Computers, Materials & Continua CMC, vol.64, no.2, pp.777-796, 2020

A Comparative Analysis of RAD and Agile Technique for Management of Computing Graduation Projects

Fazal Qudus Khan[1], Saim Rasheed[1], Maged Alsheshtawi[1], Tarig Mohamed Ahmed[1, 2] and Sadeeq Jan[3, *]

Abstract: Computing students face the problem with time and quality of the work while managing their graduation/senior projects. Rapid Application Development (RAD) model is based on continual user involvement for the process of requirement gathering via prototyping. After each iteration, the developers can validate the requirements that are completed in the iteration. Managing a project with RAD is easier but not flexible. On the other hand, Agile project management techniques focus on flexibility, agility, teamwork and quality based on user stories. Continual user involvement is avoided, which requires extensive maintenance time for fixing iteration and release of the story points. This also makes it necessary to provide onsite training to the users of the application. This research provides the pros and cons of RAD and Agile project management techniques, to help students in deciding the best approach for managing their graduation projects. For the evaluation of these techniques, similar case studies were given to the senior project students (having similar CGPAs) for building similar functionality-based applications. The two projects "Life Organizer" developed and managed using RAD and "Smart Patient Assistant (SPA)" developed and managed through Agile methodology were evaluated against the quality assurance criteria for senior projects. The study found that the project developed with RAD methodology performed 13.33% better in providing extensive and elaborated documentation than the students following the Agile technique. On the other hand, SPA-Agile based project, due to teamwork had 2.5% better implementation than Life Organizer-RAD based project.

Keywords: Rapid application development, prototyping, agile project management, smart application development, graduation project, thesis.

1 Introduction

Graduation project/Senior projects for undergraduate computing programs are mostly

[1] Department of Information Technology, FCIT, King Abdulaziz University, Jeddah, Saudi Arabia.

[2] University of Khartoum, Department of Computer Science, Khartoum, Sudan.

[3] Department of Computer Science & IT, University of Engineering and Technology, Peshawar, Pakistan.

[*] Corresponding Author: Sadeeq Jan. Email: sadeeqjan@uetpeshawar.edu.pk.

Received: 09 April 2020; Accepted: 20 April 2020.

application based. Students must complete the project in a timely manner i.e., within a semester or within an academic year depending on the institution rules. Deciding on how to go about the development of application and its documentation is a tedious job. A student must usually take a few subjects along with the project, hence making it is quite difficult to manage time and schedule for the project. In this research, we compare two approaches/methodologies of application development, in order to give senior project students an idea of the pros and cons of these methodologies.

The Rapid Application Development model for managing software application development is based on continual user involvement for the process of requirement gathering, also known as prototyping. RAD model is a quick and iterative development methodology in which the product owner feedback is a necessary element, before releasing the final version. After each iteration, the developers can validate the requirements that are completed and work on the amendments suggested (if any). RAD, is an alternative to the typical waterfall development model which was first introduced in 1991. it's now one of the most powerful development methods for managing small and medium size projects. By using a rapid application method such as RAD, designers and developers can utilize the knowledge (user feedback) they earned during the development process, to enhance and improve the design or change the software direction [Mackay, Carne, Beynon-Davies et al. (2000)]. Various systems such as generic IO-Link interface [Chavez and Wollert (2020)], Design of e-Marketplace for Village-owned Small, Micro and Medium Enterprise [Andreswari, Ambarsari, Syahrina et al. (2020)], multimedia prayer application for education [Rosyad, Syukur, Busro et al. (2020)] are developed through RAD. Agile project management has edge over other strategies due to customer engagement, reduced risk, support of team diversity, high availability and reduced methods uncertainty as described by Chow et al. [Chow and Cao (2008)]. The success of large-scale applications is more with agile methods [Jorgensen (2019)].

This research includes in detail, the development of a smart reminder application using the two approaches. Sections 2 discusses the related work and the details of the types of applications used in our case studies, Section 3 puts forward details of the two case studies implemented using RAD and Agile project management techniques, Section 4 discusses results of our implementation and testing, and finally, Section 5 concludes the project.

2 Related work

This section is divided into two parts. In the first section, we discuss different mobile applications that exist for reminders because our case studies are based on the development of two reminder applications via RAD and agile methods. In the second section, we describe the existing literature for the two main approaches RAD and agile methods in brief.

2.1 Reminder applications

Nowadays, mobile phones have become an essential element of our pockets and hands. We use it for many routines such as communication, entertainment, health applications etc. Mobile application development is a fast-growing domain. It is essential for the different manufacturers to have their "application store" with multiple options for its

customers. Amongst all mobile platforms, the android platform was chosen for both applications, because Google has provided it with powerful features and made it open source with high scalability.

The purpose of some social networking websites like Facebook is to remind us about our friends, what they do or when is their birthday, but are people really content with that yet? Not completely. People still need some easier way to remember their important works, we carry our mobile phones everywhere we go, so why don't we use it to remind us about memories or what we should do next? Getting involvement in work makes our schedule busy, so we are prone to forget important upcoming events, fortunately social websites has come to ease our life in many aspects, such as twitter that provides a list of friends who have birthday on a particular day. Therefore, these websites have solved the problem partially, but they have some constraints, as they need active users, user must log-in every time to greet and chat with his friends and to see their upcoming events. However, what will happen if the Internet is not working that day? Or the user is so busy to check out his friends list? Or he is totally unaware of the latest social websites? [Khan, Khan and Basheri (2017)]. One such attempt towards such an application, was the reminder application known as the "Greeting Reminder" [Modi, Chauhan, Rana et al. (2013)]. It takes the list of important upcoming events such as birthday, important meetings or anything else the user needs, tasks that must be done on that day. The application will give greeting notifications of friends on the specified date and time, it will also have a list of contacts with their upcoming events, luckily that the application will have a contact reminder system so that the user doesn't have to remember each phone number, so the user can directly select which one of the contacts does need to be sent a greeting message-wishes-through SMS. This research [Modi, Chauhan, Rana et al. (2013)] is applicable and useful in many ways, but it lacks the flexibility of the general use for everything in daily life such as medication, sport or anything else in general.

The most common problem the out-patient face, is the use of their medication at home without guidance or supervision. Here are some of the most common problem: (a) forgetting about some medication dose due to the patient's busy life, (b) difficult scheduling of the medication time due to a large number of medications, (c) different drug react negatively to each other, (d) lack of appropriate use of doses, (e) lack of consultation with healthcare providers (f) lack of guidance that keep track of patient's medicine schedules. In order to solve the previous problems an application "Wedjat smartphone application" was introduced. The application guide and help patient by providing two main functions: the first function is setting an automatic reminder that remind the patients, the medication time and to display how much the patient should take and from which medicine. The second function is can revise the medication time and medication dose in case the patient skipped a dose. The application gives a good solution to the "forgetting about medication time" problem, but the system lacks of appointment cancellation/reservation and appointment reminder features make it less worthy of installing [Zao, Wang, Tsai et al. (2010)].

Home Telecare System (HTS) supports an emerging model of care. In this system, chronic diseases are managed by remote supervision of the patient rather than at home. The HTS lead the healthcare unit to better supervise and manage the patient. It also maintains schedule of appointments, the regular recording of clinical measurements and an

emergency alarm button for the patient via a computer connected to a clinical workstation at his home. As this application make use of the connectivity with the workstation at home for critical patient, any electrical failure (Internet or computer power) can have serious consequences, hence this application is not reliable [Magrabi, Lovell, Huynh et al. (2001)].

Forgetting important occasions/events (not in our daily routines) such as birthdays, wedding anniversaries, invitations etc., is a common human flaw. Relying on technology to be reminded about these events has become a necessity for most of us. e.g., we rely on reminders to alert us about lecture timing, an assignment deadline, airport arrivals of family and friends etc., but what if incase we are overwhelmed with so many appointments at a given day and/or we have our sticky notes lost? To avoid such problem, Facebook application has a reminder alert, that reminds us about our friends' birthdays, special occasions such as wedding day, when a person was added with you as a friend and so on, but even then, these alerts are not enough. Utilizing smart phones with a reminder application will helps us to keep everything in order. Problem with these applications is that sending a greeting message directly to friends and family is not considered.

There exist several birthday reminder applications to maintain social contacts, designed for Android and linked with Facebook, for retrieving birthday's information, and to give notifications about the people in the contact list for birthdays. The user can personally post a congratulation message from his phone or either send a prepared message called a global message to them automatically. The application will be able to extract the friends' list and their birthday dates and by giving the users of the app the opportunity to write an instant birthday message through an automatic birthday messages mechanism. With all the advantages, it lacks the flexibility of the general use for organizing other life activities. In [Ertugrul and Onal (2014)] GPS based mobile reminder application "RemindMe" is developed. This reminder system requires GPS to be enabled all the time. User tags the areas utilizing the applications such as Google Maps or Foursquare or by means of the implanted sensors of the Android gadget. At that point, he makes updates for the tagged areas and when he gets near to this area, the framework informs the client.

2.2 Methods and techniques for managing software development

Software development can be single product development or development of a product line [Khan, Musa, Tsaramirsis et al. (2017)] gives in depth overview of the model and meta model used in the software product line engineering. For the development of software product lines (SPLs), Collateral analysis make sure all the elements of an eco-system are considered, while KAOS goal analysis make sure to have all the goals of the product line defined and Feature modelling is used to model the features of the software product line [Khan, Musa, Tsaramirsis et al. (2018)].

In this research, the focus is on single software development, comparing agile and RAD based software application development. while Agile software development (feature-oriented software development model), that can prove to be market oriented in the development of software product lines. Different modeling techniques are discussed for accurately measuring the design and implementation time of the software product lines. Below we discuss in brief the two approaches/methodologies that were used in this research for software development of the reminder applications i.e., Life Organizer and

SPA. i.e., RAD and Agile software development.

2.2.1 Rapid application development (RAD)

It is a type of Incremental Software Development which is based on a set of short iterative development cycles. RAD is considered as a faster solution, compared to the waterfall model. RAD model is used for projects with clear, understandable, structural structures. RAD requirement determination is done through prototyping and the use of Computer Aided Software Engineering (CASE) tools [Mackay, Carne, Beynon-Davies et al. (2000)]. With RAD the duration of the software development does not exceed 2-3 months. RAD phases are: Requirement planning, user design, construction, and cutover. This model focuses on collecting customer requirements through group of experts, and allowing customers to do the testing process early, by using the concept of prototyping. RAD can rapidly provide customers with a prototype application, usually, visually identical to the future system. This allow customers to provide feedback regarding the delivery and their requirements [Andreswari, Ambarsari, Syahrina et al. (2020)]. It can utilize the Components-based approach i.e., the existing software components and models. RAD staff are usually made up of small teams of developers and industry experts. Automated technique that can produce a prototype from an Ontology Chart using RAD methodology can be a speedy way of delivering the product to the user within the estimated time [Poernomo and Tsaramirsis (2008)].

2.2.2 Agile software development

Agile methodology refers to the approach that is used to develop software, it concentrates as well on the rapid development of software, where the requirements and solutions are determined through cooperation between project teamwork. What discriminates agile methodology is the ability to respond quickly to change that will increase the quality and customer satisfactions. Agile approaches promote highly iterative processes for application development. With agile all parties can provide feedback as the development progresses in an efficient and effective manner. All agile methods have five phases: Define and analyze requirements, Design, Implementation, Testing, Release and Maintenance. There are many methodologies for applying agile, the most widespread and used are: Scrum, Kanban, Extreme Programming (XP), Crystal, Dynamic Systems Development Method (DSDM) and Feature-Driven Development (FDD) [Tsaramirsis and Basahel (2016)] examined teams that include both autonomous intelligent systems and humans, describes Scrum (agile software development) as the best fit methodology to achieve the purpose of building quality applications. A detail comparison of the two-software development methodologies is depicted in the Tab. 1.

Each method has its own advantages and disadvantages. RAD tend to be better for smaller simple projects with very clear requirements and dedicated customers. Agile approaches are better in larger and/or more complex projects.

Table 1: Comparison of the features for RAD and agile models

Features	***Agile Process (XP)***	***RAD Model***
Definition [Sharma, Sarkar and Gupta (2012)]	Project management method, that enables teams to manage themselves.	Software development methodology, that starts by building a prototype and then keep adding functionality to it.
Adaptability [Geambaşu, Jianu, Jianu et al. (2011)]	Able to accept regular change requests, at the beginning of each new release meeting (Iteration planning).	Change is possible, but it may require significant rework.
Coding approach	Test driven development	Prototype generation
Testing Phase [Alshamrani, Bahattab and Fulton (2015)]	Unit testing, Integration testing, System testing	Unit testing, Integration testing, System testing
Length of iteration	2 weeks	2-3 weeks
Knowledge Required [Sharma, Sarkar and Gupta (2012)]	Product and domain	Domain
Status of Development Team [Sabale and Dani (2012)]	Pair programing	Individual development
Communication between customers and developers [Geambaşu, Jianu, Jianu et al. (2011); Raval and Rathod (2013)]	The developers can ask the customer directly.	The project manager will call for a meeting when the prototype is ready. The client will then have the chance to evaluate it and ask for any modifications.
Accurate initial estimation of costs [Sabale and Dani (2012); Khurana and Gupta (2012)]	The development team will evaluate how many tasks they can take on, estimating based on previous projects.	Managers can estimate the cost using experienced or algorithmic based approaches.
Development costs [Alshamrani, Bahattab and Fulton (2015); Raval and Rathod (2013)]	It tends to be cheaper than other approaches, as it cuts down on extensive analysis/documentation cost but more expensive than RAD in smaller projects.	RAD tend to be faster and cheaper in smaller projects. However, in larger projects the prototypes become difficult to extend increasing the cost extensionally.
Response to change [Geambaşu, Jianu, Jianu et al. (2011)]	High	Medium
Speed of change [Geambaşu, Jianu, Jianu et al. (2011)]	Medium	High

2.2.3 Decision support systems

Since deciding on the appropriateness of a project methodology is a concern, decision support systems help businesses in supporting the critical decision-making process. A decision support system for emergency operations, like Knowledge management system (KMS) for China's Emergency Operations Center (EOC) in which big data analytics is applied for decision-making, management, and collaboration within EOCs and with the public will be effective [Ma and Zhang (2018)]. Another emergency decision support system discussed in Wang et al. [Wang, Liu, Zhang et al. (2018)] in which, based on the scenario-response method, the strategical plans are realized. These systems enable the early response to public health emergencies by detecting and responding to them at a very early stage. A comparison of different methodologies, techniques, algorithms and tools is better at the start of any project in a selection of an appropriate fit for the good solution to the project. Studies like genetic algorithmic selection among different algorithms like Ant colony optimization (ACO), African Buffalo Optimization, Bat Algorithm, Particle Swarm Optimization, etc., for feature-oriented software development (FOSD) must be read and carefully understood, before applying agile methodology (FOSD) on any project [Mohmad, Noraziah, Zarina et al. (2018); Raju, Milton and Mahadevan (2018)]. A decision support system like these discussed for the emergency response can be made for methodology selection as well, in which students can take quick help in deciding which methodology to adopt at an early stage. Researchers are using different techniques to automate team selection using fuzzy logic [Goyal and Gupta (2020)] and the methodology selection, one such attempt was to use the bayesian analysis which is a statistical technique for the selection of an appropriate methodology for a project using cost, risk, size, and time duration factors [Mahapatra, Chandra and Goswami (2020)].

3 Implementation of case studies via RAD and Agile models

3.1 Case study 1: managing "life organizer" via RAD

A smart reminder application has become a necessity for everyone nowadays, it's important because we don't have to remember our works, events, appointments or meetings. Writing task/s are not practical as it's prone to loss and we must keep track of each paper. The reminders are a good solution because they provide a 'set & forget' feature, so when the time comes a person is reminded. This feature is important because any living being can be busy, whether he is a social worker, businessman, doctor or student. Anyone among us can get involved in their routine life schedule and forget about important events in their life and when the time comes, the important event, which was to be attended is passed by. Since the smart-phone market is growing every day, it is an opportunity to utilize this and help people to facilitate remembering their important works and events. By using a rapid application method such as RAD, designers, developers and even the client, can visualize the actual product and modify, adopt or propose functionality. Testers can test the early version of the system and identify defects early, allowing more time to the developers to fix them. Each iteration gives all the stakeholders the chance to repeat this process until the project is finally complete. RAD is an established methodology capable of delivering successful software applications. Fig. 1 shows the methodological way of the Life organizer application development.

At first interviews and surveys were conducted, then based on that prototypes were made and tested with students and some faculty members (acted as family member and teaching staff) for confirming the requirements. Based on the suggestions, the applications prototypes were modified and tested again with the users for making sure about the applicability of the application. After the prototyping, implementation and testing were done, where a lot of tests (Unit, Integration and System) were performed to make sure the application is operational and function properly. After that the application was developed and deployed on android play store.

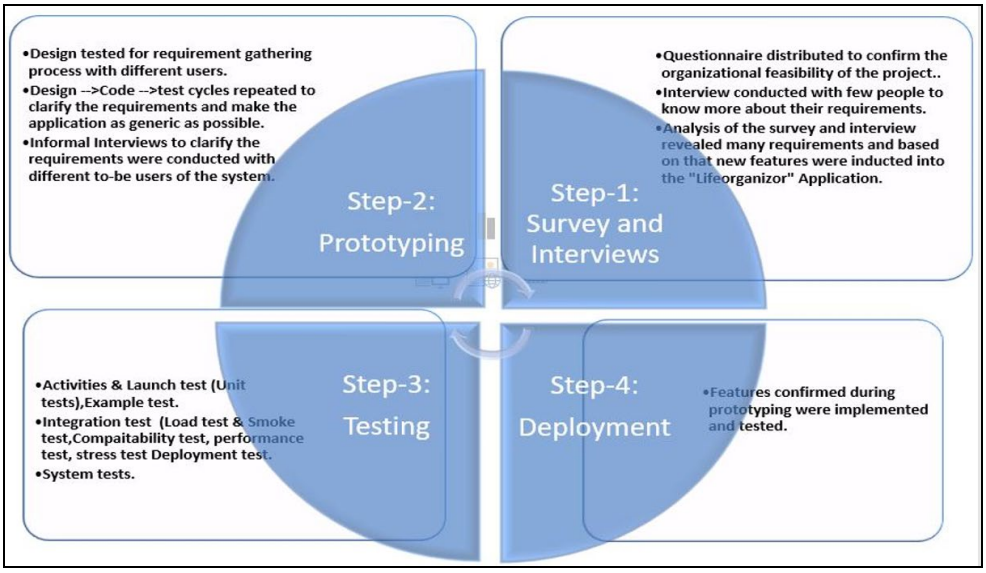

Figure 1: Rapid application development model for the "life organizer" application

Fig. 2 depicts the ecosystem's elements (launching and constructing) of the successor system i.e., the tools, libraries and User interface components that are used during the implementation of the "Life organizer" application [Khan, Musa, Tsaramirsis et al. (2018)]. It was built on making use of the RAD model. The Rapid Application Development platform is used to create web-based and mobile applications. It shows the "Life-organizer" architecture. Open standards, easy customization and rapid prototyping are central to the platform. Life Organizer is a notification-based application which relies on push notifications to remind the user what tasks he/she has for the moment. What makes it special is the ease of use, whether you are a lecturer, employee, student or any member of the society, you are going to find it easy and efficient to use. Life organizer is mainly developed for android based smartphones, with and added functionality to push your notifications to your smart watch via Bluetooth. Your reminders will be stored in the database, so you can update or delete them as per your requirement. Fig. 3 shows an application for Android based smart phones and watches.

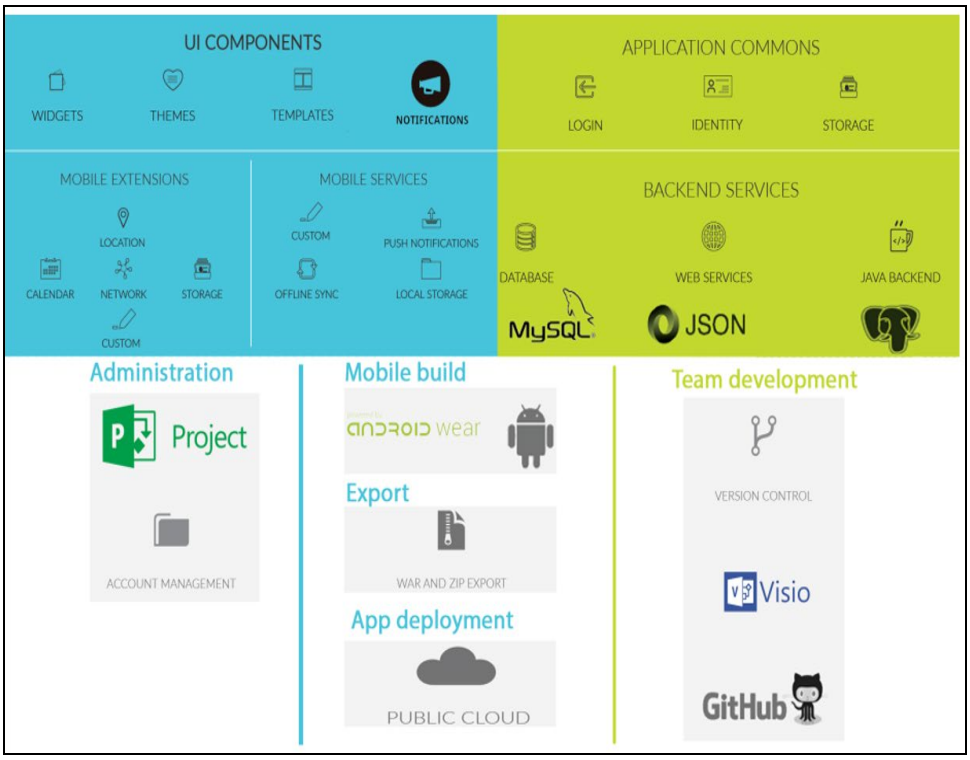

Figure 2: "Life-organizer" RAD platform architecture

An application to notify you about your events and tasks. The user inserts his tasks on the smart-phone, and when the time comes, the notification will be ready to remind him. If the user has a smart-watch, the notifications can be pushed to his smart-watch. Requirement determination for the system prototype was done using small scale throwaway-prototypes. Requirement structuring was done through UML diagrams like use case, class, sequence, communication, state chart and object diagrams. Below we explain the steps taken to manage the application along with the Use case and Class diagram based on the prototypes.

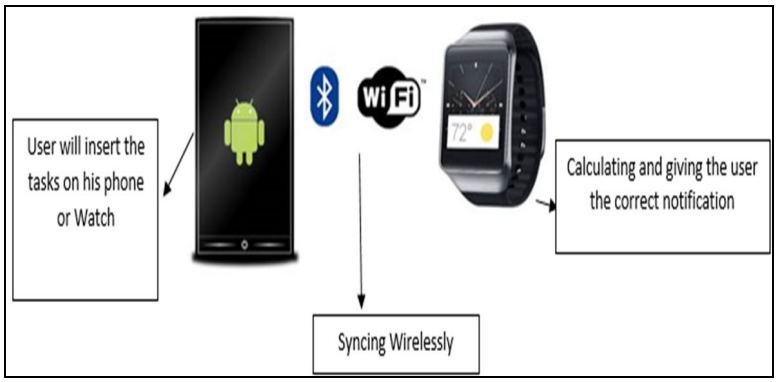

Figure 3: Application in your android smartphone and smart-watch

3.1.1 Interface design prototyping

For requirements analysis and design of the life organizer project, prototypes were prepared and discussed with some students and teachers to get feedback on the requirements. Once the requirements were put into the final shape, the system implementation started. Prototyping follows the rules and guidelines of HCI for effective communication as mentioned in Khan et al. [Khan, Khan and Basheri (2017)].

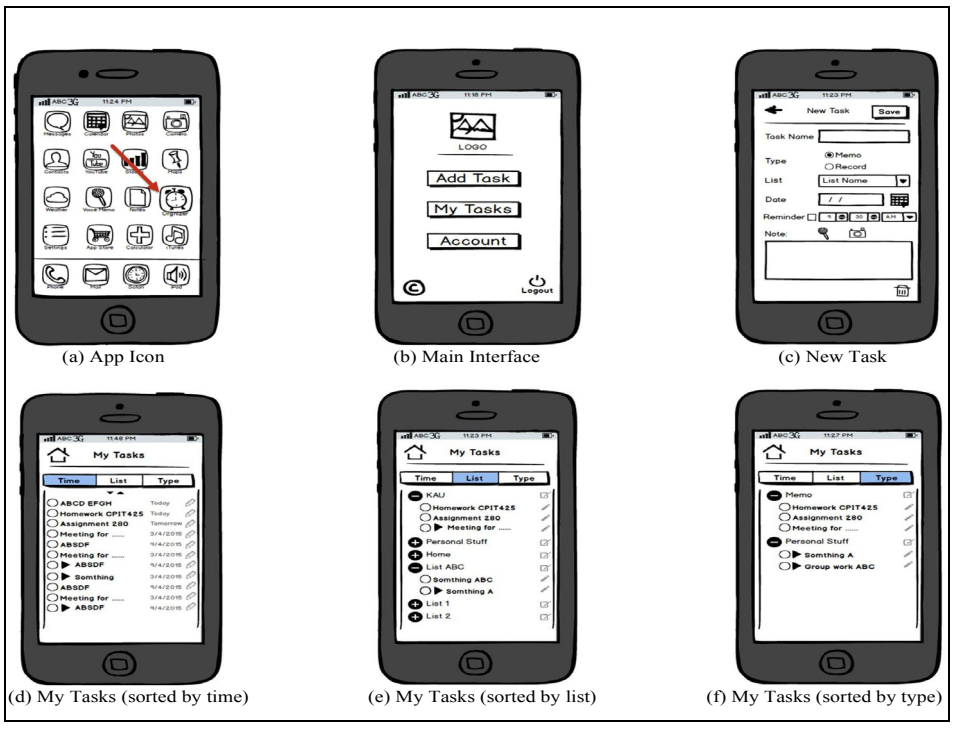

Figure 4: Prototype for requirement engineering process

3.1.2 Use-case diagram

The end user of the Life Organizer (android application) have the following set of functionalities. Fig. 5 shows use-case diagram.

- **Registration (use case 1):** the user must insert his own information and choose what categories he wants.
- **Update Information (use case 2):** the user can update information and categories.
- **Adding (use case 3):** when the user clicks on the "add Button", then a form will be displayed to insert information of the task, and he can choose one of the categories that was checked before.
- **View (use case 4):** the user can see his/her tasks based on the categories he chose.
- **Editing (use case 5):** the user can update his/her task information.
- **Deleting (use case 6):** Tasks or categories can be deleted by the end user and the user can optionally choose to delete any one of his/her tasks or categories. A notification will be displayed to show that the process has been done successfully.

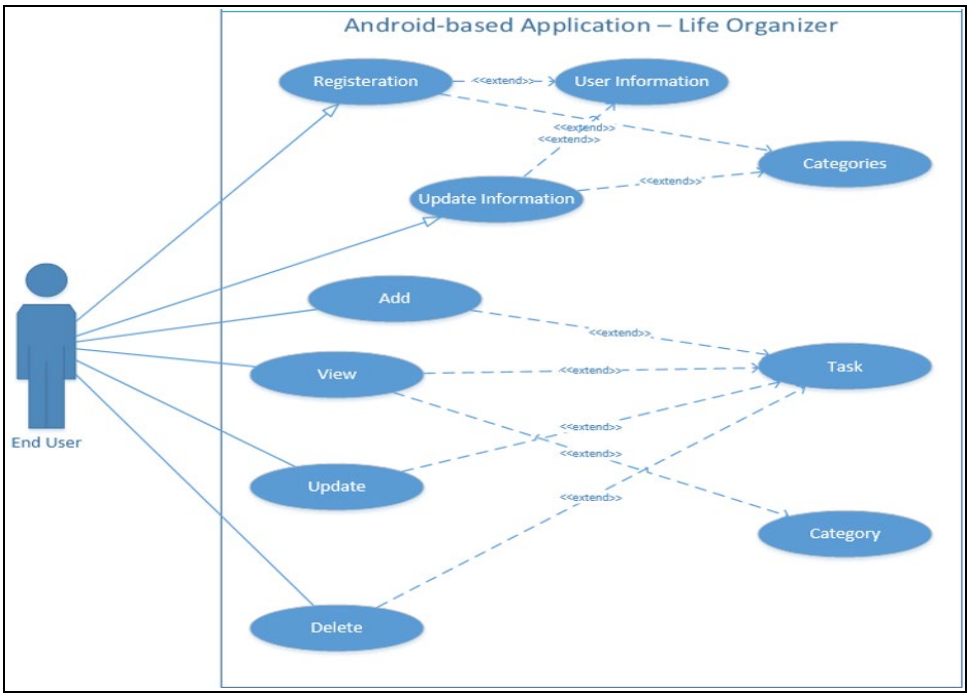

Figure 5: Use case diagram of the system

3.1.3 Class diagram

The classes explanation is given below and a class diagram is shown in Fig. 6.

- Add Class to insert task information, it has a DatabaseOperations object to store information in the database, and some attributes for task information with methods

used to set information from the user and to validate the information that was inserted by the user.

- *ShowMyTasks*: A class that's used to display all tasks that the user added before.
- *UserInformation:* this class is used to set user information (name, height, weight, age) and make him chooses his categories.
- *UpdateInformation:* To update task information.
- *Delete_Task:* To delete any Task.
- *DatabaseOperations:* A database class used to store task information.
- *UserDatabase:* A database class used to store the user information, also the categories he chose.
- *SampleAlarmReceiver:* to use it for setting alarms and send notifications.
- *AppCompatActivity, SQLiteOpenHelper* and *WakefulBroadcastReceiver:* These are built-in classes we must use them.

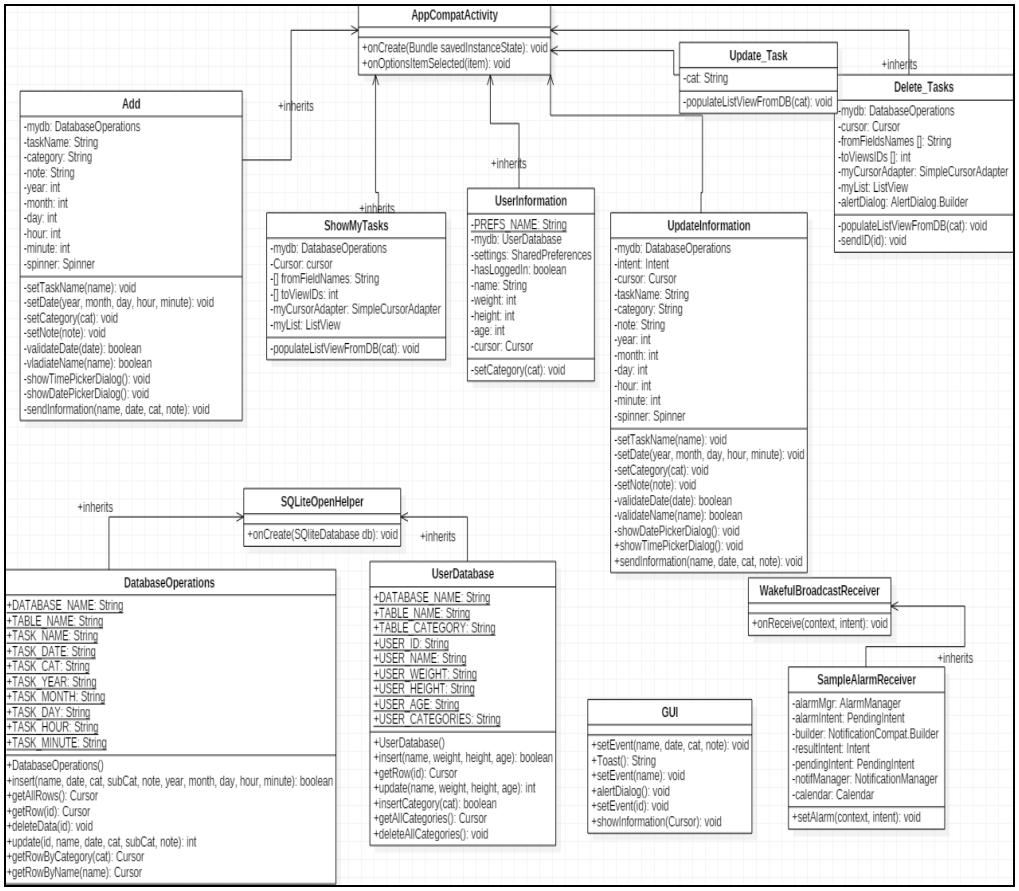

Figure 6: Life organizer class diagram

3.2 Case Study 2: Managing "Smart patient assistant" via Agile

Agile project management is purely based on agile manifesto [Fowler and Highsmith

(2001)]. Teamwork is a key in agile methodology, most of the undergraduate/senior projects are group work, so agile methodology is something that can best fit for managing the projects. User stories from the different stakeholders of the customers were taken into consideration. These user stories were carefully grouped for iteration and release. Some of the user stories can be seen in Tabs. 2-4.

Table 2: Story Card-1 for SPA

Title: Medication time and dose	**Priority:** 1- High	**Estimate:** 1.0 pnt
User Story: As a patient, I want the system to remind me of my medication date, time and dose so that I will not forget my medication **Acceptance Criteria:** - Patients need to add medication scheduled information - Reminder is displayed and alarm beep.		

Table 3: Story Card-2 for SPA

Title: Prescription Submission	**Priority:** 2-High	**Estimate:** 1.0 pnt
User Story: As a patient, I want the system informs me that my turn has come to receive the medication so that I should not have to wait the queue in the hospital' pharmacy **Acceptance Criteria:** - Patients need to submit the paper prescription to the pharmacy - Doctor sends the prescription to the pharmacy - Notification is displayed informing the patient that his turn has come, or his medicine is ready.		

Table 4: Story Card-3 for SPA

Title: Prescription information	**Priority:** 3-High	**Estimate:** 1.0 pnt
User Story: As a patient, I want the doctor to write a readable prescription so that I will get the exact medication from the pharmacy **Acceptance Criteria:** - Doctors need to write the prescription through the system - prescription is displayed and sent to pharmacy.		

It is evident from the story cards that Smart patient assistant (SPA) application will be addressing such as; patient forgetting about medication time and the dose. To disremember medication time, can cause compilations for chronicle-disease patient. The second problem SPA will address is that a patient waiting in a long queue in the hospital pharmacy with a paper prescription to get medication. A third problem SPA will handle,

is the wrong medication given to the patient because of the doctor's sloppy handwriting in the paper prescription. Fig. 7 shows the architectural diagram of SPA.

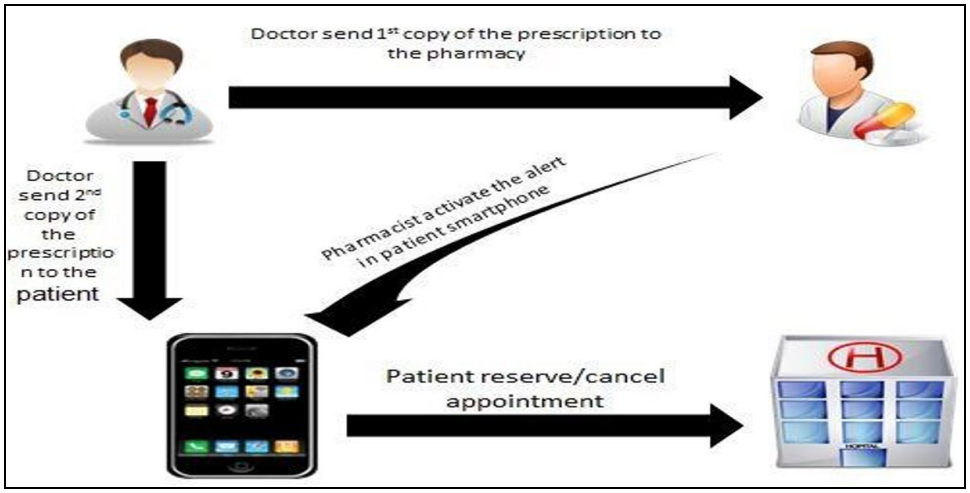

Figure 7: Architectural diagram of SPA

The system will address three main issues, the first one, SPA will solve by implementing a function that allows the patient to make or cancel an appointment using his smartphone (android OS) .The second problem SPA will be addressed by E-prescription and will send a copy to the patient smartphone and another copy to the pharmacist. The third problem will be solved by having an automated reminder in the patient smartphone to remind about the medication and the dosage.

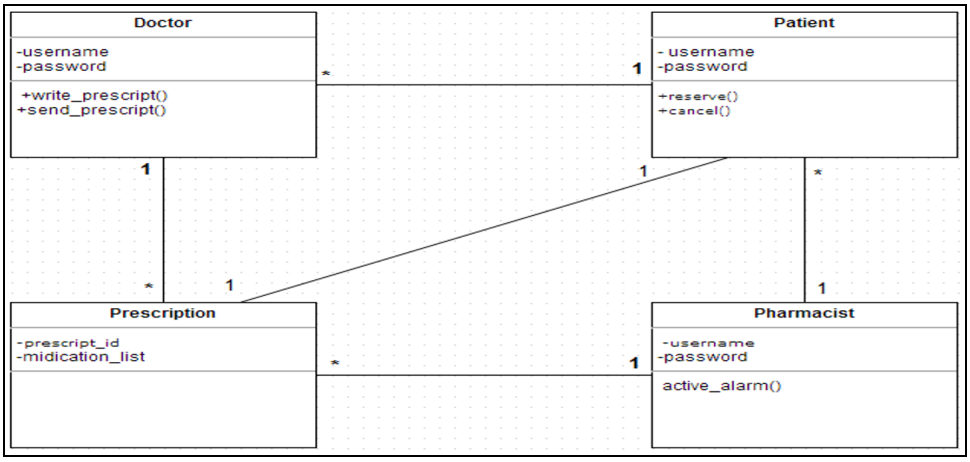

Figure 8: Class diagram for SPA

Fig. 8 depicts the static structure of the SPA system in a class diagram. The diagram is very simple, the reason being that, the focus is not on documentation but on the actual working system with high quality code. The system functionality is tested, and user

feedback is taken into consideration with the agility to accommodate any further changes according to market trends.

4 Results and discussion

4.1 Data about the subjects, group formation and supervision

Two groups of students with good programming skills and having similar grades in software engineering courses were taken for the study. Group averages in software engineering-1 (CPIT-251) courses were 91.5% (Life Organizer Group) and 92% (SPA Group) respectively, which was taught by Dr. Fazal Qudus Khan. The Group Smart patient assistant in 2013 while "Life Organizer" in 2016 was supervised by the same supervisor i.e., Dr. Fazal Qudus Khan in their senior projects.

4.2 Samples selection and grading

For the purpose of data evaluation, sample grades of "Code Demo" and "Final Report" were taken into consideration. As Final report has a comprehensive document of the whole project elements, while Code-Demo is the test of the functionality and usability of the application. Therefore, these two assessments were chosen for the evaluation of the effectiveness of the two techniques. Rubric of Code-Demo and Final report with details are given in Tabs. 5 and 6. The grading is done by a panel of three judges (Faculty member with the rank of Lecturer, Assistant Prof and Associate prof) in case of the Smart patient assistant, while the same panel in 2016 graded the Life organizer project. The panel similarity is not a coincidence though, as the senior project committee for undergraduate usually is changed every five to six years.

4.3 Documentation and the analysis of the effectiveness of the two approaches

Tab. 5 shows code demo Rubric averages from the three evaluators of the senior project committee while Tab. 6 shows rubric for Final Report grading. The rubric for code demo comprised of exception handling, testing of Code, problems faced and solution, design to code Mapping, backend connectivity and reporting the limitations of the work. The rubric for final report consisted of abstract. Chapter 1-Introduction, Chapter 2-Literature Review. Chapter 3-Analysis and Design, Chapter 4-Implementation and Testing and Chapter 5-Conclusion and Future Work. Code demo grading was out of 10 and final report was out of 30. The reported analysis is based on the average grades of the two groups for both assessment tools. Mean of the Code demo and Final report grades of the two groups, were calculated based on the standard formulae.

Mean: $\bar{x} = (\Sigma xi) / n$ (1)

where $\bar{x}$ =sample mean, Σ = Summation, xi = all the x-values and n = the number of items in the sample.

As can be seen in Tab. 5, the code demo results show similar score for both students of the SPA (Agile) team members.

Table 5: Average of the three evaluators grading (on the rubric for Code Demo)

Average of the "Code Demo" Grading of three evaluators (Code Demo Grading Criteria/Rubric)

			Semester Fall-2016					
Project Title	**Student IDs**	**Design-Code Mapping (2)**	**Backend connectivity (2)**	**Exception Handling (1)**	**Testing of Code (3)**	**Problems faced and solution (1)**	**Limitations of the work (1)**	**Total**
Life Organizer (RAD)	1323xxx	2	2	1	2	1	1	9
	1008xxx	2	1	1	1.5	0	1	6.5
			Semester Fall-2013					
Project Title	**Student IDs**	**Design-Code Mapping (2)**	**Backend connectivity (2)**	**Exception Handling (1)**	**Testing of Code (3)**	**Problems faced and solution (1)**	**Limitations of the work (1)**	**Total**
SPA (Agile)	1009xxx	1	2	1	3	0.5	0.5	8
	1009xxx	1	2	1	3	0.5	0.5	8

As interaction between programmers is quite high and there is a high-quality production, as the focus is on the working prototype rather than extensive documentation, hence the grade average of the SPA (Agile) project is 2.5 percent higher than the Life Organizer (RAD). It is worth noting that although this assessment grading is an individual and not a group activity, however, it is seen not only in this case, but in many cases of the senior project's students, that the students grades are similar (if its lower or higher). The teamwork proves to be a decisive factor in following the agile approach, hence it can be deduced that while following the agile approach, the student must make a careful decision in choosing the team member. Fig. 9 shows the marks difference and similarity of grades between the groups.

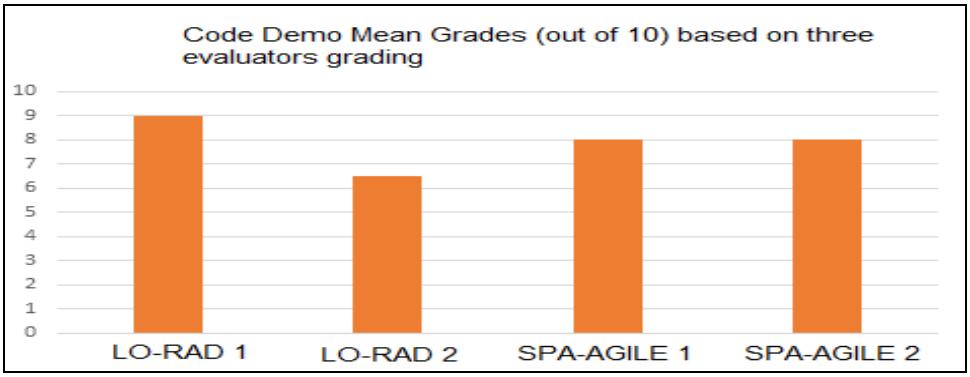

Figure 9: Marks difference and similarity of grades between the groups

As can be seen the average score of Life organizer- RAD based was 77.5%, while for SPA-Agile based was 80%. The difference in means of the two group was 2.5%, which gives an edge for Agile based approach for quality coding.

Table 6: Average of the three evaluators grading (on the rubric for Final Report)

Average of the "Final Report" Grading of three evaluators (Final Report Grading Criteria)								
Semester Fall-2016								
Project Title	**Student IDs**	**Abstract (2)**	**Chapter 1- Introduction (2)**	**Chapter 2- Literature Review (6)**	**Chapter 3- Analysis and Design (10)**	**Chapter 4- Implementation and Testing (8)**	**Chapter 5- Conclusion and Future Work (2)**	**Total**
Life Organizer (RAD)	1323xxx	2	2	6	9.5	8	2	29.5
	1008xxx	2	2	6	9.5	8	2	29.5
Semester Fall-2013								
Project Title	**Student IDs**	**Abstract**	**Chapter 1- Introduction**	**Chapter 2- Literature Review**	**Chapter 3- Analysis and Design**	**Chapter 4- Implementation and Testing**	**Chapter 5- Conclusion and Future Work**	**Total**
SPA (Agile)	1009xxx	2	2	5	7.5	7	2	25.5
	1009xxx	2	2	5	7.5	7	2	25.5

As can be noted in Tab. 6, the score of SPA (Agile) is on the lower side i.e., mean of 25.5 against the mean of 29.5 score of the Life-Organizer project. A mean difference of 13.33% for the scores, was seen between the two approaches. Fig. 10 shows the marks difference between the groups for the final report.

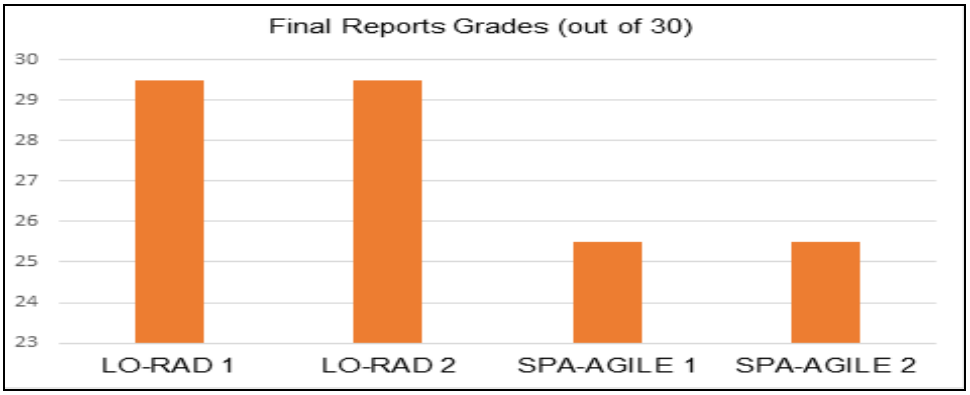

Figure 10: Marks difference between the groups for the final report

Mean of the Life organizer-RAD based was 98.33% while it was 85% for SPA-Agile based project. A quite significant difference in the mean of the two groups, suggests that if the institution has higher grades for the reports, then RAD is the better.

5 Conclusion and future work

From the analysis, it can be concluded that it is better to follow RAD methodology if the total grades are higher for the final report of the graduation project. On the other hand, if

the quality of the application software is concerned, then it is better to follow the agile based approach for senior projects. This study found that the project developed following RAD methodology performed 13.33% better in providing extensive and elaborated documentation than the students following the Agile technique. While it was noted that the project, SPA-Agile based, due to teamwork had 2.5% better implementation than Life Organizer-RAD based project.

The sample size was small for the case studies and only two assessments were chosen for comparison. For the evaluation of the two case studies, the "mean" of the two groups was compared and used. However, as a future work, sample size can be increased to many projects and more assessments must be taken into consideration for the purpose of better evaluation. Moreover, *T*-test. *F*-test and Annova tests can be performed to have quality statistical analysis to evaluate the hypothesis.

Acknowledgement: Thank goes to Mr. Thamer Dajjam, Mr. Akram Hussain, Mr. Obay Bukhari and Mr. Talat Jaada, who worked under Dr. Fazal Qudus khan's supervision for developing the "Life organizer" and "SPA" applications.

Funding Statement: The authors received no specific funding for this study.

Conflicts of Interest: The authors declare that they have no conflicts of interest to report regarding the present study.

References

Alshamrani, A.; Bahattab, A. A.; Fulton, I. A. (2015): A comparison between three SDLC models waterfall model, spiral model, and incremental/iterative model. *International Journal of Computer Science Issues*, vol. 12, no. 1, pp. 106-111.

Andreswari, R.; Ambarsari, N.; Syahrina, A.; Puspitasari, W.; Novianti, A. et al. (2020): Design of e-Marketplace for village-owned small, micro and medium enterprise using rapid application development. *International Journal of Innovation in Enterprise System*, vol. 4, no. 1, pp. 35-43.

Chavez, V.; Wollert, J. (2020): Arduino based framework for rapid application development of a generic IO-link interface. *Kommunikation und Bildverarbeitung in der Automation. Technologien für die intelligente Automation (Technologies for Intelligent Automation)*, vol. 12, pp. 21-33.

Chow, T.; Cao, D. B. (2008): A survey study of critical success factors in agile software projects. *Journal of Systems and Software*, vol. 81, no. 6, pp. 961-971.

Ertugrul, A. M.; Onal, I. (2014): RemindMe: an enhanced mobile location-based reminder application. *International Conference on Future Internet of Things and Cloud, Barcelona*, pp. 425-430.

Fowler, M.; Highsmith, J. (2001): The agile manifesto. *Software Development*, vol. 9, no. 8, pp. 28-35.

Geambaşu, C. V.; Jianu, I.; Jianu, I.; Gavrilă, A. (2011): Influence factors for the

choice of a software development methodology. *Accounting and Management Information Systems*, vol. 10, no. 4, pp. 479-494.

Goyal, M.; Gupta, C. (2020). Intuitionistic fuzzy decision making towards efficient team selection in global software development. *Journal of Information Technology Research*, vol. 13, no. 2, pp. 75-93.

Jorgensen, M. (2019): Relationships between project size, agile practices, and successful software development: results and analysis. *IEEE Software*, vol. 36, no. 2, pp. 39-43.

Khan, F. Q.; Khan, A. I.; Basheri, M. (2017): An HCI principles based framework to support deaf community. *International Journal of Engineering and Applied Computer Science*, vol. 2, no. 2, pp. 47-59.

Khan, F. Q.; Musa, S.; Tsaramirsis, G.; Bakhsh, S. T. (2017): A study: selection of model metamodel and SPL tools for the verification of software product lines. *International Journal of Information Technology*, vol. 9, no. 4, pp. 353-362.

Khan, F. Q.; Musa, S.; Tsaramirsis, G.; Khan, S. (2018): A novel requirements analysis approach in spl based on collateral, KAOS and feature model. *International Journal of Engineering & Technology*, vol. 7, no. 4, pp. 104-108.

Khurana, G.; Gupta, S. (2012): Study & comparison of software development life cycle models. *International Journal of Research in Engineering & Applied Sciences*, vol. 2, no. 2, pp. 1513-1521.

Ma, Y. F.; Zhang, H. (2018): Enhancing knowledge management and decision-making capability of China's emergency operations center using big data. *Intelligent Automation and Soft Computing*, vol. 24, no. 1, pp. 107-114.

Mackay, H.; Carne, C.; Beynon-Davies, P.; Tudhope, D. (2000): Reconfiguring the user: using rapid application development. *Social Studies of Science*, vol. 30, no. 5. pp. 737-757.

Magrabi, F.; Lovell, N. H.; Huynh, K.; Celler, B. G. (2001): Home telecare: system architecture to support chronic disease management. *23rd Annual International Conference of the IEEE Engineering in Medicine and Biology Society*, vol. 4, pp. 3559-3562.

Mahapatra, H. B.; Chandra, V.; Goswami, B. (2020). Selection of software development methodologies (SDMs) using Bayesian analysis. *New Paradigm in Decision Science and Management*, pp. 11-18.

Modi, K.; Chauhan, U.; Rana, J.; Patel, C.; Rana, A. et al. (2013): Greeting reminder application based Android. *International Journal of Innovative Research in Computer and Communication Engineering*, vol. 1, no. 2, pp. 339-344.

Odili, J. B.; Kahar, M. N. M.; Noraziah, A.; Zarina, M.; Haq, R. U. (2018): Performance analyses of nature-inspired algorithms on the traveling salesman's problems for strategic management. *Intelligent Automation and Soft Computing*, vol. 24, no. 4, pp. 759-769.

Raju, L.; Milton R. S.; Mahadevan, S. (2018): Application of multi agent systems in automation of distributed energy management in micro-grid using MACSimJX. *Intelligent Automation and Soft Computing*, vol. 24, no. 3, pp. 483-491.

Raval, R.; Rathod, M. H. (2013): Comparative study of various process model in

software development. *International Journal of Computer Applications*, vol. 82, no. 18, pp. 16-19.

Rosyad, R.; Syukur, A.; Busro, B.; Rahim, R. (2020): Multimedia prayer application for education with rapid application development method. *7th International Conference on Cyber and IT Service Management*, pp. 1-4.

Sabale, R. G.; Dani, A. R. (2012): Comparative study of prototype model for software engineering with system development life cycle. *IOSR Journal of Engineering*, vol. 2, no. 7, pp. 21-24.

Sharma, S.; Sarkar, D.; Gupta, D. (2012): Agile processes and methodologies: a conceptual study. *International Journal on Computer Science and Engineering*, vol. 4, no. 5, pp. 892-898.

Tsaramirsis, G.; Basahel, A. (2016): Scrum to manage humans and intelligent systems. *3rd International Conference on Computing for Sustainable Global Development*, pp. 1353-1356.

Tsaramirsis, G.; Poernomo, I. (2008): Prototype generation from ontology charts. *Fifth International Conference on Information Technology: New Generations*, pp. 1177-1178.

Wang, X.; Liu, Y.; Zhang, H.; Ma, Q.; Cao Z. (2018): Public health emergency management and multi-Source data technology in China. *Intelligent Automation and Soft Computing*, vol. 24, no. 1, pp. 89-96.

Zao, J. K.; Wang, M. Y.; Tsai, P.; Liu, J. W. S. (2010): Smart phone based medicine in-take scheduler, reminder and monitor. *12th IEEE International Conference on E-Health Networking, Application and Services, Healthcom*, pp. 162-168.