A Framework for Efficient Delivery of Software Products

Thesis submitted in partial fulfilment of the requirements for the award of degree of

Master of Engineering

In

Computer Science and Engineering

Submitted By

Diksha Hooda

(801532012)

Under the supervision of:

Dr. Rinkle Rani

Associate Professor

COMPUTER SCIENCE AND ENGINEERING DEPARTMENT

THAPAR UNIVERSITY
PATIALA – 147004

JULY 2017

CERTIFICATE

I hereby certify that the work which is being presented in the thesis entitled, "*A Framework for Efficient Delivery of Software Products*", in partial fulfillment of the requirements for the award of degree of Master of Engineering in *Computer Science and Engineering* submitted in Computer Science and Engineering Department of Thapar University, Patiala, is an authentic record of my own work carried out under the supervision of Dr. Rinkle Rani and refers other researcher's work which are duly listed in the reference section.

The matter presented in the thesis has not been submitted for award of any other degree of this or any other University.

(Diksha Hooda)

This is to certify that the above statement made by the candidate is correct and true to the best of my knowledge.

(Dr. Rinkle Rani)
Associate Professor
Computer Science and Engineering
Department

ACKNOWLEDGEMENT

The successful completion of any task would be incomplete without acknowledging the people who made it possible and whose constant guidance and encouragement secured the success.

First of all I wish to acknowledge the benevolence of omnipotent God who gave me strength and courage to overcome all obstacles and showed me the silver lining in the dark clouds. With the profound sense of gratitude and heartiest regard, I express my sincere feelings of indebtedness to my guide **Dr. Rinkle Rani**, Associate Professor, Computer Science and Engineering Department, Thapar University for her positive attitude, constant encouragement, keen interest, invaluable cooperation, generous attitude and above all her blessings. She has been a source of inspiration for me, I am grateful to **Dr. Maninder Singh**, Head of Department and **Dr. Ashutosh Mishra**, P.G. Coordinator, Computer Science and Engineering Department, Thapar University for the motivation and inspiration for the completion of this thesis. I will be failing in my duty if I do not express my gratitude to **Dr. S. S. Bhatia**, Senior Professor and Dean of Academics Affairs in the University for making provisions of infrastructure such as library facilities, computer labs equipped with internet facility, immensely useful for the learners to equip themselves with latest in the field.

Later but not the least I would like to express my heartfelt thanks to my parents and my friends who with their thought provoking views, veracity and whole hearted co-operation helped me in doing this thesis.

Diksha Hooda
(801532012)

ABSTRACT

Software Delivery is about bringing a piece of software into the market. Software delivery accounts to a considerable premium to an organization. With the invention of internet, each day more and more number of people are getting connected to the internet. It is found that around 28% of the world population is connected to the internet. Greater number of people carry internet connected devices and gadgets with them with each passing day. The advent of internet has great impact upon the lives of the people around the world. A majority of online retail businesses rely on the internet connectivity to be operational. This era is about how technologically advanced the world is becoming. Taking advantage of these advances and aiming to make the world more interconnected and intelligent is altering the lives and businesses all around the world. But, these advancements in technology are possible only with better software systems. Software is considered to be the intangible entity that enables smarter services and paves the way to smart enterprise software solutions. This leads to increase in customer demand which further enhances the complexity of software and thus, gives rise to the need of delivering better software solutions to the market consistently. This brings in software delivery into the picture to ensure that better solutions are delivered to the businesses in the form of software products over the network.

An astute approach towards delivering software products paves the way to overpower competitive risks. The major hindrances faced during the process of software delivery include limited bandwidth usage, poor performance, handling uncertainties at various phases of delivery and economics of the software distribution process. The size of delivery is a major area of concern for the software delivery organizations. Here, we provide a localized methodology of delivering software products which has an upper hand over the centralized portal technique where all the delivery requests by the end users are directed. There is a lot of network congestion in the centralized approach which brings down the overall performance. The local tooling provide security layer by implementing authentication mechanisms for authorized access. Encryption techniques are adopted to enhance security. We also package the delivery in a compresses format for the ease of network transfer.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

Chapter 1 INTRODUCTION

Software Engineering encompasses all aspects of building up a software product from the very beginning of induction of an idea and eventually reshaping it into the final artifact from scratch. It is a formalization of the path to be adopted to develop an intangible piece of product [1]. Software Engineering is a field that defines a set of activities carried out to deal with management, development and maintenance of software products which are governed by efficient methods and approaches [2]. These methods are formalized to obtain an artifact of high importance conforming to the predefined specifications.

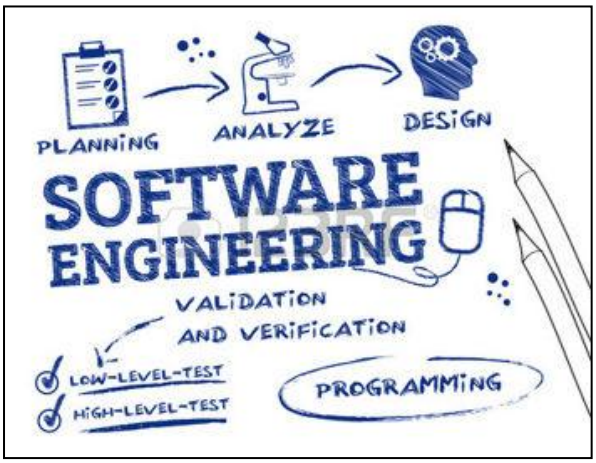

Fig. 1.1: Software Engineering Aspects

The formalized methods form up various software developments modeling processes. The quality of the software development modeling procedure applied affects the quality and texture of the developed software. So, to result in a qualitative piece of product in the end, a deeper understanding of the evolution process is required to successfully manage and cope up with the new software development trends. The improvement in the modeling process lays down the platform for monetary gains in the long term perspective for the organization. Through simulation and dynamic modeling, improvement in the development process is achieved in wider terms. This improvement is reflected in the quality of the artifact obtained and the extent to which it matches up to the expectations and the requirements. The research carried out on the traditional techniques and approaches gave birth to the processes that pave way to a higher capability and quality software development and evaluation process [3].

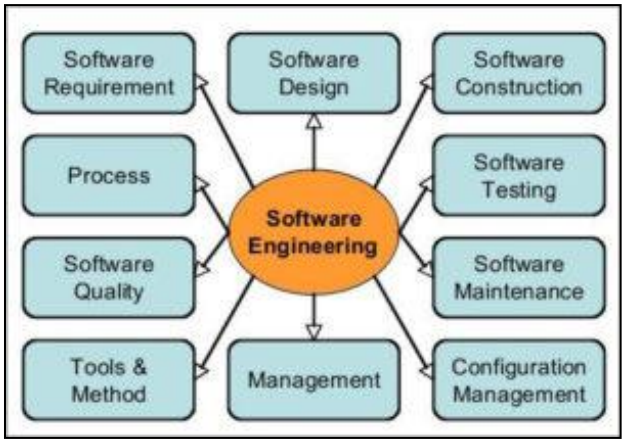

Fig. 1.2: Software Engineering Process

The approach to an efficient development [4] of software is phasing the entire process into various events. This modular perspective is a success story in the development of software [5].

Software distribution [6] is an integral part of the software production cycle. The delivery mechanism should be based on a standard infrastructure that encourages and lends support to multisource component assembly. The entire setup put to use should be adaptive and comes with a support to reconfiguration in terms of delivery so as to match up with the constant change in demands and requirements of the users and market. A collaborative infrastructure builds a bridge to the internal customers of the organization as well as with the global teams, overcoming the barriers of geography and diversity in organization.

1.1 Software Evolution

The software product supplier faces the issue of providing the product to the users in an efficient way to enhance performance. The conventional product distribution is carried out in a tedious manner by the manual transfer of a piece of software enclosed within an external media such as CD-ROM, hard disk and drive etc. It is considered as an inefficient way of distribution as there is no assurance of timely and intact delivery to the customer.

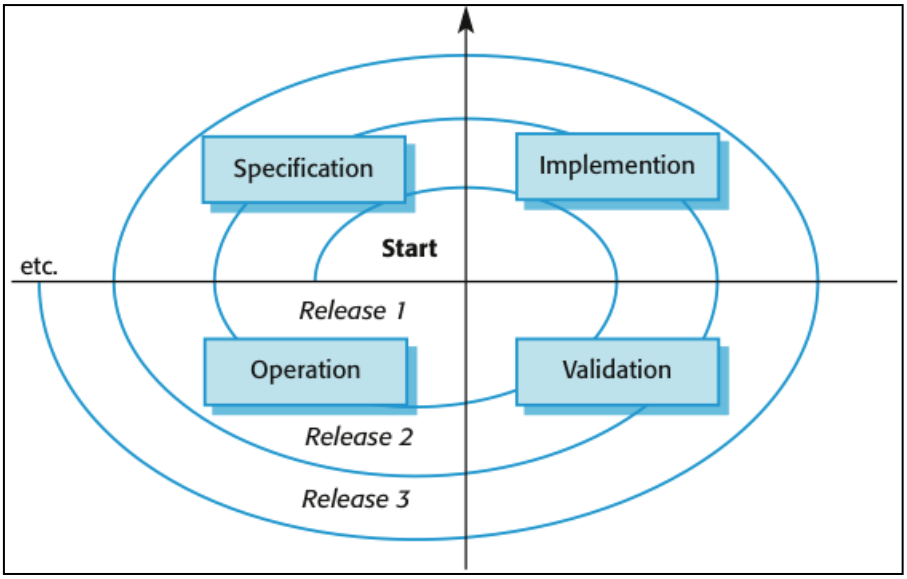

Fig. 1.3: Software Evolution

With the advent of network [7] connectivity, it has become feasible to distribute product over diversity in geography. Due to this high availability of computers and new technologies to access the internet, boom of electronic era has started. Electronic services have drawn great amount of attention towards them since they have made the world a much easier place to live in. All the geographical limitations have been put to an end due to the services provided electronically. E-services can be provided to the customers conveniently and in the economically feasible way.

It diminishes the fuss of the manual transfer. This automation in the distribution has brought about significant improvement in performance and time of delivery. The overall efficiency is enhanced and the entire process of software distribution has benefited from this era of automating the process of distributing software, products and libraries.

Security is considered to be a very sensitive issue in content delivery processes. Any security breach can lead to sensitive information in the hands of unauthenticated agents. So, there is a great requirement of introduction of protocols that take care of the secure transfer of software products.

The proposed architecture of Digital Delivery Builder which uses a localized tooling in which the user or the customer takes part in the process of packaging and

facilitating delivery [8] of the product. This demands an environment to be sourced at the user's end and with proper authentication mechanisms for requesting delivery, hence, followed by encrypting of the required piece of product and finally, initiating the delivery. The package is transferred over the network [9] and uploaded on a dedicated server using the File Transfer Protocol [10]. This server acts as the main delivery contact point from where the encoded and compressed piece of product is downloaded by the internal or the external users of the organization and installed in their respective local workspaces for further usage. The delivery package is encrypted for security concerns associated with the transfer of product over the network which may be open to various ill practices.

1.2 Software Delivery

Software Distribution is considered to be a very crucial part of the software production cycle. The main issue faced by different software providers is about adopting the mechanism that is most suitable, appropriate and inexpensive for facilitating efficient delivery of products. The delivery mechanism is devised considering the type of software to be delivered and the requirements of the clients. With the power of Internet, it has become possible to go about distributing a software product to the customers electronically [11].

Software organizations aim to build the process and technological support for facilitating efficient delivery through the following:

- A standardized infrastructure to go about delivering products supporting a multisourced component based assembly model.
- Adaptive system that has scope for reconfiguration in the times of changing needs and requirements
- Feedback and review systems allowing the scope of improvement wherever required.
- A collaborative architecture of distribution system that connects teams geographically separated creating various software services and together providing solutions.

The main limitations with this online software delivery process are the limited

bandwidth concerns. Due to bandwidth burdens, scalability of the delivery system is compromised. There is a must need to improve the delivery mechanism in order to achieve greater performance rates and better throughput. Companies need to consistently keep a check on the quality of product distribution and need to revisit their decisions in order to become a good software deliverer.

1.2.1 Changing Needs for Software Delivery

Several convergent factors are acting as driving forces behind the significant need for alteration in the process of Software Delivery. Those factors are listed below:

- **Requester requirements and expectations:** The requesters need the synchronized information about the delivery process anytime anywhere without any downtime required by the system. This demand for increased access has changed the scenario of reporting and management. The solutions under development must be able to provide the ease of usage by a broader community and huge target base on multiple platforms and devices.

- **Cost of developing solutions:** As per constantly changing needs, there is a significant emphasis on the cost cutting of the entire process which shows the inefficient side of the prevalent development processes and practices.

- **Speed of Change:** The more demand in the market to deliver solutions in order to cope up with the fast pace of changing business needs [12] to a broader network of stakeholders which results in acceleration in the delivery mechanisms with the objectives of rapidly producing solutions in the form of software products. The pressure of fulfilling the demands and requirements has led to an inclination towards achieving increased flexibility in delivery mechanisms by deploying a better and phased approach.

- **Adaptable business platforms:** Due to the highly adaptable business architectures, there is a constant need of solutions which need to be delivered in a shorter span of time. Software Delivery organizations need to excel in the tasks of introducing, adapting and optimizing the different business platforms by investing in a great amount of resources.

1.3 Software Development Life Cycle

A life cycle revolves around the chores carried out in order to develop a software product and making these activities to follow a sequence for better results. The need for development of a product starts with its requirement. The development process is initiated by the request made by the customer. This initial request for a product is called product conception. Then, the product goes through phases and undergoes transformation to be the final product. After the product is fully developed, it is then delivered to the customer. The customer uses it and then it retires when it is no longer useful. So, a software life cycle [13] is a series of phases that a software product goes through in its lifetime. After the product conception, feasibility study becomes the very first phase in the lifecycle of any software product. The requirements gathering and analysis, design, coding, testing and maintenance are the subsequent phases of a software product lifecycle. In each of these phases, certain type of activities needs to be carried out and the work done is documented for future references.

A software lifecycle model is a detailed and graphical representation of the different stages a software product undergoes in its lifetime. A life cycle model categorizes the different activities carried out on a software product in its lifetime into different life cycle stages. Different models [14] have different ways of carrying out these activities and they classify these activities into phases differently. Though each life cycle model moves around the same basic activities but their order is altered according to the model [15].

1.3.1 Why use a Life Cycle Model?

In the present world scenario, each enterprise follows some software model for their projects. It is always better to follow a formalized procedure for software projects rather than the adhoc build and fix model. The final products built through these well defined models are of good quality and time-cost efficient. It drives the development process in a systematic manner. Whenever a product is to be developed by a single developer, the entire sequenced of steps is defined by that developer. But, in the scenario which involves an entire team of developers, coordination and deep understanding are a must. If the team doesn't follow a formal procedure for development process then all this would seem to be a perfect failure. There has to

exist a model that guides the process of development of a product.

In a situation, in which a developer starts developing a part of the product, while another is working on the requirements specification and post that works on coding that, so, the first developer would now have to wait for the second part to be developed. After both the portions are developed, only then the integration of the product can be done. This practice compromises with the efficiency of the team. So, adhering to a software life cycle model [16] is crucial for successful completion of the project.

1.3.2 Phase Entry and Exit Criteria

A good life cycle model should not only clearly segregate the different phases in the software product life cycle, but also, specify the entry and exit criteria of each phase. By phase entry and exit criteria, it means before a phase starts, its entry criteria or conditions should be met. Similarly, before a phase successfully completes, its exit criteria should be fulfilled and satisfied. The entry exit criteria of phases should be well defined. If in case, it is not defined nicely, it results in ambiguity in starting and completion of various phases and confusion among the different developers of the team. It becomes very difficult for the project manager as well to monitor the progress of the project in absence of phase exit and entry criteria because it is not known when the phase starts or ends. Such situations lead to a state of problem in the world of software engineering known as 99% complete syndrome. In this syndrome, the assessment of the actual progress of the project becomes impossible. The team members think that their work is almost towards completion but in the actual scenario, they are far away from the completion of the project. This makes the predictions, estimations and the projections of the project manager irrelevant and inaccurate. If there exists a clear definition of the phase exit condition, then completion of the phase can be clearly recognized and stated to the project manager. This makes the project manager monitor the progress and accurately state the developmental process. As an example, consider the Requirements specification phase, it is considered to be complete only after all the requirements are gathered and reviewed and accepted by the customer. This is the exit criteria for the Requirements gathering and analysis phase of the software lifecycle.

1.3.3 Documentation of a Life Cycle Model

A life cycle model develops a sense of understanding among all the team members of the project. It helps the development process to mature in a disciplined manner. These days, all the software organizations document the software life cycle model that they follow. Such organizations make all the team members to master the act of documentation. They document all the outputs achieved on completion of phases, methodologies used, techniques adopted etc. into a coherent framework known as organization's software development model.

A well documented life cycle model leads to identification of redundancies, omissions and inconsistencies existing in the development process and then counter them with proper measures and planning. It also prevents unnecessary misinterpretations occurring due to lack of documentation of the process. The documentation helps to closely monitor each activity, its importance and usage. It also helps in the refining of the activity. It enhances clarity of the process going on in the organization. The process documentation paves way to customization and to tailor the process as per requirement and projects.

1.3.4 Classical Waterfall Model

This model was the first technique developed to formalize the process of development of a product. The Waterfall model forms the basis of all the SDLC models. Waterfall methodology consists of a sequence of phases which are different from each other and are cascading in nature. Each phase accepts the outputs from the previous phase as inputs and it cannot begin until the preceding phase successfully completes.

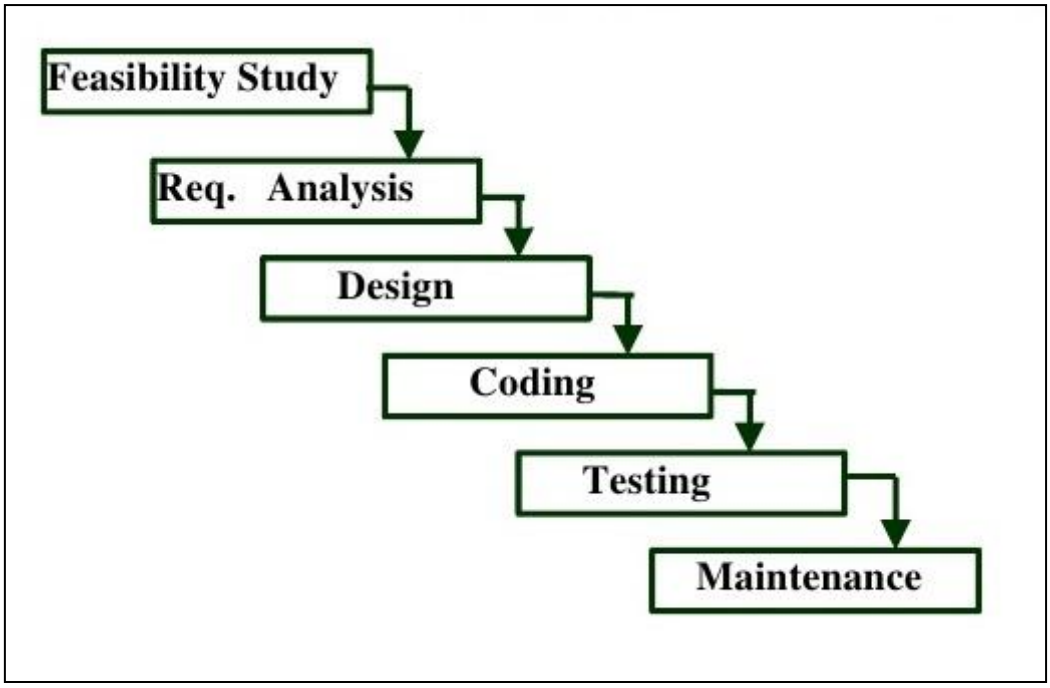

Fig. 1.4: Classical Waterfall Model

It is intuitively considered as the most generalized and obvious methodology used in the development of a software product but it is not taken to be as a practical model. It is considered to be more of a theoretical methodology of development rather than an actual development technique which can be implemented in a real world scenario. It is not practically used but forms as a basis to all other models, since all other models are in some or the other manner dependent on the waterfall model. The graphical representation of this model resembles a cascade of waterfalls, and hence, it gets its name as the Classical Waterfall Model. The waterfall model constitutes of a set of well defined and identifiable phases. The different phases in this model are:

- **Feasibility study**: This phase is about carrying out a feasibility study analysis. It is done to figure out if it is technically and financially feasible to develop the proposed product.

- **Requirements Analysis and Specification**: This involves gathering of all the requirements and specifying all these in the Software Requirement Specification document which are to be reviewed and accepted by the customer.

- **Design**: This involves designing activities which are giving the requirements

a structural form that is suitable to be implemented in a programming language.

- **Coding and Unit Testing**: The coding phase is about the transformation of the design into the source code. This coding phase is known as the implementation phase. The unit testing phase involves testing each module of the coded product. The end-product of this phase is a collection of individual modules which are separately unit tested.

- **Integration and System Testing**: This consists of integration of previously tested unit modules in a planned manner. All the coded modules are never added in one go but are incrementally added and testing of the resultant product is carried out at each step.

- **Maintenance**: This is about carrying out maintenance activities like correcting bugs and errors, enhancing functionalities of the system, improving implementation, making the software product adaptive in a new environment. Studies have shown that the effort made in the development phases is much less the effort put in the maintenance phase of the entire life cycle. The ratio of this is found out to be 40:60 respectively.

1.3.5 Iterative Waterfall Model

This model is an enhancement of the classical waterfall model. Iterative waterfall model is viewed as altered waterfall model which involves making necessary changes to the standard waterfall model so that it can be applied in a realistic scenario. The iterative model provides feedback paths to each previous phase which provides the scope of correction of errors in a phase, when detected in another phase.

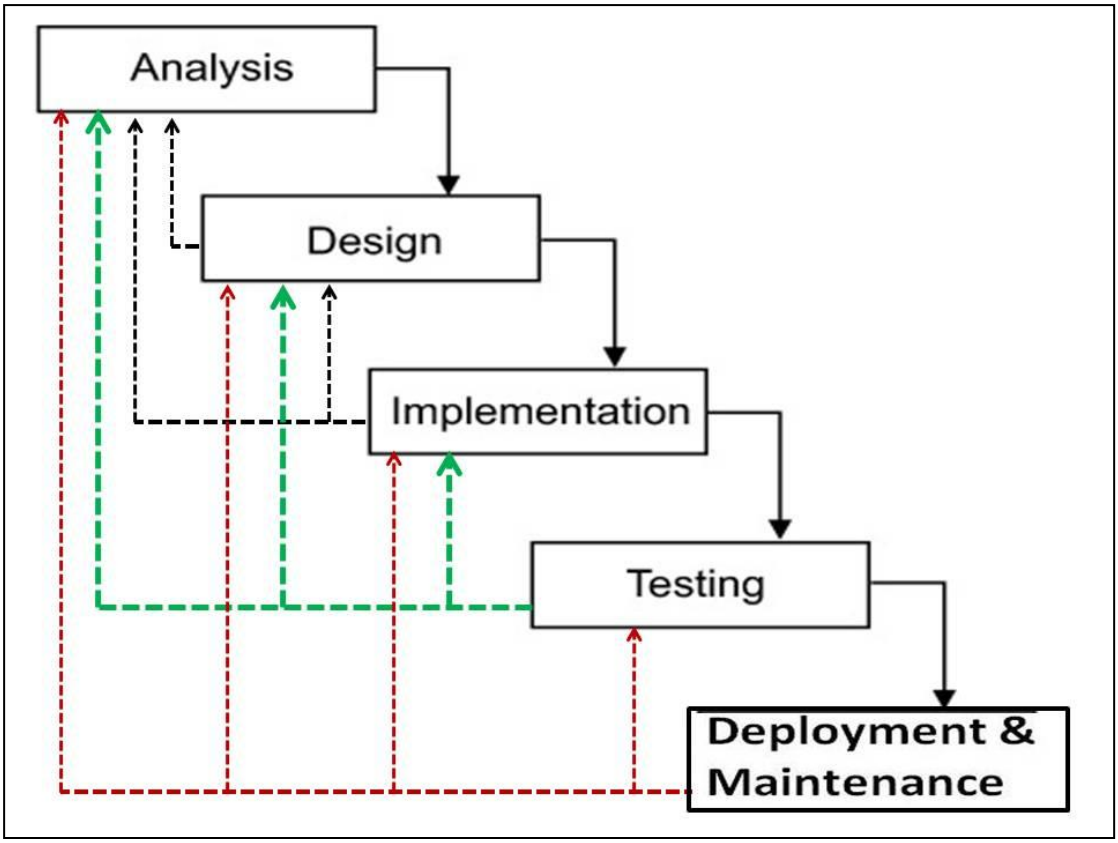

Fig. 1.5: Iterative Waterfall Model

For example, an error committed in the design phase can be corrected by reworking with design activities and documenting the changes. There is no feedback path to the first phase of the life cycle model which is Feasibility study. This means that any error encountered in feasibility study phase while currently being in another phase, cannot be dealt with.

Phase Containment of Errors

Though errors are inevitable, but it is always considered to be efficient to detect errors in the same phase in which it has occurred. This reduces the efforts required to deal with the errors to a great extent. For example, if a design bug is detected in the design phase itself, then it greatly saves time and effort of the entire team. The changes can be made there and then and subsequent phases can be carried out efficiently. Phase Containment of errors principle states that it is so much better to detect a bug as close as possible to its point of induction. A coding phase bug should be detected and corrected in the coding phase itself. Its detection in later phases increases the amount of effort required in order to handle it later on.

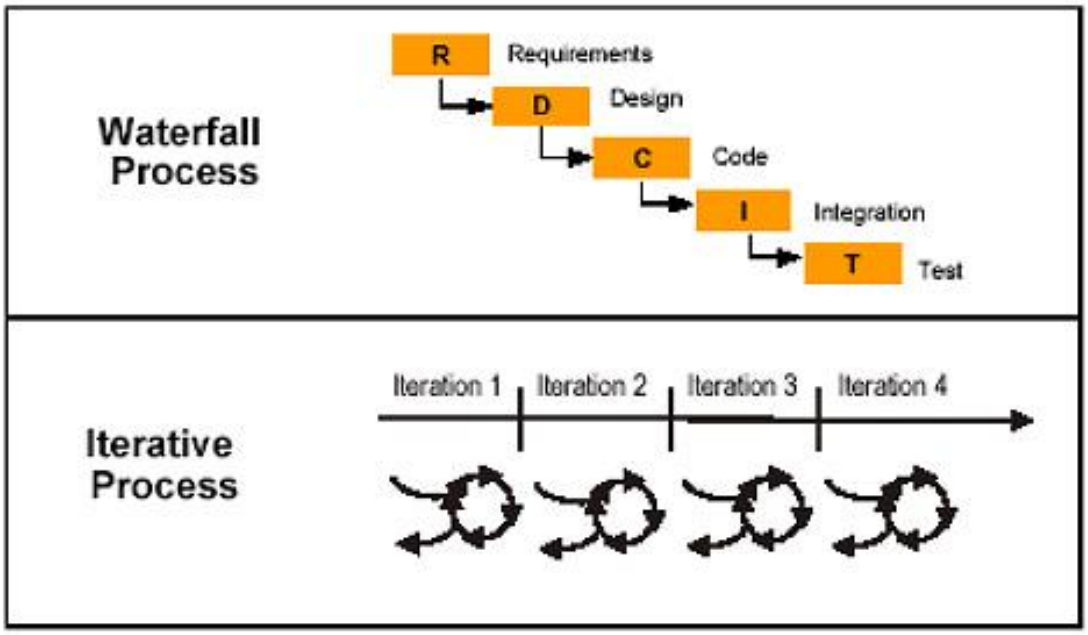

Fig. 1.6: Waterfall Vs Iterative Waterfall

In order to achieve phase containment of errors, at each stage reviews need to be conducted. In such reviews, bugs are caught and handled in the same stage before proceeding to another stage along with the erroneous product.

1.3.6 Prototyping Model

In this methodology, it is required to make a working prototype of the proposed system even before the developmental activities of the product have even started. A prototype is a toy-like representation of the system. It is usually created by making use of shortcuts. Shortcuts are inaccurate, inefficient dummy functions. Shortcut functions use very simple method to solve a problem rather than the efficient complex computations. A Shortcut function may use a table lookup to solve a problem and not bother to solve it through calculations and computations. A prototype is an unrefined version of the entire system implementing limited basic functionalities. It exhibits lesser efficiency and lower reliability in terms of performance than the actual software product. The prototype is built and then illustrated to the customer. The reviews are gathered so as to gain a better understanding of the demands and requirements of the customer at an initial stage. Gaining a good understanding of the needs of the customer at an early stage helps to do work in the right direction from the very

beginning. It becomes quite easier for the customer to give his opinion through working with the prototype rather than just imagining the product which is to be built in future.

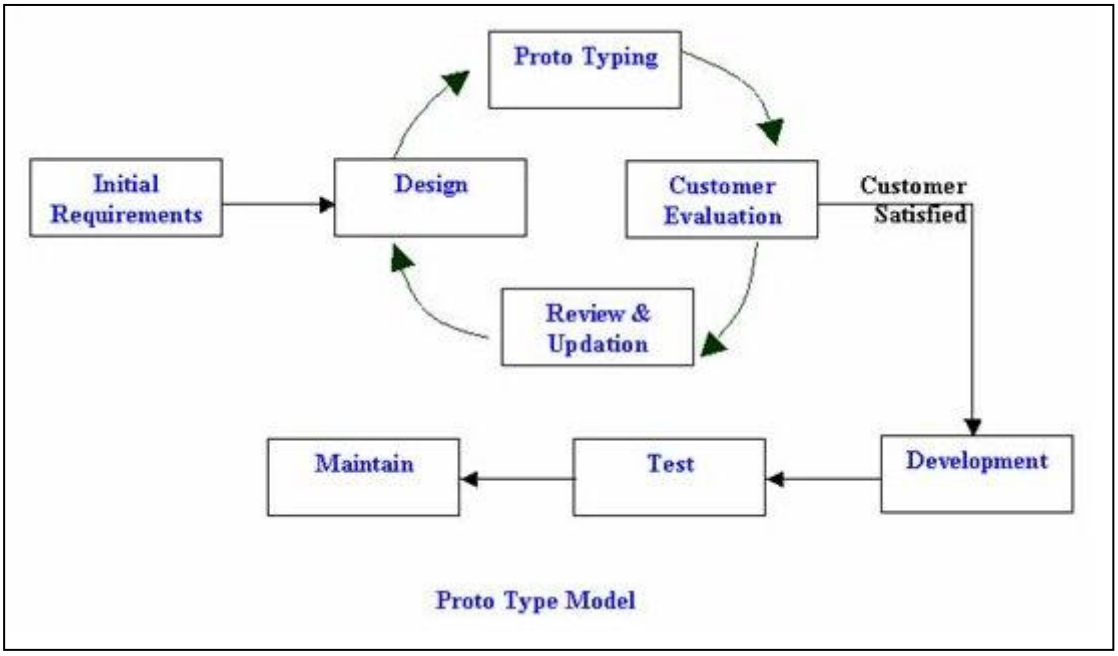

Fig. 1.7: Prototyping Model

The prototyping model can be applied in another situation where the development team doesn't have a clear understanding of the exact technical solutions to be adopted. A prototype built helps the team to deal with the issues associated with the development of the entire system. In such circumstances, it is best to deal with the issues of the prototype first and then handle the real issue while developing the system with experience.

In prototype model, the first phase is about building a prototype of the software product, followed by iterative cycles of development. The code of the prototype is thrown away and not used while the development of the actual system. The experience and the knowledge gained while working with the prototype is utilized in developing the real software product. Though, construction of a prototype is an overhead, since, it is of no use in the actual development, but it is best to use the prototyping model in situations where the customer requirements are ambiguous and the technical issues and risks are unknown to the developing team.

1.3.7 Spiral Model

This model is named as spiral because its graphical representation resembles that of a spiral with multiple loops. The number of loops is not fixed and can vary. Each loop in the spiral is phase or stage of the development process. This model provides flexibility to development as it can have any number of spirals or phases.

At each phase, some of the features are recognized and analyzed and incorporated into the product. The risks are detected and solved over each loop. Each phase completion gives a prototype which exhibits better features and functionalities than the previous one. This way, all the identified features are implemented and the software product is built.

Spiral Model is the most appropriate model to follow in situations where unknown risks are encountered as the development goes on. It is considered to be much more powerful than the prototyping model where all the risks need to be known at the start of the project and all the issues are resolved even before the development starts. In this model, the risks show up as the work proceeds, phase by phase, and resolved to take the development further.

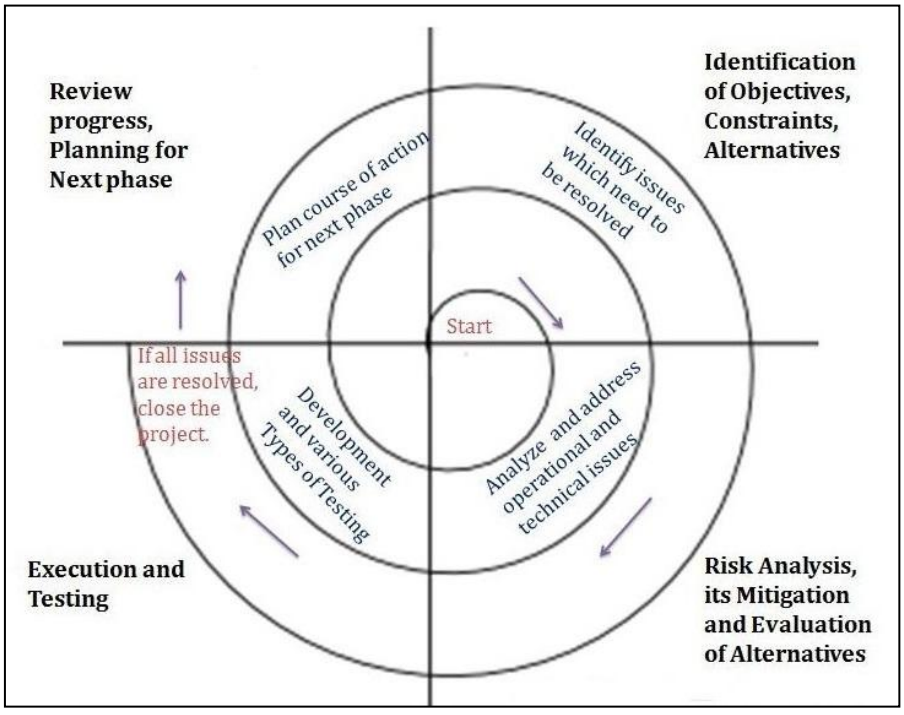

Fig. 1.8: Spiral Model

Each phase of the spiral model is categorized into four quadrants. First quadrant is about planning and identifying the features and finding out technical solutions for the same. In the second quadrant, all the risks associated are identified and work towards resolving them is done. The third sector is about implementation and execution of the planned objectives. The last quadrant allows reviews to be conducted so that errors or bugs could be identified.

Risk Handling

In this model, risk is handled by building up a prototype and then work upon analyzing all the risks and adverse circumstances. Finally, experimenting on these risks to resolve and learn from experience. Spiral model provides the scope of building a prototype at each spiral and evaluating and resolving the risks.

1.3.8 Rapid Application Development

Unlike other models, this model is time-driven rather than requirement-driven. In this model, time to deliver a requirement or functionality is supreme than anything else. It is a kind of incremental model. Here, the requirements are shaped into functionalities in the product parallel as if there are small sub projects.

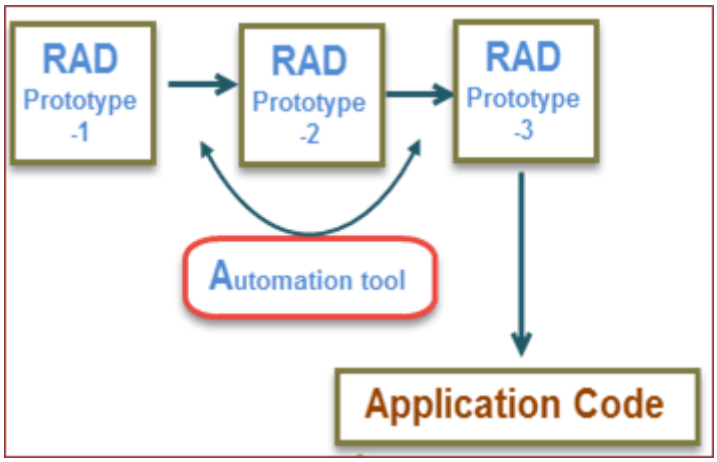

Fig. 1.9: RAD Development Model

Here, the progress is time boxed. A time box is a predefined time slot after which a certain requirement is delivered. The developments are completed in time boxes and assembled into a prototype for the customer to review it. The user checks the working of the prototype and provides feedback about the features of the prototype. The

requirements are considered priority-wise and high priority features are implemented first with working down the list. RAD model can deploy multiple time boxes in an incremental fashion to gain add on functionality. It can also deploy evolutionary methodology to achieve evolved functionality and requirements over time.

The RAD model is adopted when the system needs to be implemented in a very short span of time. It requires all the requirements and demands to be known beforehand. This model keeps the customer involved in the process all throughout the development to ensure the delivery of the required product ultimately. This model makes use of automation tool for generation of code to save time. So, when the budget is a huge factor for the usage of this methodology. The projects with high budget can accommodate RAD model.

Though not all applications are compatible with RAD model of development, but used whenever the team wishes to reduce the overall technical risk and issues associated with the project. This involves usage of code generation and cod reuse, hence, reducing the risk of bugs and errors due to lesser manual coding on the project. Each phase of the RAD model promises to deliver high priority functionality to the customer. This model surges productivity in a shorter span of time due to the fact that is deploys lesser number of people but it definitely requires highly skilled developers and designers. A RAD model goes through development in the form of a prototype, evolving into a finished application by adding functionality at each phase incrementally.

1.3.9 Agile/Extreme Model

This model is also a time driven model rather than requirements driven. This is what gives this model a departure from other SDLC models. The agile model [17] is used to reduce the overall development time by making use of Stories, Situations or Scenarios, Use cases. These are used as replacements for documentation of the entire process which is very time consuming process. The time is further compressed in the form of time boxes which are predefined time slots to carry out development and test activities.

This model is adopted in circumstances where the final product is expected to be delivered in days rather than months or years. The goal is achieve the implementation

as early as possible. The developmental activities are carried out in parallel manner in time slots to govern time factor. Evolutionary methods are applied to gain additional features. Software application is developed and given the form of a final product in rapid phases or cycles. The developmental process is in the form of a release and each release has an added functionality to the previous one. Each release undergoes rigorous testing to ensure that the quality of the software is maintained. It is deployed in time bound applications.

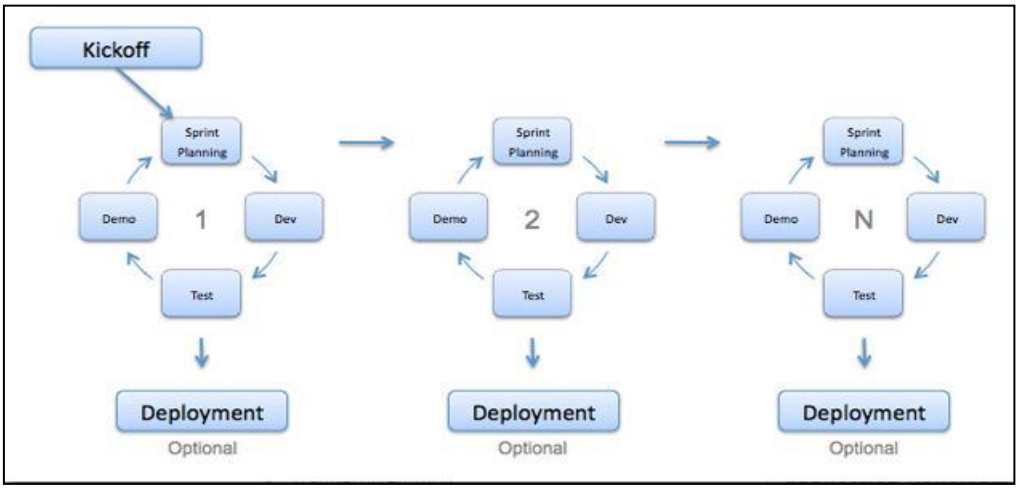

Fig. 1.10: Agile Methodology

The agile model [18] is deployed in scenarios where new changes need to be added to the application. The new changes can be incorporated into the product at little cost due to the production of new incremental releases of the product at high frequency. Unlike Waterfall model, the agile model assumes that the requirements of the customer are dynamic in nature and are not fixed. The agile model is started with very limited planning. As per the reviews and feedbacks of the user, new features are added to the product and useless functionality is removed from the application.

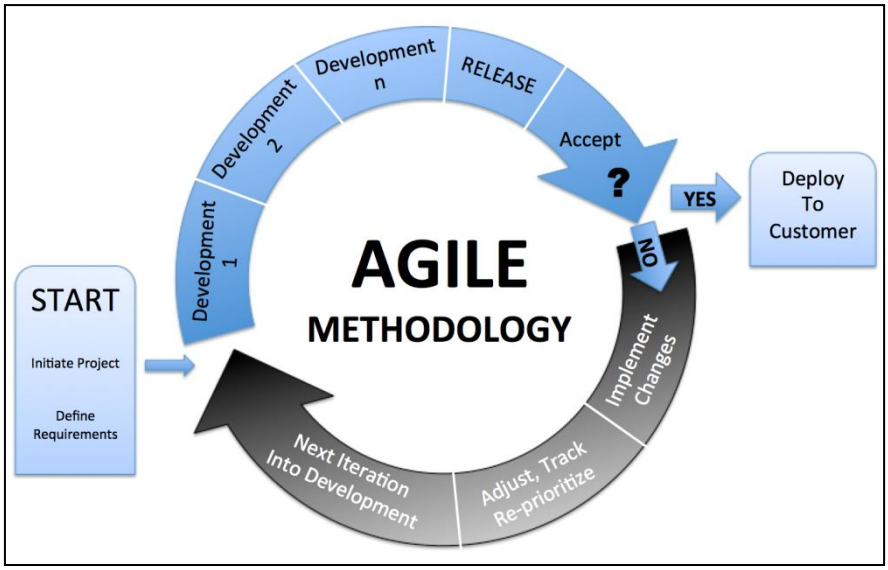

Fig. 1.11: Customer Satisfaction in Agile Model

Customer satisfaction comes along with the Agile model adoption due to the fact it ensures continuous delivery of product. This model gives emphasis on people and interactions instead of tools and processes. Working form of the software is delivered to the client weekly and not in months or years. The changing circumstances during the development process demands adaptive nature which is provided by this model in the form of regular adaption to alterations and changes. This is what agility is about here.

Chapter 2 LITERATURE REVIEW

Software distribution is considered to be important in terms of the efficient delivery of the core strength and potential of a business and it drives the innovation and introduction of new software amenities and products to the market or the outside world. Enterprises are now aiming towards enhancing their software distribution organizations through the use of system integrators and collaborating with technological enhancements to go about strengthening and improving ability to gain expertise in bringing value to business aspect. This is directed towards the enterprise software delivery. Excellency of software delivery process is dependent on the potential of the organization in building a bridge of balance between the agility and the efficiency of the mechanism. There are different proposed technologies and approaches that are prevalent for electronic software distribution.

2.1 Secure Multicast Software Delivery

In [19] Han and Shahmehri proposed a model for secure and a multicast mechanism for delivering software products. The process of multicasting is about a single sender and multiple recipients. The software product is delivered to the entire list of multicast group recipients or subscribed end-users. In the scenario of an update release, the update files are delivered to all the subscribed end users at the same time.

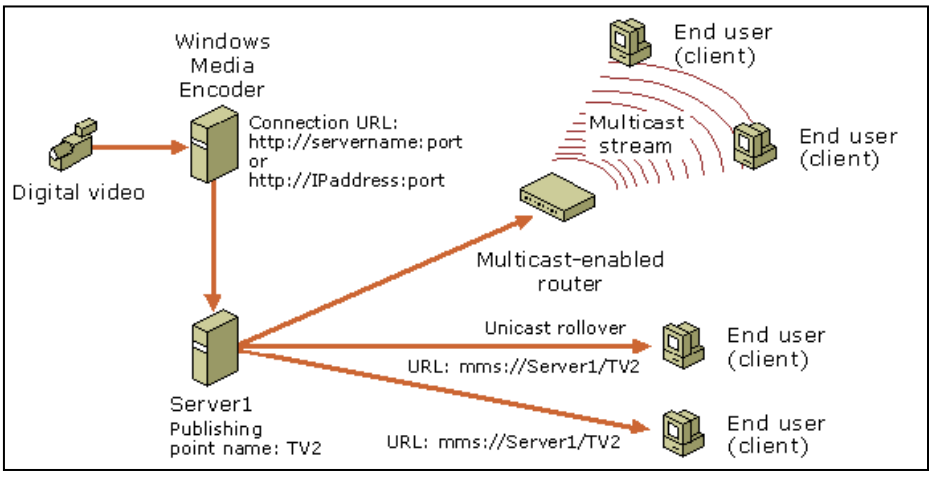

Fig. 2.1: Multicast Delivery

The authors have proposed an agent based architecture for the multicast delivery. The architecture defines a security agent that deals with collection of the subscriber's information and then performing authentication and authorization mechanisms. This security agent creates and issues session key to the recipients for authentication purposes. The recipient group establishment and the data sharing phase is based upon the multicast protocols.

The multicast delivery takes place in two phases. One phase is about the establishment of the recipient group and the other phase is the data delivery phase. The author describes the first phase as declaration of a delivery session by the sender and the recipients register as the subscribers. These subscribers are interested in the data to be delivered. Secure data distribution is based on the recipient group establishment. After this establishment is done, no registration of users is allowed. All the subscribers are considered as trusted and put under the agreement that they wont disclose their session key to anyone even after they are no longer a part of the recipient group. After the group establishment phase is completed, the data delivery phase starts. This phase is driven by a multicast protocol to deliver data. Before initiating the transfer, the data delivery agent performs encryption on the software so as to maintain its confidentiality with the session key. After successful encryption mechanism, the encrypted software is sent to the multicast recipient group address. In case of missing packets at the receiver agent, the delivery is resent. This is taken care by the deployed multicast protocol. If the subscriber is successfully authenticated, then subscriber has the authority to decrypt the software and deploy it.

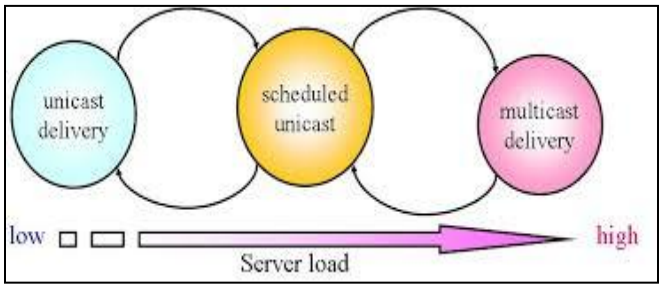

Fig. 2.2: High Server Load in Multicast Delivery

The shortcomings associated with this approach are that the infrastructure which is required to support the multicast protocols is not completely supported by the internet.

2.2 Distributed Agent Based Delivery

The proposed approach [20] is based on a system that is supported by a distributed architecture. This mechanism is driven by a software dock which is constituted of various docks or components such that each component exhibits different functionality. The Software dock has an agent based technology. The dock system provides an interface for different constituent agents to register for any event which may be any software distribution operation. The access to sensitive information is controlled by the security dock agent. The issue with the described methodology lies within the use of distributed servers that deal with data replication and geographically distributed processing. The distributed nature of the entire approach is difficult to picture in complex systems. Another shortcoming is about the huge demand for network bandwidth to go about the peak hours of downloads of the software products. The high bandwidth consumption is creates a difficult scenario to manage in real world.

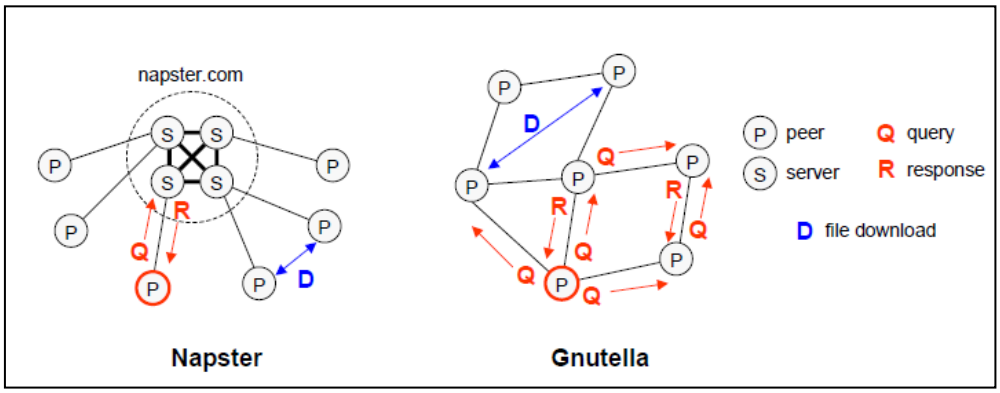

Fig. 2.3: File Location in Napster and Gnutella

2.3 Peer-to-Peer File Sharing System

Peer-to-Peer File Sharing Systems such as Gnutella and Napster [21] lack a dedicated and centralized architecture. Such systems are about voluntary participation of hosts referred to as peers in the architecture. The organization of the peers into a network of dynamic and ad-hoc communication is a challenging task in the real world scenario. The cooperation between the hosts to together result in providing a service to the community of users that is useful enough in terms of software distribution and its deployment. Whenever a file was requested, the central servers searched for the all

the copies of the file and returned them as a result. It had a limitation that it could only deal with the sharing of music files.

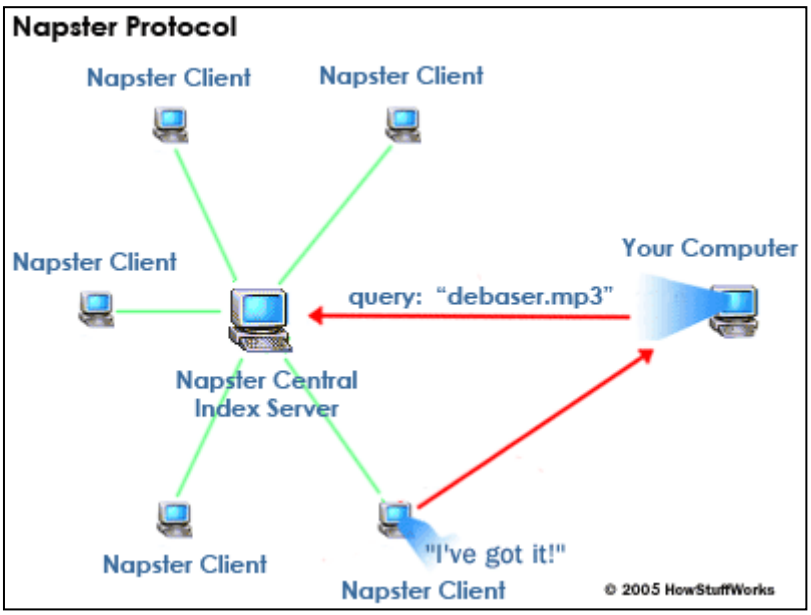

Fig. 2.4: Napster

Peer-to-Peer File sharing system [22] gained popularity in the late 1990's with the introduction of Napster. Napster was a file sharing system consisting of a set of central servers that built a link between those requesting and sharing files in the network. It has a central index server which contains metadata about the users sharing content and information about the content that is being shared. Napster and eDonkey2000, based on a centralised architecture, are classified as the first generation of P2P systems. These systems rely on the functioning of the centralised servers completely and were vulnerable to the central failure which would lead to shutdown of the entire system. Then this shortcoming led to the development of another category of systems called second generation of P2P systems which eliminated the chances of failure due to the centralised architecture.

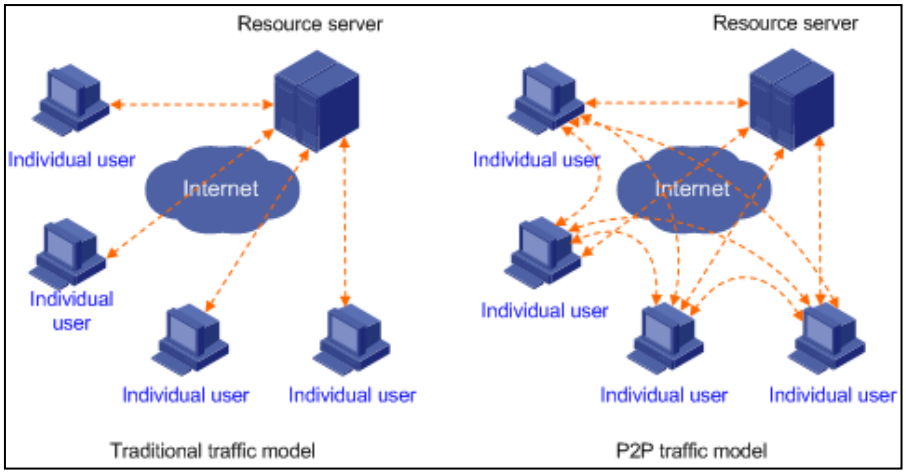

Fig. 2.5: P2P File Sharing System

The overall performance of the P2P file sharing system is about determining the suitability of a peer host to accomplish a task before allocating it to the host. Once the delegation of the task to the peer is done, the resources of the peer are utilised in carrying out the sharing job.

2.4 Content Delivery Networks

The content delivery networks [23] exploit the property of locality of reference. The frequently accessed data needs to be kept near to the requester so as to decrease the transfer time and diminish the unnecessary network latencies. These systems deal with web content in the form of HTML files and images. These systems deal with incorporating different non-origin servers to deliver content to the requesters on behalf of the origin servers. Thus, decreasing load on the original server and achieving load balancing. The non-origin servers may or may not be at the same location to that of the origin server. The CDN [24] nodes are deployed at various geographically distributed locations. These nodes aim to reduce bandwidth costs, reduce delay in page loading times and overall increased availability of the content globally. The requests for content is directed to the servers/nodes in such a way that optimises the overall service request time and process. The locations that deliver content with optimised performance are chosen to serve the request of content. In an optimal scenario, the servers that are closer to the end users or requesters tend to show better results. Such servers which are geographically less distance away are called edge non-origin servers. CDN uses different protocols of content delivering.

Content service protocols: The Internet Content Adaption protocol is a content service protocol which was developed around the year 1990. This provides a standardisation for the connection of different application servers. Another protocol known as Open Pluggable Edge Services protocol which defines requests in the form of application to be executed on the OPES host or on some remote server known as callout server.

Peer-to-Peer CDNs: In such systems, there is volunteer assistance of peer that form part of the network in servicing of the requests meant for the origin server. These content networks show better results in measures of performance and setup is running for distribution of content with very less running costs for the original distributor of content [25].

Private CDNs: The content owners create a personal delivery network to deliver better performance in case of heavy load on the network. The private network contains nodes which carry out content transfer only for the owner. A network with simply two caching servers can also be a private content delivery network.

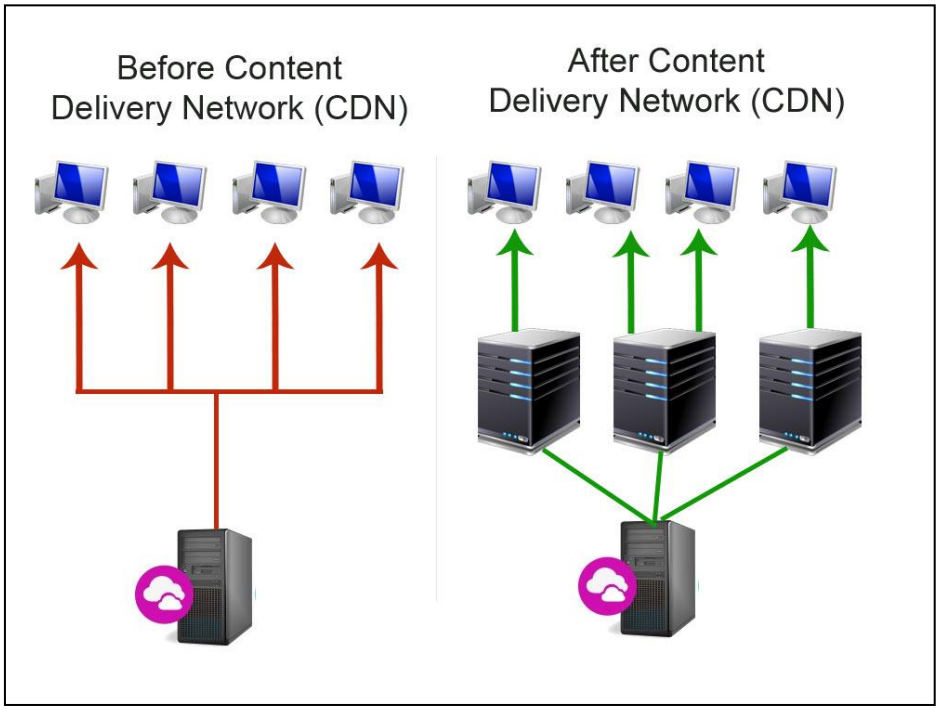

Fig. 2.6: Content Delivery Network

Although, theoretically CDN are suppose to enhance performance but after a certain threshold duration of time, there is a certain amount of overhead associated with it

which accounts to DNS lookup [26]. The performance of the origin servers is considered to be better in a larger picture. Even in the performance analysis, it was concluded that DNS lookup time incremented the response time as compared to the scenario of a fixed server to respond to requests of web content. It was observed that even the response time in the worst case situation is not improved by the DNS lookup process to find a new server to provide the service.

There are systems that have entered the commercial world through the use of Peer-to-Peer technology for streaming of real-time media content. The author [27] defines a variation in the concept by introducing the concept of virtual multicasting. A virtual multicast layer is implemented on the application layer. This revolves around organizing the peers and giving them a form of a tree-like structure in a dynamic way. This arrangement is not predefined and organization varies as the peer host jump in. The peers when organised in the form of a tree, streaming [28] is carried out such that stream of data is send to only some root nodes which ,further deliver to the leaf nodes. The technology backing such systems is seen to stand out and is considered to be better than Content delivery networks in terms of performance measures and reduced cost.

2.5 JXTA Technology

JXTA [29] is an open source technology which is supported by a peer-to-peer protocol specification and started and promoted by Sun Microsystems in the year 2001. The JXTA protocols facilitate communication of peers in the network through the transfer of messages and data between them in the form of XML messages.

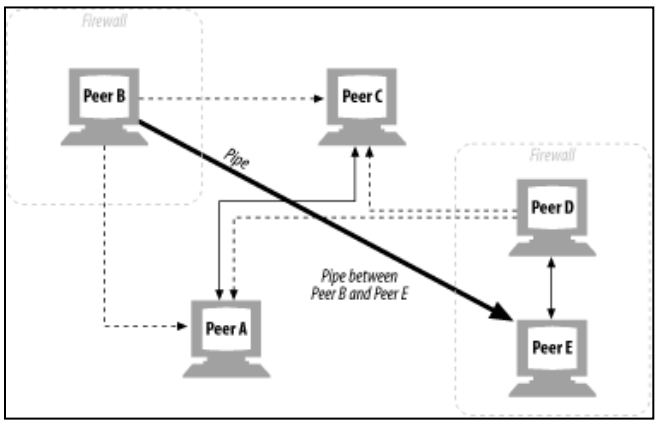

Fig. 2.7: JXTA Delivery Network

This technology allows the peers [30] to create a virtual network that allows the peers to communicate with peers that are hidden behind firewalls and NATs. The data transfer is possible even between those peers that use different network transports. Each peer in the network is identified with a unique identification number. This number uniquely identifies the peer host even in those cases where a peer changes its local address. JXTA categorizes its peers into two types: edge peers and super-peers.

- Edge peers: The peers which have transient network connectivity and have quite low bandwidth allocation. These peers are usually hidden behind firewalls and be a part of the network through non-dedicated connections.
- Super peers are further categorized into two categories:
  - Relay peers: These peers allow peers behind firewalls or NAT to be a part of the JXTA communication network. This makes use of protocols which can cross the firewalls.
  - Rendezvous peer: It is a special type of peer which allow communication and coordination of all the peers in the JXTA network. These peers facilitate the data communication and message propagation. The network should have at least one Rendezvous peer, if the peers are part of different subnetworks.

Any peer [31] in the JXTA network can be relay or rendezvous peer after it has fulfils the memory, CPU, network or storage requirements.

Chapter 3 RESEARCH PROBLEM

3.1 Problem Statement

Software Delivery accounts a considerable premium to an organization. It provides a definition to the fundamental expertise and the core competitiveness. An appropriate approach towards software distribution paves the way to overpower competitive risk and issues. Software delivery is considered to be very crucial for the efficient and effective delivery of the fundamental and basic competence to the business. It acts as the driving force to the induction of new ideas in services and products in the market. Software delivery component of an organization brings in value to the enterprise. So, organizations these days are keen in enhancing and improving the software distribution process so as to create centres of excellence and eminence that work towards bringing back value and worth to the organization. The driving factor of the successful software delivery lies behind the organization's potential to create a balance between the agility and efficiency with which the software distribution is ensured.

Software Delivery is considered to be a very vital part of the corporate world. The software product supplier faces the issue of providing the product to the users in an efficient way to enhance performance of the overall process. The conventional product distribution is carried out in a tedious manner through manual transfer of a piece of software enclosed within an external media such as CD-ROM, hard disk and drive etc. Though still prevalent, this technique of transfer is considered to be an inefficient way of distribution as there is no assurance of timely and intact delivery to the customer. With the advent of network connectivity, it has brought evolution in the field of software delivery, as it has become feasible to distribute product over diversity in geography. It greatly diminishes the fuss of manual transfer. This automation in transfer and distribution has brought an era of evolution which has greatly increased performance and time of delivery.

The main challenges faced by software supply chain delivering products over the network include the bandwidth limitation, poor performance, delivery time and

economics of distribution and handling uncertainties at various levels of delivery. The problems are aggravated with the elevation in size of package delivery.

An efficient approach of delivering software products is needed which gives good performance and high throughput. This research domain works in achieving new efficient ways to facilitate distribution of software products.

3.2 Research Gaps

- A lot of approaches have been defined but research is going on to develop an astute approach that deals with the issue of limited bandwidth usage for network connectivity and transfer of products over the network.
- There is a need to produce software delivery mechanisms which can deal with heavier package delivery sizes in the most efficient way possible and fails to lead to any loss of information or data over the network. Security is a great research domain which is a vital part of the software distribution approaches.
- Giving out better delivery solutions which involves work in the field of cryptology so as to ensure safe delivery of the package after undergoing some encryption for security purposes. It is crucial to ensure that the data is not attacked and accessed by some unauthenticated entity.
- The delivery time improvement is a measure of good performance. Good performance ensuring distribution mechanisms need to be devised for efficient delivery of products.

3.3 Research Objectives

The following objectives have been formulated in the direction of the above mentioned research gaps:

- To develop an efficient software delivery approach for distribution over the network and perform the implementation.
- Designing a secure system that facilitates software delivery using the bandwidth allocated.

- Applying encryption techniques to enhance the network security of the data being transferred over the internet.
- Designing test cases and perform testing of the implemented software delivery system.
- Making the delivery system to be a local tooling process rather than the centralized system which jams up the network and eats up the network bandwidth.
- The system should be made resilient towards any failure and should be using proper authentication mechanism for restricted access.

3.4 Research Methodology

In my research work, PERL and UNIX shell scripting will be used. Perl is used as the basic programming language for implementation of all the modules of the application. The wrapper module is implemented in shell scripting which governs the deployment of the tool in the environment and also manages the firing of the delivering process.

The initial iteration of development cycle gives a prototype implementing the basic functionality and requirement. We have built the tool on UNIX [32] operating system and chose Perl as the implementation programming language for the development of the tool. Perl is chosen because Perl has huge library support that makes functional requirements easier to incorporate. Perl is a strong language for text processing. A very important feature that makes Perl a very talked about scripting language is CPAN [33] .It a central repository of huge number of Perl modules. Since, it is an open source language; Perl has a module written for every programming requirement. It provides the developer with the flexibility to import the required module and hence implement it in the source code. The wrapper module over the PERL [34] modules is written as shell script. The overall library flow is governed by the wrapper module built over the internal functional module.

This approach performs product distribution in local mode [35]. The tool is deployed in the environment which is to be sourced to make the tool conduct the distribution of the product. Our goal is to design a tool that provides flexibility in the distribution leading to enhanced performance. It can be used as a test model for software distribution in various scenarios. It can be regressively tested in various conditions and then the results can be analyzed to determine the accuracy and efficiency of the tool. The functionality of the tool can be subjected to test to simulate its working in an artificially created environment [36].

4.1 Architecture of the Delivery Builder Tool

The architecture of Digital Delivery Builder is composed of various components as defined below:

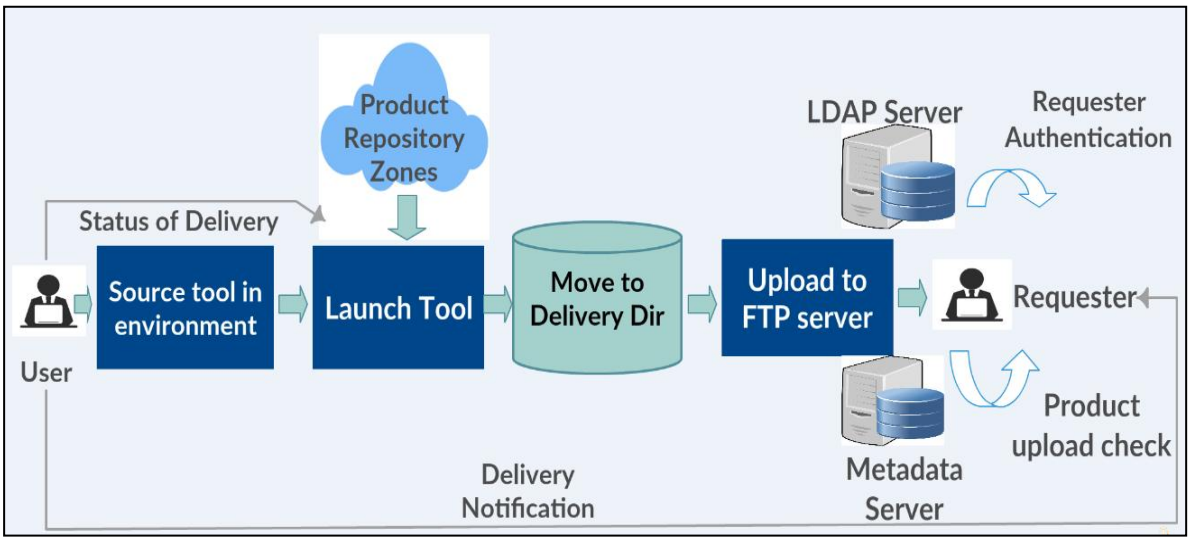

Fig. 4.1: Localized Delivery Builder Tool Architecture

User: The user is responsible for firing the command to launch the delivery builder tool. The full support in case of failed delivery or any other issue is provided by the user to the requester of the delivery. The user makes the environment ready for running the tool by sourcing all the prerequisites and then the tool takes over. A user is an authenticated person who has permissions to run the tool and has knowledge to work with it.

Requester: A requester is the one who sends a delivery request mail to the user to be uploaded over the server which would later be downloaded and installed in the requester's machine by the requester. This forms the basis of the concept of localized delivery. A requester is provided with a license to request and access deliveries. This license is valid for a certain licensing period after which it needs to be renewed to gain access again.

Sourcing the environment: The first major component of the product distribution tool is to make the environment available to the user in which the commands to initiate the process can be fired. This indicates to the deployment in the system by carrying out the installation of the tool. This tells the operating system of its existence in the environment and the path from where the executable is run when the tool API is used to initiate the process. Another important task done by this component is that all the prerequisite products are made available in the executable mode for the Builder tool. This makes sure all the environment variables hold within them the required value for the tool to run. This is setting up the environment for the delivery to start its functioning.

Launch API: UNIX API command is fired which checks for the requested product in the software repository or the database. The database query is fired which checks for the requested product and displays the result to the user who launched the check product query command. The database search operation is executed in the backend and hence, returns the result of the query. It is encrypted by applying encryption techniques for ensuring safety of the software product and then packaged into a compressed form. It is then ready to be transferred over the network.

Product Zones: These zones represent the discs that act as repositories for storing the software products. These product zones are the heavy databases that hold numerous products and libraries and are the knowledge assets of the organization. The API provides the facility to query about the products present in these install zones. Whenever a request for software is received, the availability of the distribution request is checked and further processing is carried only if the product of required version is present in the install zones. These product zones are considered to be the assets of the organization.

Delivery Directory: This is the path or location where the product in the encrypted and packaged form is moved, ready to be delivered to the requester. This is the target directory which the tool recognizes as the delivery to be built. This phase involves transferring the packaged product into the target directory which signifies that the next step of uploading it on the FTP server.

FTP Server Upload: This is the main task of product distribution. This uploads the delivery on the server which can later be downloaded by the requester after confirming the access to be authorized. The delivery requester using its credentials logs in to the upload server and after the authentication process, can access the data. The delivery can be downloaded and the licensing is taken care of for the requester. The algorithm used in the uploading of delivery on the FTP [37] server is shown in the figure below:

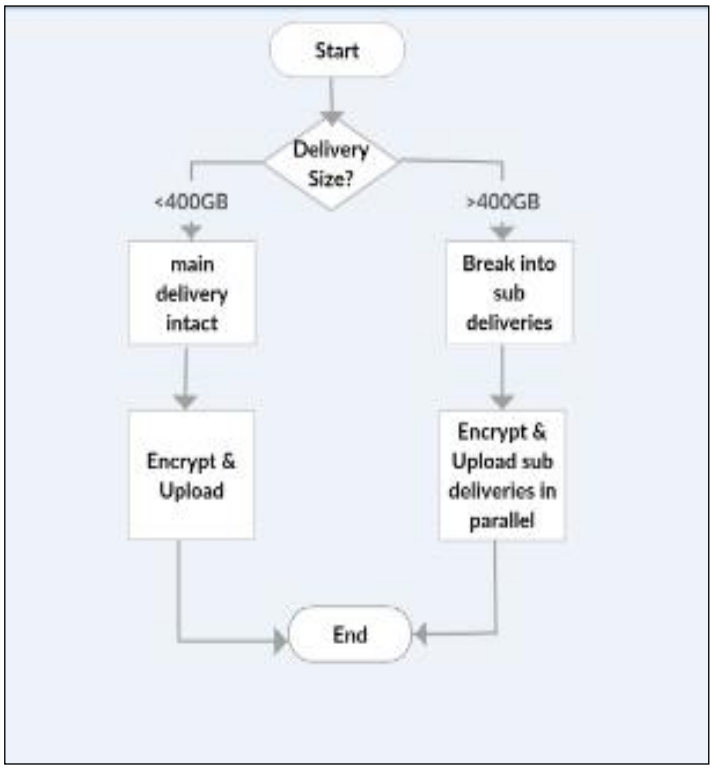

Fig. 4.2: Algorithm of FTP Uploads

This algorithm is based on the Bin Packing Algorithm [38] which aims at utilizing minimum number of bins to accommodate the total weight such that each bin cannot hold weight more than a certain amount. This distributes the sub deliveries in the most efficient way to achieve even distribution. This algorithm is implemented in the

delivery builder tool in a customized format. The bins here become the subdeliveries that are a part of the parent delivery package.

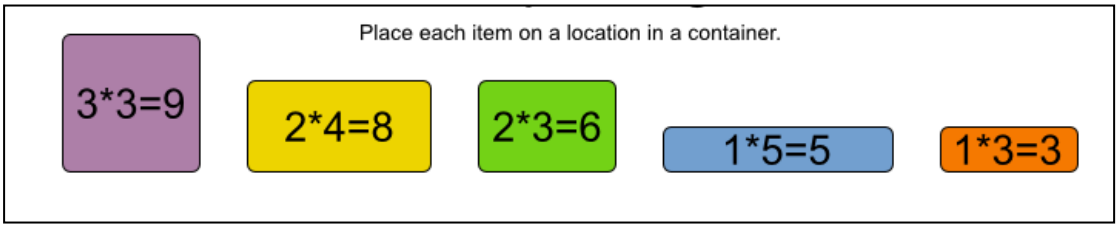

Fig. 4.3: Bin Packing Problem

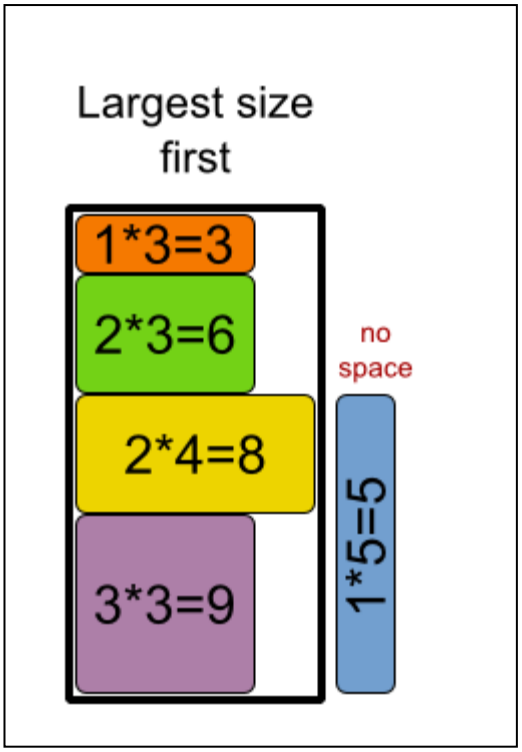

Fig. 4.4: Largest Size First Solution

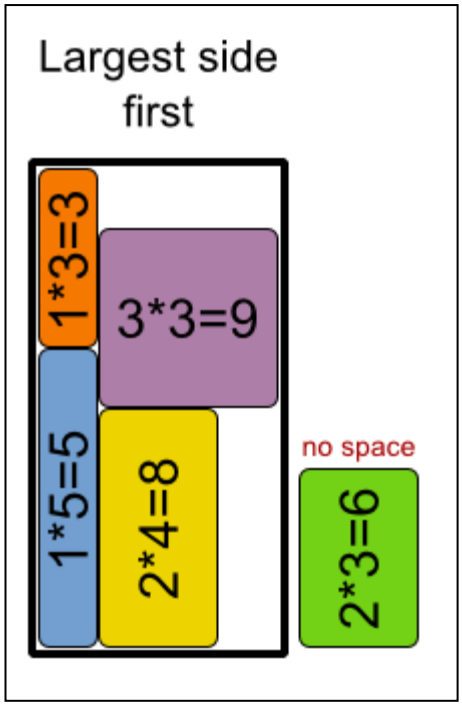

Fig. 4.5: Largest Side First Solution

The tool checks if the requested delivery is within a certain amount which is within the capability of the infrastructure to handle and transfer over the network, then the delivery is sent over the network or else it is broken down into smaller deliveries to scrap out the possibility of longer waiting times and failures due to unavailability of the required bandwidth. This enhances the chances of successful upload over the network on the target server and reduces the overhead of handling failures during upload and transfer.

Delivery Status: The user can anytime enquire about the status of the delivery with the help of tool API through command line. The tool returns the status of the delivery process and also gives the result of status of the query as the ongoing phase. It is a very important step which provides detailed information about the phase of the delivery process and keeps the user informed in case of any failure which has blocked the delivery. This is helpful in cases when the failure goes unreported and the requester keeps on waiting for the delivery upload phase to be completed.

LDAP Server: This server in the infrastructure is dedicated to storing credentials of the users/requesters so as to ensure proper authentication mechanisms are exercised

upon the users working with the tool and requesting a delivery. This is the authorization layer implemented in the form the LDAP server. The LDAP Server stores all the authorization details of the all the concerned personnel in order to maintain transparency and add an additional layer of security which is a crucial factor for every organization. It is very crucial to maintain this login information which distinguishes the authorized and the unauthorized users who possess the potential of doing harm to the entire organization by gaining access to the restricted data. The confidential information is the asset which earns profit for enterprise. It adds value to the organization.

Metadata Server: This store all the Meta information about the content uploaded on the FTP server. This is updated with the requester and the user details along with the timestamp of the upload. It keeps track of what all is uploaded along with the versioning and size of the uploaded products. It has an authorization layer associated with it which ensures that only authenticated users are allowed to access the data about the upload. The authorization layers works by asking the user to login with the assigned credentials and if there is a match with the ones stored in the database then, the access is given. If there is no match, the user/requester trying to login is blocked to further actions. This authorization layer enhances security and restricts unauthorized entry into the delivery systems. Any sensitive information or data is not compromised, since, no attacker can gain access into the system.

Requester Authentication: The authentication of the requester is a crucial step in maintaining the security of data by avoiding unauthorized access. The authorization layer ensures the cross checking of requester details with the data stored on the LDAP server. Only the authenticated requesters are further given authorization to access the uploaded delivery. This rules out the possibility of attackers getting into the system and dealing with sensitive information causing a security breach into the system.

Upload check: This is a check done to ensure the requested delivery is uploaded and there is no scope of any error or failure during transfer over the network or while uploading on the FTP server. These checks are necessary in order to make it easy for the requester as well as the user to keep track of the status of the requested delivery. Upload checks prevents the situation of deadlock within the process and

unnecessary latency in the delivery time and response time for the requester.

Delivery Notification: The requester of the delivery is notified by the user of the status of the delivery. After the final phase of delivery builder, user sends an email to the requester informing about the successful upload of requested product. Post this, requester accesses the delivery by authenticating. The requester can install the product only if he/she has a valid license. The license key is required by the installing agent to accomplish download and installation of the software product.

4.2 Localized Approach of Delivery

The localization of the delivery process is about involving the requester in the delivery process. In the centralized scenario, the requester just sends a delivery request mail and then waits for the software package to be delivered. The request is sent to centralized portal of distribution of products which is handled by the centralized server. This is a case of too much of bandwidth requirement and faces the issue of single point of failure due to the centralization approach. While the proposed localized approach, facilitates the delivery in a local mode in which the requester files a request, the package undergoes some scrutiny and then uploaded to a server. The requester has got work to do and takes part in the software delivery mechanism by downloading from the server which follows mirroring architecture.

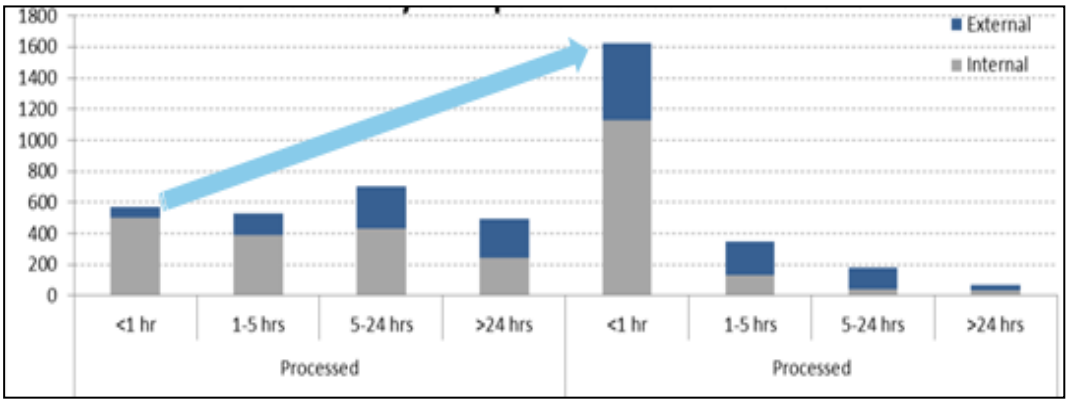

Fig. 4.6: Centralized Vs Localized Approach Delivery Times

The above figure illustrates the difference in the delivery times in the centralized and the localized approach. There is huge increase in the number of deliveries that were successfully completed within an hour of request filing in case of localized

methodology. There are about 1600 deliveries that are completed within an hour and the requester is notified of the completion. This greatly boosts up the performance by decrement in the delivery time to a great extent. The external and internal terminology represent the internal i.e. the customers within the organizations but different departments and the external i.e. the clients that are external to the organization which deal with the providers on licensing terms and conditions.

Chapter 5

SCENARIO OF DISTRIBUTION OF PRODUCTS

5.1 Delivery Request Flow

This section throws light on the flow of each delivery from the very first step of requesting it to the final delivery of the product. The above described architecture forms the basis of each requested delivery.

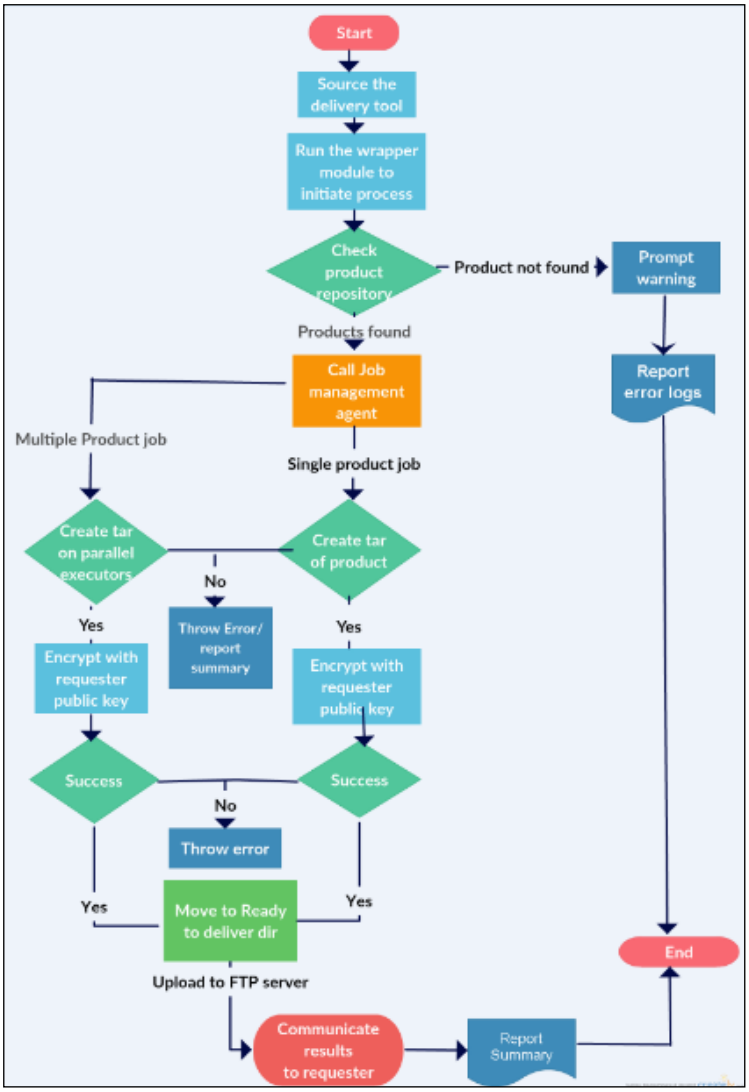

Fig. 5.1: Delivery Request Scenario

The process of delivering a software product constitutes of the following scenario:

1. The user gets a delivery request mail from the requester giving details about the product and version required.

2. The user authenticates the requester if he/she has the authority to access the request made.

3. If the authentication step of the requester fails, the requester is blocked from further access and a mail is sent notifying the requester about the credential match failure step and asking him to check his login information.

4. The user provides the requester with full support of the delivery tool.

5. After the authentication mechanism, the user initiates the building up of the delivery.

6. The delivery builder tool is sourced in the environment and all the prerequisites applications are sourced in a centralized methodology which brings them to use whenever they need to be executed in the executable form.

7. Now, after the environment is prepared, wrapper module is fired with tool API. The wrapper module launches the start command of the tool which turns on the delivery tool to start the delivery process.

8. The required product is checked with the Meta data at the metadata Server to see if the product is already uploaded on the FTP server.

9. If the product is already uploaded on the FTP server, then the access rights of the requester corresponding to the details stored over the LDAP server are checked and then given access to the FTP server for downloading the delivery and further installing it using the installer attached.

10. If the delivery is not present on the FTP server, then the tool checks if the product is present in the install zones or repository. If not found, throws an error along with the report summary to go through to the process logs.

11. If found, then the size of the delivery is checked.

12. The size of delivery determines the process of upload.

13. If the delivery size is within 400 GB, then the product is encrypted with the requester public key and moved to the target directory, ready to be uploaded to the FTP server.

14. If the size is found to be greater than 400 GB, then the delivery is broken down into multiple sub deliveries, encrypted and then moved to the target directory.

15. The sub deliveries are then uploaded to the FTP server in parallel in order to gain efficiency and limit bandwidth utilization.

16. Then the sub deliveries are assembled and acknowledged at the server. The information on the Metadata server is updated.

17. The requester is sent the delivery notification post which verifying his/her access rights, the delivery can be downloaded from the server.

The product is uploaded along with the installer using which the installing agent presents at the requester machine extracts and installs the software product.

5.2 Methodology Used

Selecting a development life cycle methodology is a challenging task. It is considered to be a crucial decision because a good methodology adds value to the organization developing the software. Due to the inclination towards the time-to-market conditions, organizations have started working to deliver the required needed solutions and services faster. Today's business world is all about accomplishing things in a better, cheaper and faster way. The methodology used determines the value proposition to an organization. It is important for the organization to utilize the SDLC [39] methodology that synchronizes with its objectives so as to achieve the desired results.

As per the Decision cube **[**40**],** a framework used for selecting the development methodology, the most suitable SDLC model is Agile Model of development. It is a amalgamation of iterative and incremental process models. The whole development of Digital Delivery Builder is divided into multiple builds and these builds are developed

in iterations. After each iteration, a functionality that is required is incorporated in the build. At the end of the final time box i.e. the last iteration, the product holds all the required features as per the product specification. The major advantage of using agile methodology is that it delivers early prototypes for initial demonstration and provides flexibility to the developers.

Another advantage to selecting this methodology [41] is that it is suitable for consistently changing requirements and minimal resource utilization.

Chapter 6

CRITERIA FOR QUALITY OF DELIVERY SERVICES

The criteria that determines the quality of the software distribution services is based on the requirements of both the software requester or the end user and the software provider i.e. the one delivers the software product. There are certain requirements that need to be fulfilled in order to ensure the quality of the products is maintained while delivering.

6.1 Software Provider Requirements

The software Provider requirements that need to be achieved in order to maintain quality of the software product are:

- It is must to efficiently deliver the software to the end user with the minimum amount of effort and cost.
- It should be ensured that good quality of service is provided to the customers in the terms of reduced transfer latencies, great response time, flexibility in the service by the provider, security and integrity in the service provided by the provider and provision of support.
- It is guaranteed that the delivery is carried out in a secure manner and the delivery package reaches the destination.
- The scope of unauthorized access and unauthenticated distribution needs to be diminished.
- The system maintenance cost needs to be lowered as a requirement.
- It should provide the provision of integration of the process of delivery with the other available complex legacy systems and certain data whenever required.
- The cost of the maintenance of the system needs to be lowered.
- The management of the system is supposed to be an easy not a daunting task.

6.2 Requester Requirements

The requirements of the end user which classify the entire process to be a good quality system are:

- The system should be void of complexity and should promote user-friendliness.
- It should have a transparent system functioning that promotes a clear understanding of the entire distribution process.
- It may allow scheduling of delivery of software packages and customization of the distribution features of the process.
- The system should consume bandwidth and not depriving other applications of the network bandwidth. There should be a limited bandwidth usage requirement which doesn't choke the entire network bandwidth.
- The distribution system should not be in the condition of resource deadlock with other applications and non-interfering in terms of hardware resource conflicts.
- It should be secure and privacy feature should be provided.
- The system lends the support for configuration changes, its management and administration.
- The system should promote integrity of the input provided and delivers it as it is with no alterations whatsoever.

6.3 Existing Software Distribution Techniques

Software Distribution can be achieved through different techniques and methods. Software Distribution began with the physical transfer of the products and now transfers by exploiting the invention of internet. Another technique of software delivery is to adopt web-based HTTP downloads. This mode of software distribution is still prevalent these days. The main issues that shake the core of the delivery process of software delivery are the bandwidth issue, performance, delivery time and other system parameters. The issues are accompanied with the bottlenecks that are

associated with the flooding of delivery request from the end users to the providers. The performance of the delivery system is determined even by the increase in the number of internet users and their internet connectivity speed.

It is advantageous to deliver products without involving the third party transfers [3]. Software Providers keep track of the information of the clients and the requests in order to be able to provide better support and maintenance services. Studies have shown that there is a sense of dissatisfaction amount the clients due to the fact that typically the annual cost of maintenance and support service is 20% of the licensing cost which accounts to a lot of amount. Investing so much of amount and still not getting the required support for maintenance is reason for resentment among the customers. With the capability to lend tremendous amount of support and maintenance, the satisfaction levels of the customers can be hiked. This paves the way to loyalty of the customer and, hence, their retention. Studies have shown that even a mere 5% increment in the retention of the customers can lead to increment in the profits of the organization of about 85-90%. So, it is very crucial to work towards the satisfaction of the customers. Hence, satisfaction leads to the retention of customers to the organizations.

Another technical issue associated with the software delivery is about the complexity about the client-specific configuration of the software. This requires each customer specific configuration and tuning which allows personalization of the system as per the requirements of the client. This can be practiced through an effective collaboration between the client and the software provider. Hence, it can be rightly said that it is so advantageous to not involve the third party for the delivery mechanisms. However, it is very crucial to be able to gain performance and scalability while hand in hand with not consuming the entire bandwidth and choking the network. At times, involving a third party for delivery transfers exposes the system to certain risks of leaking of sensitive information.

Table 6.1: Comparision of Existing Software Delivery Techniques

	Physical	E-mail	HTTP	File Network Transfer
Bandwidth Optimization				
Scalability	✓			
Security	✓			
Tracking Data			✓	✓
Customer support				
Performance				
Reporting				
Timeliness				
Reliability				
User notifications				
File size issues handling	✓		✓	✓
File type issues handling	✓	✓	✓	✓
Ease of system integration			✓	✓
Load Balancing				
Web self service facility			✓	✓
Flexibility/Convenience				
Ease of use		✓	✓	✓
Availability				
Scheduling of delivery		✓		
Low maintenance costs		✓	✓	✓
Privacy	✓	✓		
Speed of delivery		✓	✓	✓
User friendliness		✓	✓	✓
Documentation	✓			

It illustrates the criteria that determine the quality of the software distribution process. Usually, very few conditions that keep up the quality of distribution are met. It can be deduced from the above table that the delivery techniques don't fulfill certain

conditions that are relevant to maintain the quality of the delivery process. Many of the conditions are blank indicating those criteria are not met and achieved by the above mentioned delivery methodologies.

It is evident that important conditions like Fault tolerance and reliability are compromised by the above mentioned delivery techniques. There is a lack of customer support which lead to failure in achieving contentment of customers and hence, losing customers. The overall performance graph of the techniques does not seem to be in a good shape as described above.

All the above conditions describe the scenario of software distribution and highlight the need to work towards the invention of efficient delivery mechanisms with high performance and customer satisfaction.

6.4 Software Delivery Classification

Software Delivery can be accomplished through physical means and electronically. With the advent of internet connectivity, it has become feasible to achieve the distribution of the software product over the network. The software product from the software provider reaches the customer: through physical transfer and by electronic means. The old fashioned means of physically transferring the software product is still widely used across the world but is considered to be an ill practice due to the overhead incurred in terms of delay and additional cost.

Table 6.2: Comparing Different Software Delivery Techniques

	L. Han [19]	Joanne Correia [42]	M. Dillinger [43]	D. Stolarz [44]
Security/Privacy	✓	✓	✓	
Tracking data/Reporting	✓	✓		
Customer Support		✓		
User Notifications		✓		
Centralized handling			✓	
Fault Tolerance				✓
Reliability				✓
Performance/Timeliness	✓		✓	✓
Bandwidth & Scalability	✓		✓	✓
Speed	✓		✓	✓
Digital rights management		✓		✓
Maintenance Costs	✓	✓	✓	✓
Personalization/Customization			✓	✓
Inventory/Repository control		✓		
Load Balancing potential			✓	
User Convenience	✓			✓
Availability	✓			
Scheduling delivery	✓			
File Split			✓	
Reinstallation of product	✓			
Ease of use/User friendliness	✓			

Chapter 7 CONCLUSION AND FUTURE SCOPE

7.1 Conclusion

The proposed approach leads to an efficient delivery of software products by using the algorithm that improves performance and throughput. The number of failures is dragged downwards and efficiency is elevated. It aims at providing a hassle free, well monitored and a localized way of delivering software products to customers. The requested deliveries are uploaded over the server followed by downloading of products from the server after the authorization of the requester. The overall mechanism leads to a better solution for facilitating delivery.

The bandwidth related failures can be decreased to a great extent as the heavy deliveries are well taken care of by breaking the main delivery into child deliveries and then dealing with each as a different one. This reduces the waiting times of requesters as delivery uploads are done in parallel. The local approach of the tool is also a beneficial aspect of the delivery. The overall load of requests on the central delivery request portal led to freezing of the network and low availability of the network bandwidth. This makes the local tooling approach much better way to go about the electronic software delivery process.

The overall delivery time is reduced to a great extent. The localized approach has decreased the delivery time for requests on an average to an hour. This greatly improves the performance of the entire software distribution process. The entire process is documented so as to understand the entire process in a broader sense and even look back at the previous deliveries whenever the need arises. The authentication layer added in the process restricts the entry into the system. The unauthorized access is not given to any entity which strengthens the security and privacy aspect of the delivery mechanism. The ease of use is promoted for the easy operatibility for the customer and clients. The user friendliness of the system gives confidence to the user and then sows the seeds of content.

The scalability criteria are fulfilled by this methodology as there are very minor chances of failure in case of heavy load of requests. The requests are sent through mail but the requests are serviced in a local environment rather than on a centralized

server. This improves the performance measure of the delivery process.

Failure reporting is an important aspect of the local builder tool. Unlike other methodologies, it doesn't block in case of the situation of failure in the process. The failure is detected and the point of failure is reported back to the requester. This rules out the possibility of infinite loop of waiting for the delivery to reach the finish line. The user is also provided with the functionality to query the state of the system at any point of time. The query returns the currently running phase of the delivery process. This status of delivery is a check which improves the overall experience of the customer.

7.2 Future Scope

The type of the methodology used in the development of a software product directly determines the quality of the software product developed. The agile methodology used for the development of the digital delivery builder tool turns the product into a decent quality of software.

It can be further explored to develop the product using new techniques which support the development procedure to get the desired results. Agile model can be backed by Scrum technique. The agile methodology along with scrum technique can be applied to the development process. Scrum technology is based on real world scenario and results and not backed by speculation. Scrum technology is based on the fact that customer requirements are dynamic in nature and tend to change consistently. Scrum comes into picture when it is known that unpredictable issues will be coming in the way of development and any planned or decided approach doesn't seem to work. Scrum understands that it is not always possible to understand all the requirements of the customers, hence, adopts an empirical approach that drives the team members to deliver rapidly and to handle situation of changing need and demands.

REFERENCES

[1] R. S. Pressman, *Software Engineering: a practitioner's approach*. Palgrave Macmillan, 2005.

[2] N. Juristo and S. Vegas, "Analyzing software engineering experiments: everything you always wanted to know but were afraid to ask," *in Proceedings of the 38th International Conference on Software Engineering Companion*. ACM, 2016, pp. 900–901

[3] S. H. Kan, "Metrics and models in software quality engineering", Addison-Wesley Longman Publishing Co., Inc., 2002.

[4] I. Jacobson, G. Booch, J. Rumbaugh, J. Rumbaugh, and G. Booch, The unified software development process. *Addison-Wesley Reading*, 1999, vol. 1.

[5] C.-C. Huang, "Overview of modular product development," *Proceedings-National Science Council Republic of China Part a Physical Science and Engineering*, vol. 24, no. 3, pp. 149–165, 2000.

[6] LY. Ilan, "The economics of software distribution over the internet revisited," *First Monday*, vol. 6, no. 12, 2001.

[7] W. J. Mirville, D. G. Cole, and M. L. Nelson, "A network-based software distribution system for smaller organizations," *Journal of Computing Sciences in Colleges*, vol. 20, no. 2, pp. 311–317, 2004.

[8] V. S. Sharma, V. Kaulgud, and P. Duraisamy, "A gamification approach for distributed agile delivery," in *Proceedings of the 5th International Workshop on Games and Software Engineering*. ACM, 2016, pp. 42–45.

[9] T. J. Collins III, S. R. Anderson, S. J. McDowall, C. H. Kratsch, and J. P. Larson, "System for software distribution in a digital computer network," Dec. 1 1998, US Patent 5,845,090.

[10] J. F. I. Donald, "Implementing a parallel file transfer protocol," Jul. 4 2000, US Patent 6,085,251.

[11] L. Han, “Secure and scalable e-service software delivery,” Ph.D. dissertation, Link¨opings universitet, 2001.

[12] M. A. Cusumano, “The changing software business: Moving from products to services,” *Computer*, vol. 41, no. 1, 2008.

[13] S. Alter, “Service system fundamentals: Work system, value chain, and life cycle,” *IBM systems journal*, vol. 47, no. 1, pp. 71–85, 2008.

[14] A. Mishra and D. Dubey, “A comparative study of different software development life cycle models in different scenarios,” *International Journal of Advance research in computer science and management studies*, 2013.

[15] R. L. Kissel, K. M. Stine, M. A. Scholl, H. Rossman, J. Fahlsing, and J. Gulick, “Security considerations in the system development life cycle,” *Special Publication (NIST SP)*-800-64 Rev 2, 2008.

[16] M. Tuteja and G. Dubey, “A research study on importance of testing and quality assurance in software development life cycle (sdlc) models,” *International Journal of Soft Computing and Engineering (IJSCE)*, vol. 2, no. 3, pp. 251–257, 2012.

[17] J. Highsmith and A. Cockburn, “Agile software development: The business of innovation,” *Computer*, vol. 34, no. 9, pp. 120–127, 2001.

[18] A. Cockburn, *Agile software development*. Addison-Wesley Boston, 2002, vol. 177.

[19] L. Han and N. Shahmehri, “Secure multicast software delivery,” *in Proceedings of IEEE 9th International Workshop, Enabling Technologies: Infrastructure for Collaborative Enterprise, 2000 (WET ICE 2000)*, pp. 207–212.

[20] R. S. Hall, D. Heimberger, A. Van Der Hoek, and A. L. Wolf, “The software dock: A distributed, agent-based software deployment system,” COLORADO UNIV AT BOULDER DEPT OF COMPUTER SCIENCE, Tech. Rep., 1997.

[21] P. K. Gummadi, S. Saroiu, and S. D. Gribble, "A measurement study of napster and gnutella as examples of peer-to-peer file sharing systems," *ACM SIGCOMM Computer Communication Review*, vol. 32, no. 1, pp. 82–82, 2002.

[22] K. P. Gummadi, R. J. Dunn, S. Saroiu, S. D. Gribble, H. M. Levy, and J. Zahorjan, "Measurement, modeling, and analysis of a peer-to-peer file-sharing workload," *ACM SIGOPS Operating Systems Review*, vol. 37, no. 5, pp. 314–329, 2003.

[23] B. Krishnamurthy, C. Wills, and Y. Zhang, "On the use and performance of content distribution networks," *in Proceedings of the 1st ACM SIGCOMM Workshop on Internet Measurement.* ACM, 2001, pp. 169–182.

[24] K. L. Johnson, J. F. Carr, M. S. Day, and M. F. Kaashoek, "The assured performance of content distribution networks," *Computer Communications*, vol. 24, no. 2, pp. 202–206, 2001.

[25] D. Xu, S. S. Kulkarni, C. Rosenberg, and H.-K. Chai, "Analysis of a CDN–P2P hybrid architecture for cost-effective streaming media distribution," *Multimedia Systems*, vol. 11, no. 4, pp. 383–399, 2006.

[26] C. Wills and H. Shang, "The contribution of DNS lookup costs to web object retrieval," Tech. Rep. TR-00-12, Worcester Polytechnic Institute, Tech. Rep., 2000.

[27] S. Saroiu, P. K. Gummadi, S. D. Gribble et al., "A measurement study of peer-to-peer file sharing systems," *in proceedings of Multimedia Computing and Networking*, vol. 2002, 2002, p. 152.

[28] V. N. Padmanabhan, H. J. Wang, and P. A. Chou, "Resilient peer-to-peer streaming," in *Network Protocols, 2003. Proceedings. 11th IEEE International Conference on*. IEEE, 2003, pp. 16–27.

[29] L. Gong, "JXTA: A network programming environment," *IEEE Internet Computing*, vol. 5, no. 3, pp. 88–95, 2001.

[30] W. Yeager and J. Williams, "Secure peer-to-peer networking: the JXTA example," *IT professional*, vol. 4, no. 2, pp. 53–57, 2002.

[31] Z. Li, Y. Dong, L. Zhuang, and J. Huang, "Implementation of secure peer group in peer-to-peer network," *in Communication Technology Proceedings, 2003. ICCT 2003. International Conference on*, vol. 1. IEEE, 2003, pp. 192–195.

[32] O. Ritchie and K. Thompson, "The unix time-sharing system," *The Bell System Technical Journal*, vol. 57, no. 6, pp. 1905–1929, 1978.

[33] https://www.cpan.org/

[34] L. Stein, "Network programming with Perl". Addison-Wesley Longman Publishing Co., Inc., 2000.

[35] B. K. Hart, "Electronic delivery of software: implementation of a robust, effective solution," *in Proceedings of the 30th annual ACM SIGUCCS conference on User services*, ACM, 2002, pp. 177–178.

[36] S. Kaur and M. Mahajan, "Re-usability of constraints for test case generation in different applications using genetic algorithm," *International Journal of Computer Science and Network Security (IJCSNS)*, vol. 16, no. 5, p. 113, 2016

[37] G. R. Lanzy, F. A. Pflug, and G. H. Stange, "System and method for enabling and controlling anonymous file transfer protocol communications," Jul. 18 2000, US Patent 6,092,198.

[38] C.-C. Lee and D.-T. Lee, "A simple on-line bin-packing algorithm," *Journal of the ACM (JACM)*, vol. 32, no. 3, pp. 562–572, 1985.

[39] S. Balaji and M. S. Murugaiyan, "Waterfall vs.V-model vs. Agile: A comparative study on SDLC," *International Journal of Information Technology and Business Management*, vol. 2, no. 1, pp. 26–30, 2012.

[40] http://www.droancollegeuk.com/downloads/sdlc_methodology.pdf

[41] S. M. Al-Saleem and H. Ullah, "A comparative analysis and evaluation of different agile software development methodologies," *International Journal of Computer Science and Network Security (IJCSNS)*, vol. 15, no. 7, p. 39, 2015.

[42] Joanne Correia, " How Can ESD Help Cut Distribution Expenses and Retain Customers?" Gartner Research, October 2001.

https://www.gartner.com/doc/345363/esd-help-cut-distribution-expenses

[43] M. Dillinger and R. Becher, "Decentralized software distribution for SDR terminals," *IEEE Wireless Communications*, vol. 9, no. 2, pp. 20–25, 2002.

[44] D. Stolarz, "Peer-to-peer streaming media delivery," *in Peer-to-Peer Computing, 2001. Proceedings. First International Conference on.* IEEE, 2001, pp. 48–52.

LIST OF PUBLICATIONS

[1] D. Hooda and R. Rani, "A Novel Localized Approach for Efficient delivery of Software Products', In Proceedings of IEEE Fourth International Conference on Signal Processing, Computing and Control, ISPCC-2017, September 21-23, 2017. **[Accepted]**