Software Engineering for Smart Healthcare Applications

Neelu Lalband, D. Kavitha

Abstract:Software engineering has been used by software vendors and consultants in the development of quality health care applications ranging from electronic medical systems, patient record management applications to medical middleware devices. As a discipline, it has evolved over the last decade in the production of high-quality software across many industries. Healthcare applications demand unique expertise tailored to best project methodologies and software development models. Many health care providers argue that best software practices and user-centered design principles are vital to producing quality applications across all domains. Lack of focus on systematic software development process increases flaws in the implementation causing a loss regarding quality, cost, and trust. The survey paper seeks to analyze existing models in the area of Software Engineering and to propose best SDLC model for Smart Healthcare applications which focuses on quality improvement. This survey also includes identifying the research challenges of software engineering for smart applications.

Keywords:Software Engineering, SDLC, Software Applications, Smart Healthcare, Medical Services

I. INTRODUCTION

The tremendous growth in technology has led to the application of software systems in our day-to-day life. Software engineering can be defined as a dynamic discipline that aims at producing high-quality software systems through systematic project methodologies tailored to user-centered design principles. It is aided by Software Development Life Cycle (SDLC), a set of basic procedures used in design, development, and testing of Software applications. Most common SDLC models include waterfall, agile, V-model, spiral, and Rapid Application Development are the key development methodologies in software production process [1]. The software models have boosted the development of millions of software applications for desktops, mobile devices as well as smart devices. During the development process, it is essential to consider the user design principles and methodologies to ensure the quality of information systems.

Health care systems are aimed at providing best patient support as well as offer optimal support for medical care. As such, healthcare applications must ensure fair financial management, excellent medical services support, and overall delivery of excellent services [2]. In the recent past, the healthcare industry has grown rapidly both in low level and middle-level income countries. However, implementation of health systems has been met by various challenges from a choice of SDLC model, elicitation of user requirements to design and deployment of applications. Use of health information systems will improve the quality of medical services, increase the quantity of health services and reduce significantly medical errors encountered. Contrary, implementation of health systems appears to be a complicated process [3].

It is a fact that smart healthcare significantly reduces the cost and increases the accessibility, yet the use of the technology may also depend on many other factors that cannot be easily quantified. In this context, a sociotechnical perspective stands as an important indicator toward understanding the use of the technology by the end users. For the success of the technologies, it is critical to develop an understanding about existing smart healthcare opportunities and what influences on the users' decisions to use these technologies. Otherwise, lack of sociotechnical knowledge may impede the potential success of the technologies. The adoption decision of users (i.e., patients, healthcare providers) may show changes over the time considering the change in technologies and culture. [27]

Thus, there is no golden standard to assess user intentions, but a continuous investigation is required to understand the smart healthcare technologies as well as the influencing factors to use these technologies.

Moreover, engineering principles and methodologies are being applied in the health sector just like in other domains. Significantly, the IT specialists and software developers need to consider the nature of applications in the health industries during the development process [4]. Although the systems are simple in principle, it is essential to ensure that they are implemented in a structured way that addresses requirements such privacy while balancing usability requirements. The healthcare applications need to map the hospital model appropriately to fit the existing processes as well as align with practitioner's specialty.

Our paper is structured as follows: In the next section, we discuss Systematic Review. Next, we outline Software Development Life Cycle with differences then presented and our findings are discussed. We conclude by presenting a agile method is more suitable for smart healthcare applications.

II. SYSTEMATIC REVIEW

First, in this paper, a review is based on the last decade of software development life cycle models proposed by [1], to explore different SDLC methods applied in the implementation of smart healthcare applications. Next, a description of each activity is given in-depth with a description on how to achieve it.

Revised Manuscript Received on April 12, 2019.

NeeluLalband,Research Scholar, Dept of C.S.E JNTUA Anathapuram, A.P, India (Email: neelu.lalband@gmail.com)

Dr. D. Kavitha,Professor, Dept of C.S.E G.Pulla Reddy college of Engineering (Autonomous), Kurnool, A.P, India (Email: dwaramkavithareddy@gmail.com)

Step 1: Classify the necessity for the review

As discussed previously, development of quality health care systems that meet the complex nature of healthcare delivery, with streamlined management of patient information records across the extensive medical care settings is a huge challenge. In order to achieve accuracy and quality, best SDLC model should be proposed.

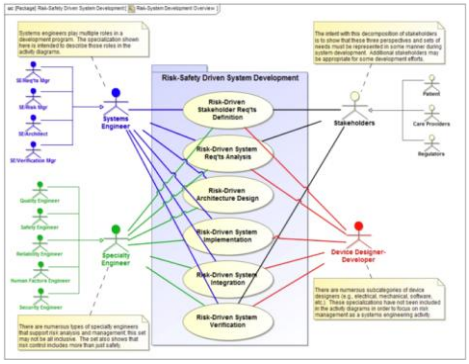

Fig. 1. Overview of System development process [5]

Step 2: Definition of research questions

In this stage, formal research questions are defined. The principal objectives of this survey are to respond the subsequent research questions (RQ):

• RQ.1: What are the different SDLC models to develop any kind of software applications?

• RQ.2: Which is the best SDLC model proposed to improve the software application quality?

• RQ.3: Which is best SDLC suitable for smart healthcare applications?

• RQ.4: What are the research challenges of software engineering for smart applications?

Step 3: Definition of search string

In this stage, the keywords to be used in preferred search tools are defined. In this paper, the next expressions of words were adopted: "Software Engineering" AND "Software development life cycle" AND "Smart Healthcare."

Step 4: Definition of research sources

In this survey, few sources and databases were utilized in information gathering. Therefore, many articles, journals, and publications were obtained from the databases and references based on applicability to the defined search terminologies. The sources or databases and corresponding URLs are described below.

Online Databases searched

TABLE 1. Online Sources Searched for Relevant Primary Studies

Database	***URL***
IEEE Explore	http://ieeexplore.ieee.org/Xplore/home.jsp
ScienceDirect	www.sciencedirect.com
ACM Digital Library	www.acm.org/dl
Springer	www.springerlink.com
Google Scholar	http://scholar.google.com/
CiteseerX	http://citeseerx.ist.psu.edu/index

Step 5: Definition of principles for inclusion or exclusion

Scope analysis for this survey paper was limited to journals, books, conferences articles, and publications published between 2012 and 2017. The keywords "software engineering" OR "software development life cycle" and "smart healthcare" were vital to be included in searched studies.

Step 6: Definition of data extraction procedure

The procedure for extracting data is defined on conventional items present in each journal consisting the keywords, and future technology works.

Step 7: Evaluation of quality studies

Evaluation of quality and relevant studies was centered on relevance examination of articles or journals about Software Engineering for Smart Healthcare Applications.

Step 8: Extraction of relevant studies

It involved the application of the data mining procedures stated in task 6 for all relevant critical studies gathered from the search process.

Step 9: Presentation of study overview

This task entailed discussion of all studies or information gathered in the task as well to categorize and illuminate them concerning the RQ defined task 2.

Step 10: Presentation of results to research questions

After an outline of primary studies in software engineering of health care systems, a discussion is given to answer the research questions defined in Activity 2. The results of this activity are presented in "Discussion" section as well.

III. SOFTWARE DEVELOPMENT LIFE CYCLE (SDLC)

The development process of system applications is primarily determined by different project methodologies [6]. The development approach defines a structure used in planning, design and manages the implementation process of computer systems. Aided by software engineering; a systematic approach to the design of software, the software methodologies result to quality software systems balanced with usability requirements. As such, SDLC provides a balanced, systematic practice for the development of high quality and reliable computer systems. The SDLC framework describes sequences of activities that are involved in the process of software production that IT specialists and software engineers must follow. It presents a standard procedure that outlines all the activities undertaken in software development and maintenance.

However, several variations of SDLC model such as waterfall, V-model, and spiral models have evolved defining different processes and guiding principles of software development [7]. The SDLC model presents a set of phases where each phase depends on the results of the preceding stage. Various SDLC models are considered for different

projects based on their suitability to project conditions such as user's requirements, project risks, cost, and development timeframe. A particular SDLC model may suit a particular project while at the same time other models may appear suitable for the elicited requirements but is it essential to consider trade-offs when choosing an SDLC model. A formal SDLC model theoretically consists of the following phases for developing and implementing computer software.

- Planning
- Analysis
- Design
- Implementation
- Testing
- Deployment and Evolution

Planning and Requirement Analysis: It's the first basic phase in which the need for a software product is analyzed. The user-requirements are gathered for development [6]. The information gathered is used in preparation for ideal project approach as well as feasibility examination in the cost-effective, functional and technical perspectives. After analysis stage is done, a clear define document for product development is prepared called as Software Requirement Specification (SRS).

Design: The next phase of SDLC is Design. It is concerned with designing a basic structural framework that identifies the significant component of the product and the communication between these components.

Implementation: During this stage, the actual process of product development is executed. The product meets the SRS is produced.

Deployment: In this phase verification and validation of the product is done.

- Verification: When the software product is built in the right way.
- Validation: When the right software is built.

Evolution: It is the last phase of SDLC. Once the product is ready after completion of testing, it is deployed into the runtime environment. Based on the feedback of the product may enhance for further development which is called as maintenance.

Common Software Engineering Methodologies: Various models of software development have been defined to be applied during software implementation process. Such models can be classified into two categories.

- Traditional and Classical Models
  - ✓ Waterfall
  - ✓ Spiral
  - ✓ V model
  - ✓ Rapid Application Development
- Modern and Popular Models
  - ✓ Agile

A. Waterfall Model

The waterfall model defines a classical and sequential approach to software engineering. It is among the earlier models of SDLC mostly applied in significant projects in corporate industries and government organizations [6]. It emphasizes on early project planning in the initial stages to eliminate design flaws before the beginning of software development. Obstacles such as time-consumption and unclear user requirements are identified earlier before other phases begin. Moreover, its intensive planning and documentation ideal for projects concerned with quality control. The waterfall model consists of various non-overlapping phases such as:

- Requirement Elicitation and Analysis
- Design
- System Implementation
- Testing
- Deployment
- Maintenance

The above stages are cascaded subsequently with the progression of development flowing downwards across the stages hence the name waterfall. In waterfall model unless until one phase is completed another phase is not started [6]. The model is not a modest rectilinear method but in entails sequential iterations defining the implementation activities. Once the product enters the testing phase, it's very difficult to do the changes as per new requirements. Production and document approval through iterations are expensive to require a lot of work. Not recommended for large projects.

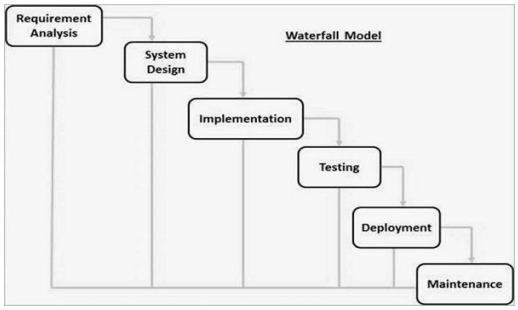

Fig. 2. Waterfall Model

Basic Principles:

- The project is divided into various stages with few overlaps between stages.
- More emphasis is given to planning phase, timeframe, budget, and implementation of the whole project [6].
- There is strict control over the model cycle of the task through broad reviews, certification and approval user acceptance by the management after implementation phase ends.

B. Spiral Model

It is the first model to explain the need for iteration and why it matters. During the development process, the iterations are scheduled between six months and two years [8]. Each stage begins with a plan objective of the design and terminates with a review from the user on the results and project progress. It emphasizes on the risk analysis as illustrated in the figure below. The entire life cycle is denoted in a spiral since the model involves a sequential flow of events with few back-pedaling from preceding action to another. Each loop is signified into the stage of software process [6].

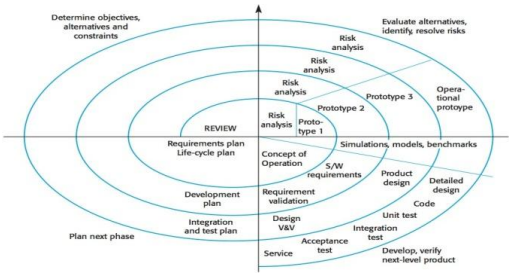

Fig. 3. Spiral Model

Each loop is divided into four segments:

- Objective Set
- Risk evaluation and elimination
- Implementation and validation
- Planning

The software process starts at the center of spiral and moves in clockwise direction. In each phase, intensive analysis efforts are applied with the primary focus on the result of the software product. Each loop results in deliverables or outcomes [9].

C. V-Shaped Model

This model resembles waterfall model in its sequential path execution of processes where each stage is finalized before proceeding to the next step [9]. Unlike waterfall method, the testing stage is emphasized in the v-model. Testing guidelines are formulated in initial stages before the implementation, and other stages are preceding coding. Prior to the development process, a system plan is designed based on the system functionality to meet all user requirements [10]. Testing of the project is planned in parallel of corresponding to phase in V-model as mentioned in figure 3.

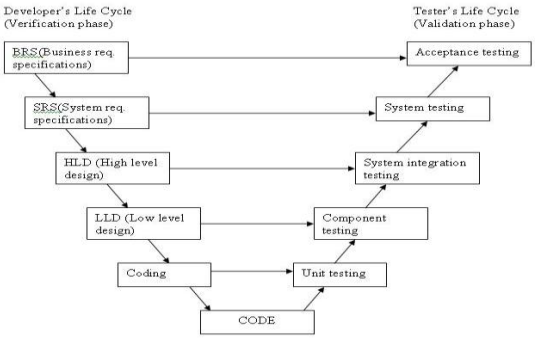

Fig. 4. V-Shaped Model

D. Rapid Application Development

This model was evolved from fourth generation language in the 80s and used for applications developed that are data intensive [11]. This set of tools allows data to created, searched, displayed and presented in reports.

The tools that are included in RAD Environment are in figure 5.

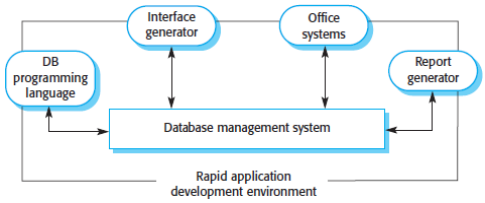

Fig. 5. RAD Environment Model

RAD enables fast development and implementation of high level and quality applications, unlike the traditional methodologies. Its main aim is to achieve a quality and efficient software at an affordable price. RAD is integrated with Computer-Assisted Software Engineering (CASE) tools and practices, to ensure user-centered design balanced with product quality [11].

E. Agile Model

It is a model for developing software applications where project implementation is done iteratively or incrementally [12]. In this model change or modification of user, requirements are allowed thereby encouraging constant responses from the customer or client. In this model, the phases don't happen in progression but are flexible for the adoption of any changes. As such, the cycle stages are executed in parallel. The figure below represents the phases of the Agile model [13].

Fig. 6. Agile Development Cycle

The model focuses on customer satisfaction through constant delivery of a functional software product. The Agile Development cycle involves the following activities [6]:

- The development cycle begins with a meeting.
- User requirements are elicited, understood and prioritized.
- A system development plan is created.
- Estimation of the complex nature of each requirement is performed.
- Design diagrams are drawn to illustrate the functionality of the system.
- Development of the project is carried out either in a team or pairs.
- Continuous test of the program code is done corresponding to an integration of system components.
- Centered on the user feedback and review, the system code is refactored to ensure the better design is achieved. It may involve the addition of features or structuring of the user interface appearance.

F. Difference between Traditional and Agile software development

Agile methods are based on adaptive software development methods, while traditional SDLC models (waterfall model, for example) are based on a predictive

approach. In traditional SDLC models, teams work with a detailed plan and have a full list of characteristics and tasks that must be completed in the next few months or the entire life cycle of the product. Predictive methods completely depend on the requirement analysis and careful planning at the beginning of the cycle. Any change that is to be included will go through a strict change control management and prioritization. The agile model uses an adaptive approach where there is no detailed planning and only clear future tasks are those related to the characteristics that must be developed. The team adapts to dynamic changes in the product requirements. The product is frequently tested, minimizing the risk of major faults in the future. Interaction with the clients is the strong point of agile methodology and open communication and minimal documentation are typical characteristics of the agile development environment. Teams collaborate closely and often are located in the same geographical space.

While agile SDLC is better suited for small and medium projects, on large scale traditional SDLC is still the better choice. Therefore it is important that the development team selects a SDLC that is best suited for project at hand. There are criteria that can be used by the development team to identify the dimension of the desired SDLC. They include team size, geographical location, size and complexity of the software, project type, business strategy, engineering capabilities etc. Also, it is very important for the team to study the differences, advantages and drawback of each SDLC before making a decision. Furthermore, the team must study the context of the business, industry requirements and business strategy before evaluating the candidate SDLCs. It is important to have a SDLC evaluation and selection process because it maximizes the chances to create successful software. Therefore, selection and adoption of an appropriate SDLC is a management decision with long term implications.

Although agile methodologies triumph over traditional ones in several aspects, there are many difficulties in making them work. One of them is the significant reduction of documentation and the claim that the source code itself should be the documentation. [16]

Thus, developers used to agile methods tend to insert more comments in source code in order to clarify and explain. It is difficult for beginner developers or new members of the team to complete their tasks when they cannot fully understand the project. They ask lots of questions to the experienced developers and this may delay completion of the iteration, which can lead to increased development costs.

On the other hand, traditional methods emphasize documentation in orientation and clarification of the project for the development team, so there is no concern about not knowing the project details or not having a knowledgeable developer. Agile methodologies are well known for the importance given to communication and client implication. [25]

For each version delivered, the development team and the clients will organize a meeting where the team will present the work done in current iteration and the clients will provide feed-back on the delivered software (improvements on current features or addition of new ones).

Most times, developers will find the periodic meeting (usually weekly) boring and tiring because they have to present the modules repeatedly, to new members and clients and, on each iteration, changes may happen as requested by clients. The time frame for each iteration is short (usually weeks). Developers will find the schedule too tight for each module, even more so for modules that require complex processing algorithms. This leads to delays in each iteration and hardships in establishing an efficient communication between team members and the clients.

On the other hand, traditional methodologies have a well-defined requirements model before the implementation and coding process starts and this acts as a reference for the development team during the coding process. Clients do not participate in this stage of the development life cycle. The development team will perform the coding according to the documentation provided by the business analysts until the system is complete and only then it will be presented to the clients as final product. Developers are not concerned about frequent meetings and have more time to finish the system. This allows them to provide a better product. The fact that agile development allows changes in requirements in an incremental way lead to two dependency problems in design: rigidity and mobility. Rigidity means a change in the system leads to a cascade of changes in other modules, while mobility means the inability of the system to include reusable components because they involve too much effort or risk. When such problems are present throughout the system, there must be a high level restructuring in order to eliminate unwanted dependencies. [26]

TABLE 2. Comparisons between Traditional and Agile Methodology [22 - 24]

Parameter	Traditional Methodology	Agile Methodology
Centered Hypothesis	Systems are completely specified and developed based on planning.	The software is specified with high and implemented by medium project teams applying best practices of development.
Management Approach	Based on control	Based on team leaders and collaborations
Management of Knowledge	Clear	Inferred
Communication	Conventional	Casual
Methodology	Either waterfall or other models	Iteratively method
Development Approach	Analytical	Adaptive
Implementation Orientation	Centered on process	Based on people

Quality	The planning phase is hard to control and achieve the excellent end result.	Controlling phases of analysis through design testing is easy.
Project requirements	Well outlined and defined	Interactive input
Project cost	High	Low
Direction of implementation	Fixed	Flexible and can be changed
Project Testing	Done at the end	After each deliverable (iteration)
Project scale	Large	Medium or at times low
Software engineers	Based on the system plan with implementation abilities.	Active and knowledgeable, collaborative.
User	Has access skills, responsive and empowered.	Knowledgeable, committed and collaborative.
Restructuring	Costly	Cheap
Team Size	Large	Small
Adaptability to Change	Change Sustainability	Change Adaptability
Documentation	High	Low

IV. DISCUSSION

A. Software Engineering for Smart Healthcare Applications

Healthcare industry is very dynamic and incorporates technologies into their processes so as to improve the quality of life of patients and efficiency. At times, it is subjected to regulatory policies and other health compliance requirements and as such, health care systems must meet the required standards. By adapting information technology, e-health was introduced, later the widespread of mobile technology introduced m-health [27]. As advancement in technology healthcare is also transforming and adapting new technology called smart healthcare. Mainly these applications are developed for chronic patients, taking care of elderly persons, physical fitness, etc. [17]. To develop a quality, suitable and most effective application, various challenges are encountered. The challenges identified are:

- Selection of process
- Software Design
- Middleware Challenge
- Software Quality
- Software Maintenance

Compared to traditional model Agile software development would be more suitable for Smart Healthcare Application development [27]. The challenges identified can be easily handled by the agile model as stated in comparison table 2.

B. Healthcare Application Issues

Health care systems are complex due to the dynamic nature of the socio-technical setting. Their concerns that smart applications may cause errors instead of correcting them [18]. When developing healthcare systems, it is ideal to consider the following issues:

- Security: Healthcare systems are different from other computer systems as it stores sensitive information such as patient records, confidential data, and financial data. Therefore, software vendors should consider security requirements when developing healthcare applications [19].
- Performance: It is a most important requirement for any system. In healthcare systems, performance is related to data; its accuracy and availability.
- Reliability: Healthcare providers operate on a 24/7 basis and as such as the health care system should be reliable in every business process.
- Availability: It entails the proportion of time when the system is running or operational. In healthcare, availability is a crucial requirement throughout the delicate and sensitive medical process such as surgery and diagnostic.

C. Final Considerations

During software development of a health care system, there must be a universal understanding among medical personnel and software developers. In each stage of development, various concerns must be considered to ensure delivery of a quality application [20].

Planning and Analysis Phase: It is most challenging phase thus a system plan should be created to guide the project team throughout the execution process of the system. Also, emphasized should be given to communication between the development team and all other stakeholders [20].

Design Phase: The requirements from the initial stage should be documented as well as the establishment of the system architecture. The architecture should define interfaces, components and behavioral characteristics of the healthcare application. The by-product; design document should demonstrate a technical plan for the system implementation [21].

Implementation Phase: The software developers build the components defined in the design stage. Issues of performance, quality, security, and reliability should be emphasized.

Validation and Delivery Phase: System validation should be undertaken by a team that knows the systems to avoid mistakes and errors. As such, it should be carried by a joint team that encompasses the system testers, practitioners, development team, health care providers and the end users [21].

V. RESULT & DISCUSSIONS

In this survey, a brief discussion about different software development lifecycle models used in the software industry is given. Although there are many development models, only a few basic models which are commonly used were discussed. Consequently, a comparison is made in between traditional and agile models to identify which is a better model. As highlighted, many software industries are shifting from traditional models to agile models to achieve quality

and save cost and time. Also, the survey featured a discussion on the growth and transformation of the healthcare industry as well as challenges of smart healthcare. Also, during healthcare software development, various issues such as reliability, security, performance, and availability should be considered. From the comparison of various software development cycles, it is evident that agile method is best suited for the development of smart healthcare systems. In the future, we can concentrate more on different types of agile models which will be suitable for which application.

REFERENCES

1. Kneuper, R. (2017). Sixty Years of Software Development Life Cycle Models. *IEEE Annals of the History of Computing*, *39*(3), 41-54.
2. Weber-Jahnke, J. H., Price, M., & Williams, J. (2013, May). Software engineering in health care: Is it really different? And how to gain impact. In *Proceedings of the 5th International Workshop on Software Engineering in Health Care* (pp. 1-4). IEEE Press.
3. Stroulia, E., Chodos, D., Boers, N. M., Huang, J., Gburzynski, P., & Nikolaidis, I. (2009, May). Software engineering for health education and care delivery systems: The Smart Condo project. In *Software Engineering in Health Care, 2009. SEHC'09. ICSE Workshop on* (pp. 20-28). IEEE.
4. Richardson, I., Reid, L., & O'Leary, P. (2016, May). Healthcare systems quality: development and use. In *Software Engineering in Healthcare Systems (SEHS), IEEE/ACM International Workshop on* (pp. 50-53). IEEE.
5. SEBoK, "IEEE Computer Society," 2017. [Online]. Available: http://sebokwiki.org/wiki/Healthcare_Systems_Engineering. [Accessed 27 01 2018].
6. Rastogi, V. (2015). Software Development Life Cycle Models-Comparison, Consequences. *International Journal of Computer Science and Information Technologies*, *6*(1), 168-172.
7. Cusumano, M. A., & Smith, S. A. (1995). Beyond the waterfall: Software development at Microsoft.
8. Boehm, B., Lane, J., Koolmanojwong, S., & Turner, R. (2014). The Incremental Commitment Spiral Model.
9. Massey, V., & Satao, K. J. (2012). Evolving a New Software Development Life Cycle Model (SDLC) incorporated with Release Management. *International Journal of Engineering and Advanced Technology (IJEAT)*, *1*(4), 25-31.
10. Balaji, S., & Murugaiyan, M. S. (2012). Waterfall vs. V-Model vs. Agile: A comparative study on SDLC. *International Journal of Information Technology and Business Management*, *2*(1), 26-30.
11. Beynon-Davies, P., Carne, C., Mackay, H., & Tudhope, D. (1999). Rapid application development (RAD): an empirical review. *European Journal of Information Systems*, *8*(3), 211-223.
12. Stoica, M., Mircea, M., & Ghilic-Micu, B. (2013). Software Development: Agile vs. Traditional. *Informatica Economica*, *17*(4).
13. Gannon, M. (2013, March). An agile implementation of scrum. In *Aerospace Conference, 2013 IEEE* (pp. 1-7). IEEE.
14. Leau, Y. B., Loo, W. K., Tham, W. Y., & Tan, S. F. (2012). Software development life cycle AGILE vs traditional approaches. In *International Conference on Information and Network Technology* (Vol. 37, No. 1, pp. 162-167).
15. Aitken, A., & Ilango, V. (2013, January). A comparative analysis of traditional software engineering and agile software development. In *System Sciences (HICSS), 2013 46th Hawaii International Conference on* (pp. 4751-4760). IEEE.
16. Bustard, D., Wilkie, G., & Greer, D. (2013, April). The maturation of agile software development principles and practice: Observations on successive industrial studies in 2010 and 2012. In Engineering of Computer Based Systems (ECBS), 2013 20th IEEE International Conference and Workshops on the (pp. 139-146). IEEE.
17. Marjomaa, S. (2015). Business model for mobile healthcare delivery in chronic disease management.
18. Al Ameen, M., Liu, J., & Kwak, K. (2012). Security and privacy issues in wireless sensor networks for healthcare applications. *Journal of medical systems*, *36*(1), 93-101.
19. Christov, S. C., Conboy, H. M., Famigletti, N., Avrunin, G. S., Clarke, L. A., & Osterweil, L. J. (2016, May). Smart checklists to improve healthcare outcomes. In *Software Engineering in Healthcare Systems (SEHS), IEEE/ACM International Workshop on* (pp. 54-57). IEEE.
20. Jalote, P. (2012). *An integrated approach to software engineering*. Springer Science & Business Media.
21. Clarke, P., & O'Connor, R. V. (2012). The situational factors that affect the software development process: Towards a comprehensive reference framework. Information and Software Technology, 54(5), 433-447.
22. Nerur, S., Mahapatra, R., & Mangalaraj, G. (2005). Challenges of migrating to agile methodologies. Communications of the ACM, 48(5), 72-78.
23. Stoica, M., Mircea, M., & Ghilic-Micu, B. (2013). Software Development: Agile vs. Traditional. Informatica Economica, 17(4).
24. Boehm, B. (2002). Get ready for agile methods, with care. Computer, 35(1), 64-69.
25. Petersen, K., & Wohlin, C. (2009). A comparison of issues and advantages in agile and incremental development between state of the art and an industrial case. Journal of systems and software, 82(9), 1479-1490.
26. Hanssen, G. K., Yamashita, A. F., Conradi, R., & Moonen, L. (2009, September). Maintenance and agile development: Challenges, opportunities and future directions. In Software Maintenance, 2009. ICSM 2009. IEEE International Conference on (pp. 487-490). IEEE.
27. Sezgin, E., Yildirim, S., Yildirim, S. Ö., &Sumuer, E. Current and Emerging mHealth Technologies.