SOFTWARE DEVELOPMENT METHODOLOGIES: A COMPARATIVE ANALYSIS

Iulia Ioana ANGHEL[1], Roberta Ștefana CĂLIN[1], Maria Lavinia NEDELEA[1], Iulia Cristina STĂNICĂ[2], Cătălin TUDOSE[3], Costin Anton BOIANGIU[4]

Software development methodologies define fine lines which ensure improved human and time management to accomplish the project needs. The current research aims to develop a comparative study on representative methodologies, seeking to identify the potential growth in the quality of the project deliverables and to decide whether the practical approach is the same as the theoretical one. To gain the needed information, a research was conducted with people from different software positions and companies. The results aim to provide a better understanding of how people feel about using software development methodologies based on personal experience.

Keywords: software project management, development methodology, waterfall, iterative, v-model, prototype, agile, scrum, feature-driven, Kanban, Lean, Extreme Programming

1. Introduction

The idea of comparative research on project management and software development methodologies started due to the increased use of methodologies in project development. Software development methodologies refer to a collection of practices that must be put together to generate a systematic procedure. Over time, development methodologies have evolved and adapted to modern times and new technologies.

The goal of the current research is to analyze whether there are correlations between the theoretical and practical aspects of each frequently encountered methodology. We will thus detail some general ideas about the traditional (Waterfall, V-Model, Iterative, and Prototyping) and modern (Scrum

[1] Student, Computer Science and Engineering Department, University POLITEHNICA of Bucharest, Romania, e-mail: angheliuliaioana@gmail.com, roberta.calin23@gmail.com, lavinia.mndl@gmail.com

[2] Teaching Assistant, PhD, Department of Engineering in Foreign Languages, University POLITEHNICA of Bucharest, Romania, e-mail: iulia.stanica@upb.ro

[3] Lecturer, PhD, Computer Science and Engineering Department, University POLITEHNICA of Bucharest, Romania, e-mail: catalin.tudose@gmail.com

[4] Prof., PhD, Computer Science and Engineering Department, University POLITEHNICA of Bucharest, Romania, e-mail: costin.boiangiu@cs.pub.ro

Agile Based, Feature Driven Development, Lean Development, Extreme Programming, and Kanban) software methodologies based on specialty literature. Furthermore, theoretical aspects will be compared with the answers received in a thorough study conducted with people working in different software companies.

2. Software development methodologies – literature comparison

Software development methodologies (SDM) refer to rules established for conceiving, planning, designing, developing, testing, deploying, and maintaining software [1]. Different SDMs have evolved over the last 50 years, from classical to modern ones, each one adding core values and principles besides previously mentioned guidelines. For the next part of the analysis, we are going to refer to pre-Agile methodologies as being part of the "traditional" type, while all post-Agile SDMs will be referred to as "modern".

2.1. Traditional development methodologies

Traditional methodologies, also called "heavy-weight", rely on strict planning, requirements definition, and documentation [2].

Waterfall

The waterfall methodology is one of the traditional models that provide linearity, based on a top-to-bottom approach [3]. Due to its origins linked to the manufacturing industry, it comes with a restrictive structure based on multiple phases, such as requirements (the needs of the project), design (architecture and technologies, according to previously established requirements), implementation, testing, deployment, and maintenance. Each phase must be executed in a queue; one should move forward to the next phase only when the preceding one is done (thus the "waterfall" terminology) [4]. This methodology comes with simplicity in a well-organized structure. It is best suited for short projects because it requires the team members to have a thorough understanding of the requirements [5]. On the other hand, it can create a gap in communication between the client and employee, which might fail in keeping up with the expectations, as a consequence of not having a model of software available until all phases are finished [6]. When it comes to the final stage, changes do not come as an easy solution, meaning tedious refactoring.

V-Model

V-model is a traditional software development and testing methodology, often considered a variant of the Waterfall approach [7]. Also named the Verification and Validation Model is used to represent a system's development lifecycle following a strict structure and being executed sequentially. Each development phase has an associated testing phase, which in this way ensures

proactive defect tracking [8]. This approach is suitable for smaller projects, as it demands a very well understanding of the requirements from the team members. Also, it is preferable for projects with vast technical resources [9]. It may be preferred as a software development methodology because it is easy to use due to its simple structure, but a project manager should take into consideration the fact that this method is risky and unstable, and it is difficult to change functionality when it gets to the testing stage [10] [11].

Iterative

This approach consists of building a product through various steps, providing a fully functional product at the end of each stage, and getting user feedback early as well. It starts with a simple prototype containing the essential requirements, and afterward, the product is constantly improved until the final stage [12]. The big advantage is that a working product can be delivered very quickly, even if it's not complete, as it may have some of the functionalities implemented. This approach also allows for earlier feedback [13], leading to issues being caught early in the cycle. The iterative model is good for large projects done by smaller teams [14], benefiting from the ability to easily adapt. However, since not all requirements might be specified from the beginning, it is possible for the architecture to change and for problems to arise when other requirements are added. When it comes to the process of each iteration, it includes planning, analysis, implementation, and testing. The final step is the feedback, with the whole process repeating until the product is finished [13].

Prototyping

The Prototyping Model is an SDM in which a prototype is successively built, tested, and modified until an acceptable result is created from which the complete system or product can be developed [15]. This method is suitable for systems with many interactions with the end-users, including online systems or web interfaces. A prototype is made before the coding proceeds, so the requirements can be changed after the end-user gets an "actual feel" of the system [16]. This may increase the complexity as the client wants more functionalities [15], but also improves the quality of the specifications provided by the customer. One of the advantages of the Prototyping Model is that missing functionalities are quickly identified. Moreover, it is easy to understand, it encourages innovation, and customers are included in the development process, helping with product improvements [15]. Besides its advantages, the Prototyping Model has also some disadvantages, including high costs spent on creating the prototype [16] due to unexpected failures during testing or errors in the development process, resulting in the need to create a new prototype and slow development time until the final project is finished.

2.2. Modern development methodologies

Light-weight or so-called "modern" technologies rely more on adaptability, small iterations, and people's involvement and communication.

Scrum

Scrum is a delivery-focused framework best suited for Agile development. It relies on a self-organizing, cross-functional team. Scrum encourages a structured way of getting work done, shaped by the company's strategies and culture, having its main focus on good communication, permanent improvement, and transparency. This framework uses visual representations such as boards for planning and tracking purposes [17]. Complex tasks are divided into small chunks of work, while small objectives and delivery in a fixed length of time make the work easier. In Scrum, there are three components: product backlog (the needs of the product to work properly), sprint backlog (selected issues for the current development cycle), and sprint goal [18]. The main advantages are customer delivery (the key functionalities might be available before the final stage of the product) and adaptability (feedback as a tool used for continuous improvement) [19]. On the other hand, Scrum may take some time to be fully grasped.

Feature-Driven Development

Feature-Driven Development (FDD) is a feature-focused, Agile method that is customer-centric, iterative, and incremental [20]. In addition, its approach is top-down decision-making [21]. As its name suggests, this methodology's core is to deliver features often and efficiently. Focusing on features offers the opportunity to address customer needs more quickly, and identify and fix problems that appear [21]. In addition, there are complementary development techniques that are driven by other specific parts of the application lifecycle. This methodology differs from other Agile-based ones by minimizing the number of meetings relying on documentation to communicate information and maximizing the support over features [20]. It is best suited for big companies which face complex projects and a large group of people. Unfortunately, it is highly dependent on developers and does not work efficiently for small projects.

Lean Development

Lean Development borrows its philosophy from the manufacturing industry, which is based on automation and speed. It optimizes people, resources, effort, and energy, creates value for the customer, and improves programming practices and the organization's performance [22]. Lean Software Development is based on a series of clear principles, it depends on self-documented code and the team's involvement. Decisions are delayed until they can be made based on facts, not uncertain assumptions, or predictions [23]. The Lean Development

methodology is suitable for small teams and projects that have to be finished quickly. When a Lean team is created, communication is the most important factor for the project's success [22]. In terms of testing, it reduces waste by testing the system continuously, from the early stages of development [24]. There are numerous advantages: minimizing waste and maximizing customer value, test automation, continuous improvement, and cost reduction. As a disadvantage, Lean Development presents low scalability in comparison with other frameworks [24].

Extreme programming (XP)

In Extreme programming, the planning is done at the beginning of each phase, and the goal is to practice completely transparent development while integrating code in small parts, avoiding long phases in the development process. Testing is also done before the code is implemented, and pair programming is highly encouraged [25]. Moreover, an important aspect of XP is the fact that the customer is always available to the team, which is preferably small but extended (including not only developers) [26]. The steps of XP are planning, managing, designing the architecture, coding, and testing. One disadvantage of XP is the focus on the code, which tends to disadvantage the design and the architecture. Also, considering the pair programming approach, different time zones may negatively impact the performance and synchronization of the team [27].

Kanban

Kanban is a flexible approach that can be used on top of other workflows and methods that already exist [28]. The first principle of Kanban is visualizing the workflow, to clearly set the stages of the development as requested, in progress, or done/delivered items. A Kanban system is defined by existing "signals" as well as commitment and delivery points, which will delimit the three categories in the workflow [29]. By having a well-delimited column for each step in the process, the state can be monitored, and problems can easily be observed. Another Kanban principle is constraining multitasking, an approach that ensures that a task will be moved to being in progress only after another one is done, managing the workload and the flow of the whole process [30]. Another principle Kanban focuses on is making the process transparent, so that everyone is familiar with it, as well as implementing cadences, or feedback opportunities. The feedback practice makes it easier to fix issues. Considering all aspects, the steps of Kanban coincide with its practices - visualizing, limiting managing the flow, using feedback loops, and improving constantly [29].

3. Practical comparison and results analysis

To pass from the theoretical aspects of SDMs to the practical ones, we conducted a study with 172 participants regarding their experience and personal opinions on SDM. The participants on whom the study was conducted are mostly in software development positions (68.68%), followed by engineering positions (15.2%), as well as management positions (9.9%). 6.22% of the respondents to the questionnaire were in Digital Analyst, Quality Assurance, or Control and Scrum Master positions. As for the teams' seniority, most participants are seniors (32.6%) and juniors (29.1%). The third-highest percentage were regulars (24.4%), and the last places were occupied by interns (11.1%) and project managers (3%).

As a result, on a scale from 1 to 5, participants' opinions about using a methodology were proven to be highly impactful: 89.2% consider the use of an SDM to have a high or very high impact on a software project. The answers highlight the relevance of our current study. Most of the software development methodologies are considered to be impactful, with V-model obtaining the maximum score from all respondents and Prototyping the lowest score. In terms of popularity, we observed a contrast when it comes to using software methodologies in practice, with modern methodologies (Scrum Agile Based, Kanban) in leading positions (Figure 1). On the same note, Scrum is also considered the most effective methodology by 68% of participants.

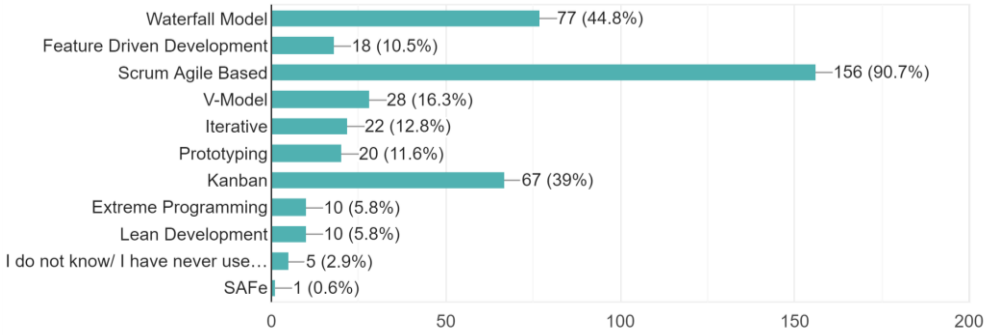

Fig. 1. Methodologies used by our respondents throughout their careers

In the following subsections, we will detail the respondents' answers related to the projects developed using that methodology, the teamwork and structure, and personal opinions and details about each individual methodology.

3.1. About the project

The current section examines the answers depicting the respondents' reflections on the software methodology used in their latest project. From the analysis, we can deduce the first observation that Scrum is spread across a large

variety of fields in the industry (Figure 2), with a majority of participants in our study choosing it as their methodology in their latest project. From the received responses, we can also conclude that modern methodologies are predominantly preferred over traditional ones (79% vs 21%).

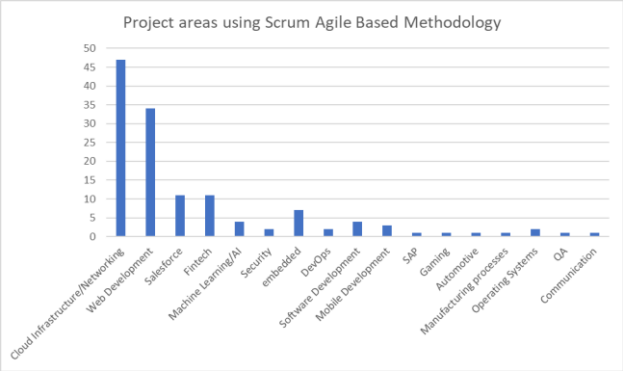

Fig. 2. Scrum-based project areas

It is also noticeable that there is a wide range of methodologies used in the Webspace, whereas in Fintech, Salesforce, or Cloud, Scrum appears to be the established choice. Moreover, it is interesting to note that the projects in the Artificial Intelligence (AI) field are only developed using Kanban and Scrum. From all the Scrum answers, 0.02% of the participants worked in this field, Kanban had 25% of the responses regarding AI. This sustains the claim that Kanban's popularity is rising within Data Science, Machine Learning, and Artificial Intelligence communities due to its focus on delivery, which is highly compatible with large repeatable processes. Regarding the suitability of the project in connection with the selected methodology, most answers were positive. However, even if Scrum's strengths include good communication (an average rate of 4.25 out of 5 for Peer-to-Peer communication and 4.21 out of 5 for supervisor-subordinate communication), our study revealed that Waterfall is in the first place in both categories, with the same average rating of 4.66.

A small percent of the respondents (9.3%) consider that the methodology used was not suited for their type of project. The dominant area of the project for which the methodology was not suitable is Cloud Infrastructure/Networking using Scrum-based methodology (61.55%), followed by Embedded and Web Development also using Scrum (23.07%), Cloud Infrastructure/Networking using Waterfall (7.69%), and Salesforce using Feature Driven Development methodology (7.69%). As for the reason why the methodology was not suitable for their respective project, the most frequent answers were that the stages were not respected, the deadline was not realistic, and too much time was spent on meetings. When it comes to the pressure felt by the participants, the majority felt time pressure (76.9%), closely followed by resource pressure (53.84%).

Moreover, Peer-to-Peer communication has an average of 3.92 out of 5, and communication with supervisors has an average of 3.84 out of 5. Time management has proven to be more feasible when software methodologies are used, with only approximately 25% of projects having issues with meeting deadlines, in the cases of Scrum (28%), FDD (16%), Prototyping (50%), and Kanban (14%). Apart from Scrum (where 86% of all the delays occur), all the other methodologies did not encounter a deadline extension of more than 50%.

The data presented in the chart below (Figure 3) is the result of pressure-oriented multiple-choice selection. While 19% of the respondents felt no pressure in the development process, both Scrum and Iterative show that productivity comes with a cost in time pressure. Over 60% (Scrum) and 80% (Iterative) of the respondents felt deadline constraints, while Kanban recorded the highest percent of budget pressure, with 50% of the respondents marking it as an issue. On the same note, the methodologies that prioritize efficiency tend to have a higher level of communication pressure, with 100% of the respondents that worked with FDD citing it as a problem. At the opposite pole, Lean Development scored last in the level of pressure employees feel, being the most balanced from this point of view. Considering these types of pressure, their impact on code quality is divided between negative (45.5%) and no impact at all (54.5%). That being mentioned, time pressure had a negative impact in 80% of cases, communication pressure in 47%, resource pressure in 45%, and budget pressure in 11% of all cases.

During the development process, some of the participants signaled either a lack of cohesion and clarity in the set of requirements or that the tasks were not being correctly assigned by their complexity in Scrum SDM. Our study also took into consideration people's positions regarding the testing process. The Scrum methodology received mixed reactions, with respondents calling it "time-consuming", "hard", and "chaotic", but also praising its practices for "continuously testing right after the implementation of each feature" and "periodic testing". Some used "manual testing", others automated ones, including "unit testing" and "integration tests". The feedback for testing in FDD was completely positive, being "straight-forward" and "Unit/integration tests working really well". When it comes to Prototyping, the process was described as "thorough" for "including not just people from the development team, but also partners in the project, the client, etc.". For the Iterative methodology, testing was viewed as "easy and effective" and managed by automation services in some cases.

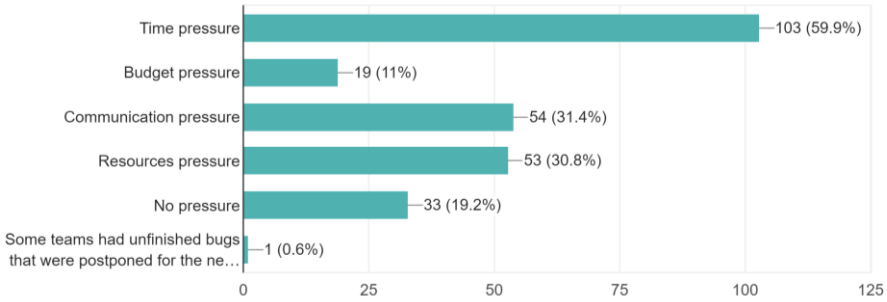

Fig. 3. The pressure felt in the development process of the project

3.2. About the team

In any development process, team structure is a valuable asset that can have a significant impact on project communication and coordination. The distribution of team structures can be observed in Figure 4.

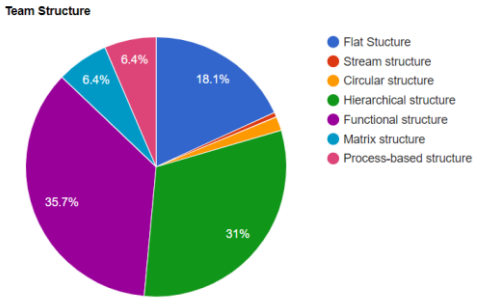

Fig. 4. Respondents' team structures

Regarding the Scrum methodology, the majority of respondents are working within a functional structure, followed by a hierarchical one. In the Waterfall model, the hierarchical structure is exclusively chosen, while in FDD flat structure and hierarchical structure are equally divided. The modern methodologies (Extreme Programming and Lean) are both headed towards a more uniform distribution of duties (flat structure). Kanban is team structure neutral, which can be observed from the results as well, which vary from functional to hierarchical or flat structure. The efficiency of using an SDM is proven by the tendency to work in smaller teams, leading to better performance management and coordination. Along these lines, 47% of the respondents who use Scrum work in teams smaller than 10 people, while 0.5% exceed the number of 100 members per team. Moreover, from the responses concerning FDD, 83% of teams were smaller than 10 people, while in Lean Development and Prototyping, 100% worked in such teams. The fact that Kanban has no team size limitation is reflected by our study as well, with 37% of respondents working in teams of less

than 10 members, 50% within teams of 10-20, and 13% within teams of 20-100 people.

3.3. About the methodology

Regarding the respected stages of the development process, almost 70% of participants followed them in both modern and traditional methodologies. The other 30% are mostly modern methodologies users. As such, even if there is much more flexibility in these methodologies, it is not enough to keep things in line.

Concerning Scrum, even if the opinions are spread across a large variety of advantages and disadvantages, 25% of the responses prefer it due to its flexibility and adaptability. While 22.6% of the respondents consider it fast, 20% complain about the amount of pressure that is being felt, especially by the software developers. Moreover, in contrast to 20% of the answers stating that too much time is wasted on meetings, 26% of the respondents praise Scrum for "keeping you involved" and for focusing on good communication and empowerment of the team. In addition, on one hand, the participants declare themselves pleased with the methodology being clear (21%) and well organized (14%). On the other hand, around 13% oppose the planning for being too thorough. The contributors who chose FDD as their last used methodology classify it as laid-back when it comes to time management as well as having well-defined requirements due to its characteristic of being structured. Moreover, the methodology is known for empowering the developer and, as revealed by our research, their "growth on multiple levels" as well. When it comes to Waterfall, its "clear set of requirements" highlights its well-organized structure. However, this plays a role in the negative perception as well, with Waterfall being considered "not flexible" and "expensive".

When it comes to a better methodology than the one currently used, the respondents who stated that they used a better methodology in the past and do not work with Scrum chose it as the most suitable (on projects which are "shorter in terms of duration"). However, out of the people who are currently using Scrum, 16% consider Waterfall to be better, especially in terms of feeling pressure and large projects where "clarity was provided" as well as when dealing with an experienced team. These statements are confirmed by the consecrated character of Waterfall concerning the clear set of requirements that are needed and the individuality of the teams as well. Some of the answers focused on Kanban as a better methodology due to its lower overhead and less frequent meetings.

4. Recommendations to select the software development methodology

Selecting the appropriate software development methodology for project management is a crucial decision, as it may overall strongly impact the success of

the project. There is no precise choice, it depends on a series of factors. Based on the analysis of the methodologies and our research, we emphasize the situations when one of the alternatives should be preferred.

Waterfall is favored when:

- The structure of the project is linear and well-organized
- The requirements are well defined from the beginning and there is less possibility to be changed during the development
- The customer is less involved in the phases of the project

V-Model is favored when, in addition to the situation described above for ***waterfall***:

- Teams are small and are able to efficiently self-manage
- Keeping the low involvement of the customer, you would still like to rely on the user acceptance and the user acceptance testing for the functionality that is implemented

The ***iterative*** approach is preferred when:

- The customer is very involved in the phases of the project and ready to provide frequent feedback
- The specifications may be frequently changed
- In addition, if the application is visual and has a lot of interaction with the end-user, you may consider ***prototyping***

Scrum is adopted if, in addition to the situation described above for ***iterative***:

- The team is small, dynamic, and able to maintain high communication inside it and with the customer
- The people from the team are highly collaborative
- If the cohesion of the team is very high and developers may adopt techniques such as pair-programming, one may adopt ***Extreme Programming***.

Kanban is preferred if, in addition to the situation described above for ***Scrum***:

- There are larger teams
- There is a need to limit the amount of the Work in Progress (WiP) functionalities

Finally, the following Agile methodologies are preferred in the following cases:

- ***Lean Development***, if the team is small and it is crucial that the project is kept on a low budget and eliminates waste
- ***Feature Driven Development,*** for larger teams and complex projects that allow the easy extraction of clear features to be implemented

5. Conclusions

In conclusion, our study aimed to prove and analyze the importance of SDM, as perceived by people working in the software development field. Similar studies have been conducted, yet they are limited in terms of the number of methodologies they analyze, or they refer to a restricted type of software development and companies. Such an analysis, presented by Molina-Rios and Pedreira-Souto [31], presents a comparison between the development methodologies used in web applications – while Agile methodologies are mentioned, their research focuses just on web-development specific methodologies. Another study focuses on the analysis of SDMs suitable for start-ups [32], concluding their preference for Agile and Lean.

In our study, as opposed to existing research, we tried to include a high number of respondents from different software companies, regardless of the technology they use. Most respondents considered them an important aspect of the development process, with the best results being achieved in teams of less than 10 people.

Scrum was, without doubt, the most popular methodology, considering its frequent inspections, leading to continuous improvements. Thus, it is ensured that a high-quality product is delivered, although quality comes at the price of being the first methodology in terms of time pressure felt by the employees. Albeit its popularity, Scrum is not considered to be suited for all types of projects, as it is reflected by the answers, but instead chosen out of commodity. Moreover, the majority of participants complained about the Scrum meetings being time-consuming, even though it compensates through its speed and automated testing. Waterfall is the traditional methodology widely used as a predecessor to Scrum, which is reflected by it being chosen as a better methodology by a significant number of respondents who currently use Scrum. It is characterized as straightforward, less stressful, and good for experienced people, but also rigid and expensive. Moreover, it ranked first when it comes to the quality of communication, both with supervisors and with colleagues. Kanban was considered a team-neutral methodology, with low overhead, less frequent meetings, and less pressure. It was deduced from the results that it is mostly used in Artificial Intelligence, due to its focus on delivery, which is highly compatible

with large repeatable processes. Feature Driven Development ranked first in terms of time management, whereas it ranked last regarding communication quality. It is thought of as a balanced methodology, offering stable deliverables and empowering the developers.

Due to the lack of sufficient respondents or data extracted, we cannot draw any conclusion regarding the other software methodologies that were studied. Our objectives were thus limited by the lack of public diversity in the applied methodologies highlighted by the questionnaire's responses. As a future improvement, the study should be conducted in an extended area of expertise, with more participants from each software development subdomain. Overall, the development process is closely dependent on the usage of a methodology, providing it with a well-defined structure and coordination. Furthermore, since our study was conducted mostly with software developers, we can state that most of the results match the theoretical ones which we expected from the beginning.

Based on the analysis and comparison of the methods and the research we conducted, we finally formulated a series of criteria and recommendations about when to favor the adoption of a particular software development methodology.

R E F E R E N C E S

[1]. *M. L. Despa*, Comparative study on software development methodologies, Database Systems Journal vol. V, no. 3, pp. 37-56, 2014.

[2]. *L. Jiang, A. Eberlein*, An analysis of the history of classical software development and agile development, 2009 IEEE International Conference on Systems, Man and Cybernetics, pp. 3733-3738, 2009

[3]. *Project Manager Website*, The Ultimate Guide…Waterfall Model, https://www.projectmanager.com/waterfall-methodology

[4]. *Smartsheet Website*, PM Methodologies – Waterfall, https://www.smartsheet.com/content-center/best-practices/project-management/project-management-guide/waterfall-methodology

[5]. *M. Martin*, What is Waterfall Model in SDLC? Advantages and Disadvantages, Guru99 Online Blog

[6]. *S. Lewis*, Waterfall Model, Tech Target – Search Software Quality, https://www.techtarget.com/searchsoftwarequality/definition/waterfall-model

[7]. *D. Firesmith*, Using V Model for Testing, Software Engineer Institute Blog, 2013, https://insights.sei.cmu.edu/blog/using-v-models-for-testing/

[8]. *N. M. A. Munassar, A. Govardhan*, A Comparison Between Five Models Of Software Engineering, IJCSI International Journal of Computer Science Issues, vol. 7, no. 5, 2010.

[9]. *D. Kumar*, Software Engineering - SDLC V-Model, Geeks for Geeks website, https://www.geeksforgeeks.org/software-engineering-sdlc-v-model/

[10]. *G. B. Regulwar, P. M. Jawandhiya, V.S. Gulhane, R.M. Tugnayat, P.R. Deshmukh*, Variations in V Model for Software Development, International Journal of Advanced Research in Computer Science, vol.1, no. 2, 2010

[11]. *M. S. Durmus, I. Ustoglu, R. Y. Tsarev, J. Borcsok*, Enhanced V-Model, Informatica – An International Journal of Computing and Informatics, 2018.

[12]. *J. Martins*, Understanding the iterative process, with examples, Asana Website, 2021

[13]. *K. Eby*, The Power of Iterative Design and Process, Smartsheet Website, 2019, https://www.smartsheet.com/iterative-process-guide
[14]. *Leankor Website*, Project Management Methodologies Designed for Small Teams, https://www.leankor.com/project-management-methodologies-designed-small-teams/
[15]. *E. R. Coutts, A. Wodehouse, J. Robertson*, A comparison of contemporary prototyping methods, Proceedings of the Design Society International Conference on Engineering Design vol. 1 no. 1, pp. 1313-1322
[16]. *L. Kent, C. Snider, J. Gopsill, B. Hicks*, Mixed reality in design prototyping: A systematic review, Design Studies vol. 77, 2021.
[17]. *Atlassian Agile Coach*, Scrum – Learn how to scrum with the best of 'em, https://www.atlassian.com/agile/scrum
[18]. *K. Schwaber, J. Sutherland*, The Scrum Guide - The Definitive Guide to Scrum: The Rules of the Game, 2020
[19]. *Agilest Website*, Why Does Scrum Work, https://www.agilest.org/scrum/why-does-scrum-work/
[20]. *R. Lynn*, What is FDD in Agile, Planview Website, https://www.planview.com/resources/articles/fdd-agile/
[21]. *Lucid Chart Blog*, Why (and How) You Should Use Feature-Driven Development, https://www.lucidchart.com/blog/why-use-feature-driven-development
[22]. *S. Saeed, I. Alsmadi, F. M. Khawaja*, Lean Development: A Tool for Knowledge Management in Software Development Process, Business Strategies and Approaches for Effective Engineering Management, 2013
[23]. *S. Jena*, Lean Software Development, 2021, Geeks for Geeks website, https://www.geeksforgeeks.org/lean-software-development-lsd/
[24]. *J. Rault*, Lean Software Development: An In-depth Guide for Every Developer, https://www.laneways.agency/lean-software-development/
[25]. *NTask Blog*, Extreme Programming in Agile – A Practical Guide for Project Managers and nTaskers, https://www.ntaskmanager.com/blog/extreme-programming-in-agile-a-practical-guide-for-project-managers-and-ntaskers/
[26]. *D. Wells*, When should Extreme Programming be Used?, 1999
[27]. *AltexSoft Website*, Extreme Programming: Values, Principles, and Practices, https://www.altexsoft.com/blog/business/extreme-programming-values-principles-and-practices/
[28]. *Kanbanize Website*, What Is Kanban? Explained for Beginners, https://kanbanize.com/kanban-resources/getting-started/what-is-kanban
[29]. *D. J. Anderson, A. Carmichael*, Essential Kanban Condensed, Lean-Kanban University, 2016
[30]. *Official Microsoft Documentation*, Set Work in Progress limits in Azure Boards, 2022
[31] *J. Molina-Rios, N. Pedreira-Souto*, Comparison of development methodologies in web applications, Information and Software Technology, vol. 119, 2020.
[32] *E. W. Tegegne, P. Seppanen, M. O. Ahmad*, Software development methodologies and practices in start-ups, The Institution of Engineering and Technology, vol. 13, 2019