Paper ID: 479

An Analysis of Applying Software Development Methodologies in Military Software Development

LM Wijethungaarachchi# and SPIU Dayarathna

Centre for Research and Development, Ministry of Defence, Sri Lanka

# lahiru@crd.lk

Abstract: The precise selection of the most suitable software development methodology is crucial to any small or enterprise level software application. Specifically, considering the development of military software applications which ranges from training, management, planning and operational scenarios, the proper usage of software development methodologies could significantly affect the decisions made regarding national security. Researches carried out in this particular domain is very limited and, in an era, where military software applications are growing and making a heavy impact on the military strategies, it is vital to understand the importance of selecting and adhering to the best methodology and the need to follow the software engineering guidelines. The main objective of the research is to study the features of existing software development methodologies and evaluate the application of such in various military scenarios. The research will be carried out primarily using qualitative methods supported by quantitative methods where necessary. The result of the research will provide a clear understanding to military software developers in planning, developing and implementing future software projects.

Keywords: Software Methodologies, Software Engineering, Defense Applications, Mission Critical Systems

Introduction

Software engineering is an ever-evolving and rapidly growing industry in modern world. From workflow management to Complex process automation and expert systems, the application of Software Engineering aspects plays a vital role. Software design and development has been revolutionizing the way businesses are carried out than any other industry. At the base of this Engineering paradigm lies the software development methodologies. Which are a set of frameworks that are defined for planning, executing, and managing the process of system development. Even the industry has been around for only few decades, there are many development methodologies that are in practice proving that the application of these methodologies impacts a vast range of subjects and multiple domains.

The military has always been a frontrunner in technology, and has always been equipped with modern equipment's ranging from hardware-based weaponry to defensive establishments. But the modern threat environments demand militaries to shift their focus from hardware to software (Hagen, 2013). The software development practices and methodologies are not explicitly developed keeping military requirements at its core. Hence, there are multiple issues that persists when development of a military software is undertaken by a team of software developers. A number of factors are in play such as; rapidly changing threat environments, cyber warfare, management of personnel and authorization. A considerable conflict with the standard software development methodologies can be observed in terms of time factor and human resources due to the confidential nature and mission criticalness. The research aims to evaluate the aptness of utilizing available development methodologies and to provide a summary recommendation on the aspects to be considered when choosing a development methodology for future military software projects.

Literature Review

The literature review conducts a detailed review of some of the well-known software development methodologies that are being practiced such as; Waterfall, Spiral, RAD (Rapid Application Development), Agile and DevOps. The review aims to present their origins and critically discuss the advantages and disadvantages of each methodology. The objective of the review is to give a better understanding of the usability of methodologies when it is applied to a military scenario. A. Waterfall Methodology Considered by many as the forerunner of Software Development Lifecycle (SDLC) methodologies. Sometimes referred to as the linear sequential model, this methodology states that fundamental processes such as specification, development, and evolution should be considered as different phases when developing a software. Those phases are considered as Requirements specification, Software design, Implementation, Testing, etc. (Sommerville, 2016) The model was first introduced by Royce W.W (1970) as a method to manage the development of large-scale software projects. The waterfall model is a classic example of a plan driven model where the activities must be properly planned and scheduled before execution. The stages of the waterfall model reflects the following fundamental development activities. (Sommerville, 2016).

i. Requirements analysis and definition.

ii. System and Software design.

iii. Implementation and unit testing.

iv. Integration and system testing.

v. Operation and maintenance.

Theoretically, a phase needs to be completed completely in order for the next phase to be started or needs to be approved (Sommerville, 2016). But the practical scenario demands a certain diversion from this theory. The phases need to be overlapped. And provide feedback to one another. (Figure 1).

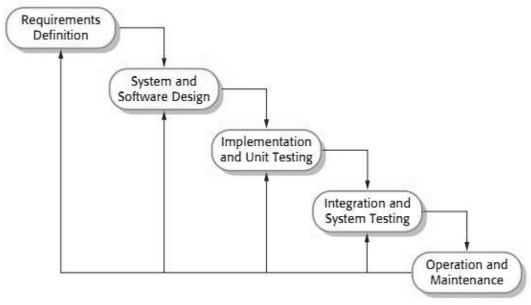
Requirements Definition
System and Software Design
Implementation and Unit Testing
Integration and System Testing
Operation and Maintenance

Figure 1: Waterfall Model (Source: Sommerville, 2016)

The Waterfall model has been introduced nearly 4 decades ago but still widely used and popular among software engineering community. Mainly due to the following notable features.

i. Simplicity.

ii. Ease of management due to each phase having clearly defined deliverables and reviews.

iii. Phases are completed one at a time. Hence managing resources is easy.

iv. Highly effective in projects that has clearly understood requirements.

v. Demands structured organization. vi. Early design changes are permitted.

vii. Ideal for milestone-based development projects.

However, due to the cost of iterations that could occur due to multiple feedback and review processes, in practice, it is recommended to lock or freeze certain phases upon reaching a milestone (Sommerville, 2016) and continues with the rest of the development process. This may result in badly structured or the resulting system fails validation from the user. When the software is put into use in the final phase. The failures and omissions occurred in the early phases are discovered. Errors emerge and user's demand new or altered functionalities. Hence for the system to remain in use, it must return to the initial phases to iterate the process. In addition to that, following disadvantages also persists with the waterfall model.

i. Lack of a working system until the final phase.

ii. High risk and uncertainty.

iii. Difficulty in applying to complex and dynamic scenarios.

iv. Difficult to measure progress before the completion of each phase.

v. Cannot adapt to rapidly changing requirements.

vi. Minimal/None user feedback during the development processes.

Even though many criticisms exist about the usage of traditional waterfall method in modern software projects, it is worthy to note that Waterfall methodology has paved way for many successive SDLC processes models.

B. Spiral Methodology

Another member of the SDLC family, Spiral methodology is a risk-driven process framework introduces by Boehm (1986). The process is diagrammatically represented as a spiral with multiple loops. Each loop represents a phase of the software development process such as, feasibility study is represented by initial loop, Requirement analysis and design is represented by subsequent loops etc. The number of loops can be varied from one project to another. The spiral model reflects changes as a result of project risk and is therefore extremely supports risk handling. In diagrammatic representation, each loop is divided into four sectors. (Sommerville, 2016) (Figure 2)

i. Objective setting.

ii. Risk assessment and reduction.

iii. Development and validation.

iv. Planning.

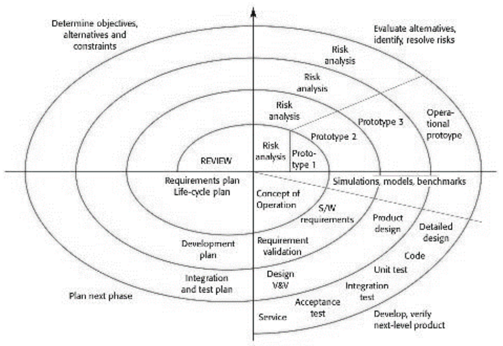

Figure 2: Boehm's spiral model (Source: Sommerville, 2016)

As stated above, due to its nature of explicit recognition of risk, the model allows adding instances of the software product when it is available or a considerably agreed prototype is developed. That guarantees that there is minimal conflict with previous designs or builds. In contrast to the Waterfall method, Spiral model also accommodates early user involvement in the development process. The advantages of spiral model are not limited to above as it also includes;

i. Proper risk evaluation.

ii. Flexible requirements are permitted.

iii. Ability to add new features and changes systematically.

iv. Space for customer feedback.

v. A working software is produced early in the process.

vi. Easy cost estimation. vii. Faster development.

On the contrary, Spiral model demands strict management capabilities and there is a risk of spirals running into infinite loop.

Furthermore, following disadvantages persists with this methodology.

i. Complex management.

ii. Cannot forecast the exact end of project.

iii. Not suitable for small projects.

iv. Excessive documentation.

v. Expensive.

vi. Difficulty in time management.

C. RAD (Rapid Application Development) Rapid Application Development is a methodology built providing a heavy emphasis on rapid development of prototypes for testing functions and features. The term was popularized by James Martin (1991) in a book of the same name. The RAD model came into existence as a solution to the problems observed in traditional Waterfall method. One of the main drawbacks that exist in the Waterfall method is he inability to accommodate changes in core functionalities once the software development is underway. By emphasizing on prototyping iterations RAD allows the accurate measurement of progress in real time. The RAD model could be broken down into multiple phases. But according to James Martin (1991) it can be divided into 4 distinct phases.(Figure 3).

i. Requirements planning phase.

ii. User design phase.

iii. Construction phase.

iv. Cutover phase.

It seems that RAD model can be utilized for all projects effectively, Nevertheless, while it can be applied to quick projects handled by small teams it's not effective in many other scenarios. A few advantages of RAD model are;

i. Ability to change requirements at any time

ii. Prioritize on customer feedback.

iii. Quick reviews.

iv. Reduced development time.

v. Early system integration.

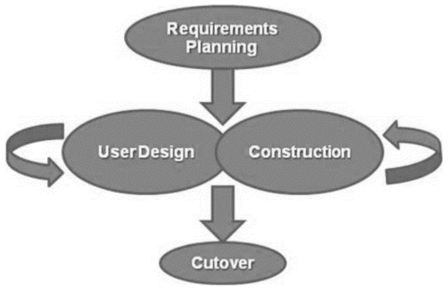

Figure 3 - RAD model (Source: Wikipedia)

While there are many benefits that implies RAD is a perfect model. Following disadvantages also persists with the model.

i. Requires the development and designer teams to be highly skilled.

ii. Constant user involvement.

iii. Can only be applied effectively in modular systems.

iv. More complex to manage.

v. Suitable only for projects requiring shorter development time.

The RAD presents strong benefits to a team that is familiar with the agile philosophy and highly skilled in the development realm and also has a relatively small project to roll out with clients willing to take part throughout the process. Nevertheless, it is not recommended to be used in projects that does not fulfill above criteria.

D. Agile Methodology

Many of the software development methodologies in 80s and 90s were highly plan-driven methods. It was believed that better software could only be achieved by careful and formalized planning, analysis, design and testing (Figure 4). However, these methods produced an unnecessary overhead when applied to small and medium sized businesses, Hence, a number of software developers proposed new “agile methods” for development (Sommerville, 2016) which allowed developers to focus on software development rather than its design. Agile methodology is itself a host or a family to another set of processes based on agile principles. Which are;

i. Customer involvement.

ii. Incremental delivery.

iii. People not processes.

iv. Embrace change.

v. Maintain simplicity.

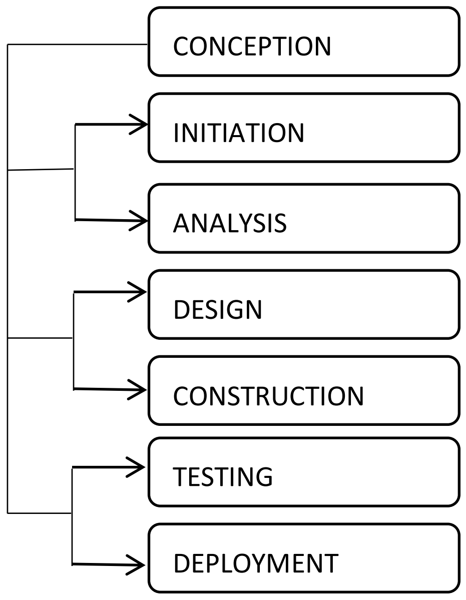

Figure 4 - Agile methodology (Source: Author Developed)

Agility in its namesake defines the flexibility and quick adaptation to changing environments. Various agile methods are;

i. Scrum.

ii. Crystal Methodologies.

iii.DSDM (Dynamic Software Development Method).

iv. Feature Driven Development (FDD).

v. Lean Software Development.

vi. Extreme Programming (XP).

While the objective of this research is not to discuss deeply on this subset of methodologies, there are number of advantages in following the agile methodologies. Which are;

i. Emphasis on modern techniques.

ii. High adaptability.

iii. Continuous customer feedback.

iv. Iterative development.

Even though agile methodologies are considered the perfect and suitable approach to modern software development, there are considerable disadvantages associated with it.

i. Difficulty to add changes within iteration.

ii. Minimal emphasis on documentation.

iii. Relies on real-time communication with users.

While agile is seemingly similar to the RAD methodology. It is widely recommended that for certain large-scale software projects application of agile could be problematic. Hence, it is recommended to adapt to a hybrid approach when using agile methodologies.

E. DevOps DevOps is a rather new and evolving software development methodology that focuses on communication, integration and collaboration among the IT practitioners to enable rapid deployment of software. DevOps promotes collaboration between development and operations teams (Figure 5). It is not just a development methodology but also an organizational culture. The combination of the development and operations teams promotes continuous integration, continuous deployment, automated testing and transparency in code repositories. (Ismail, 2018).

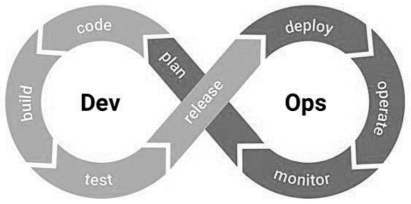

Figure 5 - DevOps methodology

The distinctive advantages of DevOps are;

i. Improved time of product release.

ii. Low failure rate.

iii. Minimize disruption.

iv. Improved customer satisfaction by continuous deployment.

v. Employee productivity and efficiency.

Even with all above benefits, major drawbacks of the methodology are;

i. Certain customers are not expecting continuous updates.

ii. Some companies have strict policies to ensure that a product goes through extensive testing before it is used in operation.

iii. If development and operational departments use different environments, unseen errors could occur.

Research Methodology

The research is conducted through a hybrid approach mainly based on qualitative research methodologies with which are slightly combined with quantitative methodologies. The research domain is fairly new, hence the authors had to consult developers and software engineers who are both serving in the military and civil industry. This was done primarily through focus groups and a questionnaire was used. Informal interviews were conducted in order to further clarify their insights. The discussions were focused on the constraints experienced by software developers when developing military systems in contrast to developing civilian systems. The questionnaire was distributed among serving software engineers/developers within armed services in the format of a google form and shared using email and social networks. Microsoft excel and Google charts were utilized to visualize the acquired data.

The questionnaire took a straight forward approach to find the proficiency of each developer about the different software development methodologies and to obtain their perspective on the constraints that are in place when developing military software. The authors defined the mostly used software development methodologies and the general constraints after focus groups sessions. Then individual anonymous responses were obtained to further clarify the findings. These findings were used as the basis for demarcating the constraints and software types which will be argued against the different methodologies in the discussion. Data from 30 respondents were used in the analysis.

Results and Discussion

This section summarized the research findings and results.

The developers were questioned on their proficiency in the software development methodologies that were outlined in the literature review. Which are;

i. Waterfall method

ii. Spiral Method

iii. RAD

iv. Agile methods

v. DevOps

The findings are represented as follows; (Figure 6)

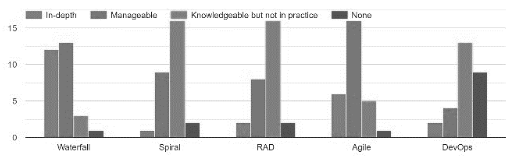

Figure 6– Proficiencyon development methodologies (Source: Author)Developed

It is observed that the most of the developers were highly proficient in early SDLC methods such as Waterfall but lacks a certain amount of knowledge in modern, versatile and flexible methodologies. This could be mainly due to lack of exposure to the evolving technologies.

Throughout the group discussions the developers were asked to group the software applications that they have developed into categories as follows.

i. Workflow automation

ii. Operational support

iii. Decision support

iv. Training

Developers were asked to provide their efforts in adapting the SE methodologies for the above applications from the initial stage (Figure 7).

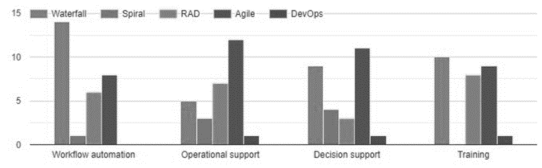

Figure 7Adaptation of SE methodologies (Source: Author Developed)

It is notable that the majority of developers opted to use Waterfall method as the preferred methodology. Mainly due to the high level of proficiency they have gained in applying the same and the ability to reuse certain components as many workflow automation applications proved to follow a similar approach. This was also done as means of time management.

Even though the standard methodologies were applied in the initial stage of system development, the developers responded that it was problematic to adhere to the methodology throughout the development process as represented below (Scale of 1-5, 1 being low adherence and 5 being strict adherence) (Figure 8).

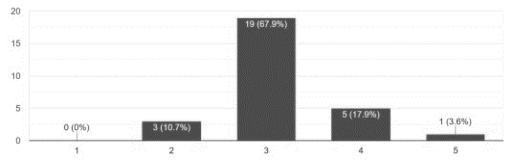

Figure 8: Adherence to methodology (Source: Author Developed)

The researches then asked respondents to provide insight on the constraints that led to the deviation from selected methodology. The defined limitations and difficulties were;

i. Time frame – The time given for the end product to be delivered

ii. Mission criticalness – The accuracy and completeness of the delivered product/components

iii. Unclear requirement identification – The Avenue to adapt to constant requirement changes

iv. Complexity of processes – Constant user interaction to verify and validate processes

v. Human resources – Avenue to assign developer teams

It is observable that the difficulty in specifying clear requirements and the limited timeframe has played a major role in forcing the developers to deviate from the standard methodology. And few state that the nature of systems and the importance of those in strategic operations is a considerable limitation. (Figure 9)

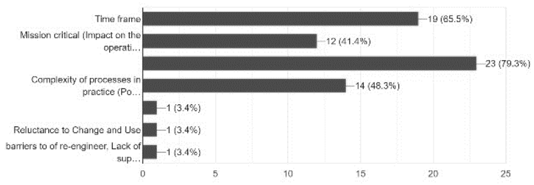

Figure 9- Constraints affecting military software (Source: Author Developed)

With the above result the authors also inquired about the factors that they define as important in developing military software. The factors were also agreed upon the focus group session and then used to obtain individual responses. The defined factors are as follows; (Figure 10)

i. Compatibility with existing systems

ii. Usability and reliability

iii. Application of State-of-the-art technology

iv. Software security

v. Good practices in coding (Standards)

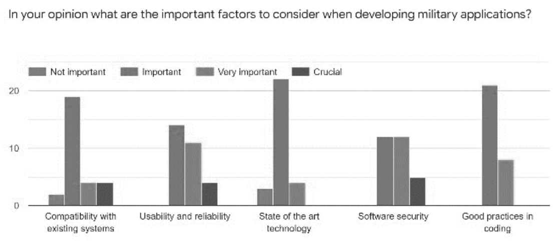

Figure 10 - Factors considered in developing military software Source: Author Developed

Many developers agree that while compatibility with existing systems is important it is equally or more important to consider the new techniques and security of the software is a crucial factor. This is a highly notable point to be discussed further when evaluating the suitability of methodologies as it narrows down to certain features offered by different methodologies.

With the above results the authors moved onto evaluate the deviation of applications from each methodology against the major factors found in the questionnaire responses. In order to effectively demonstrate the evaluation a sample project was selected to represent each of the application categories. (Table 1) (Generic names are being used Figure 10 - Factors considered in developing military software (Source: Author Developed) Figure 9- Constraints affecting military software (Source: Author Developed) Figure 8: Adherence to methodology (Source: Author Developed) to denote certain applications due to the confidentiality).

Table 1: Sample software projects developed by military (Source: Author developed)

Application name	Type of application	Category
Postman	Document management	Workflow automation
Banker	Financial management	Operational support
Warlord	Command and Control	Decision support
Red baron	Simulator	Training

Each methodology is evaluated based on their emphasis on the constraints presented on each of the above systems (figure). The developers were asked to point out the factors which deviate from the standard if the application had followed each methodology. (X) Denotes the factor forces deviation from the methodology where (Y) denotes the factor is applicable within the methodology.

A. Postman application

The objective of this application is to automate the manual process of forwarding official documents to relevant parties and keeping track of each and every document including their status. Following limitations and allowances were identified by the developers. (Table 2).

i. Higher time frame

ii. Low Mission criticalness

iii. Unclear requirements / End user awareness

iv. Low complexity of processes / Low user interaction

v. Low human resources

Table 2: Deviating factors – postman application (Source: Author developed)

Emphasis	Methodology				
	Waterfall	Spiral	RAD	Agile	DevOps
Higher Time frame	Y	X	X	X	X
Low Mission criticalness	Y	Y	Y	Y	Y
Unclear requirements / End user awareness	X	Y	X	Y	Y
Low Complexity of processes/ Low User interaction	Y	Y	X	X	Y
Low Human resources	X	X	X	Y	Y

B. Banker application

The objective of this application is to maintain a complete and systematic record of all transactions within the organization. Following are the limitations and constraints identified by the developers. (Table 3)

i. Less time frame

ii. Low Mission criticalness

iii. Dynamic requirements

iv. Complex processes / High user interaction

v. Low human resources

Table 3: Deviating factors – Banker application (Source: Author developed)

Emphasis	Methodology				
	Waterfall	Spiral	RAD	Agile	DevOps
Less Time frame	X	X	Y	Y	Y
Low Mission criticalness	Y	Y	Y	Y	Y
Dynamic requirements	X	Y	Y	Y	Y
Complex process/ High user interaction	X	Y	Y	Y	Y
Low Human resources	Y	X	X	Y	X

C. Warlord application

The objective of this application is to automate the tasks which are carried out in a battlefield coordination centre. Following are the limitations and constraints faced by the developers. (Table 4)

i. Less time frame

ii. High Mission criticalness

iii. Dynamic requirements

iv. High complexity of processes / High user interaction

v. Low human resources

Table 4: Deviating factors – Warlord application (Source: Author developed)

Emphasis	Methodology				
	Waterfall	Spiral	RAD	Agile	DevOps
Less Time frame	X	Y	Y	Y	Y
High Mission criticalness	Y	Y	X	X	Y
Dynamic requirements	X	Y	Y	Y	Y

High Complexity of processes/ High User interaction	X	Y	Y	Y	X
Low Human resources	Y	X	X	Y	X

D. Red baron application

The objective of this application is to create a virtual reality application that simulates the training environment experienced by a soldier. Following are the limitations and constraints faced by the developers (Table 5).

i. Higher time frame

ii. High Mission criticalness

iii. Clear requirements / End user awareness

iv. High complexity of processes / High user interaction

v. Low human resources

Table 5: Deviating factors – Red baron application (Source: Author developed)

Emphasis	Methodology				
	Waterfall	Spiral	RAD	Agile	DevOps
Higher Time frame	Y	X	X	X	X
High Mission criticalness	Y	Y	X	X	Y
Clear requirements / End user awareness	Y	Y	Y	Y	Y
High Complexity of processes/ High User interaction	X	Y	Y	Y	X
Low Human resources	Y	X	X	Y	X

The above results and finding indicate that while some phases of a methodology has favored the development process, certain phases have failed to provide the expected outcome. The common deviations that can be derived from the above scenarios are;

i. Demand of faster delivery.

ii. Critical components have to be delivered faster while ensuring the accuracy.

iii. End users not having a of proper understanding of the necessity of the software application resulting in vague requirements

iv. Limited Developer resources

Above problems indicate that the root cause of the deviation of methodologies mainly lies in the planning processes. The traditional software planning process should be altered/ modified to effectively allow this transition of information and to suit the demand posed by the military environment.

Conclusion

The above results and evaluations clearly indicate that due to the dynamic and variable nature in the scenarios it is problematic to apply a single software development methodology to the required systems. It is clearly visible that multiple components or phases practiced in different methodologies are required in the development of a single system.

It is best to merge in to existing military planning processes in order to produce a productive and usable plan by every stakeholder. Authors present the following recommendation as a guideline for software project planning for future military software projects. The fundamental of military planning includes the following 7 questions. These questions have tested and field proven in effective military planning. (UK Army doctrine, 2010)

i. What is the situation and how does it affect me?

ii. What have I been told to do and why?

iii. What effects do I need to achieve and what direction must I give to develop my plan?

iv. Where can I best accomplish each action or effect?

v. What resources do I need to accomplish each action or effect?

vi. When and where do the actions take place in relation to each other?

vii. What control measures do I need to impose?

The software planning commonly involves following phases. (Kate Eby, 2018)

i. Scope statement

ii. Work breakdown schedule

iii. Milestones

iv. Gantt Chart

v. Communication Plan vi. Risk Management Plan

The phases of the software planning process effectively answer the aforementioned questions. Hence, in order for the software project to be focused on the military aspects from the foundation level, the following information transfer outline is recommended to be used in military software planning.

i. Situation } System

ii. What is the requirement? } scope

iii. What needs to be achieved? - WBS

iv. When and where to achieve? } Milestones/

v. Resources available } Time plan

vi. Who does what and when? – Com plan

vii. Control measures –Risk management/ Agreements

The above outline enables the complete appreciation of military requirements and resources and transfers them into the software project planning domain. The ability to provide information in a more familiar format allows the stakeholder to be more descriptive and forces the same to do own analysis before agreeing on the software application. The information is then transferred on to the software specific criteria where they can be analyzed to select the best possible methodology. While it cannot be ensured that this planning method will be effective in civilian industry, it is recommended to apply these military fundamentals in civilian software development projects as most modern technologies has gained advantage from military practices.

References

Boehm B.W(1986) A spiral model of software development and enhancement

Hagen C. (2013) Effective Approaches for Delivering Affordable Military Software http://www.defencesynergia.co.uk/. 2010. Army doctrine operation. [ONLINE] Available at: http://www.defencesynergia.co.uk/wpcontent/uploads/2015/05/Army-Doctirne-Operations-Dec2010.pdf. [Accessed 7 September 2020].

Ismail K.(2018) Agile vs DevOps: What's the Difference?. Available at: https://www.cmswire.com/informationmanagement/agile-vs-devops-whats-thedifference/#:~:text=%E2%80%9CDevOps%20is%20a%20 methodology%20that,Product%20Manager%2C%20Raleig h%2C%20NC

Martin J (1992) Rapid Application Development. PrenticeHall,Englewood Cliffs

McGroddy J.C.(1999) Realizing the Potential of C4I

Royce W.W(1970) Managing the development of large software systems smartsheet. 2018. Demystifying the 5 Phases of Project Management. [ONLINE] Available at: https://www.smartsheet.com/blog/demystifying-5- phases-project-management. [Accessed 7 September 2020].

Sommerville I.(2016) Software Engineering.9th ed. Pearson education inc.

Author Biographies

Lahiru Maduranga Wijethungaarachchi is a graduate in Bachelor of Engineering in Software Engineering from London Metropolitan University. He is serving as a commissioned officer in Sri Lanka Army with nearly 10 years of experience in planning, designing and developing software systems for the Defense services.

Ishara Udeni Dayarathna holds a higher national diploma in computer science from Infortec International and currently reading for her Bachelor of Engineering Degree in Software Engineering. She is serving as a commissioned officer in Sri Lanka Army with over 07 years of experience in developing software systems for the Defense services