ISSN 2278-3091
Volume 9, No.4, July – August 2020
International Journal of Advanced Trends in Computer Science and Engineering
Available Online at http://www.warse.org/IJATCSE/static/pdf/file/ijatcse157942020.pdf
https://doi.org/10.30534/ijatcse/2020/157942020

Approaches to the Quality Assessment of Software System Design in the Development of Innovative Solutions

A.A. Ryndin[1], N.A. Ryndin[2], S.V. Sapegin[3]
[1]Voronezh State Technical University, Voronezh, Russian Federation; alexander.ryndin2017@yandex.ru
[2]Voronezh State Technical University, Voronezh, Russian Federation
[3]Voronezh State University, Voronezh, Russian Federation

ABSTRACT

One of the components of Industrial Revolution 4.0 is the focus on generating full cycle innovations, of which software systems are integral parts. The article deals with the approach to quality assessment of decisions made within the framework of the development of program systems in innovation activities. The research objective is to develop approaches to the assessment of quality and importance of various parameters of software systems development in the context of software development to support innovative solutions. The article uses methods of linear and nonlinear optimization, system analysis, and the theory of information processes and systems. The research finding is a development model of program systems as components of full-circle innovations, a set of ranked quality indicators and approaches to forming a paradigm of program systems under development depending on their purposes.

Key words: Industry 4.0, full-cycle innovation, multiple integration, software development.

1. INTRODUCTION

One of the most exciting trends in modern technological development is the so-called Fourth Industrial Revolution. Its meaning is the emergence of Industry 4.0 which represents a productive economy associated with the massive introduction of cyber-physical systems of various profiles. One of the integral components of Industry 4.0 is continuous innovation, the results of which are "full-cycle innovations". These are developments that have gone from basic ideas and prototypes to a reproducible and scalable business model. The popularity of such innovations can be seen in the number and development dynamics of technological startup projects, in the unprecedented measures of their support by various states and corporations, as well as in the scope of service organizations engaged in growing and promoting startup projects, including various business incubators, accelerators, etc.

Modern full-cycle innovations, around which technological startup projects are built, can both include fundamental, applied and engineering solutions of various levels of maturity and simply manage without these depending on the specifics of the project. However, within the framework of the Industry 4.0 concept, almost no innovative solution that claims to scale in the cyber-physical landscape of the near future can manage without the software being developed. Moreover, in half of the cases, it is the software being created that is the essence of the innovative solution, and in many other cases, software development is the core of innovation coordinating the process of project development from the initial idea to a scalable business model, as demonstrated in works [4, 5].

Thus, competent organization of software development within the framework of creation and development of full-cycle innovation is the cornerstone of the whole project success. As the Standish Group Chaos Reports demonstrate, the success rate of software development projects in the industry as a whole does not exceed 30%, and this indicator, by our estimates, is much lower in the field of startup projects. According to Startup Genome Report statistics, 92% of startup projects close in the first five years, moreover, at a rough estimate, more than two-thirds of them fail due to software development problems and the resulting failures in business targets. These failures range from the inability to focus on the target market segment to problems in scaling the solution leading to excess expenditure and serious organizational challenges.

It is curious that the use of any methodology from today's popular set of Agile methodologies is not a guarantee of achieving results. The study shows that success is achieved by teams using completely different development methodologies. In this case, the following arguments are stated to justify the rejection of the classic methodologies of Agile, such as Scrum, Kanban, etc.:

1. Labor costs of implementation are unjustifiably high;
2. Benefits of implementation are unobvious;

3. The dynamics of the development process is not quite adequately reflected.

Thus, a search for a methodology of the rational organization of software development within the conditions of creating full-cycle innovations represents an urgent problem for the moment.

2. SOURCE REVIEW

Let us consider the main specific features of software development methodologies. Among the total number of applied methodologies of the software development organization, it is possible to distinguish some generations connected with the orientation to various aspects of the industry, playing a key role throughout some period of time.
The first generation of approaches of software system development can be conventionally considered a set of methods and tools that provide programming in machine-independent languages and provide the possibility of transferring developed software between different models of computers and even different operating systems. In these circumstances, for the first time, the creation of software systems is considered not as one of the workflows of electronic devices development process, but as a separate task that possesses its own specific character and requires adaptation of common engineering approaches [14, 15]. Quite quickly, within the framework of this set of approaches, the developers faced the following problems:

1. Avalanche-like increase in the complexity of programs as their size increases;

2. The unpredictability of terms and volumes of software system development;

3. Difficulties in transferring knowledge about ready-made software systems from developer to developer, according to monograph [1].

As part of the solution of the abovementioned tasks, the world community has come to a generation of development tools and methodologies related to the concept of CASE (Computer-Aided Software Engineering). The specific feature of this generation has become the implementation of the structural approach in software development. As design tools for program packages, modeling methodologies have been developed, the part of which has remained fixed in IDEF set of standardized methods. Among them, it is possible to underline the methodology of ER-modelling, which is still applicable to the issues of development of data structures for relational databases. As a development methodology, the most popular and practical approach has become the use of the waterfall model. The generation of CASE tools allowed for a significant expansion of the field of IT application, as well as for designing and programming large software systems capable of covering entire segments of corporate activities. However, over time, the following disadvantages and specific features of this generation of tools have become apparent:

1. The stages of requirement analysis and system design are of great importance and have relatively high costs, as is proven in monograph [1].

2. For the overwhelming number of branches and automation objects, changing user, technical and organizational requirements is a rule, instead of an exception.

3. Successful building of systems in a structural paradigm requires developers to have a serious enough level of professionalism.

The complex of techniques, methods and means, initially developed to deal with the listed disadvantages, made it possible to form the next generation of approaches premised on methods of object-oriented design and programming, as well as on the principles of the iterative and customizable development process. The idea of the iterative approach that means work performed in parallel with continuous analysis of the results obtained and adjustment of the previous stages of work allowed for the significant improvement in the characteristics of software systems under development. From the perspective of this approach, the project in each phase of development should go through a repeated PDCA (Plan-Do-Check-Act) cycle. The specific features of this approach can be seen on the example of one of the most popular methodologies of this generation, Rational Unified Process (RUP). According to RUP, the composition of the development process stages can be revised, and within each of these stages there appears the division of the process into separate iterations, and as the result of each iteration, the existence of the finished product is implied, as demonstrated in works [2, 7].

The next generation of approaches to software design and development as well as to process management announced itself in February 2001 in the form of the so-called Agile Manifesto which defines the basic principles and approaches of flexible software development. The appearance of a family of Agile methodologies was caused by the following factors:

1. The rapid development of the Internet network has led to tightening of global competition in software development;

2. The middleware-technologies have entered into maturity. This made it possible to integrate different software not only at the level of data and files but also through function calls and even collaborative abstractions of Object Oriented Programming (OOP) such as components and interfaces;

3. The wide range of tasks solved by software development, the abundance of technologies being used and the rate of appearance of new technologies have increased the

importance of communication in projects between developers of different profiles. The practice of teamwork organization has gained great importance.

The need for Agile principles to be put into practice has stimulated the development of technologies that simplify the deployment, operation and support of software systems. The popularity of scripting languages, focused primarily on the work in the global network has significantly increased. Methodologies and technologies for the focused elimination of bottlenecks of software under development (Continuous Integration, TDD) have appeared. It was demonstrated in several works [3,11, and 13] that the need for thorough, complete software design, due to the spread of the service approach and the reduction in the software size as a consequence, has lost its importance giving way to the ideology of several most commonly used templates (MVC, etc.).

At the same time, almost all methodologies of Agile generation have the following disadvantages:

1. Implementation lead-time and results of development are unpredictable. Methodologies of Agile generation focus primarily on achieving maximum quality in terms of both the implementation of user requirements and the construction of system architecture. However, in practice, this often leads to misdirected priorities of the project and, as a consequence, the failure of the initial deadlines or poor quality solutions caused by lack of time at the end of the project.

2. The development process is strongly influenced by the human factor, namely the motivation of employees, their professional qualities and experience. Improper selection of project implementers may lead to the project failure due to insufficient qualification of its participants, or, in case of excessive qualification of employees may lead to overspending of funds and a decrease in motivation in the team.

3. In many cases, teams accustomed to organizing their activities according to Agile patterns ignore strategic planning within the framework of the project, which increases the risk of failure due to excessive attention to details.

The abovementioned specific features of Agile methodologies are quite a significant stumbling block in the environment of startup projects which often leads to the failure of general development or significant difficulties and significant budget overspending. At the same time, there are successful cases of software development for full-cycle innovations using methodologies of other generations premised, in particular, on a cascade model as well. In these cases, the advantages of the used methodologies exceed the effect of the application of Agile family methodologies which makes it possible to achieve positive results. In this paper, we will try to consider the essential features of each generation of methodologies and determine their applicability for use in startup projects.

3. METHODOLOGY

Let us consider specific features of software development in the conditions of the creation of full-cycle innovative decisions. It is possible to formulate the following specific features of the teams working in projects of such kind:

1. A short history of operational activities forces developers to use solution templates that have not been tested in the current team composition. This applies both to software development methodologies and to the selection of software implementation tools, software composition, approaches to database and architecture design, etc.

2. Bringing participants together on the basis of cultural values, on the one hand, may ensure the breadth of expertise in planning the functional and architectural features of the software being developed and, on the other hand, may significantly distort the balance of decision-making. It is not uncommon, when the team does not have the necessary competencies when working on a significant architectural issue and participants tend to choose in general non-optimal but known in other projects decisions. For example, in the selection of database, the developers who do not have competences in the fields of relational databases, replace it with any of the non-relational databases known from previous projects, even if this is extremely unjustified from the overall solution rationality point of view.

3. The specific feature is an extremely high degree of uncertainty and its specifity. The uncertainty in innovative projects arises to a lesser extent due to the specifics of communication (shared cultural values play a role), but to a large extent, the uncertainty is an integral characteristic of the innovativeness of the solution itself. The process of finding a rational solution in the framework of an innovation project is generally nonprogrammed, unpredictable and can only be planned to a first approximation.

Among the specific features of startup projects that affect software development, we can also highlight the following:

1. A rigid time schedule is associated both with the need to obtain and test on the market various solution options and the process of attracting investments which includes the need for periodic product demonstrations at certain, unknown beforehand moments of time.

2. The attraction of external expertise is necessary, and in many cases, we speak about non-core expertise, because the project area, as well as technologies of building software, can vary within a wide enough range.

3. The necessity of consistency check of decisions at various stages of the innovation development cycle and also the necessity of using the partially ready software for checking a business idea determine orientation towards the full-cycle decisions.

Proceeding from the abovementioned features, it is possible to draw a conclusion that in different projects, depending on the influence of one or another specific feature on the result of the project, it is possible to use methodologies of different generations if their features correlate with the features of the specific project as closely as possible. However, in the general case, a synthesis method of a hybrid methodology premised on different principles and most suitable and adequate for a specific project is required. To solve this problem, we will consider a mathematical description of the software development process.

Taking into account such a specific feature of startup projects as the necessity to have a full-cycle solution at key points of the project, we will consider the software being developed as a developing software system, as assumed in monographs [8, 9, and 10]. The process of the mathematical description of developing information systems (IS) is characterized by a number of specific features, which affects the formalized setting objectives for the selection of rational options, as well as methods for their solving. In particular, we have combinatorial uncertainty in the selection of the optimal option, uncertainty in the optimization objective selection due to the multiplicity of technical and economic requirements, probability of criteria for the performance evaluation of options and uncertainty of the mathematical dependence definition of the system indicators on the parameters of variable components of the structure.

The necessity to take into account system communications, as well as the joint influence of several types of uncertainties, leads to the classes of models for which it is inefficient to obtain an accurate solution. The reason is that in the course of searching for an optimal option, the possibility of analyzing a group of dominating options of IS structure is lost. At the same time, it is possible to expand the possibilities of the system approach in designing corporate ISs, if to use as a quantitative characteristic of the rational choice of IS organization options the information characteristic – the entropy of multiple integration. The entropy defines the degree of diversity of many possible integration options. In the conditions of dynamic development and changes in the structure of corporate IS, it is often the entropy that is the only indicator based on which it is possible to choose the solution which ensures the optimal development of the corporate system in conditions of uncertainty.

Proceeding from the concept of the service approach for building the IS architecture, it is possible to consider the trajectory of corporate IS development as a sequence of procedures for continuous quality improvement of existing IT services and introduction of new ones into operation. At the same time, in reality, there are complex, multistage links between the existing and new IT services of ISs. These links implement the mutual influence of services. Design, implementation and operation of each IT service, in its turn, can take place according to different methodologies, which implement different structures and approaches depending on the nature of the IT service, technologies used, approaches, IT service size, the scope of user coverage, etc., as described in monograph [6].

Let us consider the process of a software package development in the form of a design flow consisting of a task sequence of a multi-criteria selection, along with this, the result of each choice influences the trajectory of the subsequent system development. Elements of choice in the process of designing of corporate IS are elements of the vector $\vec{W}_g$, each of which, in its turn, contains a set of components making up the system. Elements of choice $w_g = \overline{1, W_g}$ as a whole specify the variant

$$S_l = (w_1, w_2, ..., w_g, ..., w_G) \in S \tag{1.2}$$

and are characterized by the vector of parameters f_{Wg}

In the transition from one implementation $w_g \in \overline{1, W_g}$ to another, vector components f_{Wg} change in an incremental fashion. An attempt of synthesis optimal from with the view of duration and cost of the design flow and the means implementing it by establishing the dependence of technical and economic indicators of the system $F_i (i = \overline{1, I})$ on the parameters of the elements $F_i = \Psi(f_{Wg})$ and determining the values f_{Wg} that implement the requirements $F_i (i = \overline{1, I})$, in most cases leads to parametric solutions. However, it is not possible to put a certain design route and the structure $w_g \in \overline{1, W_g}$ implementing it in accordance with these parametric solutions in these circumstances. Therefore, the choice S has to be made from many options $(l = \overline{1, L})$ that represent possible combinations of competing technologies and integration options $w_g \in \overline{1, W_g}$ $(g \in \overline{1, G})$. Consequently, one of the features of the problems of building large software packages is the presence of combinatorial uncertainty and uncertainty of mathematical description.

Selection of the design flow and the technical means of its implementing which are optimal from the point of view of the complexity of variant implementation and taking into account the multivariance of devising the IS architecture and integration principles is carried out according to a set of technical and economic indicators $F_i(i=\overline{1,I})$. The chosen design flow should ensure the design of corporate IS with the specified operating characteristics such as reliability, efficiency, functionality, etc. The IS operating characteristics form a subset that determines the choice of variable elements of the design flow. Thus, the use of multiple integration principles makes it possible to solve the problem of the mathematical description of developing software systems during their whole life cycle. Besides, the use of the same approach makes it possible to determine the ways, methods and means of rationalization of the software system development process proceeding from the realities of different approaches. It also makes it possible to carry out a combination of representations of these approaches to achieve the most qualitative result.

A combination of different approaches is proposed in order to investigate the engineering process of developing software systems. Such a practice will make it possible to achieve the following results:

1. Correct formulations of the tasks of rationalization from the technical and economic point of view of the software development process for each family of approaches including restrictions that have been formulated based on the development process analysis from the point of view of other representations.

2. General problem definition of the rationalization process problem of program components development, based on the principles of the system approach, taking into account different representations of the system and making it possible to find the most rational solutions by finding compromise variants or through the integral analysis. In this case, the rationality of solutions is determined based on their empirical adequacy and technical-and-economic feasibility.

4. RESULTS

In general, the multiple integration approach consists of four problems of structural synthesis of ISs and their corresponding local multiple optimization models $\mu_1 - \mu_4$ which can be used in the framework of a customizable process of optimal design of developing systems. According to works [2, 6], the problems are formulated as follows:

B1. Variety constraint of component sets at different levels of integration.

B2. Selection of the effective alternative component integration option, taking into account existing integration levels.

B3. Selection of the order of precedence of project operations.

B4. Grouping elements of sets of different integration levels into local design flows.

In cases when the design object represents the difficult, poorly formalized system, or design process initially assumes the use of the iterative approach, stage-by-stage use of multiple optimization models can essentially reduce expenses of design stages. It can be done even taking into account the use of structural synthesis procedures at separate design stages within the limits of PDCA (Plan-Do-Check-Act) model (figure 1).

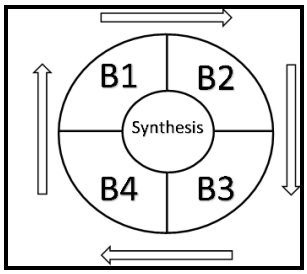

Figure 1: Step-by-step use of multiple-path synthesis procedures.

5. DISCUSSION

The need to take into account system relations, as well as the joint influence of several types of uncertainties in the process of finding rational solutions, leads to the development of models in which obtaining an accurate solution is inefficient. It happens because while searching for the optimal option, the possibility of analyzing a group of dominant solutions is lost. At the same time, it is possible to expand possibilities of the system approach for the solution of design and development problem of program components if to use as a quantitative characteristic of a rational choice of options the information characteristic – the entropy of multiple integration. The entropy defines the degree of diversity of many possible integration options. In the conditions of dynamic development and changes in the structure of rational solutions, it is often the entropy that is the only indicator based on which it is possible to choose the solution which ensures the optimal development of complex software packages in conditions of uncertainty.

In this case, as the most effective directions for expansion of possibilities of the system approach, we can specify the following:

1. Construction of the flexible, customizable architectures based on principles of object-oriented designing (OOD) and

the component approach, multiplatform realization, wide use of the weakly connected decisions based on the type parameterization, use of messages and "soft" standards of software interaction.

2. Quality improvement of software components by introducing popular templates and standards for their development as well as the increase of component manufacturing speed, ease of support, modification and adjustments, and compliance with popular design patterns due to used standards and practices.

3. Building and debugging a flexible software development process that includes continuous testing and integration using human resources as effectively as possible.

6. CONCLUSION

One of the determining factors causing problems and uncertainties in the software development process is the constant evolutionary development of both modern software systems and their environment. Proceeding from it, one of the perspective directions of searching for the unified approach, capable of rationalizing the processes of building large systems, is the consideration of program systems as developing, changing their structure and characteristics during the whole process of development, introduction and operation. It is expedient to overcome the arising complexities connected with the high dimensionality of combinatorial tasks encountered in the process of research of program systems taking into account the dynamics of their development based on the use of multiple integration methods. With the help of these methods, a multistage approach to the solution of arising tasks is developed, including estimations of the effectiveness of solution at each stage.

Thus, when using methods of complex research of the development problem of modern program systems, the search of a unified approach to their design process, maintenance, analysis and modeling, as well as the organization of interaction of programs, program systems and their components, it is necessary to take into account the context of the problem, formed by processes beyond the framework of the development itself. However, the focus on the paradigm of Industry 4.0 and the specific features of innovative projects within this paradigm may be one of the ways to improve the efficiency of software development processes for a wide variety of purposes.

REFERENCES

1. F.P. Brooks, Jr. ***The Mythical Man-Month: Essays on Software Engineering, Anniversary Edition (2nd Edition)***, University of North Carolina at Chapel Hill University: Addison-Wesley Professional, 1995.
2. B. Boehm. **A Spiral Model of Software Development and Enhancement**, *IEEE: Computer*, vol. 21, no. 5, pp. 61-72, May 1988. https://doi.org/10.1109/2.59
3. G.N. Krieg. ***Kanban-Controlled Manufacturing Systems***, Germany: Springer-Verlag Berlin Heidelberg, 2005.
4. K. Schwab. ***The Fourth Industrial Revolution***, World Economic Forum, 2016.
5. K. Schwab. The Fourth Industrial Revolution: what it means, how to respond, World Economic Forum, 2017.
6. Ya.E. Lvovich. Multi-alternative optimization: theory and application, Voronezh: Kvarta, 2006.
7. A.K. Shuja, J. Krebs. IBM Rational Unified Process Reference and Certification Guide: Solution Designer (RUP), IBM Press, 2007.
8. R. Reussner, M. Goedicke, W. Hasselbring, B. Vogel-Heuser, J. Keim, L. Märtin. ***Managed Software Evolution***, Springer Open, 2019. https://doi.org/10.1007/978-3-030-13499-0
9. D. Dell'Anna, F. Dalpiaz, V. Dastani. Requirements-driven evolution of socio-technical systems via probabilistic reasoning and hill climbing, Springer US, 2019.
10. A.A. Ryndin. ***Multiple integration: theory and applications in CAD: monograph***. Voronezh: Voronezh State Technical University, 2018.
11. Object Management Group, Essence. Kernel and Language for Software Engineering Methods (Essence), 2014.
12. E. Klotins, M. Unterkalmsteiner, T. Gorschek,. **Software engineering in start-up companies: An analysis of 88 experience reports**, *Empirical Software Engineering*, vol. 24, no. 1, pp. 68–10, 2019. https://doi.org/10.1007/s10664-018-9620-y
13. I. Jacobson, Ng Pan-Wei, P.E. McMahon, I. Spence. **The Essence of Software Engineering: The SEMAT Kernel**, *Communications of the ACM*, vol. 55, no. 12, pp. 42-49, December 2012. https://doi.org/10.1145/2380656.2380670
14. D.T. Utomo, Pratikto, Santoso, P.B. Sugiono. **Preliminary Study of Web Based Decision Support System to Select Manufacturing Industry Suppliers**, *Journal of Southwest Jiaotong University*, vol. 55, no. 2, 2020.
15. Z. Feng, J. Liu, L. Peng, Y. Zhou, P. Ning, B. Wang. **New Development of CAE Platform and Computational Mechanics Software**, *Journal of Southwest Jiaotong University*, vol. 51, no. 3, 2016.