Agile Software Development Model

For

Embedded Systems Development Environment

أنموذج تطوير البرمجيات السريع في بيئة تطوير النظم المدمجه

Student: Qais Saleh Ibrahim Al-mehmes

Supervisor: Dr.Akram Othman Al-Mashaykhi

A Thesis Submitted in Partial Fulfilment of Requirement for the master Degree of Master Science in Computer Science

Amman Arab University

Faculty of Computer Science and Information

2015

Authorization Statement

جامعة عمان العربية
AMMAN ARAB UNIVERSITY

نموذج (٩)

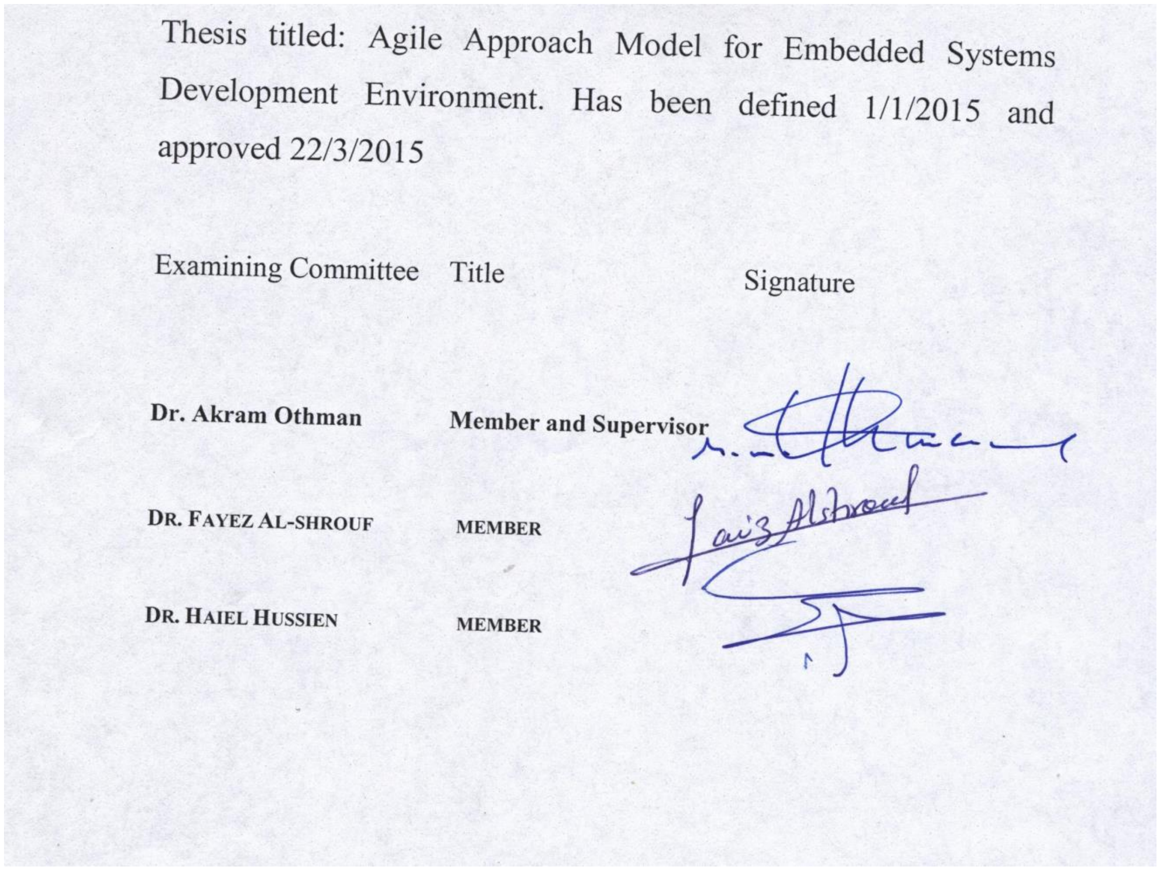
عمادة البحث العلمي والدراسات العليا

تفويض

نحن الموقعون أدناه، نتعهد بمنح جامعة عمان العربية حرية التصرف في نشر محتوى الرسالة الجامعية، بحيث تعود حقوق الملكية الفكرية لرسالة الماجستير الى الجامعة وفق القوانين والأنظمة والتعليمات المتعلقة بالملكية الفكرية وبراءة الاختراع.

المشرف الرئيس (ثلاثة مقاطع)	المشرف المشارك (إن وجد) (ثلاثة مقاطع)	الطالب (ثلاثة مقاطع)
د. أكرم محمد عثمان الشايب	د. فايز الشهري	ميس صبحي إبراهيم
التوقيع: [illegible]	التوقيع: [illegible]	التوقيع: [illegible]
التاريخ: ٣/٥/٢٠١٥	التاريخ: 3/5/2015	التاريخ: ٢/٥/٢٠١٥

Resolution of Examining Committee



Thesis titled: Agile Approach Model for Embedded Systems Development Environment. Has been defined 1/1/2015 and approved 22/3/2015

Examining Committee	Title	Signature
Dr. Akram Othman	**Member and Supervisor**	[illegible]
Dr. Fayez Al-shrouf	**MEMBER**	[illegible]
Dr. Haiel Hussien	**MEMBER**	[illegible]

Acknowledgments

In the name of Allah, the most merciful, the most compassionate all praise be to Allah, the Lord of the worlds; and prayers and peace be upon Mohammed, His servant and messenger.

Firstly, I want to express my deepest gratitude to my supervisor Assoc. prof.Dr Akram Othman for his guidance and confidence through the development of my thesis. Not only about the period preparation of the thesis, but also throughout our studies as he is always encouraged me to do my best.

I want to thank all my committee Assist. Prof .Dr. Fayez Al-shrouf, Assoc. Prof. Dr. Haiel Hussien, Assoc. Prof. Dr. Akram Al-Mshaikhi, for their guidance, advice, criticism, encouragements and insight throughout the studies in computer science

At last, special thanks to my wife kids and frinends supporting,and encouraging me during my work

Dedication

I dedicate this effort to my family. They have given me the drive and ability to tackle any task with enthusiasm and determination; without their encouragement and support this research would not have been made possible. I dedicate this work, also, to the spirit of my father and my mother , My thanks and appreciation to the Ministry of Communications and Iraqi Cultural Commission in Amman and all my friends who always encouraged me.

Contents

List of tables

Table of figures

Abstract

Agile Software Development Model for Embedded Systems Development Environment

By: Qais Saleh Ibrahim Al-mehmes

Supervisor: Dr.Akram Othman

Embedded systems became a trend and approach that are commonly used and widespread, increasingly varied in usages and in its type applications and the demand continually increased for more systems and applications at the level of individual and institutional functions, and privet and public sectors all around the world. This special stat of affair of embedded systems environment require special measurements and techniques in the development methodology and tools been adapted for development processes (initiation and planning, design and implementation) of embedded systems software Match and suitable for the nature and specificity of such systems. The thesis focuses on the problem of selecting appropriate software engineering methodology for developing embedded systems and applications allow implementation within the parameters and constraints of schedule , Budget and quality assurance and usability masseurs, all available methodologies in software engineering has been inspected and investigated for its constraints and limitations , parameters and capabilities to acquire its scope, borders and what it can safeguard for the development of embedded systems, and then draw the bases and criteria for this type of system. Then due to capabilities and advantages of the rapid software application development methodology (Agile development Methodology) into a quickly respond to changing in requirements and speed up the development in successive releases of the system and the ability to adaptive management and self–regulation of developing team and through the concepts and techniques of provided by Agile method, The thesis propose a model of Agile software development for embedded systems environment, due to the fact that the compact multifunction systems and tasks and continuously changing in light of the changed environment and working conditions in the light of changing requirements and usages that require special development Model of the agile methodology provides a transparent and efficient management to control the software project .

Keywords: Agile Method; Embedded software; embedded systems; Hardware constraint.

الملخص

أنموذج تطوير البرمجيات السريع في بيئة تطوير النظم المدمجه

الطالب: قيس صالح ابراهيم المحيمص

المشرف: د.أكرم عثمان

أصبحت الانظمة المدمجة نمطا واسلوبا شائع الاستخدام وواسع الانتشار، وتتنوع وتتزايد الاستخدامات والتطبيقات ويرتفع الطلب على المزيد من تلك النظم واستخداماتها في مجالات وتطبيقات ووظائف فردية ومؤسسية خاصة وعامة حول العالم وقد ظهرت الحاجة الى منهجية وادوات تطوير ملائمة لتصميم وتطوير برمجيات النظم المدمجة تتلاءم و طبيعة وخصوصية تلك النظم.

موضوع الرسالة يركز على مشكلة اختيار منهجية هندسة برمجيات ملائمة لتطوير النظم والتطبيقات المدمجة تسمح بتنفيذها ضمن محددات الوقت والكلفة ومعايير الجودة والاستخدامية، عليه تم دراسة حدود ومحددات وقابليات مجموعة من المنهجيات المتاحة في مجال هندسة البرمجيات للتعرف على حدودها وما يمكن ان توفره لتطوير النظم المدمجة ، وبعد استخلاص الاسس والمعايير الخاصة بهذا النوع من النظم ونظرا لما توفره منهجية التطوير السريع للبرمجيات (Agile development Methodology) من سرعة الاستجابة للتغيير في المتطلبات وسرعة انجاز النظام على شكل اصدارات متعددة والقدرة على الادارة التلقائية والتنظيم الذاتي لفريق العمل ومن خلال المفاهيم والتقنيات التي يوفرها منهج التطوير السريع للبرمجيات فقد توصل الرسالة الى اقتراح انموذج تطوير البرمجيات السريع يلائم بيئة تطوير النظم المدمجة ، نظرا لكون النظم المدمجة متعددة الوظائف والمهام ومتغيرة باستمرار في ضوء تبدل البيئة وظروف العمل وفي ضوء تبدل المتطلبات ومحددات الاستخدامات التي تحتاج الى أنموذج تطوير خاص هذا فضلا عن ان منهجية التطوير السريع توفر أدارة شفافة وكفوؤة للسيطرة على ادارة المشروع البرمجي.

الكلمات المفتاحية: الطرق الرشيقة، البرمجيات المدمجة، النظم المدمجة، يتنوع ويتزايد الانتشار والتطبيقات.

1 Chapter One

Introduction and research framework

No one can refute the significance of computer in our life, particularly in the contemporary period. Actually, computer has developed crucial in today's existence as it is utilized in countless arenas of life such as business, medication, trade, teaching and even farming. It has developed a significant component in the manufacturing and skill of progressive in addition to emerging countries. Currently, societies become more reliant on computer in everything therefore of computer machinery. Computer is deliberated as time-saving equipment and its development aids in performing multifaceted, extended, recurrent procedures in a very small period with a great speed. Additionally to consuming computer for effort, persons utilize it for entertaining and amusement.

Strikingly, the figure of firms that create software programs for the drive of easing mechanism of workplaces, managements, banks, etc., has enlarged lately which marks in the trouble of numbering such firms. During the preceding forty years, software has been industrialized from an instrument utilized for examining info or resolving a problem to an invention in itself. Though, the initial programming phases have shaped a lot of problems making software and problem to software growth chiefly those depending on computers. Software contains papers and programs that comprise a gathering that has been recognized to be a share of software engineering processes. Furthermore, the goal of software engineering is to generate an appropriate effort that concepts databases of great excellence (Caudrado, Canovaslzqierdo, & Molina, 2014).

1.1 Software Procedure Prototypes

A software procedure prototype is an intellectual picture of a procedure. It depicts an account of a procedure from some specific viewpoint as:

1. Specification.
2. Design.
3. Validation.
4. Evolution.

Common Software Procedure Prototypes are:

1. Waterfall model: Distinct and discrete stages of requirement and development.
2. Prototype model.
3. Rapid application development model (RAD).
4. Evolutionary development: Description, expansion and authentication are inserted.
5. Incremental model.
6. Iterative model.
7. Spiral model.
8. Component-based software engineering: The structure is gathered from prevailing modules.

There are numerous alternatives of these prototypes e.g. official growth where a waterfall-like procedure is utilized, but the requirement is official that is sophisticated through numerous phases to an implementable scheme (Sommerville, 2004).

1.2 Sex Models

A Software design procedure model is a nonfigurative depiction to label the procedure from a specific viewpoint. There are statistics of common models for software procedures, such as: Evolutionary development, Waterfall model, Reuse based development, and Formal systems development etc. This investigation will sight the subsequent five models:

1. Waterfall model

2. Iteration model

3. V-shaped model

4. Spiral model

5. Extreme model

6. Aspect-oriented and Agile model

These models are selected since their types agree to maximum software expansion databases.

1.2.1 The Waterfall Model

The waterfall prototype is the traditional prototypical of software engineering. This prototype is one of the first prototypes and is extensively utilized in administration schemes and in numerous main corporations. As this prototype highlights preparation in initial phases, it safeguards design faults before they grow. Furthermore, its concentrated text and preparation create it work fine for schemes in which feature regulation is a captain fear.

The wholesome waterfall lifespan contains of numerous non-overlapping phases, as exposed in the succeeding figure. The prototype instigates with founding system necessities and software requests and endures with architectural strategy, coding, comprehensive design, maintenance and (Munassar & Govardhan, 2010).

The waterfall way does not forbid recurring to a previous stage, for instance, recurring from the plan stage to the necessities stage. Though, this includes expensive revise. Each finished phase needs official appraisal and wide certification growth. Consequently, mistakes done in the necessities stage are costly to spot on later.

As the real growth originates late in the procedure, one does not get outcomes for a protracted time. This postponement can be disturbing to organization and clients. Numerous persons also think that the quantity of papers is extreme and strict.

Even though the waterfall model has its faintness, it is educational since it highlights essential phases of project growth. Albeit one does not relate to this prototype, he must contemplate each of these phases and its association to his personal task.

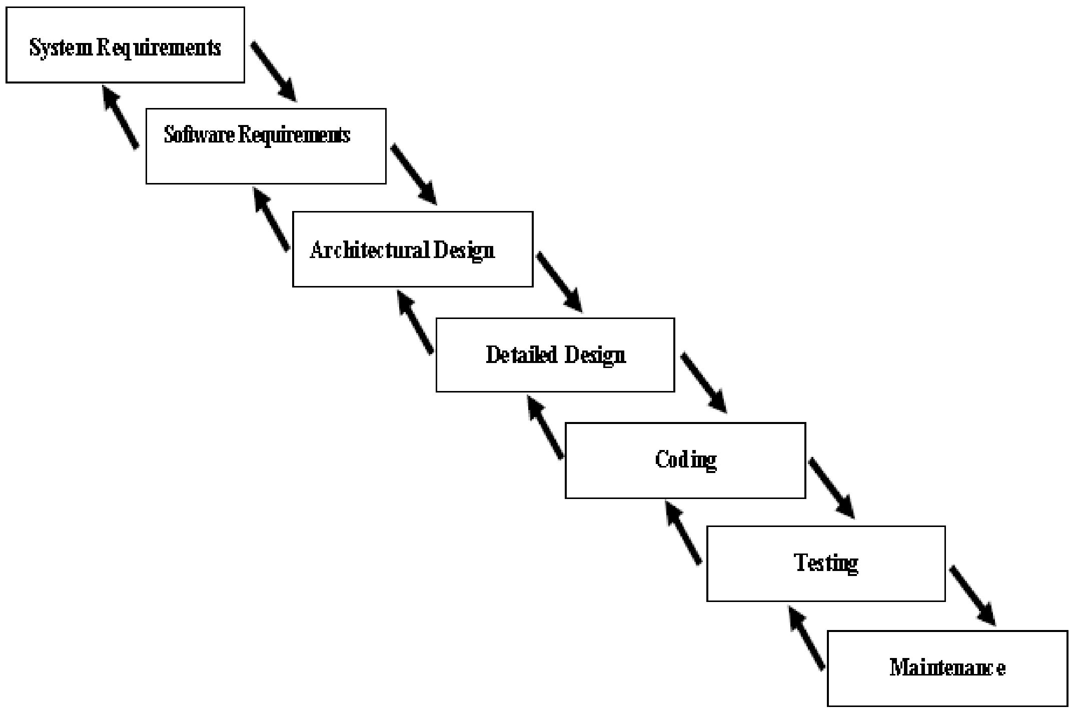


Figure 1-1: Waterfall prototype (Munassar & Govardhan, 2010)

1.2.2 Iterative Development

The difficulties with the Waterfall prototype shaped a request for a novel technique of emerging systems which could deliver quicker consequences, needs fewer straightforward data and offer better tractability. With Iterative Growth the scheme is distributed into minor parts. This lets the growth team to establish consequences previous on in the procedure and get valued response from scheme operators a lot, every repetition is really a mini-Waterfall procedure with the response from one stage offering dynamic info for the strategy of the following stage. In a difference of this prototype, the software products, which are made at the culmination of every phase (or sequence of phases), can go into creation instantaneously as incremental discharges (Verma, Bansal, & Pandey, 2014).

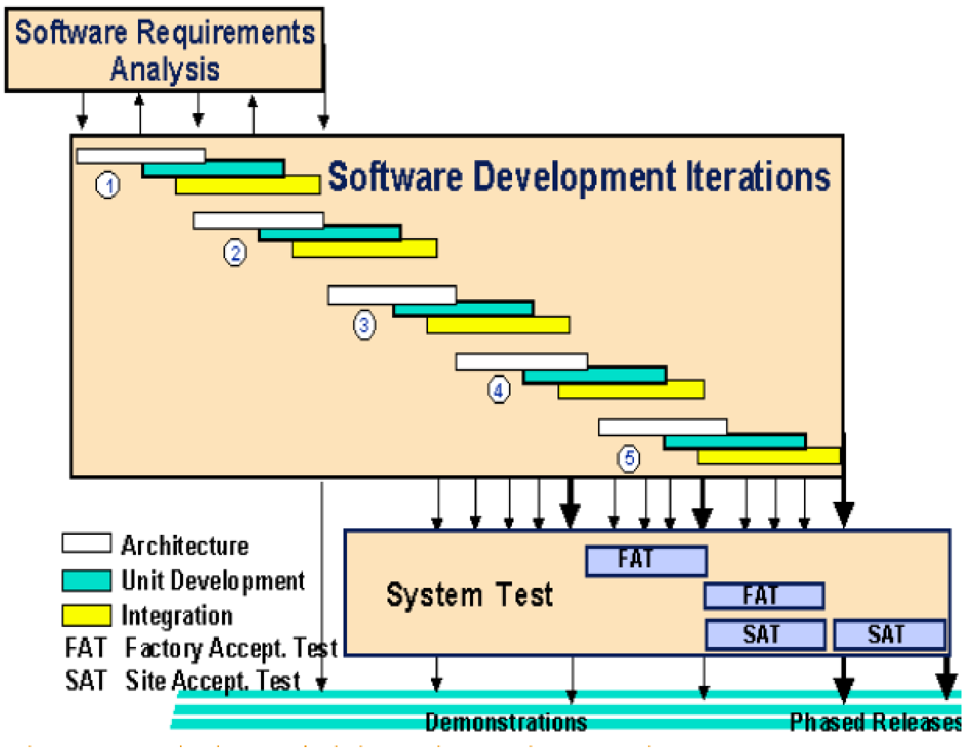


Figure 1-2: Iterative Growth (Verma et. al., 2014)

1.2.3 V-Shaped Prototype

Just like the waterfall prototype, the V-Shaped lifespan is a consecutive trail of performance of procedures. Each stage requisite to be accomplished before the subsequent stage instigates. Testing is highlighted in this prototype extra than the waterfall prototype. The difficult measures are industrialized initially in the lifespan in advance any coding is finished, through every one of the stages previous operation. Necessities instigate the lifespan prototype just like the waterfall prototype. Beforehand growth is underway; a scheme test strategy is shaped. The examination strategy emphases on gathering the functionality stated in necessities meeting.

The complex design stage emphases on system building and planning. An incorporation test plan is shaped in this stage with the intention of testing the bits of the software systems capacity to work organized. Though, the low-level strategy stage lies where the real software modules are intended, and component examinations are shaped in this stage also. The application stage is, once more, where all coding is done. When coding is done, the track of implementation endures up the exact lateral of the V where the exam strategies industrialized previous are currently put to utilize (Thomas, Adam, Hassan, & Blostien, 2014).

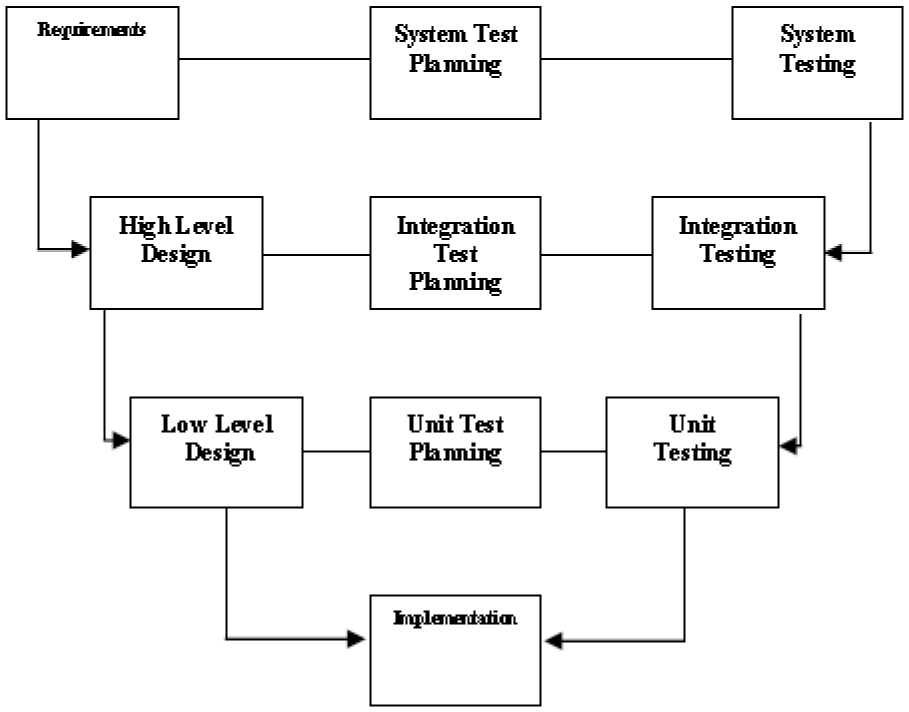


Figure 1-3: V-Shaped Life Cycle Model (Thomas, et. al., 2014)

1.2.4 Spiral Model

The twisting model is analogous to the incremental prototype, with more stresses positioned on risk study. The twisting model has four stages: Preparation, Risk Study, Manufacturing and Assessment. A software project recurrently permits through these stages in repetitions (called Twists in this model). The starting point twisting, preliminary in the preparation phase, necessities is collected and danger is measured. Each following twisting builds on the starting point spiral. Necessities are collected during the preparation stage. In the risk examination stage, a procedure is assumed to classify risk and alternative explanations. An example is shaped at the conclusion of the risk study stage. Software is shaped in the manufacturing stage, along with analysis at the end of the stage. The evaluation stage permits the client to assess the production of the scheme to date beforehand the scheme lasts to the following twisting. In the twisting model, the angular module signifies development, and the range of the twisting characterizes cost (Sommerville, 2004).

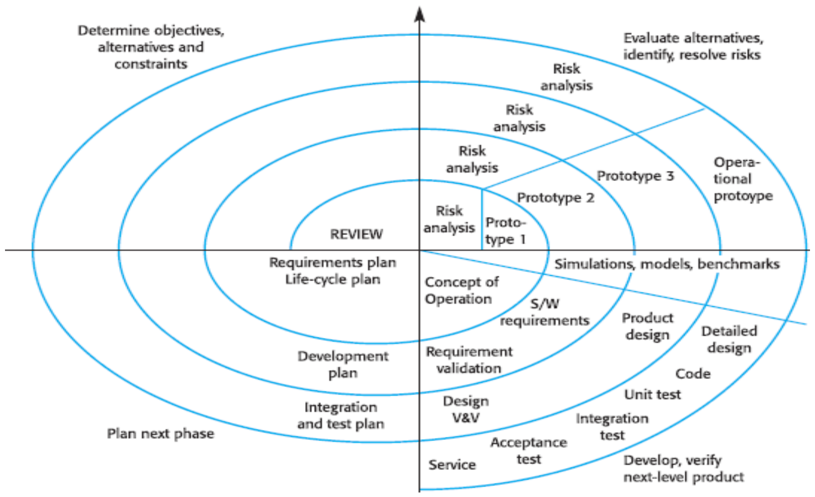


Figure 1-4: Spiral Model of the Software Procedure (Sommerville, 2014)

1.2.5 Extreme Programming

A method to growth, founded on the growth and distribution of very minor increases of functionality. It depends on continuous code development, user participation in the growth team and couple wise encoding. It can be problematic to save the attention of consumers who are tangled in the procedure. Team associates may be unfitted to the penetrating participation that typifies supple approaches. Listing variations can be problematic where there are manifold participants. Upholding straightforwardness needs additional work. Agreements may be a problematic as with other methods to iterative growth (Verma, Bansal, & Pandey, 2014).

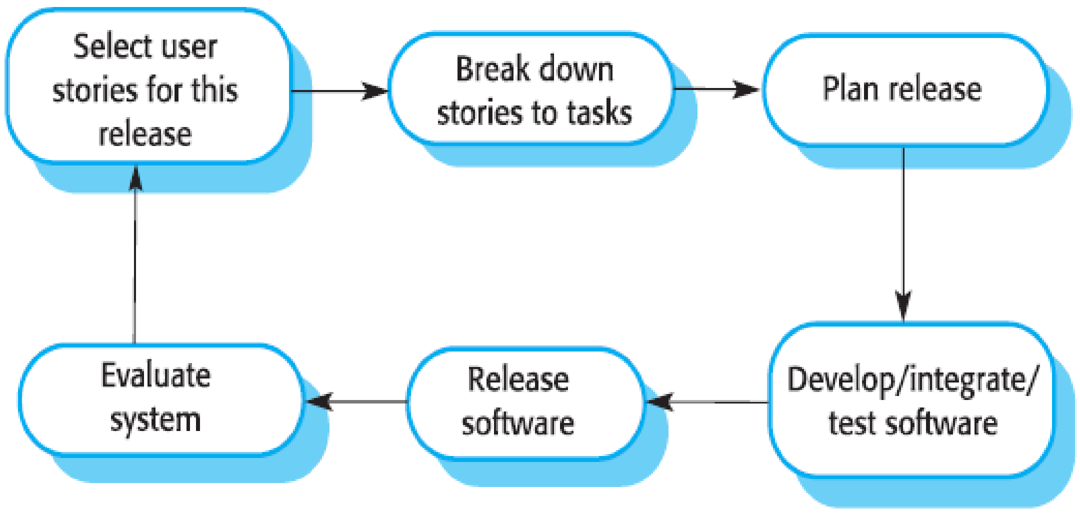

Select user stories for this release
Break down stories to tasks
Plan release
Develop/integrate/ test software
Release software
Evaluate system

Figure 1-5: The XP Release Loop (Verma, et. al., 2014)

1.2.6 Aspect-oriented and Agile

Aspect-oriented software design syndicates useful opinions of software (for instance use cases) with object or module opinions (for example class drawings). At dissimilar phases of the software lifespan, either a useful or article based opinion (or together) can be utilized, according to any which is more suitable at the instant. Largely speaking, a useful opinion is finest for necessities collecting, as operators bear in mind what the system does instead of how it is controlled. Both purposeful and constituent opinions are significant during system construction. Thorough plan, coding, and component analysis incline to emphasis on constituent opinions so that the software can be built in wieldy bits. Through incorporation, both module and purposeful opinions are vital, concluding in reception tests' emphasis on purpose. Both useful and module views are significant throughout upkeep, as the functionality of novel topographies or injections to current structures is measured, and influence examination and reversion challenging are smeared to modules. Aspect-oriented software design not only enhances an organized lookout again into component-based schemes, but also delivers a higher-level useful image. The useful "features" frequently crosscut organized procedures in addition to modules. Aspect-oriented programming depends on an outline of "link points", stipulations of diverse features that could be applied at the join opinions, and information for which feature(s) to really use. A "feature weaver" then supplements the simple or default functionality by joining in the counseled performance at the junction points. Responsive software expansion procedures are a modification of iterative procedures, in which repeatedly altering necessities are acknowledged or even cheered. Fast repetitions retain the system receptive to variations, and close interaction both amid consumers and creators and amongst designer's safeguards that the rising and transforming system properly accounts for the variations. Test-driven expansion safeguards that merely the smallest compulsory software is built, that it is built properly, and that it rests precise during the course of constant variations (West, Grant, Gerush, & Disilva, 2010).

Table 1-1: Differences between the Six Models

Factors	Waterfall	Iterative	Spiral	V-shaped	Extreme programming	Agile
Project Cost	Determined During planning	Set during project	Determined During planning	Determined During planning	Set During project	Set During project
Responsiveness to environment	Planning Only	At the end of Iteration	Planning primarily	Planning	Throughout	Throughout
Probability of Success	Low	Medium	Medium Low	Medium	High	High
Completion Date	Determined During plan	Set during project	Partially Variable	Set during project	Set during Project	Set during project

1.3 Embedded Systems

Embedded coordination has turned out to be a buzz expression in the previous five years, but embedded coordination and computers have been everywhere for considerably extended than that. One only wants to look round to perceive embedded systems ubiquitously: alarm clocks, cell phones, automobile subsystems such as ABS, personal data assistants (PDAs) and cruise regulator, etc. This segment takes an aspect at embedded systems, the subjects and outfits tangled in their strategy, existing tendencies, and how they can advantage from the investigation performed for this study (Baynes, et al., 2001).

1.3.1 Real-Time or Present Operational Systems

Real-Time Operational Systems (RTOS) are usually utilized in the growth, productizing, and distribution of embedded systems. Contrasting the sphere of common determination calculating, real-time organizations are typically industrialized for a restricted amount of jobs and have dissimilar necessities of their operational systems (Francia, 2001).This Segment first provides the necessities of present operative systems, then breakdowns the interior of RTOSs and clarifies them in part. This segment accomplishes with how the emulator industrialized in this study would support in the assessment of RTOSs.

1.3.1.1 Real-Time Operating Systems (RTOS): The Requirements

According to (Sha, et al., 2004), an efficient RTOS not only deals services and mechanisms efficiently to perform real-time preparation and source organization but also preserves its particular time and reserve depletion foreseeable and responsible. A RTOS is answerable for proposing the subsequent amenities to the customer programs that will execute on top of it.

The chief concern is that of preparation: a RTOS wants to propose the customer a technique to plan his errands. The second duty is that of scheduling upkeep: the RTOS wants to be accountable in both offering and upholding a precise scheduling technique.

The third duty is to propose consumer errands the capability to do system calls: the RTOS propose amenities to do specific responsibilities that the customer would usually have to plug-in himself, but the RTOS has them encompassed in its archive, and these organization calls have been augmented for the hardware arrangement that the RTOS is executing on. The final mechanism that the RTOS wants to deliver is a technique of allocating with interferes: the RTOS wants to propose an apparatus for usage interrupts professionally, in a appropriate way, and with an higher bound on the period it receipts to package those intrudes (Sha, et al., 2004).

There are numerous ideas that essential to be clear in any conversation of RTOSs. The chief idea is that of prevention. Actual operational systems are preventive and non- preventive. If a present operational system is preventative, it means that an undertaking presently being executed by the RTOS can be intermittent by additional job with an advanced importance or an outside interrupt. The intermittent job's state is protected, and this state will be reinstated when it is execute again, letting it to endure along from the similar theme that it was interjected. RTOSs that are non-preventive cannot be disturbed. If a job is presently running when another job wants to run, that additional task should postpone for the principal task to end running before it can initiate to run (Stepner, Rajan, & Hui, 1999).

One more significant idea is that of firm real-time contrasted with lax real-time. Firm real-time means a job always requisite to be accomplished by a particular period. The reliability of the system intended with firm real-time errands will be cooperated if such a limit is misused. An instance of this is the message device from the arena of a profitable aircraft to the embedded system supervising the wing blinders. If an aviator is impending in for touchdown, and jerks up on his blinkers to gentle his decline, that conversation should work — for if it doesn't, the whole airplane has the likelihood of booming. Lax real-time systems are whichever kind of system that is not a firm real-time system, implicating that if a job is late, the system will endure to retain running (Stepner, Rajan, & Hui, 1999).

There are numerous changed kinds of job preparation for today's real-time operative systems to select from. There is the boundless round scheduler, that is essentially though (1) loop that unceasingly runs a part of code. Doings inside the loop are performed in order and as countless times as conceivable. The following level of job preparation is that of the straightforward recurring decision-making scheduler. In a straightforward recurring preparation algorithm, the notion of the boundless ring is lengthy in that creators can discrete the program to be performed into distinct responsibilities. These errands perform in a normal arrangement in an enormously reiterating loop. This sort of preparation is frequently named round-robin preparation. Similar to the boundless loop, all of the responsibilities run as frequently as imaginable. Time determined recurring preparation, the following level of job preparation, fluctuates from elementary recurring in that rather than running every of these responsibilities as frequently as imaginable, it familiarizes the impression of a time interposes. This regulator awakens up the chief job in line, and the moment that first job is over, the subsequent job starts. All of the responsibilities in line should end before the following clock interrupt. Subsequent the period determined recurring scheduler is the multi-rate recurring decision-making scheduler. This is a growth of the stint determined recurring scheduler in that it permits manifold times, so extended as greater occurrence responsibilities part numerous of the base job's incidence. This is completed by introducing a job more than one period into the restraint or into manifold restraints. The multi degree administrative for intervallic responsibilities scheduler enhances the aptitude to have manifold times by founding a clock that is the lowermost common manifold of all of the times of all of the responsibilities. At every impulse of this clock, jobs can be prepared to perform. All of the exceeding preparation procedures frequently tackle with intrudes by introducing responsibilities and all of the before mentioned preparation procedures are non-preventive. A multi degree decision-making with disturbs permits outside interrupts to halt into present implementation and be repaired. The job intermittent is then resumed when the intervene is completed. Lastly, the importance founded preventative managerial scheduler is the identical as the multi degree decision-making with interjects excluding that it lets not only interjects to cessation into the present program, but responsibilities with greater importance too (Kalinsky, 1999).

1.4 Problem Statement

The chief tricky statement is how responsive approaches would be used in the expansion of embedded coordination, and what are their welfares, trials, and boundaries.

The main research questions as following:

1. What the dissimilarity amongst software models (iterative development, agile model, waterfall, V-Shaped, extreme program, spiral model?)

2. What the embedded system (design issues, trend, real-time systems, development tool?

3. How fit do agile approaches appropriate into embedded systems?

4. How do be relating agile approaches to embedded systems expansion?

5. What are the tests and policies that lead to effective embedded software growth?

6. How affect an agile expansion practice smeared to embedded device Software under severe hardware limitations?

1.5 Importance of This Research

Embedded systems are extensively utilized in various parts, such as consumer electronics, avionics, and medicinal apparatus's, producing a substantial influence on contemporary culture. As these systems occasionally deal unswervingly with mortal lives, and necessitate a substantial level of excellence, their growth would be theme to a hard procedure. In additional viewpoint, agile approaches (or agile procedures) have been accepted by the software engineering as a trivial, iterative, and cooperative method for emerging software coordination. Though agile approaches do not appear to be proper to embedded coordination, they have been positively utilized for constructing such schemes. Though, there occurs no thorough and logical impression of the usage of such approaches in the embedded organizations area.

1.6 Goals and Objectives

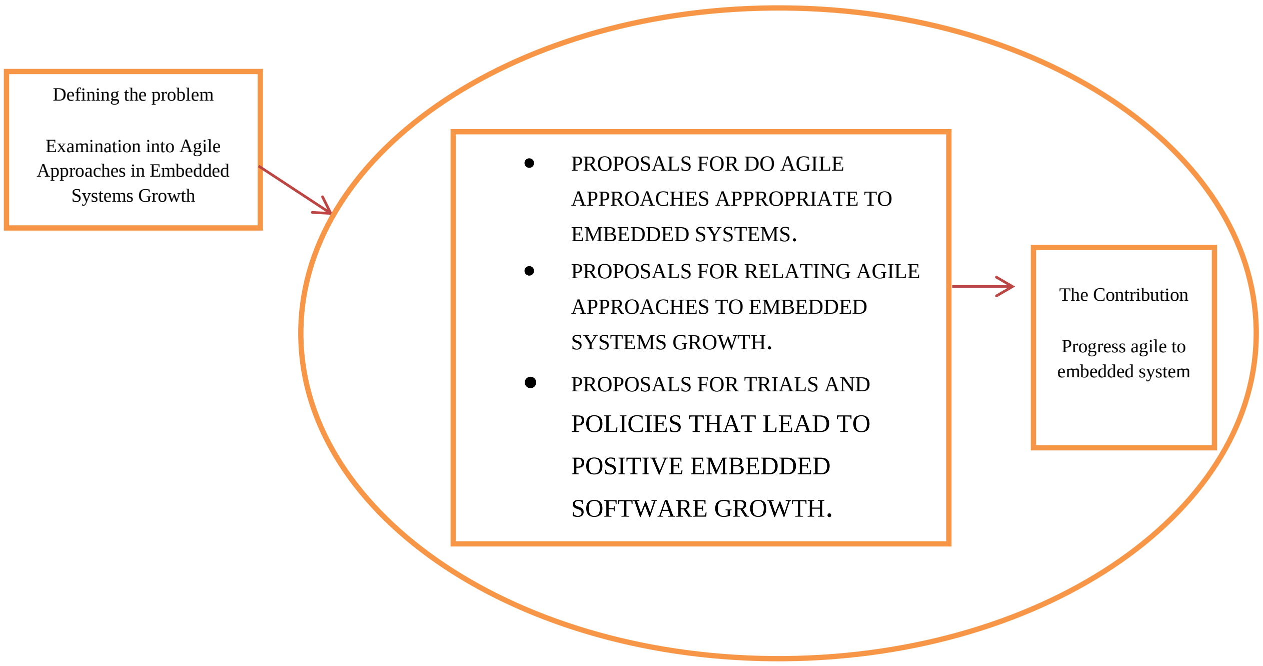
The aim of this research is to acquiring the comprehension of the trials and policies that leads to positive embedded software growth, furthermore accepting how agile technique features relates to embedded coordination software growth. The aim of this study is to discover the flaws of expending agile approaches in embedded software expansion that will offer an improved appearance and assumed the healthier conducts to evade tests if there are countless hardware limitations.

1.7 Methodology of the Proposed Solution

In this study, the way out of study problem will be resolved in two phases. Phase number one is to describe the issue, and phase number two is the role of this research.

Phase 1 (Defining the Problem): Proposal of An Study into Agile Approaches in Embedded Systems Growth

Phase 2 (The Role): discovering the proposal of suggestions that aid to surmount relating agile approaches to embedded systems growth.



Defining the problem
Examination into Agile Approaches in Embedded Systems Growth
PROPOSALS FOR DO AGILE APPROACHES APPROPRIATE TO EMBEDDED SYSTEMS.
PROPOSALS FOR RELATING AGILE APPROACHES TO EMBEDDED SYSTEMS GROWTH.
PROPOSALS FOR TRIALS AND POLICIES THAT LEAD TO POSITIVE EMBEDDED SOFTWARE GROWTH.
The Contribution
Progress agile to embedded system

Figure 1-6: Procedure of the Suggested Solution

2 Chapter Two

Literature Review

2.1 Introduction

The following is a literature review which I intend to use in my thesis. The journals are well authenticated sources that are well referenced thus making them ideal reference materials. Some of the characteristics I was looking at while searching for the reference material is the ability to answer my research questions. I have included a couple of journals that will give me information on how embedded systems are designed, the current market trends and issues. Some of the journals do have information on the benefits of using agile methods in the development of agile systems and their possible limitation.

The difference between software models is also well articulated as well as the application of agile methods in embedded system development. There is also an article by (PWC, 2013) that talks about strategies that can be used in developing successful embedded software. In addition, the articles choose have more information of developing embedded software that can be used in real times systems The researched articles gives me certainty that I will be able to come up with a succinct literature review that will support the basis of my research.

2.2 Literature Review

Eklund, U., Olsson, H. H., & Strom, N. J. (2014). Industrial challenges of scaling agile in mass-product embedded systems. (pp. 30-42). Springer international publishing.

This paper explores the challenges available in scaling up agile software practices and how companies can plan themselves to adopt such practices. The authors note that there is a significant change in the embedded systems industry which is brought about by the complex ever changing customer requirements. (Eklund, Olsson, & Strom, 2014) are of the opinion that, in order for companies to meet this changing customer demands and be economically viable in the market, there is need to scale up agile methods in embedded systems.

The researchers highlight some of the benefits of using agile methods and note that the major reason for the wide adoption of these methods is their efficiency in relaying products to the market at a much faster rate thus increasing the company's chances of meeting customer demands faster and easily. In addition they state that companies use agile methods to increase frequency of new products and features released besides improving the efficiency of their software engineering.

The authors also discuss in detail some of the challenges of using agile methods. One of the main challenges cited in the use of agile methods for large scale software development is that companies often practice agile methods in a way that is not compatible with the initial agile ideas and therefore making it difficult to incorporate them in their normal ways of working. The researchers further cite some of the characteristics that influence the adoption of agile methods under strict hardware constrains as adopted from (Abrahamsson, Warsta, Siponen, & Ronkainen, 2003) as follows; the use of detailed task specific information between teams making documentation difficult, the difficulty in implementation of embedded software because its development is mostly test driven ,there are also issues with performance of the software in meeting targeted specifications and lastly actualisation of the technology is difficult to ascertain as the software is developed experimentally.

The researchers conclude that although companies might want to adopt agile methods, up scaling them becomes a problem as the software developed only affects subsystems and not the whole product.

In addition prioritizing agile is made more difficult by the fact that there is limited functionality on product specific introduced software. This journal will be of importance to my research especially in literature review as it highlights the benefits and challenges of adopting agile methods. The writers have referenced widely thus making their work a credible citing source.

Chhya, A. S. (2008). A new process model for embedded systems control for automotive industry. The proceedings of the 2008 International Arab Conference on Information Technology, (pp. 16-18). Tunisia.

The researchers' purpose was to come up with a software model that is modified using the following three processes; Team software process, six sigma and Personal Software process. They further tested the new software process in an automotive embedded systems project which resulted in 70% defects improvement. The authors are of the opinion that software's quality, efficiency and effectiveness are the major setbacks in companies. The authors further note that there needs to combined efforts by teams in the development of software system to enhance coordination of the development process. They do propose a systematic process that is easy to grasp by the development teams for this to be effective. In addition, the authors suggest that although software processes enhance efficient service delivery and communication among the developers, customers and end users, the developed software should be efficient and effective to the company as well as reliable and of good quality to the chain members downstream.

The spiral model, PSP, TSP and sigma processes are described in their research. According to(Chhya, 2008), in the development of a spiral model a combination of features of the waterfall and prototyping model are used. They further give a detailed process of coming up with the spiral model whereby there is a detailed definition of the requirements of the new system followed by the creation of a preliminary design which is then used to develop a first prototype. After the first prototype development it is then evaluated for its effectiveness and then the second prototype developed after defining the requirements of this new prototype. If the customer feels that the prototype has a higher risk the project is immediately terminated and this necessitates development of another prototype using the results from the evaluation. This process is repeated until the customer requirements are fully met to get the final product which is tested for any errors before embarking on mass production.

The authors also give a brief description of the PSP process which is an individual process of software development for a defined activity. They state that in order to ensure the quality of the software developed the individual has to adhere to the standards and principles of PSP. This implies that the individual has to conduct a baseline survey and have planning steps. This will help the developer have a quality product developed in relation to available resources. The author’s further note that in this process the developer needs to have a way of detecting any defects in the development process as well as how to enhance the development capacity.

The authors do describe six sigma as an improvement tool where defects in a product are easily identified and eliminated. This reduces any wastes therefore making the process cost effective.

The authors do conclude that in an embedded control system the final product quality is a factor of the quality of hardware, software and system controls as a whole therefore putting emphasis on one part does not guarantee the quality of the whole system. Therefore it’s paramount to use 'Modified Spiral model using PSP, TSP and Six Sigma’, processes right from product development to come up with a wholesome quality product.

The authors of this paper have given a detailed explanation of different software methods before coming up with their proposed model which they later test. This a good way of presenting their work and there it is a credible reference source. The paper will be of paramount importance to my research work as it gives an explanation and difference in software models and their applicability.

Francisco Assis M. do Nascimento, M. F. (2006). ModES: Embedded Systems Design Methodology and Tools based on MDE. (pp. 67-76). Brazil: 07 proceedings of the fourth international workshop on Model-Based methodologies for pervasive and embedded software.

The researchers are of the opinion that the inefficiencies in the current existing system designs are pushing for better embedded approaches to be adopted. Their research maps out a meta-model that captures the processes functionality and communication of the embedded systems right from application to implementation. They present their approach using the features of the 'Eclipse Modelling Framework' and actualise their development through a real case. The researchers proposed model has a non-limited design space making it possible for any changes to be incorporated during implementation. The researchers go ahead and describe the methodology for their design that consists of all the steps in software design; application, mapping and implementation. They also test their tool in a real case study; in an automated wheel chair; thus authenticating their design. Lastly they compare their design with other MDE-based approaches thus making their work unbiased. From past researches the authors conclude that most of the works do not take into account the MDA principle of making the design space unlimited so that specific software can be changed as it deems during implementation. Their proposed new design model addresses this limitation. The research can be generalised as it has taken into consideration past work and done a good comparison. This paper is of value in my research in explaining how agile methods can be used in developing embedded systems.

Kopetz, H. (2000). Software Engineering for Real-Time: A Roadmap. Austria: Technische Universitat Wien. . In Emerging Technologies and Factory Automation, 1999. Proceedings. ETFA'99. 1999 7th IEEE International Conference on (Vol. 2, pp. 1557-1565). IEEE.

The author of this paper is of the opinion that, with the current highly depended applications, computer systems will replace mechanical and hydraulic control because of their efficiency and cost effectiveness. The author discusses in his paper the trends that will make companies shift their systems into embedded systems. In addition he describes some of the requirements that are supposed to be addressed by software developers to fast track this massive shift. Some of the requirements that the author proposes are; a good design that can be changed during implementation as well as validating the software before up scaling to reduce on any risks which he gives a detailed overview. The author further describes an ideal system component as that which is time efficient and testable. In addition he proposes that validating the software product is better done by evaluating the development process as well as testing the final product thus making it simpler to validate an independent product before it’s embedded in a system for mass production. In conclusion, the writer is of the opinion that the changes in the software industry will cause a shift in system validation by ensuring that individual products are validated rather than processes alone. This journal will be used in my research in getting more information on real time embedded systems and the trends that are currently changing the industry.

Goswami, A., & Bezboruah, T. a. (2009). Design of an Embedded System or Monitoring and Controlling Temperature and Light. International Journal of Electronics Engineering Research, 1(1), 27-36.

The authors of this paper come up with a description of an embedded system design that controls temperature and intensity of light in a single system as monitoring is done continuously in the process. The authors do appreciate that for efficient and effective operation in engineering most companies are now adopting embedded systems. The authors of this paper further come up with a description of an embedded system design that controls temperature and intensity of light in a single system. In this system they explain that monitoring is done continuously in the process. The researchers do appreciate that for efficient and effective operation in engineering most companies are now adopting embedded systems. The authors motivation comes with the fact that the proposed system will reduce a company's cost in terms of energy savings as the computers numbers used in the daily operations will be reduced. In the development of their software the author's emphasis is on the online monitoring and controlling features of the software but do not discuss offline analysis leaving this for future research.

Although the researchers give a detailed experiment of their processed system, they do not take it to trial therefore the system is not actualised. The authors may be biased in a way, in that they just need to promote their system but these needs to be backed up by previous research. Further, they also do not state the limitations of their proposed systems for example redundancy among others. This paper will not form a major basis of my research but it will be part of my bibliography in providing general knowledge about embedded systems.

Fernandes, J. M., & Machado, J. R. (2007). Teaching embedded systems in systems engineering in software oriented computing degree. 37th ASEE/IEEE frontiers in education research, (pp. 27-36).

The authors of this paper describe the topics that should be included in the curriculum design Masters in software engineering. They are of the view that embedded software design is important to be introduced in this curriculmn. In addition they state that students whose computer degrees are software oriented need to acquire effective skills in embedded systems engineering, so as to increase their professional competencies. The authors also go ahead and give a detailed definition of an embedded system.

According to(Fernandes & Machado, 2007), an embedded system is ‘a system developed to perform a specialized function by combining computer hardware, software and other mechanical parts’. This system is developed to be used in controlling systems like cars, planes, industrial systems among others therefore it’s not a computer system rather it is a computerized system.

The authors then further give a curriculum that can be adopted in delivering a Masters in software engineering program. They conclude that it is important for students to be taught embedded systems as currently the industry is moving towards use of embedded systems in their operations because of the systems efficiency and cost effectiveness. These researchers have made a wide reference and internal consultation therefore it’s a credible source. Although the paper discusses more of a curriculum in development, it will be useful in my research especially in the definition of terms like embedded systems.

Laanti, P. K. (2006). How to steer an Embedded Software project: Tactics for scaling agile software process Models. IJAM, 9(1), 59-77.

The researchers' major purpose is to investigate how to select an agile software process for the development of a market driven embedded software in the telecommunications industry. They propose a model selection frame that can be used by developers basing their knowledge on the currently available agile software process models like RUP, XP, ASD and FDD. They do this by using real problem issues in the available software processes. They are of the opinion that some problems in a project are as a result of the process model used. The authors do find out that no single system can be a solution to available problems. Therefore, the major aim of the authors is to present a project manager with ways of choosing an agile software process to use in a certain project depending on the available conditions and constraints and the reasons as to why the proposed process will be effective. They do this from literature reviews and their own practical experience. They propose that a project manager can come up with an effective agile process to use by evaluating the intended project, the current problem, associated risks and failures. They then propose the use of a comparison model matrix where the manager looks at the basic alternatives and how the different agile processes can tackle the specific problems. They conclude that using combined agile methods achieves more effective results than relying on only one agile process. They state the major limitation of their research as not being able to test the matrix on a real situation rather they used examples.

Their research work is not biased because past literature is widely consulted. The only limitation of this research is that the research may be applicable only to telecommunications hence cannot be generalized. This paper will form a basis of my research especially on supportive literature on finding out how well agile methods can fit embedded systems development.

Kaisti, M. R. (2013). Agile methods for embedded systems development - a literature review and a mapping study. EURASIP Journal on Embedded Systems .1 (15).

The main purpose of the study is to explore the current available knowledge of agile methods as used in embedded systems and to find out if using agile methods in embedded systems is suitable. The researchers explore their study to include both embedded software and hardware development. They do find out that agile methods can be used in embedded systems but these need to be product specific. In addition, the authors appreciate that, with the diversity of products serving different purposes there is need of adopting different agile methods in different situations. The authors further define an embedded system as ‘a computer system that is specifically designed for a specific function that combines both hardware and mechanical components of a system’. They give examples of where embedded systems are applied including cars, phones among others. They also define agile methods as ‘a combination of practices developed by software developers with a lot of planning and documentation’. The researchers conclude that for agile methods to be successfully applied in embedded systems, there is need of solving some product specific constraints.

They also propose more research on the benefits of using agile methods. This paper is well written and can be generalised as it has use a lot of references. The only limitation that I find with this paper is at times using one reference in explaining a specific point, it will be better if a particular view is supported by two or more references. Otherwise, it is a good paper and will add value to my literature review especially when looking at how well agile methods fit embedded systems and benefits of using agile methods as well as adopting some of their definitions.

Poulhies, M., Pulou, J., Rippert, C., &Sifakis, J. (2007). A methodology and supporting tools for the development of component-based embedded systems. In Composition of Embedded Systems. Scientific and Industrial Issues (pp. 75-96). Springer Berlin Heidelberg.

This research paper focuses on developing a methodology that can be used in developing an embedded system that is component based. They combine the BIP and THINK frame work in their methodology to come up with this system. The researchers validate their results by developing and testing a software MPEG encoder on an iPod. The authors are of the view that an integrated system preserves the structure and semantics of the system model. They also give the benefits of an embedded system in that it fosters control of developing and implementation of a product specific component. The authors of this paper appreciate that optimal use of resources and the quality of the final product are paramount in the development of embedded systems. In addition, they are of the view that cost efficiency and meeting the customer delivery requirements should also be an important consideration.

They therefore propose the use of component based methodologies that are easy to implement, execute and upscale. Further, the authors are of the view that it is important to detect any defects early in the development processes by doing a thorough evaluation of the process and the final product. The research may not be generalised at the methodology is only tested on iPods. Otherwise, the paper is well presented in a detailed form. This paper will form part of my bibliography in my research work as it has general information on embedded systems.

Khanjani, A. (2011). Comparison between four software engineering Approaches: Component based software engineering, agile methods, aspect oriented and mash-up. International Journal of Advances in Computer Science, 2(4), 20-26.

The author of this paper focuses on giving a comparison of techniques used in software engineering. He compares component based software engineering, agile methods, Aspect oriented and Mash-up in different applications. The comparison is done in terms of usability, cost efficiency, security and reusability. He finds out that the agile method is more effective when used in the development of small system software whereas in the development of large scale systems the component-based and Aspects Oriented methods are a better option.

The Mash-up method can be used to develop either small or large systems. He further finds out that all these techniques are cost effective. His study also finds out that component based and Mash-up has security issue problems. In addition the author points out that the agile method cannot be easily reused while the rest of the techniques are easily reusable. The author further gives a description of this technique alongside their benefits. He states that the agile methodology is customer oriented, easy to develop and implement within a short time, is of a higher quality and improves communication and coordination as well as efficient. He further gives the limitation of agile methods like; it is hard to upscale it, has no specific schedule, limited documentation and needs several teams to coordinate development. The author goes ahead to highlight the benefits and challenges of the other three techniques. The writer concludes that component based system is the best technique for real time systems because of its flexibility, efficiency, timeliness and cost efficiency. On the other hand the agile method is better placed in developing small systems as it is more secure. This journal is of relevance to my research as it gives a description of different embedded systems models alongside their limitations and benefits.

PWC. (2013). Accelerating embedded software development via agile techniques; the nine strategies that lead to successful embedded software development. Technology institute.

The purpose of this paper is to equip project managers with strategies that will help them apply agile practices in the development of embedded software.

The author is of the view that although agile practices are cost effective it has challenges in its implementation when used in the development of embedded. Further the author states that, for companies to be able to differentiate their products and cut out a market niche for themselves they need to use embedded software which is more flexible and adaptable. The paper presents strategies that can help companies benefit more as they rely in use of embedded software. The strategies are summarised into nine steps which the author believes that if followed carefully it will lead to the development of effective embedded systems. The writer also highlights the characteristics as well as benefits of the agile methodology in developing embedded systems.

The writer concludes that challenges in using the agile method in embedded systems can only be overcome if the design, development and implementation are carefully handled with evaluations done at it stage. This paper was written for a specific audience although anyone can use it (Chhya, 2008). The language used is simple and the techniques can be used on a wider scope as it is not industry specific. The paper will form part of my literature review especially when looking at agile methods versus embedded systems.

Cordeiro, L., Mar, C., Valentin, E., Cruz, F., Patrick, D., Barreto, R., &Lucena, V. (2008). An agile development methodology applied to embedded control software under stringent hardware constraints. ACM SIGSOFT Software Engineering Notes, 33(1), 5.

This paper focuses on a research done by several scientists with the aim of developing an agile methodology to be used in developing an embedded control software. The authors do design a methodology of developing an embedded system using agile principles. The methodology focuses on the systems constraints, safety and applicability. To ensure the efficiency and effectiveness of the methodology the authors have a well laid out testing procedure. The authors are of the conclusion that their proposed methodology is both cost and time efficient as well as reliable. This proposed methodology which they call TXM follows the principles of agile methods making software development easy.

The case study used to validate the methodology is the development of a digital soft starter and induction motor simulator. They are still doing more experimental studies with the methodology. This paper has a well explained methodology and validation procedure. The paper will form part of my review in looking at how to apply agile methods in developing embedded software systems.

Gomaa, H. (2008). Model-based software design of real-time embedded systems. IJSE, 1(1), 19-41.

The author of this paper describes how to design real time embedded systems using a model based software design method using the UML notation. The author appreciates that the market trend is currently moving towards more microcomputer based system because of their efficiency and effectiveness. The author gives a description of real time systems as those systems whose decisions are state dependent. This systems need to be developed using current designs as they process concurrent inputs from several sources. He also states the design should be able to develop a software that can run concurrent tasks with minimal constraints. The author gives a detailed description of the Model-Based software design. In addition, he gives a highlight of the COMET model-based software and how it works. He further proposes that a wholesome view in the design of a system is paramount in meeting the intended objectives.

The author concludes that the methods used to design software will soon be of importance in future. This paper will be used as a bibliography in my research to gain more knowledge on the development of embedded systems.

Rajawat, P., & Rajendra, P. (2011). A servuey of embedded software profiling methodologies. International journals of embedded systems and applications, 1 (2), 19-40.

The purpose of the authors is to give an overview of different ways to use in software profiling tools and methodologies that can be applied currently and in future. The authors are of the view that, in order to ensure the application requirements are fully met during the development of an embedded system a software developer has to perform an exploration of the design. Achieving this can be through choosing different methods or dividing the tasks among teams. The authors also identify some of the available common principles from their observations as well as evidence from past research.

Then they propose a way of classifying the tools as well as a comparison of these profiling tools. They classify the tools into software based profiling, hardware based profiling and FPGA based profiling. The authors further give the benefits and limitations of adapting the mentioned profiling methods. The authors of this paper have widely researched in this area and used credible reference sources. The major strength of this paper is that it gives future direction that can be used by upcoming researchers interested in this field. They do so by highlight some of the research gaps and what needs to be done in future. This paper is related to my study in that I will be able to get more insight on how an embedded system can be profiled and the limitations of each approach.

Apvrille, L., & Roudier, Y. (2014). Towards the Model-driven engineering of secure safe embedded systems. GraMSec, 15-30.

The authors focus is to present a SysML-based Model driven engineering environment which they term as SysML-Sec which will improve the relationship between software designers and security experts in the development of an embedded system. They propose that this methodology will evaluate any security requirements earlier on in the development process.

The authors appreciate that privacy and security issues have recently been a major problem in embedded systems. Their methodology includes three stages; analysing the system so that security threats are identified in unison with functional features of the system, focus of the system design on software-implemented security mechanisms and lastly validating the system by formally testing it. The researchers do test their proposed approach to make sure it is effective through a case study.

Thy aim to further evaluate the validity of this methodology through more experiments. This paper is presented in well detailed manner with validation and limitations of the said methodology well-articulated. Therefore the research can be generalised and adopted by other researchers as it is well argued with evidence. The relevance of this paper to my study is paramount as it will provide me with more information on using agile methods to develop embedded software systems.

Kaisti, M. R. (2013). Agile methods for embedded systems development - a literature review and a mapping study. EURASIP Journal on Embedded Systems 2013, 1 (15).

The authors' objectives are to come up with a collection of agile methods which teams can adopt in the development of embedded systems. The origin of the practices discussed is from the agile principles as presented in the agile manifesto which are envisaged to increase team productivity and software development.

The authors also try to come up with ways of enhancing collaboration between software developers and product users as well as between traditionally and agile working teams. The authors appreciate that agile methods are largely being adopted in software development nowadays because of their efficiency and productivity. They also examine the effects of adopting agile methods in terms of productivity and effectiveness. The authors state that the adoption of agile methods is sometimes challenging as many of the practices are transformational. They are also of the view that this methods are expensive to fully adopt as the hardware solely depends on embedded systems.

The researcher therefore embarks on a research of agile methods that follow the agile principle methodology. They find out in their preliminary analysis that different companies adopt different agile methods. This paper is still in development and only preliminary findings are highlighted. Since it's a project it might take a lot of time to be completed. This paper will not form a major basis of my research because it's not well referenced.

2.3 Conclusion

The above articles have a lot of similarities with few differences(Eklund, Olsson, & Strom, 2014), (Francisco Assis M. do Nascimento, 2006) and (Koptez, 2000) are taking into consideration how agile methods are beneficial and their limitations. They are of the opinion that prioritizing agile is made more difficult by the fact that there is limited functionality on product specific introduced software. These journals will be of importance to my research especially in literature review as they highlight the benefits and challenges of adopting agile methods as well as getting more information on real time embedded systems and the trends that are currently changing the industry.

The writers have referenced widely thus making their work a credible citing source. The common limitations in some of the articles which are proposing a methodology is the lack or limited validation of the methods. An example is (Goswami & Bezboruah, 2009), where although the researchers give a detailed experiment of their processed system, they do not take it to trial therefore the system is not actualised. The authors may be biased in a way, in that they just need to promote their system but these needs to be backed up by previous research. In the seventh article this limitation is also depicted as the research may be applicable only to telecommunications hence cannot be generalized. The only limitation that I find with (Kaisti, et al., 2013) is the fact that most of the time the authors have used one reference in explaining a specific point, it will be better if a particular view is supported by two or more references. Further on(Poulhies, Pulou, Rippert, & Sifakis, 2007), study, the research may not be generalised as the methodology is only tested on iPods. The remaining journals do talk about development of embedded systems and the methodologies are well validated. In conclusion, all this journals are of relevance to my research if the ones that have limitations, some of them will be used in the literature review while some will form part of my bibliography.

3 Chapter Three

Agile Software Development and Embedded Systems in Large Concept

This Chapter will discuss two main subjects; the first is background about Component-based development, the second concerns with the embedded system and how software engineering model especially agile model.

3.1 Introduction

In the software engineering, component-based development is of excessive attention and has attained substantial achievement in numerous engineering and implication spheres. Component-based development has been comprehensively utilized for some years in desktop situations, e-business and office applications and at large in web- and Internet-based dispersed uses. In several other areas, for instance real-time and embedded systems, component-based development is used to a smaller amount. It has been practiced that it is challenging to utilize the similar module machinery in diverse areas as of different system necessities and limitations. The newest tendencies display that diverse component expertise are being industrialized for dissimilar fields. Likewise to object-oriented (OO) example that is subjugated in diverse object-oriented languages, a CBD model founded on specific shared values is gradually constructed and utilized in dissimilar constituent tools (Crnkovic & Larsson, 2002).

The purpose of this search is to provide an outline of values of component centered software engineering and their operation in growth of embedded methods. The purpose is to display that the CBD method can effectively be utilized in growth of embedded methods though the diverse apprehensions, desires and boundaries are usable then for methods that effectively have utilized component-based development. Straight use of general purpose component based tools is typically not viable; instead precise implementations are obligatory, or novel component prototypes that openly address the chief apprehensions of embedded methods need to be established.

3.2 General Concept of Embedded Systems

Computer systems that are share of greater structures are named embedded systems and they fulfill several of the necessities of these structures. Certain instances of such systems are vehicle regulation systems; mobile phones, manufacturing procedures control systems, or minor device switches. A huge variety of computer systems are covered by embedded systems ranging from very small computer based machines to big systems checking and regulating multifaceted procedures. The awesome amount of computer systems goes to embedded networks: ninety-nine percent of all calculating components go to embedded networks nowadays.

Maximum of such embedded networks are likewise considered as physical systems, which show that the present features for instance reaction time, adverse case implementation period, etc., are significant project apprehensions. These systems frequently should fulfill severe conditions for security, dependability; obtain ability and other qualities of trustworthiness. Because of trivial scope and necessities for flexibility, but likewise tremendously low manufacture prices these systems necessitate lesser and controlled reserve usage, and have restricted hardware capability (Crnkovic & Larsson, 2002).

The enlarged difficulty of embedded present methods leads to swelling difficulties relating to necessities engineering, complex strategy, initial error recognition, efficiency, incorporation, confirmation and preservation, which upsurges the standing of a well-organized organization of life-cycle features for example portability, maintainability, and inflexibility.

3.2.1 Fundamental Ideas for Component-based Embedded Systems

In standard engineering domains a component is an independent share or sub system that can be utilized as an element in the strategy of a superior structure. There are numerous dissimilar proposals for a description of components in CBSE. In the software engineering world, the greatest perception of component is founded on (Szyperski, 2002) explanation.

After this description it can be expected that a constituent is an executable element, and that positioning and arrangement can be achieved at execution time. In the areas of embedded methods this description is mostly used; this is particularly factual for the split-up amid component application and boundary. Though the loads on the double or executable after is not directly used. A constituent can be carried in a method of a basic program written in a sophisticated linguistic, and lets shape-time (or plan-time) arrangement.

The features of the constituent that are outwardly noticeable to the other fragments of the system can be summarized by the component interface. As per embedded systems extra functional features are as significant as practical there is an inclination to comprise requirement of extra functional features in the component edge (for instance timing properties). This permits additional system features to be described when the system is calculated, namely such interface allows confirmation of system necessities and forecast of properties of system from components properties. Overall for component expertise, the interfaces are typically applied as purpose interfaces backing, by latter binding, polymorphism. Although delayed binding permits linking of components that are totally oblivious of all other alongside the linking interface, this elasticity originates with a presentation consequence and amplified danger for system disappointment. Likewise the obviousness of the scheme's presentation or other characteristics cuts as the configuration of the modules happens at run time.

Consequently the vibrant component placement is not possible for minor embedded methods. Owing to the restraints for actual and imperfect means there are numerous motives to perform component placement and arrangement at intention period instead of run time: This permits configuration tools to produce a monumental firmware for the scheme from the component based strategy and through this accomplish healthier functioning and improved expectedness of the system performance. This too allows worldwide optimizations of a still component configuration, contacts amid components might be interpreted into straight purpose calls in place of, by means of vibrant event announcements and confirmation and forecast of system desires which can be completed statically from the assumed component features. Koala is a typical instance of a component prototypical from embedded methods industrialized and utilizes at Philips. For great embedded structures the reserve limitations are not the main apprehensions. The difficulty and inter-operability play considerably extra significant part. Likewise owing to complication, the growth of such method is very costly and decreasing the expansion prices is extremely important. Therefore general purpose component expertise is additional curious than in a circumstance for minor organizations.

The schemes utilizing these skills go to the group of soft actual systems. Frequently component machinery is utilized as a foundation for extra generalization level provision, which is quantified one or the other as collection of values or branded explanations. One positive instance of acceptance of a component based expertise is the enterprise OLE procedure control Foundation, an association liable for a requirement that describes an array of regular boundaries for procedure mechanization founded upon OLE and lately .NET. Additional instance of a component-based method is expansion and usage of the regular IEC 61131.

A set of languages in which purpose chunks can be observed as constituents and interfaces amid chunks are out by linking out-ports and in-ports are defined by IEC 61131. Great embedded structures that should accomplish firm actual necessities frequently do not usage general purpose component-based expertise. Though in particular circumstances, a condensed form of a component prototype is utilized on a upper of a present operational method.

3.3 Agile Approaches

Agile approaches Plan driven growth means were almost the single substitute for administrations till the 1990. (Royce, 1970) Familiarized a prototype that in the 1970s turned out to be recognized as the waterfall model. It has usually been deliberated that the accessible single pass waterfall sequence is a model growth prototype when, in effect, Royce utilized it as a basic instance earlier happening to iterative representations that he really favored. The substance of agile approaches is in the incremental and iterative growth. Agile approaches have increased swelling approval subsequently the 1990s when the agile crusade instigated and numerous software manufacture procedures progressed for example the renowned Scrum procedures and Extreme Programming. The sides are usually minor, co-located and self-organizing, employed thoroughly composed that aids in creating first-class software. Recurrent response from carefully cooperating consumers is too endorsed as a ways of satisfying client requirements(Larman, 2003).

The agile approaches are a collection of performs shaped by knowledgeable software designers. The agile approaches can be understood as a reply to plan-driven procedures that highlight widespread preparation and certification with severe procedures, and have an opinion that foreseeable and specifiable explanations to difficulties occur (Dyba & Dingsoyr, 2008). In contradiction of plan driven approaches, Agile Platform www.agilemanifesto.org determines the subsequent: persons and connections over procedures and implements, functioning software above all-inclusive documents, client association over agreement cooperation and replying to a substitution subsequent a strategy. Although the platform highlights the standards of the substances on the left-hand in excess of the ones on the right-hand, it does not leave those on the right-hand either.

In Figure 3-1 the chief dissimilarities amid the agile growth standards and the plan driven are exemplified. In plan driven growth, there are consequential stages which are typically controlled by phase-detailed sides. Data amid the stages is transported by means of widespread documents. In agile practices, the growth is completed in incremental repetitions in combined sides. Necessities for the industrialized coordination are kept in the manufactured goods build up.

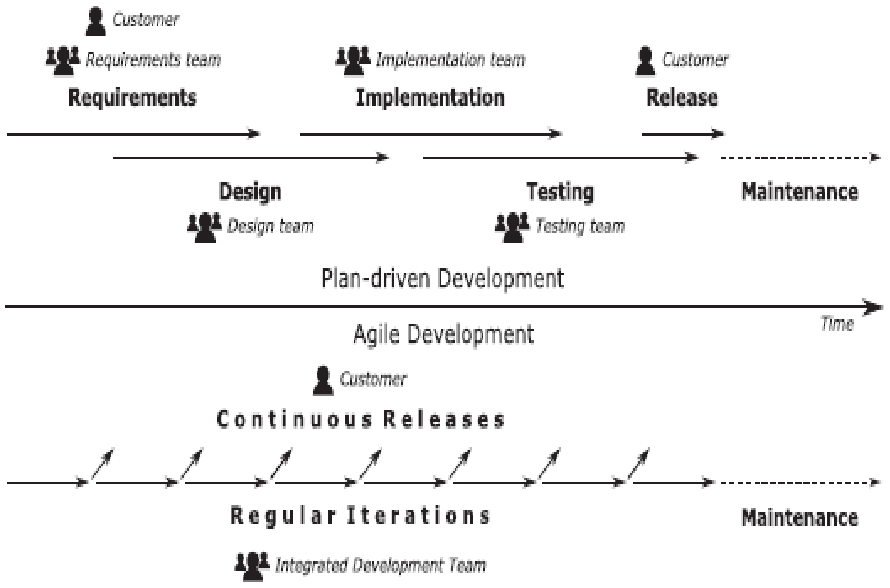


Figure 3- 1: The foremost differences amid the agile development methodologies and the plan-driven.

3.4 Embedded software growth Challenges

Five challenges have been recognized which have been supposed is needed to be overwhelmed to pause project growth trade-offs. These tests have been categorized Tools, Complexity, Interdependency, Verification, and Optimization.

1. Complexity

- **Explanation:** complexity is the consequence of numerous tendencies. It rises because of the amalgamation of additional and extra working onto a particular system, progressively multifaceted values (predominantly in media and wireless uses), and the obtain ability of additional and extra transistors to plan with. Also, as networking abilities are flattering convincing in embedded methods, a design develops an organization of organizations, adding yet additional layer of difficulty (mosterman, 2006). To estimate the ITRS "…composed, the silicon and structure difficulty trials suggest super exponentially swelling difficulty of the strategy procedure"

- **Effect:** the ITRS recognizes complexity as one of the key drivers behind increasing system design costs .Currently increased system complexity can lead to increased project delivery times and quality issues.
- **Solution Characteristics:** some means of affordably coping with increased complexity must be developed. As complexity is increasing exponentially, this requires any solution to scale sub-linearly with complexity if it is to be affordable.

2. Optimization

- **Explanation**: optimization is essentially a cycle of design, evaluation, and design again. It becomes a difficult issue because of the length of time taken to build a functioning system model or prototype to evaluate. An embedded system contains both hardware and software elements that interact in complex ways with one another which can make selecting the 'best' combination difficult. Currently the only way this can be achieved is by prototyping the different solutions or by guesswork.

- **Effect:** it is often the case that only one design can be optimized (due to the cost of building a functioning design), hence severely limiting the optimization possibilities. Choosing an incorrect or sub-optimal software/hardware combination can lead to project delays or even project failure.
- **Solution Characteristics:** it must be easier and faster to develop functioning system models or prototypes that enable a wide range of design options to be quickly and accurately evaluated. It must be possible to simulate the interaction of hardware and software.

3. Verification

Explanation: verification is a multifaceted and extensive reaching theme. At its humblest it requests whether the structure as realized fulfills the requirement or not, though much additional multifaceted difficulties are likely. For instance for security dangerous structures there might be a condition to establish that the structure never changes through a hazardous condition or to show that certain system circumstances are inaccessible below specific operational situations.

- **Influence:** verification prices are well-known to be increasing abruptly, with certain writers upholding that confirmation will use 50-70 percentage or additional of plan growth period (El-far & Whittaker, 2001).This condition is probably to deteriorate except confirmation practices alter. Regardless of automatic confirmation the occurrences of state space bang (Design + MEDEA Automation Roadmap, 2005). For multifaceted structures will endure to create mechanization a problematic job and decrease the probability of attaining complete model and code treatment in analysis. Present design procedures also incline to discover mistakes late in the growth procedure when they are greatest costly to accurate (Dabney, 2004).

- **Solution characteristics:** some procedure of confirmation computerization by means of the system requirement is virtually positive. This in sequence has insinuations for the procedure of the system requirement as any automatic examinations need to be resulting from an extra proper explanation or MOC which might be a runnable description. A runnable requirement (or the more overall usage of Model Based Scheme) will aid with the initial discovery of mistakes (Executable specifications: creating testable, 2001), and is assessed to have a considerable influence on scheme efficiency (MEDEA+ Design Automation Roadmap, 2005).

4. Tools

    - **Explanation:** the growth tools utilized by embedded software designers are considerably fewer complexes than those utilized by desktop uses creators for instance. At large, present Integrated Development Situations are upright at small level growth, but don't propose additional high-level growth methods for example re-factoring. Also these implements deliberate on merely one phase of the scheme procedure or on one scheme area and there is frequently restricted incorporation amid tools, for instance amid confirmation tools and an IDE.

- **Influence:** the ‘point solution’ behavior of present scheme tools avoids intelligible workflow and tool chain incorporation, for instance creating cross area optimization (e.g. digital and RF) problematic or difficult. The whole thing of the encounters itemized in this paper will need variations in the behavior and kind of expansion implements utilized for embedded software growth. Fiasco of implements to change will end development.
- **Solution characteristics**: tools will be vital to deliver high-levels of inter-operability. This can be attained at the ideal material level by the growth of inter-operability values. Otherwise, at the arithmetical interface stage, a execution groundwork might permit the addition of the undercurrents demonstrated by diverse tools. In reaction to this test there has been a change in investigation significances in the arena of demonstrating and reproduction in the direction of the elevation of tool inter-operability and additionally overall workflow matters (www.spiritconsortium.org).

4 Chapters Four

Agile and Embedded Systems

4.1 Embedded Systems Characteristics'

A system that is not primarily a computer but contains a processor is known as embedded system. However, it is more important to highlight the common aspects of most embedded system to some extent (Noergaard, 2012).

- **Price and Size**

To offer an appealing price point and portability to end users, embedded systems are engineered to possess just enough resources, such as processor power, size of on-chip / off-chip memory and number of peripherals. Many embedded systems products also have physical size constraints, size and weight for example, to be as portable as possible (Deng, 2014). There are some embedded systems that have physical constraints on form factor and not much expensive. These embedded systems are also using small components to optimize the performance at the cost of maintainability. To operate the embedded systems “clarity, portability or modularity” needs an optimization factor by using low intensity language. For example in place of C language or C the code is generated by the UML model. This modification is usually applied only on small components of the system, recognized by means of “90/10” principle as being the major performance bottlenecks (Dahlby, 2004).

- **Power Requirements and Limitations**

Compared to desktop computers, embedded systems often contain far less computational power and memory. The available processing capabilities, memory and peripherals are just enough for the tasks that need to be performed (Deng, 2014). System software utilizes the battery, moreover during emergencies and continually. Hence, consumption of power is chosen for most embedded systems although it comes at the rate of sophistication and maintenance requirement (Dahlby, 2004).

- **Real-Time Processing**

A real-time system is one whose correctness involves both the logical correctness of outputs and their timeliness. It must satisfy response-time constraints or risk severe consequences including failure. A real-time system is one in which the correctness of the computations not only depends upon the logical correctness of the computation but also upon the time in which the result is produced. If the timing constraints are not met, system failure is said to have occurred. These systems respond to a series of external inputs, which arrive in an unpredictable fashion. The real-time systems process these inputs, take appropriate decisions and also generate output necessary to control the peripherals connected to them. The design of a real-time system must specify the timing requirements of the system and ensure that the system performance is both correct and timely (Vijay, 2001). The real time constraint also gives favor in performance aspects over maintenance perspectives. These operating systems are mainly using “prioritized scheduling” to make sure that either deadlines are meeting or not, but “careful thought” needs to be separate into the execution stage, control /organize the data flow between the implementation and to set the levels of contexts (Shih, 2014).

Use Custom Hardware

Embedded systems are often designed to perform a small set of dedicated tasks continuously and repeatedly. The hardware is customized specifically to fulfill the requirement of targeted tasks. Thus, the development of embedded systems usually involves a software and hardware co-design pattern which requires developers to possess detailed knowledge on both software and hardware of the system as a whole (Deng, 2014). The software of a system is often composed of “off the-shelf processors” coupled with “off-the-shelf peripherals”. Components of the software must be in ordinary size, because the convention integration and similar needs requires high amount of unity among the software and hardware of the system. This software is frequently accessible for purchasing and for using purpose. The software of embedded system is mostly custom developed in houses, or to attach together “off-the-shelf” software in a tradition arrangement. The efficiency of a system totally based on numerous examines processor or slave/master processors. Careful thought is essential for processing the tasks among “processors, method, extent and timing of communication between processors”. However, software takes advantage of specific “FPGAs or ASICs”. Due to this reason this software must interact with the hardware (Dahlby, 2004).

- **Hidden From View**

By character, much software naturally has a restricted interface with their “user” that is true user or other component of the super system. The main objective of the embedded system software is to develop to meet, instead of the requirements of the user.

- **Monolithic Functionality**

Many software is using for the particular main reason. These systems are decaying into components and these apparatus have low fractious consistency power and fractious coupling. That means, each constituent has a discrete purpose. The connections among the mechanism are limited to a small number of fine distinct points. Each components of a system must be operational to perform the functionality. All the components must be function properly so that system can achieve useful functionality so that this system can be called as the "monolithic system". That creates the "non linear jump" in the efficiency of the software. The maintainability of the system is in distinction to some other types of software, where the 50 percent the software must be completed and 50 percent of the software must be function properly.

First example, "space probe" is invented to travel the distances by or to other planets and to send the data back about them. In space probe, many components which perform low level functions such as "landing, targeting, deploying sensors, communications and deploying solar panels".

This "space probe" will be ineffective, in case any of the mechanism is not working properly or missing. Second example, cell phone has all associate features such as "cellular base station selection, the user interface, and the communications protocols". These features have vital aspects to transfer the audio data among the precise remote nodes and user. In comparison, software's such as "desktop tools, web services in which low level of responsibilities, contribute as independently to the aggregate system functionality". The components of embedded systems are much distinct and combined into a "monolithic functionality". Embedded systems will combine software components for low rank device driver I/O, signal processing direction, control and communications protocols. Each component requires different skill set (Shih, 2014).

- **Development Tools Limitations**

Unlike enterprise system software development environments that enjoy rich development tools, embedded counterparts often use basic editors and compilers in many development environments because embedded systems use custom hardware from different vendors, which have limited tool support. Low-level software and hardware interactive debugging tools (Deng, 2014). Some software regimes have tools to support the development process. Software development of embedded systems is limited and use only basic compiler tools. Embedded systems uses custom hardware, which may not contain tool support and through this embedded systems perform unnaturally and to make it difficult to freeze the "entire execution" context under the control of a debugger and the data which lies between the host-based tool and the embedded target. Many embedded systems create their own tools to use for testing and debugging, this is only because of the limited commercial tools (Dahlby, 2004).

- **Robustness Requirements**

Many systems are using in medical purposes, in ruthless environments and for operations which are critical. For that reason, necessities for "correct exception handling, reliability and mean time between failures" are usually more rigorous for embedded systems software as compared to other software. This translates into testing requirements and stringent development processes. Most embedded systems are subject to dictatorial necessities which decrease the liability rates by obligatory the process of extension, or at slightest specifies the records, which come with the software. Moreover, it is impossible to upgrade firmware of several embedded systems, which is very important to get necessitate to "get it right" in the system's primary commercial release (Dahlby, 2004).

- **Durability**

The software is being used for many years. It gives the better documentation as compared to the documentation of "source code" itself, to explain the embedded systems software (Shih, 2014).

4.2 How do Agile Methods Fit with Embedded Systems?

This part, explains agile method distinctiveness which are mentioned above and to explain the development process of software.

1. **Agile Aspect: Use Regular Rapid Cycles Which Create Executable Deliverables**

   - **Implications:**

Embedded system is known as monolithic system, it is difficult to reduce the functionality into smaller pieces. There must be new feature in the embedded system within a short cycle and this feature must be coarse. Thus, large amount of work is needed for the decomposition plans. Instead of focusing on disintegration, the cycle should be modified to fit the existing decomposition dimension. Extra long cycles are needed at the project's initiation along with the development of a "simulation and/or the OS infrastructure, and the hardware bring up". If a new feature is hastily adjusted in the ongoing cycle, it may be implemented inefficiently. This will lead to the refactoring of a new feature in future cycles. Thus, the approach, although being potentially effective, may present problems (Suomi, 2014).

Due to the cycle bloat the "executable deliverable" may turn out to make it inefficient. Secondly, after some instance the product needs optimization refactoring because product goes out of cycles. Finally, the product ended up with lack of efficiency instead of few exceptional "cycle hungry" features/functions. For instance, "debugging code" must be used and finish up with memory practice and dominating cycles of the product. This convention can easily be compiled and disabled, if it designed properly.

However, if system developed ad hoc, more refactoring work will be required to immobilize it. This "death by a thousand paper cuts" situation is very hard to stick with hindsight (Srinivasan, Dobrin, & Lundvist, 2009). In comparison, a "Big Design Up Front" can put a few conditions for which features should be applied in a efficient way from the beginning. To support the cycle or memory effectiveness the coding practices must be observed across the world. Whereas, "agile practices" disagree with up-front design and it should be reduced. Good practices are those which deal with unforeseen issue and volatile requirements. However, the requirements for embedded systems are more significant than the requirements in the general software. Significant requirements are more cut, dried, and frozen. By the use of embedded systems software, the development methods are needed to bring a change.

Even if an embedded system perform efficient and convey useful performance in minute pieces to make a customer release on each iteration, where the "complexity, size and reliability" are sufficient large to form momentous system. The development process will give great benefit if these system tests can be automated. "Running load tests" requires additional analysis of hardware which is much expensive. Hence, the better rule for embedded systems is only to "invoke the overhead of extensive system for operations and commercial release qualification".

Agile aspect benefit for embedded systems: neutral

2. Agile Aspect: Focus on Coding Rather than Planning or Documentation

- **Implications:**

In software development there exists a tension between quality, cost, and time. Delivering cost competitive quality software in today's time constrained market is a difficult task. Many traditional software processes are top heavy with documentation and rigid control mechanisms making it difficult applying them to different software projects. New families of processes, referred to as Agile Processes, are making headway into the software industry. These processes focus on code rather than documentation calling themselves agile because, unlike the traditional processes, they are adaptable (Deng, 2014)

Placing more focus on coding and less on framework and documentation has several enormous benefits. Initial cycles and memory usage metrics are brought into consideration. Taking measurements beforehand is not a good indicator of complete product but has an edge over the rough calculations. Planning a project accurately and detecting errors at an early stage are the outcome of memory usage projections and having cycles. If the projections of the cycle are not in line with the sample processors capability then another prototype with robust processor and a bigger memory can be produced. Attached with this is the condition that the original prototype should posses the utmost capability.

If it is not desirable to launch second spin of the sample you will know beforehand the time needed for optimization. Design and documentation hold immense importance in embedded system. The advantages of timely feedback need to be measured against the design and documentation of embedded systems. “Upfront design” is very significant due to obligatory limits that how much this design can easily be downstream. As compared to other software, deprived designs in a product are leisurely but still it functions. These systems have deadlines and these deadlines must be authentic and if this system is leisurely in function than it is ineffective (Abrahamsson, Warsta, Siponen, & Ronkainen, 2003)

Embedded systems gives importance on its functionality and portability, its portability can be compromised and this makes the first preference of “operating systems, processors and compiler”. This system requires well design, portability and performance to change the lifetime of product. The partition of work among processor cannot be changed, so that’s why it is vital to design the “mapping of work and tasks up front”.

Therefore, the design of product applies additional interest in the systems. There is an extra requirement for documenting in an embedded system. Software’s can rely on self-documenting codes but this does not hold true for embedded systems since they want to have an optimized code. Moreover, embedded systems have a long life which makes it difficult for the original developer to be there any time in the problem in the system arises. Original developer won’t be available during the time the product is in the maintenance period. There are certain regulatory laws to which the embedded systems are exposed in respect to documentation.

Although the agile methods aid in the production of different artifacts but according to the agile viewpoint these artifacts may not be of much importance since this view states that the software must be well documenting. Moreover, the software must be written down first and then the next step must be creating the software. Working software is the irresistible goal that is the very basic and one dimensional. The foremost deliverable of a software development project is the working system, however, there are some more deliverables which are secondary in nature and they must be taken into consideration. Some of the secondary goals are stated below (Lindvall, Muthing, Dagnino, Walliam, Kiefer, & Stupperich, 2004)

4.2.1 Objective Proof to Ensure That the Software Works Appropriately

Some examples of the artifacts are "requirements traceability, records of test coverage and records of test and source code quality metrics". These "artifacts" can be needed to ensure that the software systems work appropriately. There are several tools inculcated in the agile process. These tools will help to generate the documentation for user automatically. The focus of the "working software" must not preclude documentation. This needs to be made sure (Dahlby, 2004).

(a) To Make Sure That the Working of the Software Is Smooth and Free From Errors in the Upcoming Period

In order to advise the maintainers of code so that they can comprehend the complex structure of the underlying system more artifacts which include architecture documentation may be required. Source code quality metric, decision relating the designs, and design documentation aids the code maintainers to understand the level of system. Most of the customers are unaware of the arrangement about future commitments; therefore customers may not demand the artifacts. Irrespective of the fact, those customers directly ask for documentation or not it is the professional duty of the developer to consummate it for the complex program (Dahlby, 2004).

(b) User Documentation

Agile methods are capable at establishing fine usability in the software. The usability of the software continues afar from the software. Antiquity of the documentation done by user include a number of things such as errata list, guide needed for installation, service provided to customers, summary, and a guide of user giving him directions. In many cases customers will decidedly ask for it but in several cases customers will be unaware of it unless the issue arises. The agile process give rise to the production of artifacts, extracted from the existing occupied software, however this process needs to be managed with care. Agile philosophy states that occupied software are really important however user documentation denies it (Dahlby, 2004).

Agile aspect benefit for embedded systems: neutral

3. Agile Aspect: Refactor Continually To Improve Code

- **Implications**:

Refactoring practices for design and code, this strategy identifies Process smells and targets the worst of them with specific agile practices drawn from several popular agile processes.

In order to improve the modules of code it is a common practice to refactoring. However, refactoring is not that easy. One of the issues of refactoring is to choose the software module to improve the iteration.

Furthermore, it is difficult to refactoring the chain of tightly coupled modules. Moreover, when refactoring adopts a substantive approach and do not indulge in religion related wars. More focus needs to be placed on management so that working can improve et cetera. Several benefits of refactoring are that refactoring minimizes “entropy growth” of the aging software system.

The coding knowledge can be improved along with the style. This also leads to similarity for different developers in the software module. It is observed that refactoring embedded system is much more challenging that refactoring the development of software regime. Firstly, Stressing on performance as compared to clarity hinders, refactoring to comprehend and modernize old codes. Fortunately documented information is available. This together with good architecture helps to refactoring. The quality of architecture is compromised by agile methods. Even though if there is a good architecture available that takes into account adequate coding, it may be a gross let down due to poor documentation. Secondly, the cost of modifying previously constructed architectural plans might be too high, hence weighing down the idea to change it. Example of such is a compiled work by several professors. In such cases, refactoring may not be advantageous. In fact, it is more effective to make a large scale improvement in the design than small scale localized changes. As a generalized rule, an initial expert guess might just be an excellent answer key to several costly problems. Thirdly, the tremendous increase in software specializations and its sub-branches has made refactoring ones code by other just impractical.

It is not possible for any one individual to excel in all the fields of software specialization, hence, expertise of an individual of that particular field may be required to master a particular code/set of instructions. Thus, the idea of an all-in-one programmer may sound ridiculous.

Code ownership may sound a promising idea but it is also immensely important to take into account the level of individual skillfulness, which may limit the whole deal. One successful strategy may be pair programming where a poor programmer’s work may be reviewed by a more skilled person. In summary bringing changes to an original design is always a good idea unless the degree of alteration is limited and expense is bearable. And in process of going through the code repetitively, limited error and maximum optimized work is ensured (Dahlby, 2004).

Agile aspect benefit for embedded systems: beneficial

4. Agile Aspect: Communicate Continually And Extensively Within The Engineering Development Team

- **Implications:**

According to the developers, “agile system” needs to exchange a few words personally instead of using other modes of communication such as messaging or documentation. Code reviews and pair programming are one form of personal communication. The personal communication among the developers is enhanced by embedded systems that produce obfuscated codes, thus leading towards the payment of high dividend. Personal communication through embedded systems tends to pose complication that is the availability of hardware. There is a shortage of supply of hardware systems due to their high cost and fragility. The reliability testing of hardware and integration of hardware and software are in competition with the development of software. The prototype systems that are functional must be fully utilized. In order to achieve this objective developer need to work like a tag team. This means that one developer needs to work the entire day(early schedule) and then hand over the work to the second developer who will work on it overnight(late schedule).

Agile teams consist of multi-skilled individuals. The development teams also have on-site customers with substantial domain knowledge to help them better understand the requirements. Multiple short development cycles also enable teams to accommodate request for change and provide the opportunity to discover emerging requirements. The agile approach promotes micro-project plans to help determine more accurate scheduling delivery commitments (Dahlby, 2004).

This practice will enable the development team to overcome the hardware shortage but there must be expense of face to face time among developers. Furthermore, there is an additional shortcoming of depending lying on personal communication that is lack of proper records for the developers who have to sustain the system. Moreover, system maintainers often find the documentation compiled for communication among real owners invaluable. Therefore, in individual communication the need of appropriate documentation is needed by potential maintainers. The difficulty to make the code self-documenting and the long term nature of embedded systems poses a problem. There are several advantages of in person communication. In person communication is very useful if it does not create a problem in documentation on the embedded systems to be used by future developers and maintainers or the system. (Suomi, 2014) (Srinivasan, Dobrin, & Lundvist, 2009).

Agile aspect benefit for embedded systems: beneficial

5. Agile Aspect: Communicate Continually And Extensively With Customers

- **Implications**:

Encouraging the communication between customers and the team who has developed the system is little rewarding since most of the embedded system software are not seen by customers. The stories such as "replace Gaussian elimination with L-U decomposition", "hand-code the inner loop of foot in assembly", and "add in-band control feedback to the channel X data stream" are of little use to customers with the intention of selecting stories for the upcoming iterations. Since these decisions are hard to make it would be more feasible to provide the real customer with all the necessary information to help him make accurate decisions. It would be suitable to set up a "tech-savvy marketing team/ a business-savvy engineering management team" which will act as an alternative for the actual customers and aid them in deciding the features which are important in the long run as well as in near time. It will also help customers in choosing the optimal development path for the iterations. Engineering team must possess business skills and extraordinary technical skills for embedded systems. The team should use proactive approach thereby utilizing their skills when needed, setting out course for the underlying project and correcting themselves continually.

Using customer proxy for the embedded systems to deal with customers, who are not educated, is called a counter note. Under such circumstance it would be wise to use agile capability to reinvent the ways of developing system and evaluating the systems in order to cater for difference between two parties that are the real customers and alternative (Dahlby, 2004).

Agile aspect benefit for embedded systems: unbeneficial

6. Agile Aspect: Continual Measurements, Planning, Projections, and Adjustments by Management

- **Implications**:

Performance constraints are faced by embedded system. Furthermore, they also face time constraint, budget constraint as well as staffing issues. These problems are common to all software developers. It is very important to manage the delivery date of project, staff issues as well as the feature set. Moreover, it is also very essential to obtain a timely feedback on actual performance of the underlying partial system which will help mangers to decide whether and when there is a need for performance optimizations. The projections made by mangers must take into account that the work of the embedded systems is not so predictable. There are several reasons contributing towards this variation. They include unusual timings to find bugs and the need for performance optimizations.

The bugs found in the embedded system can cause a lot of disruption due to the interaction and several race conditions in a real time system when multitasking that is to perform several tasks at a time. It has been debated that the application of debugging methods in an embedded system is hard. Statistical prediction and measurement of effort required in debugging is one of the best means to cater for variation. Debugging, integration with coding, combine testing and code development are all part of agile method thereby facilitating in making predictions. Brook conundrum states that including new developers to a project will make the project delayed because of (a) the size and (b) its complexity. Agile methods provide quick and timely feedback which helps the managers analyze if there is need to hire extra staff. Advantage is that managers can analyze the impact of hiring staff and increment in the manufacture rate (Dahlby, 2004).

A*gile aspect benefit for embedded systems: beneficial*

7. **Agile Aspect: Test-Driven Development And Regression Testing**
   - **Implications**:

Agile method focuses on testing the system at beginning stages in order to identify a bug to rectify as soon as possible. This practice is considered beneficial universally as the expense incurred on removing the bug grows exponentially once the entire system is developed and then the bug is found. It is simpler to identify the bug at unit level and remove it rather than waiting for the entire system to complete and then debugging from the whole system. Due to various reasons it has become difficult to debug an embedded system. The codes and log used for debugging may be limited my memory constraints. Using breakpoints to make the system stop working or system execution through single step are not possible due to time shortage and performance of multiple tasks at a time. Inserting the code used for debugging at a different time may conceal the actual problem (bug) that needed to be solved (removed). There are two types of processors. One of them is the target processor which is being debugged while the host processor is the one on which the commands used for debugging are operated. A unique type of hardware is required to deliver the commandments needed for removing bugs along with transfer of data between these two types of processors. In order to debug and test a system a unique system must be developed resulting in access contention. Systems used for testing may be needed to uncover bugs. Even if a company has a highly efficient testing mechanism debugging will be required when it comes to amalgamation of the components of software and unification of hardware and software. Unit testing as well as additional integration tools may remove many bugs. A stimulator can be developed for the embedded system to help in debugging and consolidation of hardware and software (Douglass, 1997) (Thüm, Kastner, Bendun, Meinicke, Saake, & Leich, 2014). In order to use a stimulator you first need to design stimulator framework (especially when within a processor multiple task are being controlled by the stimulator, when there are several systems and within a system there are more than one processors), hardware abstraction interface needs to be designed in order for the stimulator framework to correspond with the application, to organize software for target and real stimulator, and to deal with the speed issue of the stimulator. However, there are several advantages provided by the stimulator. First of all it provides attractive returns on the investment made by disuniting the hardware bring up and the development of software. It also gives the facility of removing bugs which is convenient. Furthermore, stimulator encourages the testing of software on a grand scale. Stimulators also provide the advantage of load testing which may not be available in physical systems.

It is difficult to perform regression testing for embedded systems. Specialized test harness and the equipment needed for testing are difficult to find. It is also very difficult to examine the internal state of system to rectify wrong and certify the right. Test system shortages also pose a problem. It is advisable to perform the regression testing on the stimulator and real systems desirably from a framework that combines the execution of test and the logging of test record for both the real and stimulated systems. Therefore, despite all the hurdles faced in the testing and removing bugs in system it is feasible to use the "agile method" by experimenting the system and to identify the problems in a timely fashion so that the wrong can be rectified early within the scheduled time (Eklund, Olsson, & Strom, industrial challenges of scaling agile in mass-product embedded systems, 2014). Running the tests over and over again gives you confidence that the new work just added to the system didn't break or destabilize anything that used to work and that the new code does what it is supposed to do. Running the tests over and over (particularly acceptance tests) can also help you understand what portion of the desired functionality has been implemented. Together the set of automated tests can form a regression test suite. Regression testing is selective retesting of a system or component to verify that modifications have not caused unintended effects and that the system or component still complies with its specified requirements (Dabney, 2004).

Agile aspect benefit for embedded systems: beneficial

5 Chapter five

An Agile Development Methodology Applied to Embedded Control Software under Stringent Hardware Constraints

5.1 Background

In the present era, it has been seen that people are trying to rely on micro-controllers for their work related to computers. There are applications where many of the people are able to sell the microcontrollers which are mainly of different bit size, primarily, 4-, 8- and 16-bits (Koopman, 2006).

Approximately all the people have been processing with some wide range of machine systems which are monitored with a better controlling development. As there is increase in the complexity, the development is also affected which also has a great impact on the different developing methodologies which are being worked upon. The development is mainly applies in order to take care of the size of the team as well as meeting the demands of the projects. The scope and the requirement is generally based on the constraints of the project which will develop and control all the discredited controlling systems for the software to run properly on the PC's (Personal Computers). It is appropriate to develop and work with the functioning of the software which is developed depending upon sharing different characteristics based on the ideas related to real time issues.

At a certain age, it becomes important that the applications have a great algorithm which follows the output as a result. With the control over the different complexities, the system is able to share the most embedded systems to get run on all the hardware and the software devices. It becomes important to reach dedicated software which will be able to reach a common based system approach leading to better energy consumption as well as managing the time of execution. With the different footprints, one has the ability to match all the risks and critically manage with all the failures which try to affect the embedded control over the systems.

Hence, it becomes important to plan all the technologies and try to match with all the contextual based investments which would not lead to project failure. As per the contextual part, there is a development mainly on the different methodology like the TXM which is named as the next Methodology. Depending upon the agile principles, it mainly work for a better planning and flexible approaches along with interaction, approach for incremental development too. The embedded software are generally to compose all the practices which comes under the part of Software Engineering and the Agile Methods (Scrums and the XP) aim for a composed approach for minimization of all the problem set which could lead to different development in the software context. With the requirement to achieve all the volatility and manage the risk, it is important to practice and achieve some of the hardware as well as (Sangiovanni-Vincentelli & Martin, 2010).

Software developments primarily based on the design of platform based development. The goal is mainly to focus on certain features which will have a better impact on the designing methodology and lead to changes in the software along with embedding with better technologies as well. Some of the major works that have been proposed in these times are:

- There is a flexible approach which has a better trade off and performance development to adapt to different changes done under high platform programs.
- With the different processes under it, there are practices for the development of the embedded software which are under main constraint problems.
- A support to software always drives the hardware and approach towards a flow which could have better specifications for implementations.
- The techniques are handled under the proposal which would be in combination and has never been able to analyze it before.
- The iterative approaches and the incremental sets are there to offer all the process to the designer where one can validate all the specifications and try to manage with the process.
- The experimentation on the results depends on the applications which have a better proposal to the strategies as well as control over the systems.

5.2 A Brief Look at the Agile Methods and Patterns

The below are the details over all the principles and the methods which have been proposed to manage the presentation and identify approaches to the product development and its management on all grounds of practices.

5.2.1 Extreme Programming

With the most recognizable methods, it is one of the important methods which focus on how to manage and orient communication along with team orientation. (Sangiovanni-Vincentelli & Martin, 2010). The XP is generally composed of those 12 practices related to core which have been integrated and composed under those methodology which have the features relating to:

- The process of continuous changing generally has a way that it is able to alter all the behavior, thereby, trying to manage with the codes along with improvement in the structure as well as integration process.
- The code has been compiled to be tested under the process which is checked every time the work is done.
- In the test development driving process, one can estimate that they are mainly written for the developers who want coding as their major goal and the units are automated under the system which completely affect the functionality of the code piece.
- One has the project which needs to follow all the code style which is for maintaining the standard practicing and performs consistent formatting for the source code which is readable with the programming language worked upon.

The specifications of XP mainly promote an approach which has a way to choose all the designing systems along with describing all the main benefits which could help to increase and aim to reduce all the risk of the project and lead to uncertainty at an early stage.

5.2.2 Scrum

It is one of the simple approaches which is able to manage the process of the software development process and has been worked upon both the environmental as well as technical approaches for different variables depending on the process.(Schwaber & Mike, 2002). The Scrum is mainly composed of 14 practices which are under a consistent feature which could integrate and manage the proposal of better methodologies which mainly include:

- There is some iteration in the organization of the calendar of 30 days.
- The planning practice by sprint consisted of two meeting which contains the list of all the features and the backlog stage. The use case are the best to enhance and promote a better refinery for the system approach. This would generally reprioritize the ownership of the product and help the stakeholders for the next iteration process. As per the second meeting agenda, there are certain Scrum figures which are able to achieve all the requests that have been made to create a better backlog which contains mainly a detailed task which needs to be accomplished in the current iterative process.
- The practice of Sprint reviewing is mainly that team which presents all the results that have been obtained under the main iteration which shows the work of the software and the owner of the product and customers, stakeholders.
- In the practicing of daily scrum, there are certain meetings which are being held up with time for answering some of the important questions which are answered by the team of Scrum people. They employ all those process which are for process to control the model

and try those which aim to inspect all the conditions which have the activities which empirically determine what needs to done higher in the next order.

In order to produce a better expected result, there is a great productivity which strongly depends on mainly the skills and the motivation of the people and how they are involved in the process. With the different sections, section 4 is mainly showing how the practices for Scrum came into existence and they were adapted for a better proposed technology.

5.2.3 Patterns for Agile Software Development

The agile patterns represented generally is (Coplien & Harrison, 2005) combined with different patterns of XP and thereafter one could start working in a way to achieve the development of the code for the organizations. These patterns are split generally into categories which are completely different and provide a better collection of different patterns, overall.

With some better languages, one is able to provide a collection which could facilitate the organization and develop the merchandise to follow all the patterns to clarify the needs as well as coordinate the activities of the project which mainly depend on the development and the clarification of the merchandise needs. With the clarification and managing the project activities, one could generate a system build which could develop and clarify all the needs to coordinate the different activities of the project, thereby, trying to generate a better system built up in order to clear and concentrate on the primary goals of the team. With the development of the languages, it becomes important for the organization to outline and try to manage with the high-level approaches which can quantifies all the details as per the projects and is able to guarantee that the client would be satisfied with the communication needs and the system too. The major aim is to focus on a better vision which has a concerned development related to all the team work which would build up a better structure as well as provide to a collection which could facilitate the overhead of the project and realize that the collection is mainly for compatible reasons and able to withhold different merchandise styling.

The latency is to work and focus on the different patterns of the code where the individual is able to provide with different collection in the organization to keep a better approach and make assurance that the implementation is done materially. The major features for the non-trivial solution comes after configuring all the integrated structured patterns which would have better impacts on the planning methodology as well as on their integrations. The patterns are generally outlines to supply a better facilitation to the different mechanism teams which are able to work on different versions completely and try to re-define all the difference codes which are able to parallel manage of all the link ups and the development stages. The code has been formed up to determine how the projects are there to manage the main and all the important parts.

5.3 Proposed Development Methodology

The proposal of the methodology generally looks forward for all the aims which could re-define all the major possibilities where the role is to manage the lifecycle and all the tools which could be under the employed way to manage the system projects. There are different processes which could be categorized under the process of developing a system namely: system platform, product development and management. With the different platforms, one is able to aim and work on different products where the user/designer could aim for better component and lead to a better architecture approach. The platforms for API are mainly to maintain the library which could manage the system approach and the parts are generally chosen to lead to better encounter of the architecture and the platforms. There is one of the major possibility that the architecture needs to meet all the major constraints which need a customization process to be carried out. The configuration of the logical approach is mainly to integrate and run all the successive implements which would have a better way out to all the programmable designs and other methodologies. The product development mainly tries to offer all those practices which could help in developing better applications as well as helping integration on a different platform level. With the major functionalities rising up, the products are generally partitioned under those areas which could manage all the tasks and lead to a better consumption in energy as well as execution time. With the intact memory size for all the major application components to be worded upon. The techniques are majorly applies in the incremental and the iterative matter and one has to manage all the scope of the paper as per the mechanical design. The parameters are calculated and managed under the different group processes which are able to control and mainly influence the different platform systems. When there is an initiation of the project, it is mainly worked upon the major developments which could lead to a start-up and then process is continues to manage the plan. The corrective answers are carries in the sense which could aim for tracking the project which would assure the different parameters of the project. With the development of different processes, one is able to group and practice all the Scrum methods which are able to design the agile patterns and describe better promotion (Schwaber & Mike, 2002) . To the next subsections, they mainly emphasize on description of the groups processes which are held with different and major roles and responsibilities, and the processes lifecycle of the proposed methodology

5.3.1 System Platform Processes Group

With the different compose of the process, the system groups are generally focused on the major requirement of the product as well as different system platform, managing the product line, and trying to optimize the system. With all the requirements, the process tries to aim for major requirements of the system which are generally categorized under the functional as well as non-functional categories and look forward which are relevant enough to determine that the system platform which is being built for the product is proper or not. The platform instance is that process which tries to help in better development of the team, defining the system platform by making use of a set to design all the tools and benchmarks which are essential for the better platform support of the group and the process.

To define the platform, it becomes important for the product to develop and step a position which could help in categorizing the process, allowing developing a team to integrate and implement all the major functionalities which could be helpful for a better release of the system. The major versions of the different products are mainly for the product which is developed and worked under the different functionalities of the system to help and process the development line. The optimization process provides a great set-up which could optimize and make the assurance of the different variance of the system products as well as better consumption of energy. The program size and the memory size are mainly enough constraints to outline the development of the product.

5.3.2 Product Development Processes Group

The product development processes group generally is composed of some of the major following processes:

- The implementation of the different functionality implementation
- The integration of the different task processes which could refactor the system approach
- The optimization of the system which could handle all the functionality as well as process which could lead to different test case which are in process.

The quality of the product is determined by a continued test which could create and manage with all the functions complex or simple. One could create all the testing mainly which could apply towards validation of different layers of API. With the creation of different functions, one need to properly manage with the different applications of the software which try to integrate and the task has the process to suit all the implementation functionalities as well as complex functionalities too.

The integration of the task generally means to integrate all the new and major implementation which could develop into a better line of product and have the drive to force all other team members to work with same zeal. The process of refactoring mainly focused on the development where the major opportunity is to improve all the coding part and change it without any major alterations in the behavior related to the external matters. When the code is refactored, one could find that the development team is able to optimize it and make it useful for the teams to work for monitoring support as well as profiling them to understand the major instance too (Junior, Neto, Maciel, Lima, & Rib, 2006). Once needs to manage the energy and the memory space which has a better appearance and could meet the software needs too according to the constraints in the system.

5.3.3 Product Management Processes Group

The product management processes cluster consists of all the subsequent processes:

- product needs
- project management
- bug following
- sprint needs
- wares
- implementation priority

The methods which merchandise needs are mainly those which belong conjointly to those systems which are platform dependent process groups. Generally the standards aims to meet all the requirements which are important for the major functional and non-functional requirements and which have a major part to manage and permit the events which could implement a better team of work related to the system needs as well as managing the different blocks of merchandise. The backlogs and the activities which are coordinated are under the different build and are majorly implemented depending upon some bugs where the project could generate a better system built and lead to a lifecycle which has been properly managed and led to a resourceful lifecycle of different problems of the project.

To reach and correct the bugs, there are tasks to enhance the activities and supply all the required information which would support through some of the major qualities, thereby, investigating all the methods which are important to be evaluated for begging a project sprinting. This method conjointly helps the event team to unharness new product versions into the market. The implementation priority method helps the merchandise leader manage any quite interruptions which will impact the project's goals. This method guarantees that the project's tasks square measure one hundred pc completed once initiated.

5.3.4 Roles and Responsibilities

The projected methodology involves four totally different roles and also the responsibility of every role is represented as follows: Platform Owner: Platform owner is that the one who is formally accountable for the product that derive from a given platform. This person is accountable for shaping quality, schedule and prices targets of the merchandise. He/she should additionally produce and rank the merchandise backlog, opt for the goals for the sprints, and review the merchandise with the stakeholders. Product Leader: Product leader is accountable for the implementation, integration and take a look at of the merchandise guaranteeing that quality, schedule, and value targets outlined by the platform owner square measure met.

He/she is additionally accountable for mediating between management and development team additionally as taking note of progress and removes block points. Feature Leader: Feature leader is accountable for managing, dominant and coordinative scheme comes, pre integration comes, external suppliers that contribute to an outlined set of options. The feature leader additionally tracks the progress and standing of the feature development (deliverables, integration and take a look at standing, defects, and alter requests) and reports the standing to the merchandise leader. Development Team: the event team which can accommodate programmers, architects, and testers square measure accountable for functioning on the merchandise development.

They need the authority to form any selections, do no matter is important to try and do (according to the project's guidelines), and raise any block points to be removed. If the merchandise to be developed is little, i.e. it\'s composed of few elements (less than fifty KLOC) and doesn't need different development groups to implement the merchandise's functionalities then one product leader and also the development team square measure enough for the product development. On the opposite hand, if the merchandise consists by many elements (more than fifty KLOC) and needs different development groups to implement the product's functionalities then the Feature Leader role should be concerned within the processes. During this context, one product leader needs feature leaders to manage, management and coordinate components' comes. Therefore, for medium and bigger comes, one product leader and several other feature leaders and development groups could also be concerned within the processes.

5.3.5 Processes Lifecycle

The projected agile methodology consists of 5 phases:

- Exploration
- Planning
- Development
- Release
- Maintenance.

With better sections to explore, there are certain necessities which could primarily be held for unleashing and manage all the enclosed necessities for the merchandise to backlog and own the platform.

The estimate that one wants with no item larger than three person-days of effort. During this section, the event team identifies the platform and application constraints and estimates the system's metrics supported the merchandise backlog things. With this info at hand, the event team is able to ready to outline the system platform which will be wont to develop the merchandise within the next phases. Within the coming up with section, the platform owner and customers establish a better estimate for the decomposition of the sprint tasks into many other backlogs which could take the form of different styles as well as prototypes. One needs to clarify all the necessities which could help in managing the different teams to help them clarify the important needs of a system.

Within the development section, the team members implement new functionalities and enhance the system supported the things of the sprint backlog. The daily conferences are command at identical time and place with the aim of observance and adapting the activities to supply the specified outcomes.

At the top of the every iteration, major units are additionally to improve the sections which would be able to manage all the software which could maintain a place which has a better place to be used in. Throughout this section, it always involves the major identification of errors which is able to improve the different services of the customer, thereby, unleashing all the important changes which could deliver and aim for the required documentation. This section aims to deliver the discharge product and required documentation to the client. the upkeep section might also need additional sprints so as to implement new options, improvement and bug fixes raised within the unleash section. Consecutive subsections describe solely a set of processes of the projected methodology that focuses on achieving the aims of the embedded management systems.

6 Chapters Six

Discussions and Conclusions

6.1 Discussion

This study mainly addresses the major components of the engineering studies namely:

- Software engineering
- Agile development
- Embedded system

The study has given a chance to come over all the solution and usage of the agile methods which provide better solutions to the researches. The chapters are mainly for the software engineering which could help in modeling the entire major difference and trying to figure out all the advantages which will have a major impact on the pictures of the embedded systems and which look forward for the systems operating in real time operating systems (RQ1).

After that we discussed all important issues for embedded system as (trends, design issues, development tool, real-time system), as (RQ2) , there is a proposal for mapping all the embedded systems which will have an impact on the process of the software and one is able to know all the methods related to different agile methods development of embedded systems (RQ3). We proposed all challenges which have a successful reach to all the development in the software as well as implementing all those optimization studies which lead to development too as (complexity, optimizations, interdependency, verification and tools) then studied the impact and solution properties for each challenges (RQ4).

After that we pointed for how well agile methods fit embedded systems. Certain proposals are there for the different aspects of agile which could benefit with the better system (RQ5) and ne needs to develop different methodologies for the same to enhance a stringent approach for different constraints on the hardware (RQ6).

In our thesis, we have been able to manage and analyze how the different method of agile are able to be implemented in the embedded software and systems (RQ3) which are found in the diversified methods as well as different developments. One needs to mainly emphasize on those points which could lead to a better characterization also with different viewpoints.

The process of beneficiation (RQ4) has been mainly pointed which could guide all the characteristics for embedded system which would help in measuring all the important beneficiaries in the process with different agile process schedules. There are different multiple development which have been proposed to suit and manage the constraints leading to the embedded domains which have their different ideas for scaling the embedded systems and the methods of the agile approach. The third research question was whether the agile methods are suitable for the development of embedded systems and embedded software. For example (Ronkainen & Abrahamsson, 2003) lay out requirements which could be addresses and are under the major development of the product which would substantially help in differing with what the agile products are originally about.

There is a substantial change which is targeted to bring a major change in the development, leading to the target which mainly needs to meet the demands where the development and the requirements were to point all the embedded systems and target towards a support which would enhance the different roles for better up gradation, confronting and pointing to all those characteristics which are able to support all the important architecture and also the different up-front designs. The requirements of the techniques are mainly to manage the different amount which is able to document and specify that the requirement s is important to manage all the documentation as well as architecture.

For agile methods that need to be addressed when used in embedded product development. The characteristics of embedded product substantially differ from what agile was originally targeted for. Meeting real-time requirements of embedded systems is pointed out to be the most important difference that new agile methods should be able to support. In embedded systems, the major confront comes when the designs are avoided and lead to finding better techniques which could roll into the account and lead towards a better and suitable amount of documentation and specification. Furthermore, it is pointed out by Ronkainen and Abrahamsson that top level documentation is important as there are different stakeholders who are indulged in the working and the coordination of various research methodologies. The main view is to analyze the support which is given for a better development of the product which could embed and lead to the major support by (Drobk, Noftz, & Raghu, 2004), (Gul, Sekerci, Yücetürk, & Yildirim, 2005). In (Punkka, 2013) a document-driven development approach is proposed where the major importance and focus is on the real time systems which could outline and shine over the major and effect sharing of information, thereby, realizing that there is a need to maintain the coherent ideas which would be able to solve all the problems.

Our thesis has been achieved with better goals and objectives which are able to solve and minimize the problems statement questions, gaining a better coordination and relationship between the processes of better methods for Agile. The inputs are very helpful to analyze what is better for pointing towards changes and using all agile methods in embedded system and pointed of the Agile Development Methodology Applied to Embedded Control Software under Stringent Hardware Constraints.

6.2 Future Work and Recommendations

Embedded software development is generally going to remain always challenging domain where there are going to be assured developments which could lead to the different technologies and methodological approaches. One has to yield a better benefit which would sustain a development and benefit from the different approach which is certain to bring success in all the upcoming years to develop and manage all the success in the adoption for the better designing process. It is important to develop the different process of the software which has a major effect on the adoption on the agile methods leading to a contextual approach and development of systems namely in the:

- technical issues(requirements, and testing);

Organizational issues (process tailoring, knowledge sharing & transfer, culture change, and support infrastructure development).

7 References

Abrahamsson, P., Warsta, J., Siponen, M. T., & Ronkainen, J. (2003). New directions on agile methods : acomparative analysis . *international conference on software engineering 25th* (pp. 244-254). IEEE.

Apvrille, L., & Roudier, Y. (2014). towards the model- driven engineering ofsecure safe embedded systems . *GraMSc* , 15-30.

Baynes, K., Collins, C., Fiterman, E., Ganesh, B., Kohout, P., Smit, C., et al. (2001). the performance and energy consumption of three embedded real-time operating systems. (pp. 203-210). in proceeding of the 2001 international conference of compilers architecture and analysis for embedded systems ACM.

Caudrado, J. S., Canovaslzqierdo, J. L., & Molina, J. G. (2014). applying model- drivern engineering in small software interprises. *science of computer programming , 89*, 176-198.

Chhya, A. S. (2008). A new process model for embedded systems control for automotive industry . *proceeding of the 2008 international arab conference on information technology* , (pp. 16-18). Tunisia .

Coplien, J., & Harrison, N. (2005). organizational patterns of agile software development. 171. Upper Saddle River: Pearson Prentice Hall.

Cordeiro, L., Mar, C., Valentin, E., Cruz, F., Patrick, D., Barreto, R., et al. (2008). An agile development methodology appliued to embedded controlsoftware under stringent hardeware constraint. *ACM SIGSOFT software engineering notes , 33* (1), 5.

Crnkovic, I., & Larsson, M. H. (2002). *building reliable component-based software system.* upper saddle riverpper : prentic Hall.

Dabney, J. (2004). *return on investment for independent verification and validation-phase 2B final reort.* NASA.

Dahlby, D. (2004). Applying agile methods to embedded systems development. *Embedded Software Design Resources* , 41 (2004): 1014123.

Deng, D. (2014). Reliable Embedded Systems Development. *master thesis* . university of calgary, department of electrical and computer engineering.

Douglass, B. P. (1997). *Real-time UML: developing efficient objects for embedded systems.* Addison-Wesley Longman Publishing Co., Inc..

Drobk, J., Noftz, D., & Raghu, R. (2004). Piloting XP on Four Mission-Critical Projects. *IEEE Software , 23* (6), 70-75.

Dyba, T., & Dingsoyr, T. (2008). Empirical Studies of agile software development: systematic review. *information and software technology , 50* (9-10), 833-859.

Eklund, U., & etal. (2014). *Industrial challenges of scaling agile in mass-produced embedded systems Agile Methods.Large-Scale Development, Refactoring, Testing, and Estimation.* Springer International Publishing..

Eklund, U., Olsson, H. H., & Strom, N. J. (2014). industrial challenges of scaling agile in mass-product embedded systems. (pp. 30-42). springer international publishing.

El-far, I. K., & Whittaker, J. A. (2001). model-based software testing . Encyclopedia on software engineering Wailey.

Fernandes, J. M., & Machado, J. R. (2007). teaching embedded systems in systems engineering in a software oriented computing degree. *37th ASEE/IEEE frontiers in education research*, (pp. 27-36).

Francia, G. A. (2001). Embedded systems programming. *journal of computing science in colleges , 17* (2), 217-223.

Francisco Assis M. do Nascimento, M. F. (2006). ModES:Embedded Systems Design Methodology and Tools based on MDE. (pp. 67-76). Brazil: 07 proceedings of the fourth international workshop on Model-Based methodologies for pervasive and embedded software.

Gomaa, H. (2008). model-based software design of real-time embedded systems. *IJSE , 1* (1), 19-41.

Goswami, A., & Bezboruah, T. (2009). Design of an Embedded System for Monitoring and Controlling Temperature and Light. *International Journal of Electronics Engineering Research , 1* (1), 27-36.

Gul, E., Sekerci, T., Yücetürk, A., & Yildirim, C. (2005). Using XP in Telecommunication SoftwareDevelopment. *International Conference on Software Engineering Advances* (pp. 258–263.). The Third, Sliema, Malta.

Junior, M., Neto, N., Maciel, S., Lima, P., & Rib, R. (2006). Analyzing software performance and energy consumption of embedded systems by probabilistic modeling: An approach based on coloured petri nets. *In Petri Nets and Other Models of Concurrency-ICATPN* (pp. 261-281). Springer Berlin Heidelberg.

Kaisti, M. R. (2013). Agile methods for embedded systems development - a literature review and a mapping study. *EURASIP Journal on Embedded Systems 2013 , 1* (15).

Kaisti, M., Rantala, V., Mujunen, T., Hyrynsalmi, S., Konnola, K., Makila, T., et al. (2013). agile methods for embedded systems development-a literature review and amapping study. *EURASIP journal on embedded systems* (1), 1-16.

Kalinsky, D. (1999). *"A survey of task schedulers". In Embedded systems conference.* San Jose CA.

Khanjani, A. (2011). comparison between four software engineering approaches: component based software engineering, agile methods, aspect oriented and mash-up. *international journal of advances in computer science , 2* (4), 20-26.

Koopman, P. (2006). Embedded system design issues (the rest of the story). *Computer Design: VLSI in Computers and Processors* (pp. 310-317). ICCD'96. Proceedings., 1996 IEEE International Conference.

Koptez, H. (2000). Software Engineering for Real-Time:A Roadmap. *IEEE international conference on factory automation* (pp. 1557-1565). Austria: IEEE.

Laanti, P. K. (2006). How to steer an embedded software project: Tactics for scaling agile softrware processs moels . *IJAM , 9* (1), 59-77.

Larman, C. (2003). *Agile and iterative development: A manager guide.* Boston: Addison Wesley.

Lindvall, M., Muthing, D., Dagnino, A., Walliam, C., Kiefer, D., & Stupperich, M. (2004). Agile software development in large organizations. *Computer , 37* (12), 26-34.

media design automation Roadmap, 2205.Version 5.

mosterman, P. J. (2006). *network embedded systems.* Punkka: proceeding of beyond SCADA: cyber physical systems meeting pittsburgh.

Munassar, N. M., & Govardhan, A. (2010). *A comparision between five models of software engineering.* IJCS,93, Newness, Butterworth-Heinemann,Boston MA.

Noergaard, T. (2012). *Embedded systems architecture: a comprehensive guide for engineers and programmers.* Newnes.

Poulhies, M., Pulou, J., Rippert, C., & Sifakis, J. (2007). A methodology and supporting Tools for the development of component-based embedded systems. *in composition of embedded systems.scientific and industrial issues* , (pp. 75-96). springer berlin heidelberg.

Punkka, T. (2013). Agile hardware and co-design.

PWC. (2013). *Accelerating embedded software development via agile techniques;The nine strategies that lead to successful embedded software development.* Technology institute.

Rajawat, P., & Rajendra, P. (2011). A servuey of embedded software profiling methodologies. *international journals of embedded systems and applications , 1* (2), 19-40.

Ronkainen, J., & Abrahamsson, P. (2003). software development under stringent hardware constrait: do agile methode have a chance in . *4th International Conference on Extreme Programming and Agile Processes in Software Engineering*, (pp. 73–79).

Royce, W. (1970). *managing the development of large software systems: concept and techniques* . Los Angeles: in proceeding of the IEEE westcon.

Sangiovanni-Vincentelli, A., & Martin, G. (2010). Platform-based design and software design methodology for embedded systems. *IEEE Design & Test of Computers , 18* (6), 23-33.

Schwaber, K., & Mike, B. (2002). gilè Software Development with Scrum. *First Edition, Series in Agile Software Development, Prentice Hall.*

Sha, L., Abdelzaher, T., Arzen, K. E., Cervin, A., Baker, T., Burns, A., et al. (2004). Real time scheduling theory: A historical perespective. *Real-time systems , 28* (2-3), 101-155.

Shih, W. Y. (2014). *Embedded System Software Testing Process and Criteria: Application on Consumer Products of Networking and Communication Industry..*

Sommerville, I. (2004). Software engineering. *Spirit Consortium* .

Srinivasan, J., Dobrin, R., & Lundvist, K. (2009). State of the Art'in Using Agile Methods for Embedded Systems Development. *Computer Software and Applications Conference* (pp. Vol. 2, pp. 522-527). 33rd Annual IEEE International.

Stepner, D., Rajan, N., & Hui, D. (1999). Embedded application design using a real-time. *design automation conferece IEEE, 36*, pp. 151-156.

Suomi, S. (2014). Empirical study of agile software development: a systematic review . In T. Dyba, & T. Dingsoyer, *Project Management Tools in Agile Embedded Systems Development.* (pp. 9-10). Technol.

Szyperski, C. (2002). *Component Software: Beyond object oriented programming* . New York: press and addison wesley.

Thomas, S. W., Adam, B., Hassan, A. E., & Blostien, D. (2014). studying software evolution using topics models. *science of computer programming , 80*, 457-479.

Thüm, T., Kastner, C., Bendun, F., Meinicke, J., Saake, J., & Leich, T. (2014). Featureide: An extensible framework for feature-oriented software development. *Science of Computer Programming , 79*, 70-85.

Verma, J., Bansal, S., & Pandey, H. (2014). Develop framework for selecting best software development methodology. *international journal of science and engineering research , 5* (4).

Vijay, S. (2001). A Study of Real-Time Embedded Software Systems and Real-time Operating Systems. *master thesis* . Mumbai: Indian Institute of Technology.

www.agilemanifesto.org

West, D., Grant, T., Gerush, M., & Disilva, D. (2010). agile development: Mainstream adoption has change agility. *forrester research , 2*, 41.