



COMPARISON OF REQUIREMENTS ENGINEERING KNOWLEDGE AREAS IN TERMS OF TRADITIONAL AND AGILE SOFTWARE METHODS

Kadir ÇAMOĞLU^{1*} , Rembiye KANDEMİR² 

¹Trakya University, Department of Computer Engineering, Engineering Faculty, Edirne / Turkey

²Trakya University, Department of Computer Engineering, Engineering Faculty, Edirne / Turkey

Cite this article as:

Çamoğlu, K. & Kandemir, R. (2019). Comparison of Requirements Engineering Knowledge Areas in Terms of Traditional and Agile Software Methods, *Trakya University Journal of Engineering Sciences*, 20(2), 79-91.

Highlights

- Determining requirements engineering key concepts
- Current requirements engineering approaches
- Advantages and Disadvantages of Traditional Methods and Agile Methods

Article Info

Abstract

Article History:

Received:
May 22, 2019
Accepted:
August 12, 2019

Keywords:

Requirements engineering;
Traditional software development
methods;
Agile software development
methods;
Agile and traditional comparison;

One of the important points to be considered when implementing a software project is the successful management of the requirements. The success of the software depends on the accuracy and completeness of the requirements. Knowing the positive and negative aspects, advantages and disadvantages of the existing methodologies helps for selecting the appropriate methodology and choosing the appropriate approach to the situation and the project in this direction will increase the probability of success of the project. Although there is a lot of research in the literature about the traditional and agile approaches which are the main trends of software development approaches, the publications made in specific for requirements engineering process are limited. In this study, the necessary engineering applications in traditional and agile methodology approaches are evaluated to assist in the selection of methodology.

GEREKSİNİM MÜHENDİSLİĞİ BİLGİ ALANLARININ GELENEKSEL VE ÇEVİK YAZILIM YÖNTEMLERİ AÇISINDAN KARIŞLAŞTIRILMASI

Makale Bilgileri

Öz

Makale Tarihiçesi:

Geliş:
22 Mayıs 2019
Kabul:
12 Ağustos 2019

Anahtar Kelimeler:

Gereksinim mühendisliği;
Geleneksel yazılım geliştirme
yöntemleri;
Çevik yazılım geliştirme
yöntemleri;
Çevik ve geleneksel süreçlerin
karşılaştırılması;

Bir yazılım projesi uygularken dikkat edilmesi gereken önemli noktalardan biri gereksinimlerin başarılı bir şekilde yönetimidir. Yazılımın başarısı, gereksinimlerin doğruluğuna ve eksiksizliğine bağlıdır. Mevcut metodolojilerin olumlu ve olumsuz yönlerini, avantajlarını ve dezavantajlarını bilmek gerçekleştirilecek projeler için en uygun metodolojinin seçilmesine yardımcı olacaktır. Proje için kaynaklara, bağlama ve diğer unsurlara göre en uygun yaklaşımı seçmek de proje başarısını artırır. Literatürde yazılım geliştirme yaklaşımlarının ana eğilimleri olan geleneksel ve çevik yaklaşımlar hakkında çok fazla araştırma olmasına rağmen, gereksinim mühendisliği süreci için özel olarak yapılan yayınlar sınırlıdır. Bu çalışmada, uygun metodoloji seçiminde yardımcı olmak amacıyla geleneksel ve çevik metodoloji yaklaşımlarındaki gereksinim mühendisliği uygulamaları karşılaştırılmıştır.

1. Introduction

Every organization that develops software aims to produce quality products by realizing successful projects to meet the needs of its stakeholders. The Standish Group's report published in 2015 states that considering the budget, time, scope and quality constraints of the projects, the success rate is 29%. 19% of the projects were canceled, and the remaining 52% could be completed by budget or timeout, quality loss or scope contraction (Hastle & Wojewoda, 2015). According to the same report, in the top ten factors affecting the success or failure of the projects, "clearly stated requirements" are in the third place. Similarly, the changes and / or lack of requirements is the second and third among the project difficulties, and the first among the factors causing the cancellation of the project.

Requirements engineering is a process of how the possible system should be defined. Requirements serve as a guide for the software development team. The purpose of requirements engineering is to guide software development activities to produce the right software (Lawrence, Wieggers, and Ebert, 2001). If the requirements are not achieved correctly from the right stakeholders, it is highly likely that the project will fail, even if the rest of the project is well executed. Complete and accurate requirements provide many benefits, such as avoiding errors, improving quality and reducing risk in the software development process (Brooks, 1987; Procaccino, Verner, & Overmyer, 2002). The Software Engineering Body of Knowledge (SWEBOK) defines software requirements engineering activities as the determination, analysis, specification, approval of, and management of requirements throughout the entire lifecycle of the software product (Bourque & Fairley, 2014).

The ultimate goal of the traditional project management approach is to successfully complete the project in the planned time, budget, and scope following the plan set

up at the beginning of the project (Decarlo, 2004; Shenhar, & Dvir, 2007; Wysocki, Junior, & Crane, 2007). For this purpose, the requirements, and scope should be determined and fixed at the beginning of the project. It is assumed that there will be no major changes in the traditional approaches during the project which will affect the scope. Conventional requirements engineering is structured on these assumptions and documentation is mainly weighted (Hastle & Wojewoda, 2015).

Agile requirements engineering, however, specifies how to manage requirements in software development methods which implement the basic principles set in Agile Manifesto (Agile Manifesto, 2001). In this approach, documentation is given fewer space and software requirement activities are extended to the entire software development process (Bose, Kurhekar, & Ghoshal 2014; Boyer & Mili 2014; Lucia & Qusef 2010).

In the literature, it is seen that the studies aimed at comparing the traditional and agile approaches are carried out to cover all the steps of the software development process (Batool et al 2013; Elshandidy & Mazen 2013; Palmquist et al 2013; Seda & Tarhan 2010; Shinde, Tangyde, & Kulkarni 2015; Stoica, Mircea, & Ghilic-Micu 2013). Some studies on, examining traditional and agile processes in terms of requirements engineering activities, it has been observed that certain areas are focused. For example, Elshandidy and Mazen (2013) deal with traditional and agile methods in their planning, focus, documentation, development team roles, customer roles, development model, communication, management style, and quality control topics. Palmquist et al (2013) provide requirements acquisition, prioritization, modeling, documentation, validation, and management titles.

In this study, the requirements engineering discipline is vital to perform more successful software projects, is compared to the most widely applied traditional and

agile software methods based on six of the seven fields of information specified in the SWEBOK (Bourque & Fairley, 2014).

2. Requirements Engineering

Requirements engineering determines whether a new software project is to be implemented and, if the project is to be implemented, forms the basis of project launch activities. The acquisition of requirements requires the participation of end users, process owners, sponsors, developers and all other stakeholders. Requirements engineering ensures that the requirements for the successful completion of the project and providing a quality software product are obtained in a correct and complete manner. Thus, it can be ensured that all stakeholders are satisfied at the end of the project (More, Sapre, & Chawan 2011; What are the Software Development Models 2018).

Sommerville and Sawyer, claim that the requirements engineering process consists of five main tasks: elimination, analysis and negotiation, modeling, verification/validation and management (Sommerville & Sawyer 1997). However, SWEBOK breaks down requirements engineering in seven sections: software requirements fundamentals, requirements process, requirement elicitation, requirement analysis, requirement validation, practical considerations and software requirements tools (Bourque & Fairley, 2014)

In this study, the requirements engineering process is compared and discussed, on the basis of SWEBOK and previous studies, under the following sections:

2.1. Software Requirements Fundamentals

SWEBOK elaborates the software requirements fundamentals with a description of the definition of a software requirement, product and process requirements, functional and nonfunctional requirements, emergent properties, quantifiable

requirements, system requirements and software requirements (Bourque & Fairley, 2014).

2.2. Requirements Process

SWEBOK requirement process is addressed by process models, process actors, process support and management, process quality, and improvement (Bourque & Fairley, 2014). The work on the requirement process is mainly expected from requirement engineers and analysts.

2.3. Requirements Elicitation

Requirement elicitation activities are carried out to determine the scope of the system and to provide the requirements that determine the characteristics that the system must-have. It is necessary to work with project stakeholders and other requirements resources for the elicitation of requirements. During these studies, the scope of the system is determined and started to be managed. SWEBOK describes the requirements elicitation as resources of software requirements and requirements acquisition techniques (Bourque & Fairley, 2014). During the requirements elicitation process, face-to-face interviews, brainstorming, prototyping, workshops, focus groups, etc. techniques are used to carry out activities with all necessary stakeholders, including sponsors, initiators of project requests and domain knowledge experts (Batool et al 2013).

2.4. Requirements Analysis Knowledge Area

The SWEBOK requirement analysis addresses requirements classification, conceptual design, architectural design, linking requirements, negotiating requirements, and formal analysis (Bourque & Fairley, 2014). At this stage, the consistency, completeness, and applicability of the requirements obtained by the elicitation, are checked. The scope of the system is managed through negotiations and prioritization activities in this process. The scope management of the

system is also carried out through negotiations and prioritization activities at this stage.

2.5. Requirements Specification

The SWEBOK requirements specification is addressed by the system identification specification, system requirements specification, and software requirements specification (Bourque & Fairley, 2014). One or more analysis documents are created at this stage using the business analysis information. This information contains the functional and non-functional requirements, analysis, risk, constraints, dependencies, and other analysis information obtained and analyzed.

2.6. Requirements Validation

SWEBOK addresses the validation of requirements by reviewing requirements, prototyping, model validation, and acceptance testing (Bourque & Fairley, 2014). Requirements validation is performed by reviews and prototyping at the beginning of the development process. It is performed after the development process is completed, by acceptance tests over the resulting software and system components.

2.7. Practical Considerations

The topics related to SWEBOK implementation address the recurring nature of the requirement process, change management, requirements attributes, tracing requirements and measurement of requirements (Bourque & Fairley, 2014).

3. Current Requirements Engineering Approaches

Stoica et al. defines Software Development Life Cycle (SDLC) as a structure, which explains how software is developed and how maintenance and changes are carried out (Batoool et al 2013). In the literature, the defined international standard for SDLC is ISO/IEC 12207. This standard contains many software development models that demonstrate how to implement SDLC. In addition, some organizations are

creating and using organizations' own models. The most common of all these models can be listed as follows (Stoica et al 2013):

- Waterfall model
- V model
- Incremental model
- Rapid Application Development (RAD) Model
- Rational Unified Process (RUP)
- Microsoft Solutions Framework (MSF)
- Iterative model
- Spiral model
- Scrum
- Kanban
- Extreme Programming (XP)

Each model has advantages and disadvantages, and the model to be implemented should be selected according to the organization's capacity, needs, and project (What are the Software Development Models 2018; Software development process 2018; SDLC – Overview 2018).

3.1. Traditional Approaches

In the traditional software development, approaches such as waterfall and spiral, the requirement analysis, architectural design, software development, test and stabilization, and deployment phases required for revealing the software, are carried out in successive order. The basic motivation of the traditional method can be defined as determining a clear scope of the project, making a project plan and adhering to this project plan to reveal the project without departing the out of scope (Babok 2015; Bourque & Fairley 2014; Withall 2007).

Traditional approaches are based on the following assumptions (IEEE recommended practice for software requirements specifications 1998; Lawrence et al 2001; Software Requirements Engineering; Agile Manifesto 2001; More et al 2011):

- From the very beginning of the project, the clients know exactly what they expect from the system;
- The development team understands the needs of the customer accurately and clearly;
- Only one or more stakeholders have the responsibility to detail the requirements;
- There is a strict separation of different functions in teams.

The basis for traditional requirement specification approaches comes from the IEEE standard 830-1998 (Recommended Practice for Software Requirements Specifications). This standard describes how to document the requirements and any other business information that requires for the project to be accomplished. The standard generally refers to the requirement and the associated components to be expressed as texts. In addition to this, it is stated that user interfaces should be composed as prototypes except for the textual specification of the requirements (IEEE recommended practice for software requirements specifications 1998).

3.2. Agile Approaches

It is impossible to freeze software projects within a certain scope due to constantly changing priorities and competition. Therefore, the agile approach, being open to change, aims to develop the highest value-added product that can be produced within a certain period of time and with a specific source and budget. For this purpose, it focuses on the most effective way to manage the changes can occur during the software process. The

most effective way of responding to changes is possible with a system, based on close and face-to-face communication of the stakeholders and minimum documentation (Boyer et al 2014; Norton 2008; Patton et al 2014; Rubin 2013).

Although a specific specification method is not officially defined as an agile requirement specification, “User Story” is common in the literature and applications (Bose et al 2014; Norton 2008; Rubin 2013; Patton et al 2014)

4. Requirements Engineering in Traditional and Agile Approaches

In our study, we have discussed the differences between the traditional and agile approaches considering application similarities, advantages and disadvantages over six of the seven topics mentioned in SWEBOK. In Table 1, the basic concepts related to the requirement are considered especially in terms of the types of requirements. Comparisons related to the requirements processes, including process actors, are given in Table 2. In Table 3, requirement acquisition is discussed over resources, techniques and roles. In Table 4, requirement analysis is evaluated through concepts such as scope management, prioritization of requirements, while Table 5 presents the specification of requirements. Table 6 describes the validation of requirements and Table 7 describes the comparison of requirements engineering in terms of implementation. In addition, activities and concepts that are not included in SWEBOK but encountered throughout the study are also included in Table 8.

Table 1: Comparison of Traditional Methods and Agile Methods in Terms of Software Requirements Fundamentals

Key Activities and/or Concepts	Traditional Methods	Agile Methods
Requirements	It is assumed that they are stable and are known from the beginning.	It is assumed that they will mature and become clear over time.
Product and Process Requirements	Product and process requirements are handled and documented separately.	Requirements are followed in a format called user story, without product or process separation.
Functional and Non-Functional Requirements	Functional and non-functional requirements are documented separately and in detail.	User stories are created based on functional requirements. Non-functional requirements are included in the user story.
Emergent Properties	Emergent properties are documented under a separate section.	Emergent properties concerning multiple user stories are treated as system constraints.
Quantifiable Requirements	Each of the requirements is to be measured separately.	It focuses on measuring the usefulness of the functions that fulfill the requirements.

Table 2: Comparison of Traditional Methods and Agile Methods in Terms of Software Requirements Process

Key Activities and/or Concepts	Traditional Methods	Agile Methods
Process models	Waterfall method, spiral model, Rational Unified Process are the most widely used traditional methods.	Scrum, Kanban, Extreme Programming are the most widely used agile methods.
Process actors	Business stakeholders, analyst, project manager, software developers, test professionals, software architects are the most basic process actors.	It generally includes agile coach, product manager, and development team roles.
Process Quality and Improvement	Process quality and improvement are handled outside the project. It is executed by a separate role or team. During the project execution, the lessons learned are documented in order to contribute to the improvement.	In each cycle of the software development process, a planned activity for process quality and improvement is realized.
Process Support and Management	The requirement analysis process is carried out by the “analyst” role.	The analysis process can be conducted by the product manager, developers or team. In some applications, it is also carried out by the analyst.

Table 3: Comparison of Traditional Methods and Agile Methods in Terms of Software Requirements Elicitation

Key Activities and/or Concepts	Traditional Methods	Agile Methods
Requirements Sources	All relevant stakeholders are considered as the source of requirement in the elicitation phase.	In many agile methodologies, the development team works with a product manager as a source of requirement. However, the product manager is responsible for doing the necessary work with all other stakeholders for the acquisition of requirements. In some agile applications, the customer is included in the team and is considered a direct source of requirement.
Elicitation Techniques	It is performed by more formal techniques.	Close work and face-to-face communication techniques are preferred.
Roles that fulfill requirements elicitation	System analyst, requirement engineer, domain expert.	Product Owner, development team, and customers.

Table 4: Comparison of Traditional Methods and Agile Methods in Terms of Software Requirements Analysis

Key Activities and/or Concepts	Traditional Methods	Agile Methods
Requirements Classification	Requirements are classified and documented as functional, non-functional, and transition requirements.	Requirements are not classified. They all are covered in user stories.
Conceptual Modeling	The conceptual design is done with the designation of the requirements first and the architectural design and development are expected to be done accordingly.	Conceptual design is created within the first iteration and if required it is updated within each iteration.
Architectural Design and Requirements Allocation	It is based on the creation of a formal and visual model of the entire system.	It is carried out in the form of modeling only current iteration.
Requirements Negotiation	It is based on the persuasion of stakeholders for the successful implementation of the project plan.	It aims to prioritize the work that will provide the highest added value to the product.
Formal Analysis	When the project requires, formal analysis is applied.	Formal analysis is not applied.
Scope Management	Requirements are set in the early stages of the project. The scope is determined and frozen according to these requirements.	The scope is not frozen under any circumstances. In the later stages of the project, the scope can be differentiated according to stakeholder expectations and priority changes.
Requirements Prioritization	The prioritization of the requirements is carried out by the project manager and determined by the stakeholders.	The prioritization of requirements is carried out together by the product owner, agile coach and the customer altogether.

Table 5: Comparison of Traditional Methods and Agile Methods in Terms of Software Requirements Specification

Key Activities and/or Concepts	Traditional Methods	Agile Methods
Documentation of Requirements	All requirements are written in detail.	Documentation is not mandatory. However, if the development team needs it, it can do the documentation as needed and in detail.
Requirements documents	The system identification specification, system requirements specification, software requirements specification are minimum required documents.	It is executed through a simple list called Product Backlog.
Specification methods	Requirements can be specified with formal, textual, and UML Use Case methods.	Usually a user story is used to create a common reference point among stakeholders rather than documentation.

Table 6: Comparison of Traditional Methods and Agile Methods in Terms of Software Requirements Validation

Key Activities and/or Concepts	Traditional Methods	Agile Methods
Requirements Review	Requires formal review and approval.	Informal review meetings are held under the leadership of the product manager.
Prototyping	Prototyping is used where necessary.	It is often used for reconciliation with stakeholders.
Model validation	Model validation is used if necessary.	Model validation is used if necessary.
Acceptance Tests	Acceptance tests are carried out at the end of the project, based on the requirements set that determined and frozen at the beginning of the project.	Acceptance tests are performed at the end of each iteration, based on user stories that are set for iteration.
Verification	Each of the requirements is checked individually.	Definition of Done and Acceptance Criteria is used to verify quality expectations.

Table 7: Comparison of Traditional Methods and Agile Methods in Terms of Software Requirements Engineering Applications

Key Activities and/or Concepts	Traditional Methods	Agile Methods
Iterative Nature of the Requirements Process	Each step of SDLC is performed in the form of sequential activities.	All activities of the SDLC are repeated in each iteration.
Dealing with Changes	New requirements and change requests are seen as risky and negative for the project.	It is open to changes and any change requests that add value to the product are welcomed.
Change Management	Change management is carried out through a comprehensive process with many approval phases.	The changes are considered as a natural part of planning at the beginning of each iteration.
Requirements Attributes	Additional qualifications, such as when the requirement is received, when it is received and the owner of the requirement, are documented as detailed as possible.	Attributes are not used unless they are critical.
Requirements Tracing	The relations of requirements with each other are traced and documented in detail.	Relationships between user stories are traced in critical situations.
Measuring Requirements	Each of the requirements is measured separately.	It is based on the measurement of user stories that can consist of more than one requirement.

Table 8: Comparison of Traditional Methods and Agile Methods in Terms of the other Aspects

Key Activities and/or Concepts	Traditional Methods	Agile Methods
Timing of requirement activities	All requirements are determined at the beginning of the project.	Requirements are envisioned at the beginning of the project and are considered to change in the later stages of the project.
Involvement of stakeholders	Stakeholders are expected to be actively involved during the analysis.	Stakeholders are expected to support the project throughout the project.
Communication of requirements	Written and formal communication.	Face-to-face and informal communication.
Planning	Predictive	Adaptive
Focus	Process centric	Focus on adding value.

The findings presented in these tables are summarized in Table 9, summarizing the advantages and disadvantages of traditional methods and agile methods.

Table 9: Advantages and Disadvantages of Traditional Methods and Agile Methods

	Advantages	Disadvantages
Traditional Methods	<p>Since change management is carried out within detailed formality, it is possible to prevent changes that will not be really effective.</p> <p>Due to the tracing of the relations of the requirements in detail, the impact analysis of the change requests can be carried out effectively.</p> <p>Comprehensive and formal review and verification practices allow for better quality requirements.</p> <p>With detailed and comprehensive documentation, it is possible to create corporate knowledge base.</p> <p>Budgeting and calendar management can be done effectively in large projects by pre-determined scope of design.</p> <p>It is possible to work with a large number of stakeholders in hierarchically structured and dispersed locations.</p> <p>The requirements engineering process is designed to create standards for the entire organization.</p> <p>It is easier to reach the requirements and to understand due to compliance with corporate standards.</p> <p>Due to the detailed measurement of requirements, budget and calendar planning can be done more precisely and the final product can be defined from the beginning of the project.</p>	<p>The extent of adherence to the identified scope and the fact that change management is carried out in a detailed process prevents the project from responding quickly to changing priorities and business rules.</p> <p>Due to detailed documentation, formal verification and other requirement activities, high resource costs and additional time are needed.</p> <p>In fact, it is not possible to determine the scope at the beginning of the project by 100% in many projects. Based on this, prioritizations, plans, budget studies and architectural designs cannot meet project expectations and negatively affect project success.</p> <p>The success of the project is adversely affected when the change requests exceeding a certain rate in the future stages of the project.</p>
Agile Methods	<p>The fact that the process is fully open to the new requests and changes in cycles is able to provide the highest added value to stakeholders in projects where priorities and business rules change.</p> <p>The output of the project can be more accurate and reaches high quality because of the short form of iterations and small frequent increments that delivered to the end user.</p> <p>The cost of requirements engineering is lower through the lean documentation that is gradually developed during iterations.</p> <p>Negotiations of requirements can be managed better through stakeholder collaboration and frequent engagement.</p> <p>The requirements for the relevant iteration can be achieved quickly and practically.</p> <p>It enables the management of projects that are not specific at the beginning of the project and have requirements that may change with various effects throughout the project.</p> <p>Through the defined improvement step of the process within each cycle, the team and process performance can reach the highest level.</p>	<p>A comprehensive impact analysis cannot be performed due to the inability to trace the attributes of the requirements and their relations with each other.</p> <p>Documentation is optional and the agile team and customer decide together in which detail the document will be created.</p> <p>Since the documentation is optional, a knowledge base may not be created for the product.</p> <p>The output of the project cannot be seen as a whole from the beginning of the project.</p> <p>For extensive projects, face to face and informal communication with many stakeholders in different locations cause difficulties in practice.</p> <p>Since the entire system is not handled at once, additional costs of reworks may be encountered as cycles progress.</p>

4. Conclusions

As "Standish Group" clearly shows, regardless of which method is being used, requirement activities have a great importance in the success and product quality of the software project. In this study based on the determination above, requirements engineering has been compared in terms of traditional and agile methods. Based on the SWEBOK Requirements Engineering Knowledge Area, we have discussed the basic concepts and key activities in both methods with advantages and disadvantages.

According to the findings, traditional methods may be more suitable for projects where a defined scope can be established, the requirements can be determined from the beginning and will change very little during the project period. However, if the requirements are not mature enough and / or are expected to change during the project, then agile methods may be more advantageous. While agile methods are more suitable for projects with a stakeholder structure that is suitable for face-to-face communication, traditional methods seem to be more suitable for projects with stakeholders located in different locations. While traditional methods have a prominent place in organizations that have formalities and give importance to written documentation, agile methods can be preferred in smaller organizations.

Taking into consideration all of these, the size of the project, the complexity of the stakeholders, the level of maturity of the organization and the potential of change should be considered when choosing the most appropriate method to ensure the success of the project.

Conflict of Interest:

There is no conflict of interest.

References

Babok v3: A guide to business analysis body of knowledge. (2015). Toronto: IIBA.

BATOOL, A., MOTLA, Y.H., HAMID, B., ASGHAR, S., RIAZ, M.N., MUKHTAR, M., AHMED, Comparative study of traditional requirements engineering and Agile requirements engineering., 15th (ICACT), 1006-1014, 2013

BOSE S., KURHEKAR M., GHOSHAL I. (2014, October) Agile Methodology in Requirements Engineering, Retrieved from: <http://www.infosys.com/research/publications/agilerequirements-engineering.pdf>

BOURQUE, P., & FAIRLEY, R. E. (2014). SWEBOK: Guide to the software engineering body of knowledge. Los Alamitos, CA: IEEE Computer Society.

BOYER, J., & MILI, H. (2014). Agile Business Rule Development Process, Architecture, and JRules Examples. Berlin: Springer Berlin.

BROOKS. (1987). No Silver Bullet Essence and Accidents of Software Engineering. Computer, 20(4), 10-19.

DECARLO, D. (2004). EXtreme project management: Using leadership, principles, and tools to deliver value in the face of volatility. San Francisco, CA: Jossey-Bass.

ELSHANDIDY, H. & MAZEN, S., Agile and Traditional Requirements Engineering: A Survey, International Journal of Scientific & Engineering Research, 4 (9), 2013

GRIFFITHS, M. (2015). PMI-ACP exam prep: Rapid learning to pass the PMI Agile Certified Practitioner (PMI-ACP) exam. Minnetonka, MN: RMC Publications.

HASTLE, S., WOJEWODA S. (2015, October 4). Standish Group 2015 Chaos Report - Q&A with Jennifer Lynch. InfoQ. Retrieved from

- <https://www.infoq.com/articles/standish-chaos-2015>
- IEEE recommended practice for software requirements specifications. (1998). New York: IEEE.
- Industry implementation of International Standard ISO / 1995: (ISO/IEC 12207) standard for information technology: Software life cycle processes. (1998). New York: Institute of Electrical and Electronics Engineers.
- LAWRENCE, B., WIEGERS, K., and EBERT, C. "The top risk of requirements engineering," in *IEEE Software*, vol. 18, no. 6, pp. 62-63, Nov.-Dec. 2001.
- LUCIA A., QUSEF A., Requirements Engineering in Agile Software Development, *Journal Of Emerging Technologies In Web Intelligence*, 2, (3), Ağustos 2010.
- Manifesto for Agile Software Development, Agile Manifesto (2001). Retrieved from <https://agilemanifesto.org/>
- MORE N., SAPRE B.S., CHAWAN P.M., An Insight into the Importance of Requirements Engineering, *IJIC*, ,1 (2), 2011.
- NORTON D., Agile Estimation and Planning Is Moving From a Dictatorship to a Democracy, Gartner, 2008.
- PALMQUIST M.S., LAPHAM M.A., MILLER S., CHICK T, OZKAYA I., Parallel Worlds: Agile and Waterfall Differences and Similarities, CMU/SEI-2013-TN-021, Ekim 2013.
- PATTON, J., ECONOMY, P., FOWLER, M., COOPER, A., & CAGAN, M. (2014). User story mapping. Beijing: O'Reilly.
- PROCACCINO, J. D., VERNER, J. M., OVERMYER, S. P., & DARTER, M. E. (2002). Case study: Factors for early prediction of software development success. *Information and Software Technology*, 44(1), 53-62.
- RUBIN, K. S. (2013). *Essential Scrum: A practical guide to the most popular agile process*. Upper Saddle River, NJ: Addison-Wesley.
- SEDA, G. Y., TARHAN, A., Çevik Süreç ile Kıyaslamaya Temel Olarak Artırımsal Sürecin Nicel Analizi: Bir Durum Çalışması, 2. Yazılım Kalitesi ve Yazılım Geliştirme Araçları Sempozyumu, 2010.
- SHENHAR, A., & DVIR, D. (2007). *Reinventing project management: The diamond approach to successful growth and innovation*. Boston, MA: Harvard Business School Press.
- SHINDE L.K., TANGDE Y.S., KULKARNI R.P. Traditional vs. Modern Software Engineering - An Overview of Similarities and Differences. *Advances in Computational Research*, 7 (1), 187-190, 2015.
- SILLITTI, A., CESCHI, M., RUSSO, B., & SUCCI, G. (n.d.). Managing Uncertainty in Requirements: A Survey in Documentation-Driven and Agile Companies. 11th IEEE International Software Metrics Symposium (METRICS05). doi:10.1109/metrics.2005.29
- Software development process (2018, October), Wikipedia (2018), Retrieved from: https://en.wikipedia.org/wiki/Software_development_process
- Software Requirements Engineering: What, Why, Who, When ... (n.d.). Retrieved from http://westfallteam.com/Papers/The_Why_What_Who_When_and_How_Of_Software_Requirements.pdf
- SOMMERVILLE I, and SAWYER, P. *Requirements Engineering*, John Wiley & Sons, 1997.

STOICA, M., MIRCEA, M., & GHILIC-MICU, B.
(2013). Software Development: Agile vs.
Traditional. *Informatica Economica*, 17(4/2013),
64-76. doi:10.12948/issn14531305/17.4.2013.06

Try QA, What are the Software Development Models
(2018, October), Retrieved from:
[http://istqbexamcertification.com/whatare-the-
software-development-models/](http://istqbexamcertification.com/whatare-the-software-development-models/).

Tutorialspoint, SDLC – Overview (2018, October) ,
Retrieved from:
[http://www.tutorialspoint.com/sdlc/sdlc_overview.
htm](http://www.tutorialspoint.com/sdlc/sdlc_overview.htm)

WITHALL, S. (2007). *Software requirements patterns*.
Erscheinungsort nicht ermittelbar: Microsoft Press.

WYSOCKI, R. K., JUNIOR, R. B., & CRANE, D.
(2007). *Effective project management: How to
plan, manage, and deliver projects on time and
within budget*. (Fourth Edition) New York: John
Wiley.