

An Architectural Approach to the Precedent-Oriented Solution of Tasks in Designing a Software Intensive System

Petr Sosnin(✉), Sergey Shumilov, and Aleksandr Ivasev

Ulyanovsk State Technical University,
Severny Venets, Str. 32, 432027 Ulyanovsk, Russia
sosnin@ulstu.ru, mars@mv.ru, nevskei@yandex.ru

Abstract. The paper presents an approach to architectural modeling focused on its use for the development of functional components of a software intensive system (SIS), each of which is associated with the solution of a specific design task. To ensure the independence of proposed architectural modeling from the specificity of a certain SIS, the approach is focused on creating a complex of specialized means called TASK, which is intended for solving the design tasks in the workplace of a member of the design team. The specificity of the complex consists in forming by the designer for any solved task its reusable model (model of a precedent), which is included in the experience base applied in the design process of a certain SIS or their family. The prototype version of the TASK was developed in the personified version of the toolkit WIQA, which provides the solution of design tasks at the conceptual stage of the development of the SIS. This version of objectifying the approach is specifically implemented in a prototype form so that it can be used as a sample that can be fitted for architectural modeling of any design tasks when developing any SIS.

Keywords: Architectural modeling · Designing · Precedent model · Project task · Prototyping · Software intensive systems

1 Introduction

Professionally mature development of a modern SIS is unthinkable without the mandatory construction and operational use of its architectural representation (Architecture Description, AD), which is considered to be a very important version of the existence of the SIS, demanded at all stages of its life cycle. Most importantly, this version is the first (earliest) representation of the SIS, which can be tested as a whole. Moreover, this wholeness must express the essential features of the SIS, and, usually, in forms that contribute to a constructive understanding of them by those persons (stakeholders), concerns of which should be taken into account in the SIS project.

In all subsequent versions of the SIS-existence, the requirements integrated into AD must be coordinated without fail, preserving the wholeness embedded in AD, which may change on the course of developing the SIS, but only if there are very serious reasons for this. Thus, the AD artifact in its current state captures those interconnected

© Springer Nature Switzerland AG 2019
S. Misra et al. (Eds.): ICCSA 2019, LNCS 11623, pp. 97–107, 2019.
https://doi.org/10.1007/978-3-030-24308-1_9

high-level requirements, which are prohibited from being changed at lower levels of detail, since possible changes in AD are too expensive decisions for the process of developing the SIS.

The high importance of artifacts of the AD-type has found expression in standards that integrate the experience of architectural modeling of systems with software. Among such standards, a special place is occupied by the IEEE-1497: 2000 standards and its extension ISO/IEC/IEEE 42010: 2011. The standards are based on the formation of the system S ({Vj}) of "architectural views" (Views, {Vj}), based on the "points of view" (Viewpoints, {VPk}) specified in the architectural description of the project. This description must include all specifications that should be materialized for each modeled view Vj in the developed SIS. It should be noted that behind each architectural view is its graphic (typically block-and-line) image or diagram with the necessary description.

In the development of a specific SIS, the responsibility for constructing an artifact lies with the architect, who possesses the necessary competencies and uses specialized tools that ensure executing the role of the "architect." The article proposes a version of architectural modeling, focused on its use by the designer in solving design tasks, and such use contributes to the achievement of the positive effects of such modeling not only from the activity of the architect but also on other members of the designers' team.

The proposed version of the architectural approach is aimed at using useful "architectural views" on the design process from the "point of view of working with design task." The implementation of the architectural approach has led to the development of specialized tools, the prototyping of which was implemented in the instrumental environment OwnWIQA, intended for supporting the conceptual activity of the designer in individual working with project tasks [1]. The toolkit OwnWIQA provides uniting its instances into complexes in the corporate network of the SIS development for providing the work in designers' group.

2 Question-Answer Approach to Solving the Project Tasks

For many years, we investigated the processes of conceptual designing the SISs in conditions of the operational interactions of designers with experience and its models. This stage of designing was chosen because it directly involves working with semantic (conceptual) objects, the system of which must be materialized in the form of a conceptual project of SIS before its programming. Also, in practice, this stage is usually the source of costly semantic errors due to personal and collective problems with understanding, or, in other words, this stage is subject to the negative influence of human factors, especially when the designers face new tasks in their operative actions [2].

In the course of the research, some versions of the toolkit WIQA were developed, the continuous improvement of which was facilitated by its architecture, which was based on a system of architectural views on the (conceptual) project of the SIS. In investigated views onto the system, the central place occupies the task-oriented structuring of the project and the question-answer interactions of designers with experience in the process of solving the project tasks.

A special place in the systematization of views on a project is the construction of their instances for specific projects, for which designers can use the specialized

instrumental mechanisms of mapping the main entities of the operational space of designing onto the semantic memory of the question-answer type.

As a result of mappings OF a project, a corresponding conceptual space is formed in which the construction and integration of species instances and their use is carried out. The features of the conceptual space of the project in the network version of the NetWIQA toolkit and its personalized version of OwnWIQA are described in detail in the publications [1] and [3].

To date, the potential of WIQA toolkit versions provides some representations of the tasks used in the activities shown in Fig. 1. In actions with these representations, the components are objectified in similar forms of question-answer nets (QA-nets) localized in the conceptual space of the SIS-project.

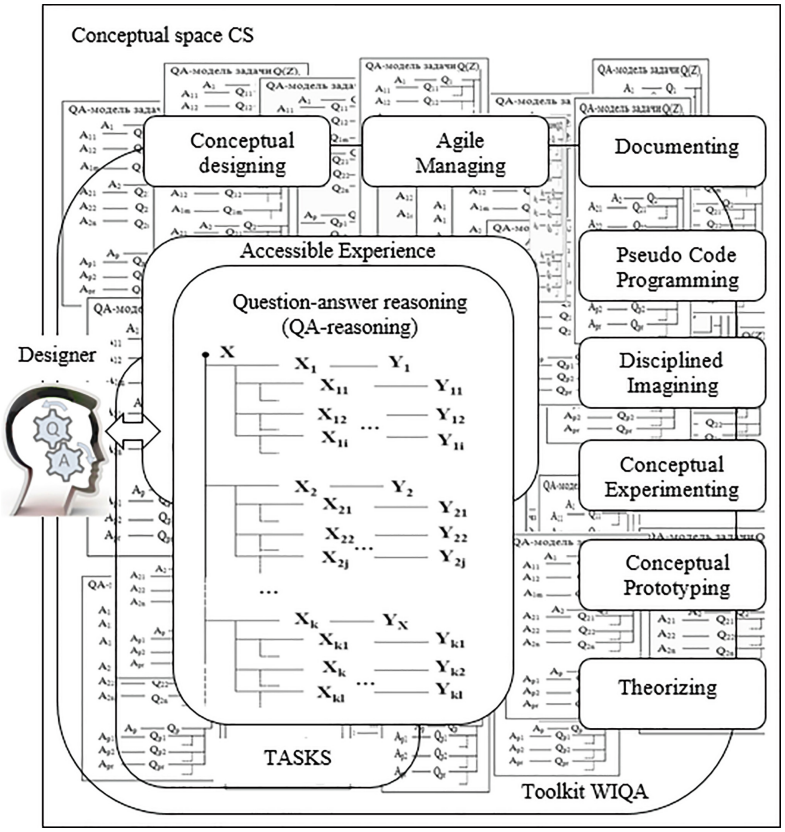


Fig. 1. Basic applications of QA-nets

To increase the efficiency of work with such components, they are integrated into the system of architectural views on that what is understood as the task/In its turn; it led to implementing the next step in the development of the toolkit OwnWIQA – to its extension by the subsystem called TASK.

The diagram, which reflects the (potential) work of a designer with a task, specifically uses the term "point of view", for which the following definition is introduced in ISO/IEC/IEEE 42010: 2011 - "a set of samples, templates, and agreements for building one type of architectural views. It identifies stakeholders whose problems are reflected in the point of view, as well as guidelines, principles, and template models for building the relevant views" [4].

Namely, in the first place, it is proposed to link the system of "points of view" with an integrated set of typical diagram types (templates) of the "task", and secondly, to integrate the subsystem TASK into the environment of the OwnWIQA for effective work with project tasks in the conceptual space of the project of any developed system.

3 Related Works

The standard ISO/IEC/IEEE 42010: 2011 envisages that any "concern" may be architecturally described with a coordinated group of useful "points of view" and corresponding "views." For example, the authors of [5] recommend considering such a construct as an "architectural decision" by using the Decision Detail viewpoint, Decision Relationship viewpoint, Decision Chronology viewpoint, and Decision Stakeholder Involvement. Moreover, the same authors in the following publication [6] proposed to expand the given set by the inclusion in it the Decision Forces Viewpoint.

Another trick is given in the publication [7], in which its authors suggest associating with the "Context Description Viewpoint" of the developed SIS the following concerns:

1. System Scope: Where is the boundary between the system and its context, and what interactions between the system and its context cross this boundary?
2. System Users: Who are the users of the system; what are their types, roles, and characteristics; and how and where do they access and use the system?
3. External Dependencies: Which external services and/or applications are relevant for the system, including their properties and providers?
4. Execution Environment: What is the expected or desired technical execution environment that the system will be running on?
5. Stakeholder Impact: Which stakeholders, including organizations and their resources, influence the system and in what way? What influence does the system have on organizations and stakeholders?"

Another possibility of accounting and materialization of "concerns" is their specification, distribution of a set of types and integration of a set of (distributed) constituent concerns within the framework of "architectural types." With such a possibility they associate an "aspect-oriented" representation and the materialization of concerns [8].

The real practice of architectural decisions is discussed in the industrial case study published in [9]. The current retrospection view on the theory and practice of architectural descriptions is presented in the paper [10].

In our version of the architectural approach to the precedent-oriented solution of the design task, their groups, and systems, it was decided to use all the cited possibilities

for recording, specifying and realizing the "concerns" and "architectural views" chosen for the development of the subsystem TASK. Note that if the project tasks of any SIS are solved in the OwnWIQA-environment, then the "concerns" and "architectural views" processed in the subsystem TASK will be taken into account and materialized in each instance of the task. So, they will be inherited by the created SIS, firstly, at the conceptual stage of its development, and secondly, in reusable applications of solutions. Thus, the version of the architectural approach presented in the article is created to solve design tasks of the SIS, but its instrumental materialization is used by the authors to build the subsystem TASK of the toolkit OwnWIQA.

4 Distribution of Tasks in the Conceptual Space of the Project

For the set of architectural views that make up the architecture of the developed SIS, it is required to be coordinated so that they form integrity corresponding to the theory of the SIS-project [3]. Since the theory of the project and the architecture of the SIS are built in parallel on the course of designing, they complement each other. Moreover, each architectural view must be the correct application of the project theory, and prompts its development, providing an interpretation of theoretical constructs and contributing architectural understanding.

Which means that it is useful to include the view from the viewpoint of theorizing in a coherent set of architectural views materialized in the OwnWIQA, the structure of which is shown in Fig. 2.

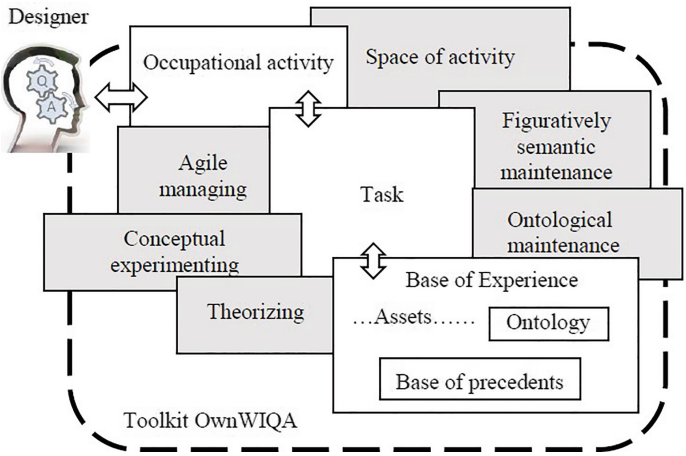


Fig. 2. Structure of the toolkit OwnWIQA from the viewpoint of working with tasks

In Fig. 2, there are no links between the blocks, but for the blocks, we use two colors (“white” and “gray”) to separate the main line of designer’ activity (white color) from the accompanying actions (gray color).

The main line of activity focuses on solving the project task, on the course of which the designer interacts with the necessary experience and its models that are accumulated in the Base of Experience. In the frame of such activity, the designer builds the reusable descriptions of the solved tasks (models of precedents) that are placed in the Base of Precedents. Indicated blocks of the main activity lead to the corresponding architectural views objectified in the OwnWIQA.

In actions of the main line, the designer uses automated support (blocks marked in gray) that help to increase the quality of working with tasks, their groups, and systems. Such blocks objectify an additional subset of the architectural views on work with project tasks in the toolkit OwnWIQA.

Below, we clarify the proposed version of the architectural approach to precedent-oriented work with project task only for the block TASK integrated all mastered reflections of the task onto QA-memory of the toolkit OwnWIQA. Namely, with this block, we bind the subsystem TASK, the architectural structure of which is schematically shown in Fig. 3.

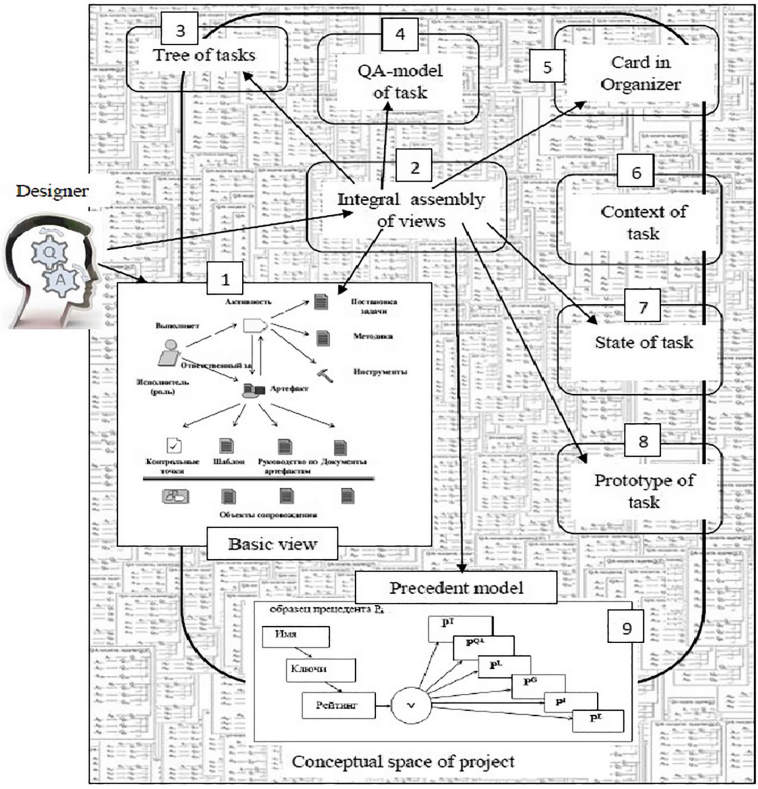


Fig. 3. The system of architectural views of the “task”

In the proposed version of the complete set of views on the task and their integration into a single whole, the following requirements were used:

1. For mastered activities noted in Fig. 1, all typical models of the task must find their expressions in the integral set of architectural views.
2. Each additional view of the task must be configured for its use in the semantic memory of the OwnWIQA environment.
3. Samples of typical views for a task and their integral set should be placed in the conceptual space [11] of the corresponding project.

Based on these requirements, a set of architectural views was selected. They are shown in Fig. 3 where names of views suggest what they are responsible for in the process of conceptual solution of the design task. Several indicated views found their informational descriptions in papers [1] and [3]. For this reason, below we concern only decisions for architectural views with numbers one, two and nine.

5 Prototyping of Architectural Solutions

A very important part of the OwnWIQA is a set of tools for programming in pseudo-code intended for two purposes:

- tools allow users (and therefore creators) of OwnWIQA to expand the functional potential of this toolkit;
- using these tools, one can build simple applications or application prototypes by programming a suitable shell above the OwnWIQA.

These possibilities helped us to develop a prototype version of the TASK linking prototypes for architectural views shown in Fig. 3. In this section, we will demonstrate

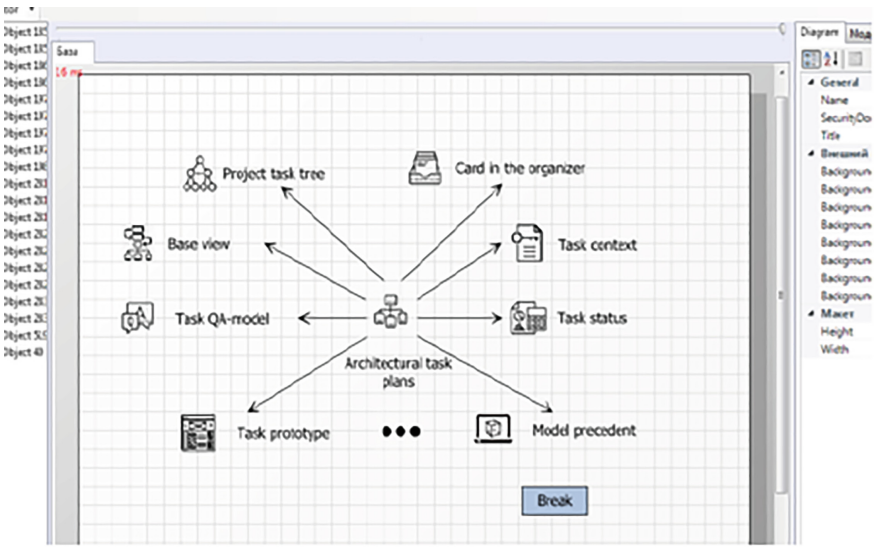


Fig. 4. Integral view of the "task"

what is behind pseudo-code prototyping for architectural views 1, 2, and 9. Let's start with the integral view, the visual (interface) form of which is shown in Fig. 4.

It should be noted, that Figs. 3 and 4 reflects only the projection of the TASK on visually accessible models of a design task, with which the designer interacts in the process of solution, applying activities and other artifacts of the scheme presented in Fig. 2. Among these models, very important places occupy those of them that belong to the main line of the designer's activity. One of these models corresponds to the basic view, the interface form of which is presented in Fig. 5.

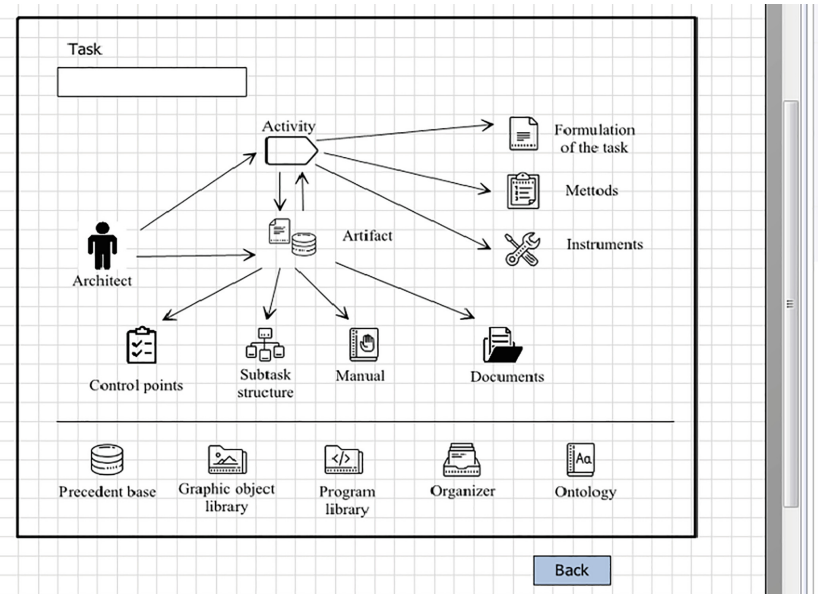


Fig. 5. Basic view of the "task"

The picture of the basic view is specially drawn with the use of the sample of the visual presentation of the task in the Rational Unified Process developed in Corporation IBM. In addition to the already noted indexing capabilities of structural elements, this view includes the navigation button "Back" and the (input/output) field for the task name, that binds the architectural view with the corresponding task. After that, the task instance is registered in the catalog of work tasks in the Organizer. For a specific instance of a task, its basic view can be easily modified if it proves appropriate.

For the considered architectural view, it should be noted that several artifacts accessible through this interface are formed when the task is resolved, and they are available when the designer go to them in their current state (for example, statement of the task, used practices or other artifacts).

All elements of the interface form of the basic view are also indexed. Some of them lead to software agents, each of which ensures the identity of the declarative components of the task that are used in other views. For example, one of these agents provides the identity of the current version of the task statement in the basic view and in the corresponding model of the precedent, which is built in the course of solving the task.

Another major artifact of the main line of designer's actions is the precedent model, the architectural view of which is shown in Fig. 6, where icons are clarified by names of components integrated into the precedent model.

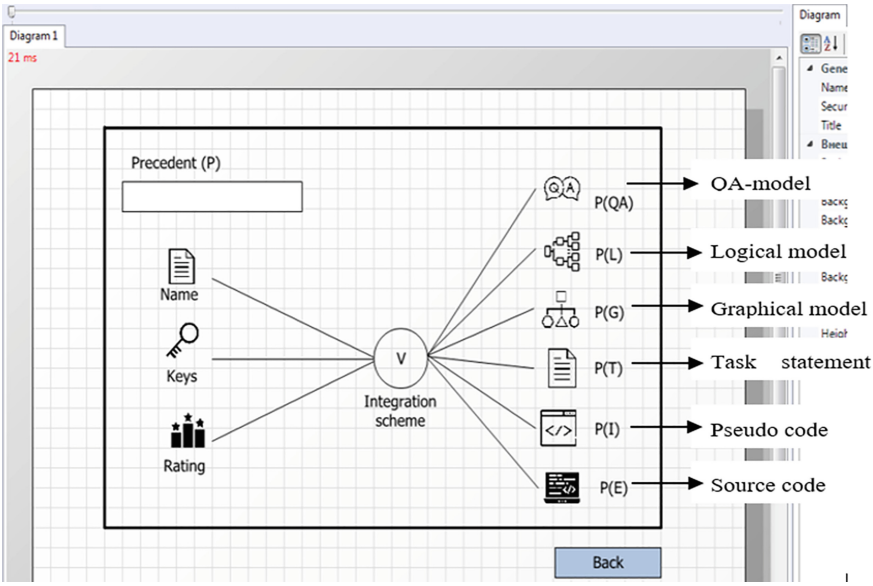
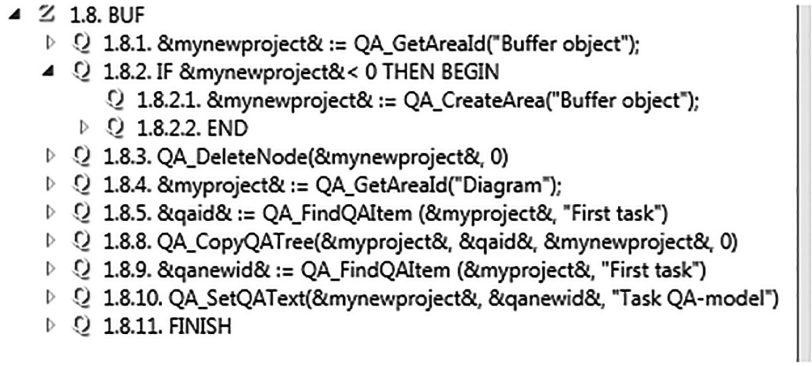


Fig. 6. Visualization of the precedent model

The structure of the model and the details of its components, taking into account their operational formation, are discussed in detail in [12]. For this architectural view, it is also important to ensure the identity of the content and structure of the available and processed conceptual objects represented by QA-nets.

To ensure the identity, we developed a utility for copying of QA-objects into a specialized buffer memory for subsequent loading from this memory onto the necessary place occupied by the corresponding view. The implementation of such utility includes operations of export and import of QA-nets with their intermediate representation in the form of XML-files. The necessary work is performed by the following pseudocode program:



The code is shown to demonstrate its syntax, allowing the use of procedures and functions that can be programmed both in pseudo-code and in other languages. In the presented code, the used functions are programmed in C#, and they selected from the library included in the toolkit OwnWIQA.

As noted above, the architectural representation of the TASK is aimed at using it as a sample for applying the proposed version of the architectural approach to the development of functional subsystems of any SIS, each of which is associated with the corresponding design task. At the same time, the TASK embodies very important functionality that is programmed on the executed pseudo code, and it is embedded in the OwnWIQA as its subsystem. For improving the quality of this subsystem, we have an intent to program it on C#.

6 Conclusion

The successful development of the modern SIS is almost impossible without the creation of its architectural description that is the source of numerous positive effects in all stages of designing. Usually, any artifact of this type is interpreted as a version of the SIS existence that discloses its structure and content in understandable forms helping to discover possible semantic errors. Building the architectural description for the SIS is a very qualified work that must be fulfilled by the architect or their group.

In any case, the architect solves a set of specialized tasks. Considering the usefulness of the architect's actions and their results, we develop the simplified version of tools that provide the use of the architectural approach in a precedent-oriented solution of project task at the individual workplace. The feature of the approach is the use of the system of architectural views on the task in the project.

The proposed version of architectural views is realized as the prototype version of means (called TASK) with the use of the toolkit OwnWIQA intended for the conceptual solution of project tasks. To simplify the possibility of adjusting the TASK by its users, we developed it with the use of drawn interfaces and pseudocode programming embedded into the toolkit OwnWIQA.

The kernel of the TASK consists of the basic architectural view integrating essential components of the task to be solved, and the precedent-oriented view accumulating the necessary models of the task for its reusable application.

Acknowledgments. This work is supported by the Russian Fund for Basic Research (RFBR) with Grants # 18-07-00989a, 18-47-730016 p_a, 18-47-732012p_мк, and Ministry of Science and High Education with the State Contract №2.1534.2017/4.6.

References

1. Sosnin, P.: Experience-Based Human-Computer Interactions: Emerging Research and Opportunities, p. 294. Hershey, IGI-Global (2017)
2. Clarke, P., O'Connor, R.V.: The situational factors that affect the software development process: towards a comprehensive reference framework. J. Inf. Softw. Technol. **54**(5), 433–447 (2012)
3. Sosnin, P.: Substantially evolutionary theorizing in designing software-intensive systems information, Switzerland, vol. 9, no. 4, paper № 91 (2018)
4. ISO/IEC/IEEE 42010:2011 Systems and software engineering - Architecture description, pp. 1–46 (2011)
5. Bedjeti, A., Lago, P., Lewis, G.A., Boer, R.D.D., Hilliard, R.: Modeling context with an architecture viewpoint. In: IEEE International Conference on Software Architecture (ICSA), pp. 117–120 (2017)
6. Van Heesch, U., Avgeriou, P., Hilliard, R.: Forces on architecture decisions - a viewpoint. In: Proceedings of the 2012 Joint Working IEEE/IFIP Conference on Software Architecture and European Conference on Software Architecture, pp. 101–110. IEEE Computer Society, Washington, DC, USA (2012)
7. Van Heesch, U., Avgeriou, P., Hilliard, R.: A documentation framework for architecture decisions. J. Syst. Softw. **85**(4), 795–820 (2012)
8. Hilliard, R.: Using aspects in architectural description. In: Moreira, A., Grundy, J. (eds.) EAW 2007. LNCS, vol. 4765, pp. 139–154. Springer, Heidelberg (2007). https://doi.org/10.1007/978-3-540-76811-1_8
9. Dasanayake, S., Markkula, J., Aaramaa, S., Oivo, M.: Software architecture decision-making practices and challenges: an industrial case study. In: Proceedings of 24th Australasian Software Engineering Conference, pp. 88–97 (2015)
10. Cuesta, C.E., Garlan, D., Pérez, J. (eds.): ECSA 2018. LNCS, vol. 11048. Springer, Cham (2018). https://doi.org/10.1007/978-3-030-00761-4
11. Sosnin, P.: Question-answer nets as a valuable source of information in designing the software intensive system. In: 25th Telecommunications Forum, TELFOR 2017 - Proceedings, pp. 1–4 (2017)
12. Sosnin, P.: Precedent-oriented approach to conceptually experimental activity in designing the software intensive systems. Int. J. Ambient Comput. Intell. **7**(1), 69–93 (2016)