Applied Computer Systems
ISSN 2255-8691 (online)
ISSN 2255-8683 (print)
May 2021, vol. 26, no. 1, pp. 38–43
https://doi.org/10.2478/acss-2021-0005
https://content.sciendo.com

Expert Survey on Current Trends in Agile, Disciplined and Hybrid Practices for Software Development

Oksana Ņikiforova[1], Kristaps Babris[2*], Linda Madelāne[3]
[1–3] *Riga Technical University, Riga, Latvia*

Abstract **– Every software development company makes software development based on a specific approach. There are a number of approaches to software development, both disciplined and agile. Each approach includes a set of different activities. Sometimes, the specific nature of a company's work requires a specific approach, but the need to make work more efficient, faster and better requires implementing activities of other approaches. Then hybrid software development approaches come in. The paper presents an expert survey to examine the most important software development activities, the combinations of development approaches that are used in software development processes and the way of upgrading existing approaches. The evaluated activities of software development process are classified according to their nature – whether they correspond with a team, organisation, documentation, development, and testing. The conclusions are also made on the practices that are required most – disciplined, Agile or hybrid.**

Keywords **– Agile software development, disciplined software development, hybrid software development.**

I. Introduction

Every customer wants to get the product they ordered as quickly as possible and at the lowest cost. The professional goal of every software developer and each development team is to provide the highest possible value to employers and customers. Software development, like many other processes, has certain types of development or approaches, which have defined stages and basic principles. There are several approaches to software development, and when undertaking the development of a project, it is necessary to understand which approach to choose for its implementation, so that it is done as efficiently as possible and both parties - the customer and the supplier - are satisfied. Often in companies, teams work using one approach to develop different projects.

Software development approaches are mainly divided into two parts – traditional or disciplined software development approaches and agile software development approaches [1]. In this paper, the term "disciplined software development approaches" will be used. Discipline in this case is considered to be the observance of certain procedures, regulations, rules. This definition best explains the nature of disciplined software development approaches, as they involve the following specific development steps and stages [2].

Although disciplined software development approaches are widely used, in today's business environment [3], the customer wants to follow the project development and receive their order as soon as possible, so agile software development approaches are becoming more common.

Agile software development approaches have become very popular over time and are increasingly used in software development [2], [3], but there are companies whose job specifics do not allow them to completely abandon disciplined development approaches [4]. There are situations when one software development approach cannot meet all project development needs, so the approaches are adapted to the needs of each project and hybrids of different approaches are created [5], which can take the form of adopting some practices or a complete merging of approaches [1].

The aim of the present study is to examine software development practices that are most important in software development and the existing combinations of software development approaches that are used in various software development companies, and to offer possible additions to the existing approaches. In order to achieve the aim of the study, the following tasks have been set:

1) to summarise the considered software development approach practices that are performed in the development process;
2) to create an expert survey on the practices that are used in the software development process;
3) to collect data on key practices and conclude which practices need to complement the existing approaches.

The paper is structured as follows. The next section givs a brief overview of the related research. The third section sets a list of activities to be evaluated and gives a description of expert survey method applied. The fourth section discusses the analysis of the survey data. Finally, the last section concludes on the results and states some areas for further research.

[*] Corresponding author's e-mail: kristaps.babris@rtu.lv

II. Related Work / Literatur Review

So far, various studies have been reported to select a software development method and approaches. For example, research that recommends a development method with higher aptitude compared with past projects has been conducted in [6], [7].

There is a method to compare past and new projects and to select a development method suitable for the new project. For example, research has been conducted to select a development method of a new project by analysing past projects in terms of agile development and a project characteristics [7]. A technique for selecting a development method using the survey result on project agility [8] has been reported in [9].

Although the experience and practices used may differ from team to team and from project to project, it is useful to get to know and evaluate the views and experiences of industry experts in the context of project management and software development methodologies [10].

Most companies have not been able to fully adapt agile software development and implement it in their projects as a universal and sole approach [11]. This is probably why many researchers argue that the best way to manage a project is through hybrid methods [12].

Analysing the survey, we can observe that most of the surveyed companies in their projects used a combination of traditional and agile methods – hybrid methods. It can be concluded that despite the advantages of the agile development approach, they are far from the only and universal solution [13].

In various specific cases, traditional process elements continue to be used in agile development scaling approaches. For example, the waterfall model, which has been frequently criticized, is still in use [13].

The past decade has seen significant changes in the software development process, mainly due to the increased focus on user-centered design as well as automation, which in turn opens up opportunities for continuous improvement in this area [4].

III. Definition of List of Activities and Description of Survey Method

In order to compile practices and create a common list, practices have been mapped. i.e., practices that are named differently in different methodologies but are essentially focused on the same result have been redefined, and thus some of the practices have not been included in the common list. A summary of the practices used below is shown in Table I. The practice of "creating a list of requirements for the system and its functionality" has been removed from the compilation, as it in a way duplicates "creating a to-do list for the product". The practice of "delivery of the function within 2–10 days" has also been removed, as it has been included in the practice of "demonstrating the system to the product customer as soon as possible". There is also practice that a sprint review meeting takes place at the end of a sprint and that any programmer has the right to change the code in any part of it. The activities summarised in Table I are used as a basis for the development of the expert survey.

TABLE I
COMPILED PRACTICES

No	**Software development practice**
1	Managers, product customers and developers are equal members of the same team in one room
2	The team is self-organised and able to find a suitable solution to solve a certain task as efficiently as possible
3	The work is organised in iterations
4	Development is divided into phases and after completing one phase it is not possible to return to any of the previous steps
5	Development is divided into phases, after the completion of the phase it is possible to return to one of the previous phases and make changes, which can be continued in the next phases
6	System development is divided into function development
7	Development of a system prototype before the analysis, design and coding phases
8	Daily meetings
9	Staff rotation (pair programming)
10	Two or more weeks in which additional hours to be worked (maximum 40 hours per week) are not allowed
11	A product backlog has been created, which includes all the requirements from the product customer
12	The requirements are prioritised and the effort required to implement them is estimated
13	During each new iteration, the project team reviews the updated to-do list, determining the scope of tasks for the next iteration
14	First, a system contour is created, to which functions are gradually added
15	The team chooses the tasks to be performed and the type of implementation depending on the business priority and technical capabilities
16	When a new task appears, it is recorded in the user's story card, which contains all the requirements of the system expressed by the customer
17	Communication via code (using comments in code) is established
18	Every day, a piece of code created on that day is integrated into the system
19	Integration can only be performed by one programmer at a time, who uses a single, unique physical object for this purpose
20	The customer of the product is always among the developers and is able to explain the details of the current task
21	Prototype is enriched based on the evaluation of the customer's existing prototype version
22	Emphasis is placed on minimising errors at the early stages of development
23	For each new functionality, a test is written first and then the functionality code itself
24	Unit tests are written before writing the unit code itself
25	The technology is tested in the form of a test
26	The system to be developed is demonstrated to the customer of the product as soon as possible and, based on feedback, the necessary changes are made
27	The development process is strictly documented
28	Written documentation is replaced by communication among people

Based on the compilation of software development practices, an expert survey has been conducted. Delphi method usually consists of several iterations as it can be used to reach consensus among experts [14]. However, if the aim of the expert survey is not to achieve complete harmonisation of the experts' opinions, but to find out the current situation, then several iterations are not necessary [15]. In this study, the survey has been conducted in one iteration.

At the beginning of the survey, experts have been given the opportunity to indicate the company for which the expert works, as well as the role played in the software development process. The expert has been asked to indicate the company in order to obtain as much diversity as possible, assuming that the opinions of experts working for the same company could coincide. However, this issue has been left as an optional part, given the expert's possible desire for anonymity. Experts should also indicate their role in gathering the views of all roles, as the survey covers different phases of software development.

In the survey, experts have evaluated 28 software development practices divided into sections – team, work organisation, development, testing and documentation. Each software development practice should be rated according to a scale based on the ratings offered in Table II and tested in the survey.

TABLE II
RATING SCALE OF SOFTWARE DEVELOPMENT APPROACH

Rating	**Description**
0	Not performed in any project
1	Performed rarely
2–3	Done sometimes
4–5	Activity can be done but it is not so important
6–7	A fairly important activity that may not be performed in some circumstances
8–9	Important activity that is rarely omitted
10	Performed in each project. Unable to skip this activity

After that, ranks $R_{i,j}$ have been calculated according to the place the activity takes in the expert's opinion. Ratings given by each expert have been ranked from highest to lowest and R_{ij} is the resulting rank of the *j*-th practice coming from the *i*-th expert.

Value of importance G_j of activity *j* has been calculated using [15], [16]:

$$G_j = \sum_{i=1}^{m} R_{ij}.$$

Mean value of importance has been calculated as follows:

$$\bar{G} = \frac{1}{n}\sum_{j=1}^{n} G_j.$$

Deviation of activity *j* from mean d_j has been calculated as follows:

$$d_j = |G_j - \bar{G}|.$$

Sum of square deviations has been calculated as follows:

$$G = \sum_{j=1}^{n} d_j^2.$$

A number of equal ratings t_i by expert have been used to calculate parameter T_i as follows:

$$T_i = (t_i^3 - t_i).$$

For every question, the group coefficient of coherence *K* has been calculated:

$$K = \frac{12G}{m^2(n^3 - n) - m\sum_{i=1}^{m} T_i}.$$

Expected range for coefficient *K* is between 0 and 1, meaning the closer *K* is to 1, the higher the level of coherence among experts [15], [16]. If the value of coefficient is close to zero, additional experiments should be conducted as the level of coherence among experts has not been achieved [17].

Sorting the practices according to their importance, they have assigned places from 1 to 28. Based on the obtained place, the importance of the practices has been calculated in the ideal case, if all experts have given it the respective place. The calculations have been performed by multiplying the place of practice by its importance with the number of experts, which in this case is 15. When the values of practice importance in the case of ideal expert agreement are obtained, the deviation of real practice importance from the ideal case is calculated by module.

IV. Results of Evaluation

The expert survey has been distributed among experts of appropriate competence. The expert survey has resulted in 25 responses from at least 11 companies. 15 experts from 10 companies have been selected to apply the Delphi method. 9 experts have noted their role as a developer, 2 as a tester, 2 as a project manager, 1 as a system analyst and 1 as a department manager.

The level of coherence *K* of the selected experts has been calculated and a value of 0.46 has been obtained, which in these 41 cases is considered sufficient to use the results obtained in the survey in further research and not to repeat the survey of experts.

The questions have been divided into five sections of the survey; however, the results have been considered for all sections together because the research does not examine the importance of practices at a particular software development phase, but studies the importance of practices used in the whole software development process. The results obtained and summarised during the survey are shown in Table III. The practices offered to the experts for evaluation are arranged according to their importance by the team, work organisation, development, testing and documentation.

The results show that the team's self-organisation and ability to find a suitable solution to solve a certain task as effectively as possible are recognised as an important practice in software development. As the first practice is from the section in terms of importance, it can be concluded that the development team, its composition and ability to cooperate with each other play an important role in ensuring a successful project development process. This is also evidenced by the relatively high evaluation of the practice, which provides for daily team meetings, where team members can exchange experience and speed up the problem-solving process.

At the top of the list of priorities, there is also the prioritisation of requirements and effort assessment, the creation of a product to-do list, the ability of the team to select tasks to complete, and the updating of the to-do list during the iteration. It can be concluded that in the development process it is important to understand the main tasks to be performed and to closely follow their fulfillment. In order to successfully plan the execution of tasks and to avoid, as far as possible, unplanned delays in delivery deadlines, the effort and time required to complete them must be accurately assessed.

Work organisation in iterations is considered to be the third most important practice. It allows the team to regularly monitor the progress of the development process, look back on what has been done and evaluate the most important tasks to be performed in the future. This practice is part of a capable software development approach to practice that involves regular communication between the development team and the product customer.

TABLE III
RANKING OF SOFTWARE DEVELOPMENT PRACTICES ACCORDING TO EXPERT ASSESSMENTS

Priority	**No**	**Software development practice (*K* = 0.46)**	**Type of practice**	**Importance of practice (G_j)**
1	2	The team is self-organised and able to find a suitable solution to solve a certain task as efficiently as possible	Team	85.5
2	12	The requirements are prioritised and the effort required to implement them is estimated	Development	103.5
3	3	The work is organised in iterations	Organization	108
4	11	A product backlog has been created, which includes all the requirements from the product customer	Development	120
5	15	The team chooses the tasks to be performed and the type of implementation depending on the business priority and technical capabilities	Development	130.5
6	5	Development is divided into phases, after the completion of the phase it is possible to return to one of the previous phases and make changes, which can be continued during the next phases	Organisation	134.5
7	13	During each new iteration, the project team reviews the updated to-do list, determining the scope of tasks for the next iteration	Development	142.5
8	22	Emphasis is placed on minimising errors at the early stages of development	Testing	148
9	26	The system to be developed is demonstrated to the customer of the product as soon as possible and, based on feedback, the necessary changes are made	Testing	149
10	8	Daily meetings	Organisation	169.5
11	17	Communication via code (using comments in code)	Development	190
12	25	Testing the technology in the form of a test	Testing	194
13	14	First, a system contour is created, to which functions are gradually added	Development	196.5
14	10	Two or more weeks in which additional hours have to be worked (maximum 40 hours per week) are not allowed	Organisation	211
15	27	The development process is strictly documented	Documentation	216.5
16	6	System development is divided into function development	Organisation	220
17	1	Managers, product customers and developers are equal members of the same team in one room	Team	231
18	18	Every day, a piece of code created on that day is integrated into the system	Development	240
19	16	When a new task appears, it is recorded in the user's story card, which contains all the requirements of the system expressed by the customer	Development	243.5
20	21	Prototype enrichment based on the evaluation of the customer's existing prototype version	Development	260.5
21	7	Development of a system prototype before the analysis, design and coding phases	Organization	282
22	28	Written documentation is replaced by communication among people	Documentation	284.5
23	20	The customer of the product is always among the developers and is able to explain the details of the current task	Development	291
24	24	Unit tests are written before writing the unit code itself	Testing	325.5
25	19	Integration can only be performed by one programmer at a time, who uses a single, unique physical object for this purpose	Development	332
26	23	For each new functionality, a test is written first and then the functionality code itself	Testing	348.5
27	4	Development is divided into phases and after completing one phase it is not possible to return to any of the previous steps	Organisation	350.5
28	9	Staff rotation (pair programming)	Organisation	371

At the end of the iteration of most capable software development approaches, the product customer has the opportunity to evaluate the product developed so far, propose changes and add new requirements, which should ensure customer satisfaction with the final version of the product.

Dividing the development process into phases and the possibility to return to one of the previous development phases during the development process are also recognised as sufficiently important practices, which allow ensuring changes in the customer's requirements without starting the project development from the beginning, as well as saving time and money. In turn, the division of development into phases, when it is not possible to return to any of the previous steps, occupies the penultimate place according to the importance of the practice and the frequency of use.

Although documentation practices are not at the top of the list in terms of importance, it can be observed that strict documentation of the development process is more important than replacing documentation with communication. This shows that although team communication is very important in the software development process, writing and maintaining documentation are also important and necessary.

Looking at the testing section practices, it can be seen that unit text writing and pre-code writing are generally recognised as relatively unimportant practices, and minimising errors at the early stages of development and demonstrating the system to the developer as soon as possible, which rank 8th and 9th, occupy a more important place with ratings of 148 and 149.

Staff rotation or pair programming is recognised as a less common practice. This practice ensures that all team members are familiar with the development process and can take over their work at any time when a team member leaves. However, team members can be kept informed through daily and iteration review meetings.

It can be seen that a number of capable software development approach practices are recognised as the most important and widely used practices. Practices that provide for the self-organisation of the team and include communication among team members are highly valued. Against this background, it can be seen that practices that review the tasks and priorities to be performed are also high in importance. It should be mentioned that communication with the product customer is also recognised as important in order to achieve the best possible result.

Of the disciplined approaches to practice, the highest places are occupied by the division of development into phases with the possibility to return to one of the previous development phases, as well as the strict maintenance of documentation. This shows that although there is an increasing emphasis on people and their communication in software development, it is also important to understand the development process and the various levels of transparency provided by both successive phases and documentation.

V. Conclusion and Future Research

The aim of the study has been to investigate the software development practices that are most important in software development and the existing combinations of software development approaches that are used at various software development companies, and to offer possible additions to existing approaches.

As a result, we have found that in disciplined software development, the emphasis is on the sequential course and transparency of the development process. Disciplined software approaches can be complemented by regular feedback meetings among the development team and the team and the product customer, increasing mutual communication and information exchange, thus ensuring a more efficient development process. Agile software development approaches are more effective in software development, as they incorporate most of the most important development practices. However, in cases where the specifics of the company do not allow it, Agile development practices can be implemented during different phases of disciplined approaches. In the Agile software development process, great emphasis is placed on people-to-people communication, team self-organisation, and prioritization and evaluation of tasks to be performed. When using Agile software development approaches, it is important to form a team of professionals and attention should also be paid to their ability to communicate successfully with each other. Agile software development approaches can include tighter phasing of the process and process documentation, thus ensuring process transparency and traceability, as well as more efficient replacement of team members. The results of this survey may be of interest to project managers and analysts who need to consider how to improve software development processes in specific projects to make project development more efficient or transparent. A summary of the practices developed and an assessment of their relevance can be used to evaluate possible additions.

Further research could include conducting the second iteration of an expert survey, reviewing other software approaches and their practices, as well as analysing the existing hybrid software development approaches.

VI. Acknowledgment

The present study has been supported by the Doctoral Grant Programme of Riga Technical University.

References

[1] M. Kuhrmann, P. Diebold, J. Munch, P. Tell, K. Trektere, F. McCaffery, V. Garousi, M. Felderer, O. Linssen, E. Hanser, and C. R. Prause, "Hybrid Software Development Approaches in Practice: A European Perspective," *IEEE Software*, vol. 36, no. 4, pp. 20–31, Jul. 2019. https://doi.org/10.1109/MS.2018.110161245

[2] M. Singh, N. Chauhan and R. Popli, "A Framework for Transitioning of Traditional Software Development Method To Distributed Agile Software Development," in *2019 International Conference on Issues and Challenges in Intelligent Computing Techniques (ICICT)*, Ghaziabad, India, Sep. 2019, pp. 1–4. https://doi.org/10.1109/ICICT46931.2019.8977654

[3] S. Komai, H. Nakanishi and H. Saidi, "Guidelines for Selecting Agile Development Method in System Requirements Definition," in *2017 7th IEEE International Conference on Control System, Computing and Engineering (ICCSCE)*, Penang, Nov. 2017, pp. 49–53. https://doi.org/10.1109/ICCSCE.2017.8284378

[4] S. A. Ruk, M. F. Khan, S. G. Khan, and S. M. Zia, "A Survey on Adopting Agile Software Development: Issues & Its impact on Software Quality," in *2019 IEEE 6th International Conference on Engineering Technologies and Applied Sciences (ICETAS)*, Kuala Lumpur, Malaysia, Dec. 2019, pp. 1–5. https://doi.org/10.1109/ICETAS48360.2019.9117324

[5] L. R. Vijayasarathy and C. W. Butler, "Choice of Software Development Methodologies: Do Organizational, Project, and Team Characteristics Matter?," *IEEE Software*, vol. 33, no. 5, pp. 86–94, Sep. 2016. https://doi.org/10.1109/MS.2015.26

[6] H. Ishii, K. Maruya, T. Habara and H. Washizaki, "Compatibility Assessment Method for Agile Development Based on Project Characteristics," SIG Technical Reports, vol. 2015-SE-187, no. 36, 2015.

[7] A. Shimoda and T. Yabuki, "Cost and Value Analysis of Software Development Method Focused on Individual Function," in *2017 Eighth International Conference on Intelligent Computing and Information Systems (ICICIS)*, Cairo, Dec. 2017, pp. 237–243. https://doi.org/10.1109/INTELCIS.2017.8260053

[8] J. Sheffield and J. Lemétayer, "Factors Associated with the Software Development Agility of Successful Projects," *International Journal of Project Management*, vol. 31, no. 3, pp. 459–472, Apr. 2013. https://doi.org/10.1016/j.ijproman.2012.09.011

[9] T. Imani and M. Nakano, "Managing Large-Scale IT Projects: A Decision-Making Flow Using Plan-Driven and Agile Method for a Hybrid Approach," *Journal of the Society of Project Management*, vol. 18, no. 3, pp. 14–19, 2016.

[10] A. Rauf and M. AlGhafees, "Gap Analysis between State of Practice and State of Art Practices in Agile Software Development," in *2015 Agile Conference, Washington*, DC, Aug. 2015, pp. 102–106. https://doi.org/10.1109/Agile.2015.21

[11] S. A. Ruk, M. F. Khan, S. G. Khan, and S. M. Zia, "A Survey on Adopting Agile Software Development: Issues & Its impact on Software Quality," in *2019 IEEE 6th International Conference on Engineering Technologies and Applied Sciences (ICETAS)*, Kuala Lumpur, Malaysia, Dec. 2019, pp. 1–5. https://doi.org/10.1109/ICETAS48360.2019.9117324

[12] B. Boehm and R. Turner, "Management Challenges to Implementing Agile Processes in Traditional Development Organizations," *IEEE Software*, vol. 22, no. 5, pp. 30–39, Sep. 2005. https://doi.org/10.1109/MS.2005.129

[13] M. Marinho, J. Noll, I. Richardson, and S. Beecham, "Plan-Driven Approaches Are Alive and Kicking in Agile Global Software Development," in *2019 ACM/IEEE International Symposium on Empirical Software Engineering and Measurement (ESEM)*, Porto de Galinhas, Recife, Brazil, Sep. 2019, pp. 1–11. https://doi.org/10.1109/ESEM.2019.8870168

[14] K. K. Lija, K. Laakso, and J. Palomäki, "Using the Delphi Method," in *2011 Proceedings of PICMET'11: Technology Management in the Energy Smart World (PICMET)*, Portland, OR, 2011, pp. 1–10

[15] L. Leimane., O. Nikiforova. "Results from Expert Survey on System Analysis Process Activities", in *Applied Computer Systems*, Vol. 24, Issue 2, pp. 141.-149., 2019. https://doi.org/10.2478/acss-2019-0018

[16] O. Nikiforova and U. Sukovskis, "Framework for Comparison of System Modelling Tool", in *Proceedings of Fifth IEEE International Baltic Workshop on DB and IS*, *BalticDB&IS'2002*, Tallinn, Estonia, Jan. 2002, vol. 1, pp. 63–70.

[17] L. Madelāne, "Analytical Review on Agile, Disciplined and Hybrid Approaches to Software Development," Bachelor thesis, Riga Technical University, 2020.

Oksana Ņikiforova received the Doctoral degree in Information Technologies (system analysis, modelling and design) from Riga Technical University, Latvia, in 2001. She is presently a Professor at the Department of Applied Computer Science, Riga Technical University. Her current research interests include Agile software development methodologies and project management methods and tools.
E-mail: oksana.nikiforova@rtu.lv
ORCID iD: https://orcid.org/0000-0001-7983-3088

Kristaps Babris received the Master degree in Computer Systems from Riga Technical University, Latvia, in 2018. He is presently the second-year PhD student at the Department of Applied Computer Science, Riga Technical University. In parallel, he is a CTO at the company, which develops Business Intelligence solutions. His current research interests include design and modelling.
E-mail: kristaps.babris@rtu.lv

Linda Madelāne is a recent graduate of Riga Technical University with progressive experience in the IT industry. She has a Bachelor degree in Computer Control and Computer Science.
E-mail: linda.madelane@edu.rtu.lv