effects in some directions of designers' activity, but other researchers count that through such applied theories it is possible to find the way to General Theory (the way from applied Grounded Theories to General Theory of software engineering).

The state of affairs in attempts to developments of Grounded Theories in the domain of software engineering one can estimate with the use of reviews [19,20]. In the review [19] published in 2013, there are no applications of Grounded Theory to a coherent set of specialized disciplines typically implemented by designers in the development of systems with software. Therefore, in the review, all investigated publications were separated into three areas - flexible design, distributed development and the formation of requirements.

In the second review [20] that was published (in 2016) by authors directly related to the SEMAT initiative, it is stated that in the vast majority of investigated publication, Grounded Theory is applied fragmentarily, either at the level of Grounded Theory procedures or to some kinds of activity of designers. Particularly useful results of the analysis are two questionnaires, allowing researchers to determine the specifics of the Grounded Theory designed by them. The first questionnaire helps to understand the general characteristics of the Grounded Theory being created, while the second questionnaire focuses on the features of the basic data (the facts over which the theory is built up): what is their source? To what extent are the data adequate for the study? Which forms and procedures are adequate for coding of data? It should be noted; the second review includes the following assertion «Grounded theory remains one of the most rigorous methods to generate new theories. This is a significant issue as the establishment of a strong theory base has been identified as an important challenge for the software engineering discipline. We believe well-conducted Grounded Theories studies can make significant contributions to our field and help to develop rich theories to inform future empirical studies in Software Engineering.

Among last attempts in this direction, we mark the following publications that were taken into account [21] in developing the suggested sub-class of SE-theories:

1. Johnson and Ekstedt [22], who "explore a theory of cognition, as a component of General Theory and found a correlation between the theory of human cognition and the empirical measurements, which express software complexity regarding the program languages".
2. Ralph [23] who estimated possible inheritances from process theories with the point of view "how and why an entity changes and develops".

3. Adolpha with co-authors [24] who indicated that "A necessary condition for the success of a software project is that there is at least one individual who is sufficiently engaged that they can detect Perspective Mismatches, and who has the personal strength to reach out and initiate the Reconciling Perspectives process".

At the second part of the discussion. It is necessary to refer to some positions that are bound with understanding and objectifying of spaces in the domain of design. In the most general case, any space is an essence or construction that consists of objects of different kinds and relations among them, including dynamic relations. The important feature of any space is a having the regularities manifest themselves in activity in this space as surrounding of people or as surrounding of the certain system or systems. In this respect, any space has boundaries that limit the area of interest of a particular activity.

In practice of designing the system with software, it actively uses the phrase "design space" that is far from new. In the report [25], this phrase points out on "*a multi-dimensional design space that classifies system architectures. Each dimension of a design space describes variation in one system characteristic or design choice*". Very interesting interpretation of the design space is specified in the dissertation [26], where this artifact was disclosed from the viewpoint of "*the construction, exploration, and expansion of a conceptual space*".

Similar position on the design space "as a conceptual space, which encompasses the creativity constraints that govern what the outcome of the design process might (and might not) be" is presented in the paper [27]. Authors of this paper interpret the design space as a dynamic artifact that is developed and changed by designers on the basis of accessible and mastered knowledge and experience.

Additionally, we mark the paper [28], where the design space is interpreted "*as a conceptual tool that can be used both for designing and understanding design processes*". Here, the author underlines that "from all work done during the design process, designers construct knowledge and experience of the design space" in the constructive form.

The above-cited works implicitly or explicitly indicate on design spaces as artifacts that belong to the class of conceptual spaces. This class of spaces includes their versions with a number of other applications. Disclosing these versions, we start with the paper [29] underlining the role of the CS(t) in creative activity "*someone who seeks to understand what creativity is, and how it is even possible, needs to consider the mental geography of conceptual spaces*".

One type of geography focuses on "Conceptual space as a Geometry of Thought" suggested by P. Gärdenfors [30]. This type of spaces can be formalized with the use of metrics and useful ordering of their objects. Any object of the CS(t) is presented as the certain domain with characteristics of object quality. These features are used for logical views on the CS(t) [31], its theoretical descriptions [32] and thorough formalization [33]. We also mark the paper [34] where the design is understood and 789 formalized with the use of three worlds (spaces)—the external world, the interpreted world, and the expected world, in which designers apply Function-Behavior-Structure framework (FBS-framework, FBS-model). In the paper [35], there is a suggestion for the answer the question "How a function is transformed into behavior?".

All papers indicated in this section were used as sources of prompts for decisions, structures of artifacts, schemes of actions and requirements to instrumental means for the creation and use of SE-theories. It also needs to note, that for the proposed kind of the SE-theories and the method of their construction, we found no analogs.

## 4. Means of Theorizing

### *4.1. Design Thinking Approach*

In the suggested approach to theorizing, we use understanding a nature of design that was described in [36] where its author K. Dorst interpreted such nature by the following implication

$$??U(t) \overset{???W(t)}{\rightarrow} ?V(t)$$

This expression reflects a situation S(t), in which initiators of a certain innovation decided to start the work with a corresponding project P, realizing of which can lead to the new SIS. The initiator estimates the arisen situation in the following way:

1. There is a goal G to achieve a certain value for potential customers, and in the current moment of time, this value V expressed with some uncertainty ?V(t).
2. Conditions U(t), in which the goal can be achieved, are vague perceived, and they can be expressed with essential uncertainty ??U(t) for reducing of which conditions can be constructed while designing the SIS.
3. The construct ???W(t) is unknown, and it can not be qualified as a problem gap between ??U(t) and ?V(t) taking into account their ambiguity.
4. Having the goal G expressed by?V(t) opens the possibility for interpreting the situation S(t) as an indicator of the task Z* that should be solved.

In creating of SE-theory, this process begins with an initial state that corresponds to a perception of the situation $S(t_0)$ at time $t_0$. From this moment of time, a conceptual stage of project P begins its life cycle in our version that supposes that designers will use Design Thinking approach (DT-approach) and other possibilities described in this paper. The general scheme of the DT-approach is presented in Figure 16.

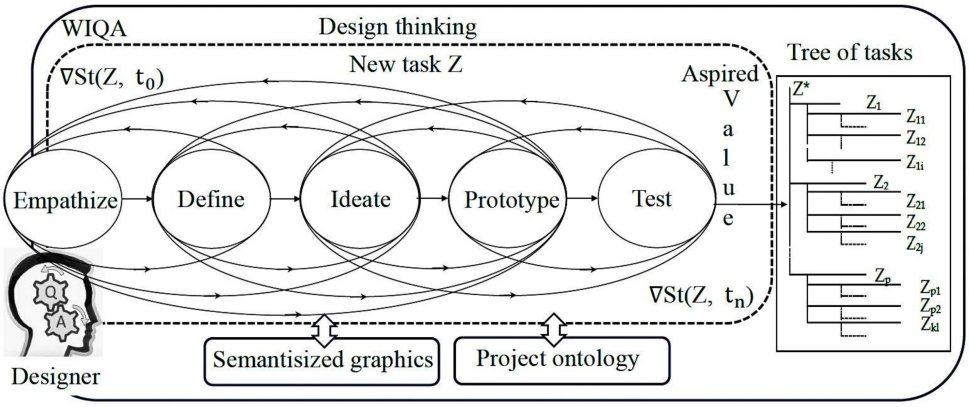


**Figure 16.** The iterative process of design thinking.

Applying the DT-approach begins with the state $S(t_0)$ corresponding the root task $Z^*(t_0)$ of the project P, but such approach will be applied to any new task $Z_j$. Therefore, an implementation of this approach can be explained by an example of any new task $Z_j$.

When the designer discovers a new task $Z_j$ in a situation $St(t_0)$, implementing this approach begins with registering this situation in a verbal form expressing an initial uncertainty $\nabla U(Z_j, t_0)$ of an appeared situation. Let us assume that this verbal expression (for example, list of keywords) registered outside the designer's brain is sufficient to restore (call to mind) the discovered task in brain's structures if it will be necessary.

Having such a list of keywords discloses the possibility for discursive expression of the task $Z_j$, and for creating the discourses (in forms of textual expressions), the designer will implicitly and explicitly apply interactions with personal vocabulary (inside the brain). It should be noted that the effectiveness of interactions will be increased if, in parallel and in reasoning, the designer will refer to appropriate dictionaries outside of brain (e.g., will use a project ontology). We note that the creations of discourses lead to outcomes, the uncertainty of which $\nabla U(Z_j, t_1)$ is less than in its initial state $\nabla U(Z_j, t_0)$.

The designer implements described actions in the first step of the DT-approach. This step is aimed at generating the discourses, in which the designer takes into account relations of "aspired value" with

its "customer" or "customers". That is why this step is termed "Empathize". In a modified version of the DT-approach implemented in the environment of the toolkit WIQA, in the transition from keywords to a necessary set of discourses, the designer registers facts, checks their lexis and potential relation, and, after that, generates the conceived discourses in understandable forms. In enumerated actions and actions applied at the next steps of the DT-approach, the designer use methods and means described in the next sub-sections.

The second step "Define" is targeted at designer's understanding of the task situation in its current state. To understand, the designer formulates and rewordings the initial statement $St(Z_j, t_2)$ of the discovered task, keeping the essence of its content with the corresponding uncertainty $\nabla U(Z_j, t_2)$. At this step, results of empathizing are processed in conditions of active using the automated imagination that includes a figurative ability of the designer in the process of understanding.

At the third step "Ideate", the designer tries to invent an appropriate idea for developing the task solution. The found idea will express the feature of an answer the question "How the task can be solved?". This answer will continue to reduce the uncertainty of the task description. It needs to underline that actions of this step are based on designer's intellectual abilities among which are especially important intuition and imagination in the context use of language means. This step also leads to reducing the uncertainty of the task description.

Steps "Prototype" and "Test" are intended for preliminary building the task solution that corresponds to the invented idea and has a checkable form for its testing.

The scheme in Figure 16 additionally demonstrates the active use by the designer of feedbacks in the iterative DT-process that reduce the uncertainty from initial state $\nabla U(Z_j, t_0)$ until $\nabla U(Z_j, t_n)$ at the end of the DT-process. This mechanism supports corrections of previous steps and the creation of alternative solutions also. Thus, the DT-approach helps the designer to build more than one conceptual solution of the new task and to choose the better solution among alternatives. In this process, the designer actively uses the project language for different aims.

From the viewpoint of theorizing, the modified version of the DT-approach plays the role of "a soft rule of inference" [37], which leads to the following outcomes:

1. Extracting and preliminary testing the new concepts (notions) and/or enriching the before-mastered concepts;
2. Revealing, wording and testing the theoretical constructs such as concept, attribute, property, attitude, postulate, statement, idea, axiom, principle, hypothesis, goal, motive, condition, cause, effect, and other constructs;
3. Revealing and checking the "cause and effect regularities" that need to take into account and embed in the ES-theory and system that is designed. Some such regularities are the models of precedents;
4. Generating the figuratively semantic schemes objectified results of the architectural and cause-and-effects understanding in reusable forms;
5. Revealing the potential directions of for the next steps in theorizing and designing (new subordinated tasks).

Some details of achieving the enumerated outcomes are reflected in Figure 17, which is the disclosed connectivity of the design thinking with SE-theory and the process of designing.

The scheme indicates that in the course of implementing the DT-approach, the designer uses conceptual experimenting [38] and an automated mental imagination when it is necessary or useful. Both these activities are bound because mental imagery helps in defining the conditions of any automated thought experiment (conceptual experiment), features of which are described in detail in publication [37]. Moreover, the automated mental imagination allows the designer in presenting the results of conceptual experiments with the use of figuratively semantic schemes [39].

In the DT-processes, designers actively use the means of question-answer analysis that occupied the central place in the instrumental support of the QA-approach.

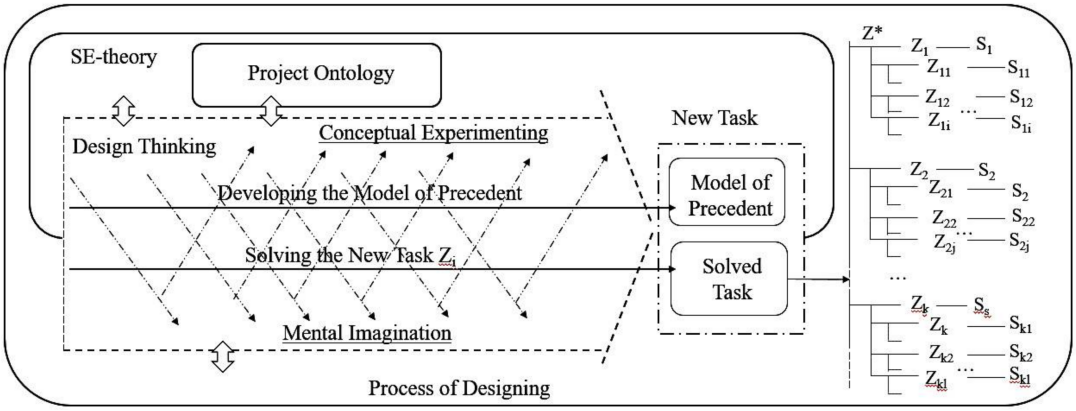


**Figure 17.** The context of design thinking.

*4.2. Figuratively Semantic Support of Conceptual Activity*

The basic aims of conceptual solving any task $Z_j$ are the following:

1. To present the statement $St(Z_j, t)$ in the understandable form, for which it needs to verify achieving the algorithmic realizability of the task solution;
2. To develop and test a conceptually algorithmic solution in conditions corresponding to the place of the task in the project of SIS.

For achieving the first aim, the designer includes the necessary textual and graphical models from $\{M^V_j\}$ and $\{M^G_k\}$ in the construct $St(Z_j, t)$, and embedded components will provide reuse of understanding in the process of which these models will repeatedly activate the mental imagery. Without such activation, the needed understanding is impossible. It should be noted, in a stimulation of the mental imagery, visual items of any text will participate as special graphics, but, in understanding, the use of models $\{M^G_k\}$ is more efficient. The toolkit WIQA supports creating and transforming the visual models for types presented in Figure 18.

A set of types include

1. Pictorial type $M^P$, models of which help to express the structure of the requested graphical construct as an interactive model $I^P$ or not. For both versions of models, the used graphical editor automatically create programmatic versions, changing of which reflected on graphical models. Such opportunity is also realized for other types of models;
2. Declarative type $M^D$ that is intended for visualizing the semantics of textual models $\{M^V\}$ that are important for the achievement of understanding. The main cause for using of this type is the check of $St(Z_i, t)$ and its verbal components via mechanisms of declarative programming. Therefore, the version $P^D$ additionally has a Prolog-like description;
3. Conceptually algorithmic type $M^A$ provides visual modeling of the algorithmic components of the task is solved. For this, the designer can use a pseudo-code language LWIQA that is built for the semantic memory of the toolkit WIQA. In the current state of WIQA, the type $M^A$ supports the work with Use-Case diagrams, Activity diagrams, and diagrams of Classes. For the transition to the version $P^A$, it is applied the automatic mode based on the model-driven approach.

The scheme in Figure 18 includes not only types but also relations among them. The set of relations consist of pairs presenting the transitions between corresponding models of different types (transitions $([R^{PD}, R^{DP}]$, $[R^{PA}, R^{AP}]$, $[R^{DA}, R^{AD}])$ and between models and their programmatic versions ($[R_1, R_2]$, $[R_3, R_4]$, $[R_5, R_6]$, $[R_7, R_8]$). Each of these transitions is realized either automatically or automated, including behavioral actions. Any program version corresponds to the executed pseudo-code program that could be changed by the designer, for instance, to correct errors or to reduce uncertainty in the created and used visual model.

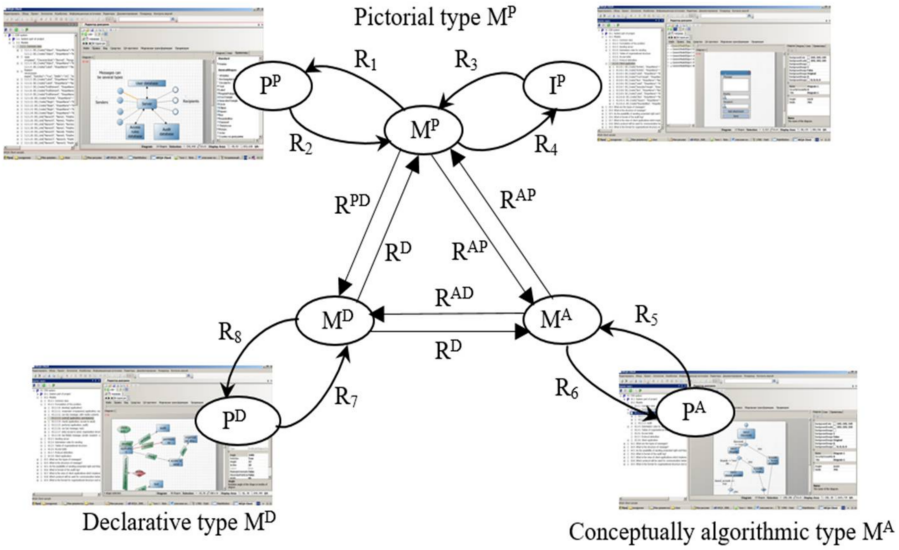


**Figure 18.** Normative types of visual models.

One of the important features of the offered approach is the use of dynamic effects and transformations that can be implemented programmatically. The designer can organize and activate the following versions of dynamic visualization:

1. Sequential visualization of any pictorial model in the step-by-step mode (or execution with the controlled delay) of the corresponding pseudo-code program;
2. Visualization of history drawing for the pictorial diagrams, which allows simulating the process of drawing (with the controlled delay or in step mode);
3. Use of different placement options. For example, when distributing the nodes of the model, it is convenient to take into account the number of input and output connections. The controlled replacement is available through a consistent application to the created diagram of different options of the placement;
4. Use of the slideshow mode for programmatic transitions from the diagram to diagram that was created, for example, in solving the task.

The offered approach to the use of figuratively semantic transformations of visual models in conceptual solving the project tasks facilitates discovering the errors in reasoning and documents and supports mental imagination and the achievements of necessary understanding.

### *4.3. Example of Starting Point of Becoming the SE-Theory*

In the previous subsection, we described the developed complex of means that is intended for figuratively semantic support of the designer's work with the task statement for expressing it in the understandable form. In the development of this complex, we applied substantially evolutionary theorizing, including its use at the stage of formation of the initial statement of the root task Z* of the corresponding project.

Applying the DT-approach to the main intention of this project began with the following discourse of the Empathy step:

D1. The statement of the root task should be expressed in an understandable form (?) that must include explicit (?) and implicit (?) traces (?) of results (?) of understanding, and such traces must be sufficient (?) for the reuse (?) of understanding expressed in the statement.

This discourse has led to questions in its point marked by a sign "?". All these questions were verbalized and got answers that were formulated at the first level of QA-analysis in in its application to discourse D1.

For example, among these answers were the following:

A1. Understanding is an artificially natural phenomenon aimed at controlling (?) the right use of naturally professional language in explicit or implicit descriptions of perceived or imagined (?) situations or events.
A2. Explicit traces of understanding are registered by concepts (notions) embedded in descriptions or pictorial constructs included into the description or accompanied it.
A3. Coordination (?) of traces and the wholeness (?) of their used set should be registered by graphics constructs (?) including the semantic traces (?).
A4. Traces of understanding must be distributed (?) in the description so that in repeatable achieving of understanding, this process will be reusable (?).

During analysis of the first discourse, it was generated the second discourse:

D2, The process (?) of achieving of the necessary results (?) of understanding and the reuse (?) of such processes must be automated (?).

Among answers built for questions extracted from the second discourse, in this article, we mark the following of them:

A5. Processes indicated in D2 are better to realize in the form of behavioral programs (?).
A6. Graphical components used in processes of understanding and their results must have program versions (?) coordinated (?) with graphics.

QA-analysis of discourses D1 and D2 was more complete than that presented above. It included expressions for all points in texts marked "?", required answers to formulated questions and not only on the first level of the analysis. Moreover, this reasoning was registered in the context of informational support with using the block-and-line schemes, evaluating, checking and decision-making. After conducted analysis with using abstracting its results, it was formulated the initial statement of the root task Z* for the project of the COMPLEX in the following view:

Root task Z*(t).

1. For creating the conditions for conducting the conceptual experiments (?) in realization of design thinking approach that is adjusted (?) on solving the project tasks at the conceptual stage of designing the SISs, it needs to develop a COMPLEX (?) of means (?) for figuratively semantic support (?) of achieving (?) and registering (?) the results of architectural (?) and cause-and-effects (?) understanding.
2. COMPLEX is to provide the iterative achieving the results of understanding for its indicated types by the use interactive block-and-line schemes (?) and corresponding them programme versions (?), adequacy (?) with which is coordinated automatically (?).
3. The development of COMPLEX should be carried out in the environment of the toolkit WIQA so that the built-in set of means is included in it as an extension.

It needs to note, on the way from discourse D1 to the initial statement of the task Z*, it was generated and analyzed 34 question-answer pairs, discovered 22 semantic errors in their texts, extracted and specified 47 concepts and built seven block-and-line schemes.

Building the SE-theory for the project of COMPLEX began from the time of formulating the initial statement of the task Z*. At that time, the project ontology already included 47 concepts with their current definitions and specifications. A part of question-answer pairs was included in QA-analysis of the root task Z*.

## 5. Conclusions

In conclusion, we mark the following:

1. The suggested substantially evolutionary approach to creating and using the projects' theories constructively involves the interactions of designers with experience and its models in the design process. For any applied SE-theory, it is implemented in parallel with other design actions, and such a mode leads to systematizing the already used units of experience in theoretical forms embedded into the current state of SE-theory. In other words, in real time, designers create a theoretical system of experience that was used in solving the project tasks and apply this system such as SE-theory, when necessary or useful.
2. Such applications lead to the additional effects in the following actions:

    - Searching for semantic errors at the level words and phrase used in reasoning or writing the textual units of memos or prescribed documents;
    - Architectural or cause-and-effect understanding in personal or collective interactions with complicated environments in the real-time process of designing;
    - Explaining that the situation appeared in solving the project tasks or solutions of tasks;
    - Creating the models of precedents for reusable project tasks;
    - Predicting a possible solution for a new task using the regularities registered in the current state of the SE-theory;
    - Managing the process of designing by discovering the questions indicated the directions of next steps of designing.

Creating and using the applied theories of SE-type are implemented through controlled intellectual actions that are intertwined with other activities of designing, and such intertwining facilitates increasing the degree of success in designing of software intensive systems. The suggested kind of SE-theories is a new sub-class of class called "Grounded Theories".

**Acknowledgments:** This work was supported by the Russian Fund for Basic Research (RFBR), Grant #18-07-00989a and the State Contract No. 2.1534.2017/4.6.

**Conflicts of Interest:** The author declare no conflict of interest

## References

1. Reports of Standish Group. Available online: www.standishgroup.com/outline (accessed on 20 November 2017).
2. Jacobson, I.; Ng, P.-W.; McMahon, P.; Spence, I.; Lidman, S. The essence of software engineering: The SEMAT kernel. *Queue* **2012**, *10*. [CrossRef]
3. Sosnin, P.; Maklaev, V. Question-Answer Reflections in a Creation of an Experience Base for Conceptual Designing the Family of Software Intensive Systems. In Proceedings of the Joint Conference on Knowledge-Based Software Engineering, Volgograd, Russia, 17–20 September 2014; Volume 466, pp. 658–672. [CrossRef]
4. IBM Rational Unified Process. Available online: http://www-01.ibm.com/software/rational/rup/ (accessed on 20 November 2017).
5. Sosnin, P. A Scientifically Experimental Approach to the Simulation of Designer Activity in the Conceptual Designing of Software Intensive Systems. *IEEE Access* **2013**, *1*, 488–504. [CrossRef]
6. Sosnin, P. Role "Intellectual Processor" in Conceptual Designing of Software Intensive Systems. In Proceedings of the 11th International Conference on Computational Science and Applications, Ho Chi Minh, Vietnam, 24–27 June 2013; Lecture Notes in Computer Science. Volume 7973, pp. 1–16. [CrossRef]
7. Young, R.M. *Mental Space*; Process Press: London, UK, 1994.
8. Sosnin, P. Precedent-Oriented Approach to Conceptually Experimental Activity in Designing the Software Intensive Systems. *Int. J. Ambient Comput. Intell.* **2016**, *7*, 69–93. [CrossRef]
9. Sosnin, P.I.; Maklayev, V.A. *Sozdaniye i Ispol'zovaniye Avtomatizirovannoy Bazy Opyta Proyektnoy Organizatsii*; Ulyanovsk State Technical University: Ulyanovsk, Russia, 2012.
10. Sosnin, P. A place and role of an ontology in using a base of experience in designing the software intensive systems. *Int. J. Web Inf. Syst.* **2016**, *12*, 62–82. [CrossRef]

11. Charmaz, K. *Constructing Grounded Theory*, 2nd ed.; Sage: London, UK, 2014; ISBN 13-978-085702914.
12. Leifer, I.; Meinel, C. Manifesto: Design thinking becomes foundational. In *Design Thinking Research: Making Design Thinking Foundational*; Springer: Cham, Switzerland, 2015; pp. 1–4. [CrossRef]
13. Johnson, P.; Ralph, P.; Goedicke, M.; Ng, P.-W.; Stol, K.-J.; Smolander, K.; Exman, J.; Perry, D.E. Report on the Second SEMAT Workshop on General Theory of Software Engineering. *ACM SIGSOFT Softw. Eng. Notes* **2013**, *38*, 47–50. [CrossRef]
14. Exman, I.; Perry, D.; Barn, B.; Ralph, P. Separability Principles for a General Theory of Software Engineering: Report on the GTSE 2015 Workshop. *ACM SIGSOFT Softw. Eng. Notes* **2016**, *41*, 25–27. [CrossRef]
15. Perry, D.E. Theories, theories everywhere. In Proceedings of the 5th International Workshop on Theory-Oriented Software Engineering, Austin, TX, USA, 15 May 2016. [CrossRef]
16. Ralph, P. Comparing two software design process theories. In Proceedings of the International Conference on Design Science Research in Information Systems and Technology, St. Gallen, Switzerland, 4–5 June 2010; Winter, R., Zhao, J.L., Aier, S., Eds.; Springer: Berlin/Heidelberg, Germany, 2010; pp. 139–153. [CrossRef]
17. Ng, P.-W. Theory-based software engineering with the SEMAT kernel: Preliminary investigation and experiences. In Proceedings of the 3rd SEMAT Workshop on General Theories of Software Engineering, Hyderabad, India, 2 June 2014; ACM: New York, NY, USA, 2014; pp. 13–20. [CrossRef]
18. Sjøberg, D.I.K.; Dyba, T.; Anda, B.C.D.; Hannay, J.E. Building Theories in Software Engineering. In *Guide to Advanced Empirical Software Engineering*; Shull, F., Singer, J., Sjøberg, D.I.K., Eds.; Springer: London, UK, 2008; pp. 312–336.
19. Badreddin, O. Thematic Review and Analysis of Grounded Theory Application in Software Engineering. *Adv. Softw. Eng.* **2013**, *2013*, 468021. [CrossRef]
20. Stol, K.; Ralph, P.; Fitzgerald, B. Grounded theory research in software engineering: A critical review and guidelines. In Proceedings of the 2016 International Conference on Software Engineering, Austin, TX, USA, 14–22 May 2016; pp. 120–131. [CrossRef]
21. Sosnin, P. *Experience-Based Human-Computer Interactions: Emerging Research and Opportunities*; IGI-Global: Hershey, PA, USA, 2017; ISBN 13-978-1522529873.
22. Johnson, P.; Ekstedt, M. Exploring Theory of Cognition for General Theory of Software Engineering. In Proceedings of the 4th SEMAT Workshop on General Theories of Software Engineering, Florence, Italy, 18 May 2015; pp. 15–24. [CrossRef]
23. Ralph, P. The Sensemaking-Coevolution-Implementation theory of software design. *Sci. Comput. Program.* **2016**, *101*, 21–41. [CrossRef]
24. Adolpha, S.; Kruchten, P.; Hallb, W. Reconciling perspectives: A grounded theory of how people manage the process of software development. *J. Syst. Softw.* **2012**, *85*, 1269–1286. [CrossRef]
25. Lane, T.G. *A Design Space and Design Rules for User Interface Software Architecture*; Technical Report CMU/SEI-90-TR-22 ESD-90-TR-223; Software Engineering Institute, Carnegie Mellon University: Pittsburgh, PA, USA, 1990.
26. Heape, C. The Design Space: The Design Process as the Construction, Exploration and Expansion of a Conceptual Space. Ph.D. Thesis, The University of Southern Denmark, Sønderborg, Denmark, 2007.
27. Kang, E.; Jackson, E.; Schulte, W. An Approach for Effective Design Space Exploration. In *Foundations of Computer Software. Modeling, Development, and Verification of Adaptive Systems*; Calinescu, R., Jackson, E., Eds.; Lecture Notes in Computer Science; Springer: Berlin/Heidelberg, Germany, 2010; Volume 6662, pp. 33–54.
28. Westerlund, B. Design Space Conceptual Tool—Grasping the Design Process. In Proceedings of the Nordic Design Research Conference, Copenhagen, Denmark, 29–31 May 2005.
29. Boden, M.A. Conceptual Spaces. In *Milieus of Creativity, Knowledge and Space 2*; Springer Science + Business Media B.V: Dordrecht, the Netherlands, 2009; pp. 235–243.
30. Gärdenfors, P. *Conceptual Spaces—The Geometry of Thought*; The MIT Press: Cambridge, MA, USA, 2000.
31. Gärdenfors, P. Semantics based on conceptual spaces. In *Logic and Its Applications*; Banerjee, M., Seth, A., Eds.; Lecture Notes in Computer Science; Springer: Berlin/Heidelberg, Germany, 2011; Volume 6521, pp. 1–11.
32. Rickard, J.T.; Aisbett, J. Greg Gibbon Reformulation of the theory of conceptual spaces. *Inf. Sci.* **2007**, *177*, 4539–4565. [CrossRef]
33. Bechberger, L.; Kühnberger, K.-U. A Thorough Formalization of Conceptual Spaces. In *Advances in Artificial Intelligence*; Lecture Notes in Computer Science; Springer: Cham, Switzerland, 2017; Volume 10505, pp. 58–71.

34. Gero, J.S.; Kannengiesser, N. The Situated Function-Behavior-Structure Framework. *Des. Stud.* **2004**, *25*, 373–391. [CrossRef]
35. Al-Fedaghi, S. Function-Behavior-Structure Model of Design: An Alternative Approach. *Int. J. Adv. Comput. Sci. Appl.* **2016**, *7*, 133–139. [CrossRef]
36. Dorst, K. The Nature of Design Thinking. In Proceedings of the Design Thinking Research Symposium, Sydney, Australia, 19–20 October 2010; pp. 131–139. Available online: http://epress.lib.uts.edu.au/research/handle/10453/16590/ (accessed on 20 November 2017).
37. Sosnin, P. A way for creating and using a theory of a project in designing of a software intensive system. In Proceedings of the 17th International Conference on Computational Science and Its Applications, Trieste, Italy, 3–6 July 2017; pp. 1–5. [CrossRef]
38. Sosnin, P. Conceptual Experiments in Automated Designing. In *Projective Processes and Neuroscience in Art and Design*; Zuanon, R., Ed.; IG-Global: Hershey, PA, USA, 2016; pp. 155–181. [CrossRef]
39. Sosnin, P.; Galochkin, M. Way of Coordination of Visual Modeling and Mental Imagery in Conceptual Solution of Project Task. In *Advances in Artificial Intelligence: From Theory to Practice*; Lecture Notes in Computer Science; Springer: Cham, Switzerland,, 2017; Volume 10350, pp. 635–638. [CrossRef]