DE GRUYTER OPEN
RIGA TECHNICAL UNIVERSITY
Applied Computer Systems
ISSN 2255-8691 (online)
ISSN 2255-8683 (print)
May 2018, vol. 23, no. 1, pp. 63–68
doi: 10.2478/acss-2018-0008
https://www.degruyter.com/view/j/acss

Evolution and Development Prospects of Information System Design Methodologies

Lyudmila Onokoy[1*], Jurijs Lavendels[2]
[1] *Financial University under the Government of the Russian Federation, Moscow, Russia*
[2] *Riga Technical University, Riga, Latvia*

***Abstract* – The article investigates different approaches to the design of information systems. Much attention is paid to comparative analysis of criteria for selecting methodologies for software development, and also to not well-known methodology of DevOps (Development & Operation) [1], [2], which aims at consolidation of software developers (Development) and IT professionals' (Operation) efforts, and automation of implementation process. In conclusion, based on the retrospective analysis and practical experience, the authors formulate regularities and prospects of information systems design methodology development.**

***Keywords* – Heavyweight and agile methodologies, lean software development, methodology of DevOps.**

I. Introduction

Twenty years ago the information systems design was more like art, rather than the result of engineering activities. This was due to the fact that the development was mainly carried out by non-formalized methods based on practical experience of the project team, expert estimates and expensive experimental validations of the quality of information systems functioning. As a rule, the design process did not fit in time and budget, and the information system did not totally match the requirements of the customer. There are data that by the end of the last century only 26 % of IT projects for information systems development were successful [3].

To overcome the above-mentioned drawbacks, IT professionals attempted to develop the methodology of information system design that provides unification and standardization of the design process based on the systemic principles featuring the industrial production (mass production, repeatability of processes and components, reliability, etc.).

One of the main stages in the design of information systems, which largely determines the quality and functionality of the future system, is the stage of software development.

Currently, there are several dozens of different methodologies of software development.

II. Agile and Heavyweight Methodologies of Software Development

One of the most well-known methods until now is the Rational Unified Process (RUP), created in the second half of the 1990s by the Rational Software company, and based on the principles of object-oriented design [4]. The RUP Methodology accumulates experience of a large number of developers and includes a set of concepts that are present in other techniques. RUP meets the standards and regulations, such as ISO 12207, ISO 9000, CMMI, etc.

The iterative process of software development described in the RUP methodology has two dimensions: the content is split into disciplines, and the time – into phases. All disciplines and phases are described in detail in the RUP documentation in the context of individual roles, activities and obtained results (artifacts). The RUP methodology contains a description of more than 30 roles, 20 activities and 70 artifacts. Thus, the design process described in the RUP methodology is strictly regulated [5], [6].

The RUP belongs to the class of heavyweight methodologies. It is intended to support the IT design process of any complexity and volume. The consequence of the universality of this methodology is its excessive complexity and inability to give high performance and usability in the design process. Therefore, in practice, an information system design process based on RUP usually starts with the selection of roles, activities and artifacts to simplify the project planning and to reduce financial costs and time.

The Microsoft Solutions Framework (MSF) is also a heavyweight software engineering methodology [7]. The MSF specifies in detail various aspects of the project, such as its phases and milestones, project team, task assignments, etc. In this methodology, special attention is paid to project preparation and risk management.

In 2001, with the publication of the Agile Manifesto [8], the agile software engineering methods emerged [9], which focused on reducing mandatory regulations and increasing in degrees of freedom for the design team. The basic principles of agile methods pursue to overcome the disadvantages of heavyweight methodologies. Therefore, agile methodology of software development gives priority to project team members over processes and tools, to working program – over documentation, adjustment of changes in requirements – over compliance with the established plans [10]. An important innovation of agile methodologies is mandatory customer's involvement in the design process, due to the necessity of periodical tests of releases and, consequently, to possible changes in requirements. Thus, agile methodologies provide the blurring of boundaries between the customer and the project team; they work on the project together, constantly interacting [11], [12].

* Corresponding author's e-mail: OnokoyLS@fa.ru

Class of agile methodologies currently includes quite a lot of specific methodologies. To the authors' opinion, the most interesting ones are the methodologies that have gained wide practical application: SCRUM, extreme programming (XP), Kanban, LEAN [11], [12].

Scrum methodology is based on time-management principles, which have been efficient for IT project development. Scrum includes the detailed description of the three main components: roles, artifacts and processes. The main roles are Scrum Master, Product Owner, and Scrum Team. Scrum Master is responsible for productive interaction between the project team members and external relations. Scrum Master is also responsible for the methodology principle implementation. In turn, Product Owner represents the interests of the customer in the IT project and is responsible for the quality and functionality of the software product. Scrum Team develops the software product and is responsible for it to the Product Owner [13], [14], [15].

The iterative process of software development involves time-limited iterations (Sprints), at the initial stage of which new requirements are introduced. Each Sprint ends with a new release for the software product increment. The specific feature of the Scrum methodology is a mandatory meeting of all project team members to discuss current problems (daily meetings), to introduce new functionality to the Product Owner at the end of the sprint (sprint reviews), and to discuss the results of the sprint (retrospective).

To its advantage, the Scrum methodology strives to satisfy the customer, shows significant adaptability to the new requirements, as well as ability to start the project when the requirements are still incomplete. The significant drawback of Scrum is strong dependence of the development process efficiency on the skills, self-organisation, cross-functionality and stability of the team. In consequence, typical Scrum projects involve high costs on the formation of the team, its training and motivation. Besides, the implementation of the Scrum principles means significant loss of time. In practice, daily meetings, sprint reviews, and sprint retrospectives take about 30 % to 40 % of project time. In real software development projects, the team members usually combine several roles (Scrum Master and Scrum Team member, or Product Owner and Scrum Team member) to reduce costs.

XP (eXtreme Programming) [16] is focused on software development in case of uncertainty and/or inconstancy of requirements. XP consists of management and engineering practices. Special attention is given to the engineering practices such as TDD (Test Driven Development) and Pair Programming, providing high quality software development. XP supports iterative design process and consists of short iterations (minimum duration of a few seconds). In XP process the participation of customer's representative is mandatory [17], [18].

Advantages of XP:

- guaranteed customer and user satisfaction by the quality and functionality of the software product;
- high-speed development and high quality of the resulting software product.

Disadvantage of XP: strong dependence of the project results on the customer representatives' participation and their ability to formulate requirements and evaluate the obtained results, as well as the skills of developers.

The Lean methodology of software development [19], [20] contains seven principles of adapting successful production management practices to the software development process. Lean methodology is described in detail in the book "Lean Software Development" by Mary and Tom Poppendieck. The authors suppose that software development project efficiency increases, when the developers focus on increasing product value for the consumer and reducing losses. Lean methodology considers as losses to project the redundant requirements and functions of the software product, unreasonable extras in development process, code error, project team downtime, etc. To eliminate losses, it is necessary to manage and put under control all the stages of software development.

The Lean methodology allows speeding up software development and improving its quality, thus reducing the costs.

In practice, the disadvantage of the Lean methodology is excessive requirements to qualification of the project team members and constant monitoring of software development processes management and effective decision making.

The Kanban methodology contains only three rules, which regulate the software development process, divided into separate tasks. Kanban uses mandatory visualisation of task performing process (Kanban Board), limits the number of simultaneously running tasks at every stage of design, optimises the design process by the analysis of task run-time [11], [12], [21].

The Kanban methodology makes a good supplement to other methodologies (Scrum, Lean, etc.) and speeds up development progress significantly.

The disadvantage of the Kanban methodology is team size limited to 5 members, whose cross-functionality is not allowed, which imposes significant limits on the scope and complexity of design.

Figure 1 shows the level of popularity of the said agile methodologies according to the survey Version One (2016) [22].

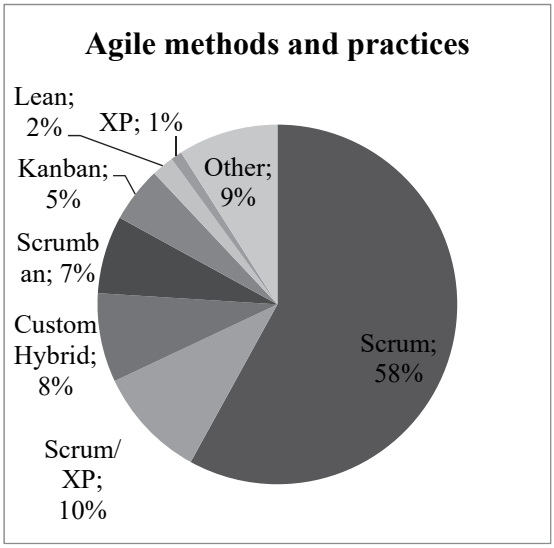

Fig. 1. The popularity of agile methodologies and practices.

In practice, the agile methodologies of software development are not universally applicable and have significant limitations for complex and important projects, which are critical in terms of confidentiality. They are mainly applied to small projects or projects that are easily split into components. They show good results for IT start-ups. In the agile methodologies increased requirements are imposed on the project team (usually small in size), and in particular, on the members' high skills, self-organisation and commitment, because they are expected to make project management decisions on their own, ensuring its efficiency and development quality [11], [12], [20].

III. Selection Criteria of Software Development Methodologies

One of the important tasks of software development project is the choice of methodological approach to its implementation. According to the authors, the methodology selection criteria should be considered in terms of (Fig. 2):

- the main characteristics of the software development project;
- parameters of the project group;
- customer's characteristics.

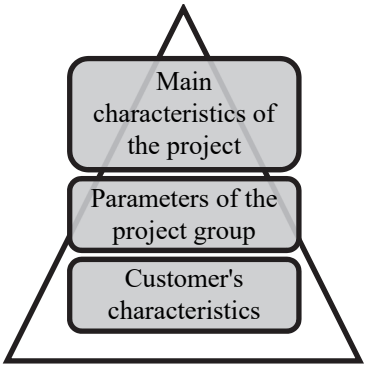

Fig. 2. Classification of selection criteria for software development methodology.

To the authors' opinion, in the first class of methodology selection criteria the following project characteristics are the most significant:

- importance and complexity of the project;
- volume of development;
- time;
- budget;
- documentary support;
- requirements for software development;
- safety and risks;
- specifics of the subject area, which impose restrictions on the choice of methodology.

In turn, the class of the project group parameters, according to the authors, should be divided into subclasses of objective and subjective criteria. Indicators of objective criteria are measured with specific values, and therefore their implementation is easy to manage. The subjective criteria have no obvious units to be measured; therefore, they are rather hard to be applied.

Objective criteria include:

- team members' number;
- qualification level of the team members;
- cross-functionality of team members;
- stability of the team;
- location of team members.

The most important subjective criteria are:

- project team members' experience in using specific methodology;
- project team commitment to teamwork, discipline and self-organisation.

The characteristics of customer's organisation, according to the authors, should also be divided into objective and subjective criteria.

As objective characteristics of the customer it is necessary to mention:

- readiness of customer's representatives to participate in the project;
- customer's high qualification;
- readiness for the introduction and testing of prototypes at a fixed frequency.

The subjective characteristic of the client, which influences the choice of software development methodology, is a trust relationship with the project team.

Table I presents the values of the criteria with regard to the selection of various methodologies. On the basis of the data provided in Table I and actual multi-criteria selection methods, it would be easy to make a substantiated choice of methodology for software development.

In practice, the increase of project team and/or of project importance means pursuance of binding regulations, and increases methodology hardness (density). The harder the methodology, the higher the project costs and its duration. The less difficult the methodology, the more autonomy a project team has in management and engineering solutions, and therefore the higher requirements for skills, discipline and self-organisation of the team. The modern design process of large and complex information systems is often implemented with the use of both heavy weight methodologies for software modules with high requirements to security and risk minimisation, and agile methodologies for software modules with frequently changing requirements.

However, the emergence of agile methodologies in software development has not solved the longstanding contradiction between software developers and IT professionals who are responsible for its operation. The essence of this contradiction lies in different tasks and different areas of responsibility of development and operation professionals. Thus, developers are interested in more releases, improving software and expanding its functionality. In turn, IT professionals are interested in stable software to ensure trouble-free operation of information system.

TABLE I

VALUES OF THE SELECTION CRITERIA FOR SOFTWARE DEVELOPMENT METHODOLOGY

CRITERION	RUP	MSF	SCRUM	XP	LEAN	KANBAN
1. Main characteristics of the project						
Importance and complexity of the project	High	High	High	Low	Low	Low
Volume of the IT project	Large	Large	Small	Small	Small	Small
Time	Rigidly fixed	Rigidly fixed	Can be adjusted	Can be adjusted	Can be adjusted	Can be adjusted
Budget	Rigidly fixed	Rigidly fixed	Can be adjusted	Can be adjusted	Can be adjusted	Can be adjusted
Documentary support	Necessary	Necessary	Not necessary	Not necessary	Not necessary	Not necessary
Requirements for software development	Fixed	Fixed	Changes are permissible	Frequent changes are permissible	Changes are permissible	Changes are permissible
Project safety	High	High	Low	Satisfactory	Low	Low
Project risks	Manageable	Manageable	Not manageable	Not manageable	Not manageable	Not manageable
Specifics of the subject area	No limits	No limits	With restrictions	With restrictions	With restrictions	With restrictions
2. Parameters of the project group						
2.1. Objective criteria						
Team members' number	More than 40–50 people	Groups of 3–10 people	5–9 people	Up to 10–15 people	5–10 people	Up to 5 people
High level of team qualification	Not necessary	Not necessary	Necessary	Necessary	Desirable	Desirable
Cross-functionality of team members	Not necessary	Not necessary	Necessary	Desirable	Desirable	Unacceptable
Stability of the team	Not necessary	Not necessary	Necessary	Desirable	Desirable	Desirable
Placement of the team in the same room	Not necessary	Not necessary	Necessary	Desirable	Desirable	Desirable
2.2. Subjective criteria						
Experience in using the methodology	Not necessary	Not necessary	Desirable	Desirable	Desirable	Desirable
Team work skills, self-discipline and self-organisation	Not necessary	Not necessary	Necessary	Necessary	Necessary	Necessary
3. Customer's characteristics						
3.1. Objective criteria						
Participation of the customer in the project	Not necessary	Not necessary	Desirable	Necessary	Desirable	Desirable
Customer's high qualification	Not necessary	Not necessary	Not necessary	Necessary	Necessary	Necessary
Readiness for the introduction and testing of prototypes at a fixed frequency	Not necessary	Not necessary	Necessary	Necessary	Desirable	Desirable
3.2. Subjective criteria						
Trust relationship with the project team	Not necessary	Not necessary	Necessary	Necessary	Desirable	Desirable

Developers perform their task in accordance with requirements of the customer, who is not necessarily the end user of the software product, but for IT professionals, the main consumer of operation services is the user. Besides, developers and IT professionals work each in their own environment, and conflict of interest often arises when the software product has to be deployed in its operation environment, different from the development environment.

IV. DevOps Methodology

To solve the above-mentioned problems, a rather new methodology DevOps is intended, the application of which was first mentioned in 2009. The main ideas of DevOps are to consolidate efforts and expand communication between software developers (Development) and IT professionals (Operation), as well as to automate software deployment processes to ensure high quality [1], [2]. The functionality of the DevOps methodology is the inter-crossing of three areas (Fig. 3).

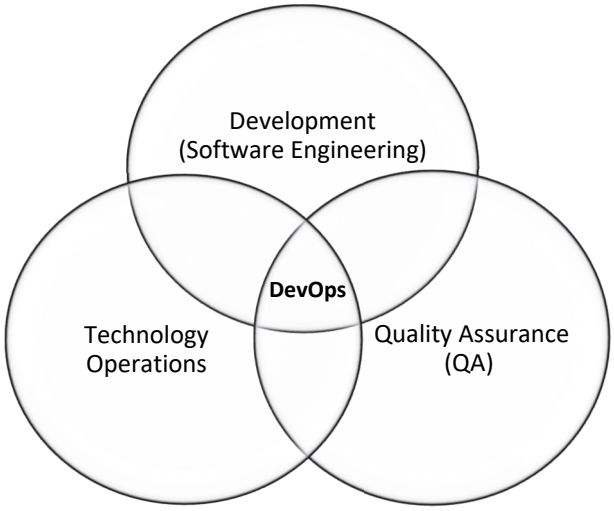

Fig. 3. Functionality of DevOps methodology.

The structure of DevOps (Fig. 4) includes three main components: people, processes and tools [1].

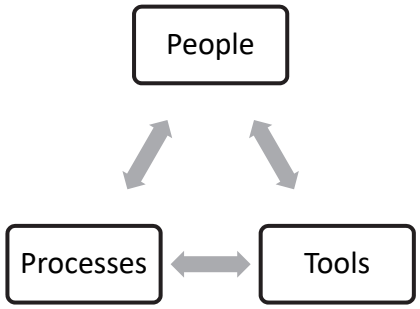

Fig. 4. The structure of DevOps.

People, and this is already mentioned above, developers (analysts, programmers and testers) and IT staff should, through continuous interaction, feel themselves as members of a single team aimed at increasing the value of the software product for users. To achieve this result, it is necessary to encourage all IT staff cohesion, mutual understanding, responsibility, and cross-functionality, usually due to periodic general meetings to discuss the received results and existing problems, and refresher courses.

Design processes are implemented as iterative with the participation of all members of the development team and IT staff, as well as customers and users. The DevOps methodology focuses on frequent issue of releases and solves the problem of their effective implementation in the operating environment, so DevOps can act as an optimal complement to any agile software development methodology (SCRUM, Kanban, Extreme Programming, etc.) [2].

The toolchain that implements the DevOps methodology provides [1]:

- automatic testing of the new release code;
- integration of a new release into a previously developed working version of the software product;
- automatic testing of the updated working version of the software product;
- management and preparation of computing infrastructure for deployment of the updated working version of the software product;
- management of the updated working version deployment;
- configuration management;
- load testing and stress testing of the updated working version of the software product.

Thus, using the methodology of DevOps it is possible to achieve the repeatability of development process and application deployment, supporting its versioning and high quality. The advantages of the DevOps methodology accelerate the implementation process and increase its reliability, as well as resource efficiency, and reduce risks.

V. Conclusion

The world-wide transition to the digital economy extends the boundaries of the use of IT technologies. From the point of view of the IT industry, this means that the volumes, complexity and variety of problems in the design of information systems are increasing. In these conditions, the importance of methodologies for the design of information systems increases, which ensure an increase in the efficiency of development processes.

As the study showed, the methods and practices of information systems design have had progress and wide practical application.

According to the authors, the main trend in software engineering is further standardisation and unification of all the life cycle stages of information systems, based on specially designed software.

The IT industry efficiency requires to implement Lean software design principles [10], in particular the transition from the predominantly unique project activity to the flow production model. In the future, the work of most IT professionals will consist in the implementation of strictly regulated actions managed by software.

Moreover, the steadily increasing IT literacy of the population requires information systems quality improvement. The modern customer is oftentimes able not only to formulate functional requirements for the IS design, but also to have a notion of how it should work. As practice shows, it is impossible to ensure the required level of quality without standardisation and unification of all stages of the life cycle of the information system using information system design methodologies.

REFERENCES

[1] J. Davis and R. Daniels. *Effective DevOps: Building a Culture of Collaboration, Affinity, and Tooling at Scale*. O'Reilly Media, 2016. 410 p.
[2] G. Kim, J. Humble, P. Debois, and J. Willis. *The DevOps Handbook: How to Create World-Class Agility, Reliability, and Security in Technology Organizations*. IT Revolution Press, 2016. 250 p.
[3] Standish Group, CHAOS Report, 1998, [Online]. Available: http://it210web.groups.et.byu.net/Tutorials/UnifiedProcess.pdf
[4] F. Kratchen, *The Rational Unified Process*. Addison-Wesley, 2002.
[5] P. Kroll and F. Kracten. *The Rational Unified Process Made Easy: A Practitioner's Guide to the RUP*. Addison-Wesley, 2003. 464 p.
[6] T. Gilb. *Principles of Software Engineering Management*, 1st ed. Addison-Wesley, 1988. 464 p.
[7] M. S. V. Turner. *Microsoft Solutions Framework Essentials*. Microsoft Press, 2006. 336 p.
[8] *Principles behind the Agile Manifesto*. [Online]. Available: agilemanifesto.org/principles.html
[9] Agile Alliance. [Online]. Available: www.agilealliance.org
[10] M. Fowler and J. Highsmith, "The Agile Manifesto," Aug. 2001. [Online]. Available: http://www.drdobbs.com/open-source/the-agile-manifesto/184414755. [Accessed: 25 April 2018].
[11] B. Volfson. *Gibkie Metodologii Razrabotki*. Piter, 2012. s. 112.
[12] B. Meyer. *Agile! The Good, the Hype and the Ugly*. Springer, 2014. 170 p.
[13] M. Cohn. *Succeeding With Agile: Software Development Using Scrum*. Addison-Wesley, 2009. 504 p.
[14] J. Sutherland. *Scrum: The Art of Doing Twice the Work in Half the Time*. Crown Business Publications, 2014. 256 p.
[15] K. S. Rubin. *Essential Scrum: A Practical Guide to the Most Popular Agile Process*. Addison-Wesley, 2012. 452 p.
[16] K. Beck and C. Andres. *Extreme Programming Explained: Embrace Change*, 2nd ed. Addison-Wesley, 2000.
[17] K. Auer and R. Miller. *Extreme Programming Applied: Playing to Win*. Addison-Wesley, 2002. 326 p.
[18] H. Kniberg. *Scrum and XP from the Trenches*, 2nd ed. 2015. Available: https://www.infoq.com/minibooks/scrum-xp-from-the-trenches-2
[19] M. Poppendieck and T. Poppendieck. *Lean Software Development: An Agile Toolkit*. Addison-Wesley, 2003. 240 p.
[20] A. Stellman and J. Greene. *Learning Agile: Understanding Scrum, XP, Lean, and Kanban*. O'Reilly, 2014. 422 p.
[21] H. Kniberg and M. Skarin. *Kanban and Scrum: Making the Most of Both*. 2009. [Online]. Available: http://www.infoq.com/minibooks/kanban-scrum-minibook
[22] Version One. [Online]. Available: www.versionone.com

Lyudmila Onokoy, Dr. of sociology. She is a Professor of Department "Business Informatics" of the Financial University under the Government of the Russian Federation (Moscow).
Scientific-pedagogical work experience is 39 years.
The main area of research is development of techniques and methodologies of software engineering.
Phone: +9154863392
E-mail: OnokoyLS@fa.ru

Jurijs Lavendels, Dr. sc. ing. He is a Professor at Riga Technical University.
His current research interests include computation methods with discretisation and algebraisation for design and modelling of physical processes.
Address: Kalku Str. 1, Riga, LV-1658, Latvia
Phone: +371 67089573
E-mail: jurijs.lavendels@.rtu.lv
ORCID iD: https://orcid.org/0000-0002-8448-4705