Planning and Management Issues in Software Development

By

Srireshmi Nittala

Submitted to the Department of Information and Communication Technologies

On June 2, 2017

In partial fulfilment of the requirements for the degree of Masters in Information and Communication Technologies

Abstract

Software Development is never an exact science, but when it combines with planning and management, you have an answer for building effective software system. Software development sector has witnessed many failures due to the lack of planning and management. There are many reasons behind the failure like over budget, over time taking, low customer satisfaction etc. This paper discusses the Management and planning issues related to time, finance, man power, and life cycle flaws involved in development of software.

Table of Contents

Introduction

Software development is the process of designing computer program or any software related project, documenting the steps involved in the life cycle, testing, and bug fixing which involves creating and maintaining applications and frameworks resulting in a software product. In a wider perspective, developing software is conception of desired model, which happens, in a planned and structured process. Software development is a process implemented for various purposes. Three most prominent reasons include, meeting the specific needs of a client or a business, to achieve precise needs of a group of potential users, and for personal use. It is not enough for executives to list out reasons to develop software. Management and Planning play a vital role in completing projects successfully.

Successful management of a software project can be achieved by utilizing popular software development life cycle models. Out of the many models, “Water fall” model is the traditional version followed by many companies over the years and even today. ‘Agile software development’ is another fail-fast methodology used across the software industry. These life cycle models help in controlling the process of development and it ensures systematic delivery of the desired software product. Some of the common steps involved in SDLC model include:

1. Gathering requirements
2. Design Analysis
3. Implementation or Development
4. Testing
5. Deployment

6. Maintenance

Almost every SDLC model follows the above steps, which ensures a methodical approach to the developed software. In software development, planning is another constraint that affects the quality of the project. Though management is one of the keys in software development, planning plays an equally important role in successful execution of a software project.

Software Development Plan is a step taken to gather all of the necessary information to control the project. It describes the approach to the development of the software, and is the top-level plan generated and used by the managers to direct the development effort. Software development plan (SDP) describes a developer's plan for conducting a software developing effort. SDP provides an insight for monitoring the processes in the software development. The planning should adhere to the contractor's preferred structure and timeline. It should document all processes applicable to the system, at a level of detail sufficient to allow the use of the SDP as the full guidance for the developers. Planning in software development must ensure that the following parameters are considered.

1. Having a proper plan introduction
2. Clear view on objectives and purpose of project
3. Relevant documentation
4. Identification of all software and software products to which the software development plan applies.
5. Timelines to perform project development activities

Evolution of Software Development

Software and Hardware are the two prominent kinds of technologies in computational world. Software development has started in late 1940's and has gone through many stages of evolution. In the book "*Great Software Debates,"* subchapter "*The Missing Piece of Software Development"* Alan. M Davis states, "Great software does not come from tools, it comes from people." The use of software development as a business process has a long human history attached to it. Indeed, the building of the Egyptian pyramids is believed by many to have been assisted by the use of simple project management and planning principles. For much of the history of management, the predominant application type similar to today's software development practices were seen in engineering and construction projects like building roads, bridges, and skyscrapers. This was still the case when formalization of software management and planning occurred in the 1960s with the help of new computing power also known as hardware.

A particularly impressive software management achievement at that time was the Apollo moon-landing project (1961-1969), which required the coordination of about 410,000 workers at a cost of $25 billion in 1961 dollars, or $154 billion in 2011. Another impressive achievement is the organization of the Olympic Games using management and planning techniques. These are examples of events, where the project deadline is fixed and violation of timelines is considered a major threat to the project. These successes were achieved for a narrow range of applications. However, the potential for effective management and planning in software development that could be applied to a much wider spectrum of applications gradually became apparent.

One important factor that boosts the development of software throughout the ages was that new and improved computers were coming out into the market at an unprecedented rate. The development of computer hardware technology demanded that the software be as good, fast, and reliable as the hardware itself. During the early ages, a computer was not built to be mobile and scalable, and rather they were huge machines and did not have the speed or the reliability of modern computers. With advancements in data processing, the need for faster, reliable, and smaller computers was felt throughout the computing world. As time went on computers began to shrink in size but expanded in speed, reliability, and performance. With the introduction of better hardware technology, software developed into new heights. These new software were not only reliable and fast, they were user-friendly too. Initially, hardware vendors freely gave away system software due to the difficulty in selling hardware without the software. With the increased popularity and efficiency of software, the methods to develop the software also slowly evolved.

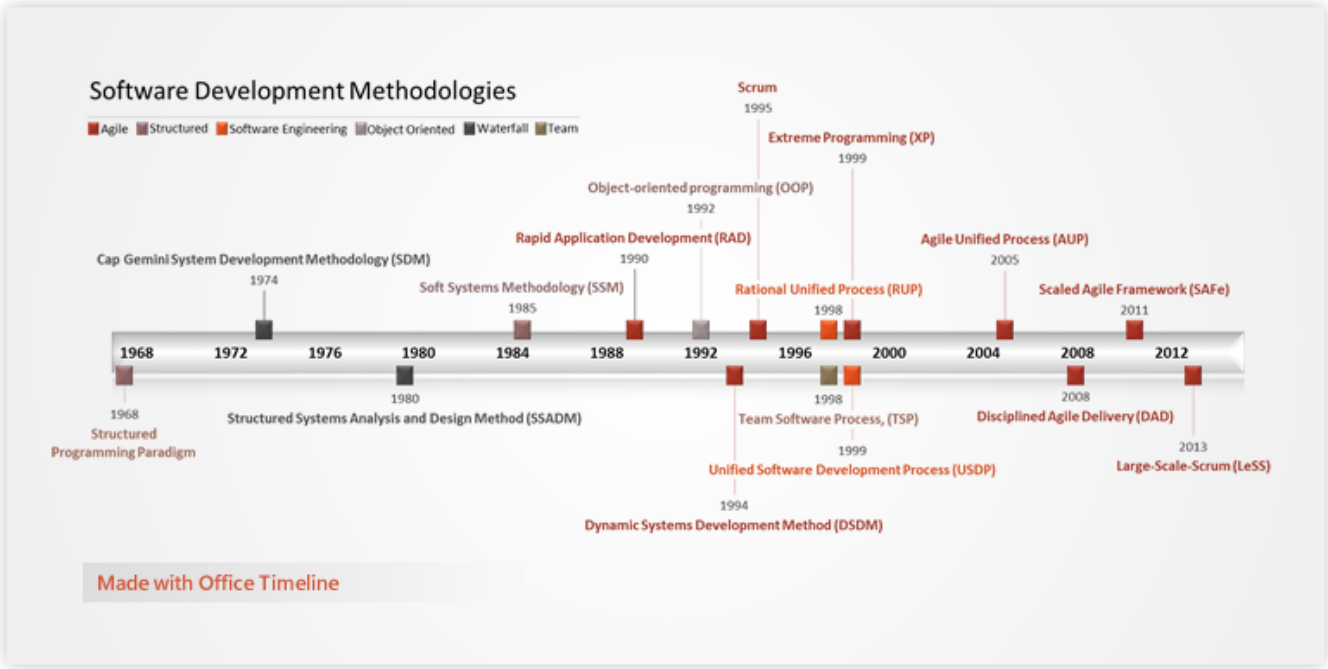

Figure 1: Evolution timeline of SDLC methodologies

List of Issues in today's software development

Software development is a dynamic industry and there are always emerging issues and challenges in this domain. Here are some of the major challenges faced by the software industry.

1. Understanding the behaviour of the customer that uses a given software product.
2. Integrating different systems and technologies that work seamlessly regardless of platforms and devices used by end customers.
3. Hiring the right skilled person is a unique challenge for companies and projects.
4. Maintenance of analogous Quality, Test, Development, and Production environments can be very difficult depending on the project size, frameworks, and software stacks used.
5. Sometimes requirements change during every phase of software project life cycle. Communication with the clients regarding dynamic requirements can be a challenge.
6. Not having a planned process for the entire SDLC increases waiting and queuing time. This will in turn lead to delays in delivery that will eventually cause a bad reputation to the software development company.
7. Lack of planning and knowledge regarding business and the value can cause failure to develop the software accurately with all the business rules.
8. Security issues: With increased risk in deploying applications to the Cloud, the developers must integrate security features into the application. Failure to do so can cause business loss due to hacking, phishing, and other cyber crimes.
9. Complexity in understanding emerging techniques and frame works.
10. Collaboration problems can occur due to utilization of legacy communication methods.

Solutions to Overcome Issues in Software Development

As we have come across a numerous problems in software planning and development, there exist balanced solutions to overcome them. Though they are not completely accurate, for ideal software, following the techniques mentioned in this segment may resolve the discrepancies in developing software projects. According to researches, 70% of software project failures are due to issues related to over budget, delay in functionality and poor execution. Due to these factors, some projects are prone to deprecation during or at the time of completion. This segment mainly triggers the best solutions for software planning and development and they as follows:

1. **Choosing appropriate development process**

   Choosing appropriate development process plays a significant role in project execution because majority of the activities in software project are reliant on appropriate development process. In today's modern software development market, most of the companies are opting for hybrid risk-driven process models such as spiral-model over the traditional waterfall model. There are various other choices including Agile, Rational Unified Process (RUP), IBM method, Global Services Method, and extreme Programming (XP). These development processes help in giving better direction to the project.

2. **Having good expertise**

   After the team decides on a particular in-house development solution or any development process mentioned above, the team should always make sure that they have good knowledge and expertise in executing the plan. Failure to have product owners with expertise can result in unsuccessful implementation.

3. Gathering fundamental requirements

Working, agreeing and gathering on requirements is the fundamental ingredient of project success. It is not necessary to meet all the requirements before the architecture, design and coding is complete. However, it is important for the software development team to know and understand the features before commencing the development process. Generally, there exist two categories of requirements - 'Functional' and 'Non-functional'. Requirement gathering for functional module involves use cases and the functional document primarily shows the various classes and objects that are to be present in the system. On the other hand, Non-functional requirements gathering document depicts the desired performance of the system and operational characteristics of the application. It is always important to gather these two modules as they have a major impact on the architecture of application, performance of the system and design characteristics.

4. Prominence of controlling cost

It is always difficult to judge the overall cost of software package. However, it is apparent that working with custom software development team can fetch estimated cost from the beginning. Additional costs may be encountered while adding extra features or when you speed up the software development timeline. There should be a room allotted to monitor overall cost flow in that project which helps the organization to cut unnecessary budget.

5. **Opting the better architecture**

   Successful companies always choose well-known industry architectures for their software development process. Organization should always ask their respective team members to review whether the project has major issues. If they find any, it is possible that the application contains a failed architecture practice. This is also known as anti-pattern concept. Many projects fail due to the 'anti-pattern' architecture of the application.

6. **Importance of communication**

   In software development, process communication is always important. Regardless of the size of the project, communication in all phases of s/w development is important. This helps in determining the progress of the process and it makes sure that it is going in right direction. Having right communication among teams will help respective teams to discuss pros and cons in the development, what is being implemented, changes in implementations, time constraints, adding of features etc.

7. **Giving equal importance to design**

   Over designing or under designing an application is a common issue overlooked during the design phase. There is a lot of probability that even with good architecture it is still possible to have bad design. The three primary rules to follow in order to attain good design are, 'keep it simple', 'object oriented patterns', and 'design good UML'. Reusing UML patterns extracted from successful models can help during the design phase.

8. **Code construction**

In total project development, code construction is a fraction of the total project effort, and it is also most visible. Other works like architecture, analysis, design, testing and requirements play equally prominent role. In case of programming, the best practices for constructing code are daily build and smoke tests, Continuous Integration, and self-testing code. In Continuous Integration, an automated process continuously looks at newly deployed code and runs builds on top of the new code. Continuous unit tests and integration tools like Jenkins have already gained popularity through the extreme programming (XP) technique. One can always use these practices on all kinds of projects. Project experts from different companies recommend using standard frame works to automate, build, and test, such as ANT and JUnit.

9. **Performing peer review sessions**

Reviewing others work in the process of project development helps in understanding the project progress. Experiences have shown that problems are eliminated earlier through peer reviews, which sometimes are considered more effective than testing. Artefacts from the development process are reviewed, including plans, requirements, architecture, design code, and test cases.

10. **Testing**

Testing should never be an afterthought. Some projects neglect testing when their deadlines get tighter or for any other ambiguities in project. It is an integral part of software development that needs to be planned. Testing should be done

proactively which means test cases are planned before coding starts, and test cases are meant to be developed while the application is being coded and designed.

11. Knowing more about configuration management

Configuration management is more than just controlling systems. Knowing about configuration management helps in managing and releasing distinct versions of the system. It can also help in separating the development and test environment dependencies so that they do not overwrite or interfere with each other.

12. Management of Defects and Quality

Establishing quality priorities and release criteria for the project is very important. This kind of planning will help the team to achieve quality software. Predetermining the potential defects can help in easy identification when they actually occur. Fix rate i.e., the number of bug fixes made to the code, is another attribute that helps measure the maturity of the code. Installing a defect tracking system to the source code can be very useful in source control management.

13. Need for planning deployment

Final stage of the releasing an application for the users is called deployment. If any project comes this far in an organization, it is a good sign of success. However, a few things could go wrong even at this phase. Planning for deployment must be a priority and teams should use a deployment checklist to resolve last minute issues and ensure a smooth deployment to production.

14. System operations and support

The support area is a vital factor to respond and resolve customer problems. Without the operations department, the project team will not be able to provide support to a new application. This could lead to unfriendly customer experience. To minimize the problems regarding support system, the issues that occur in the application database should be linked to the application defect tracking system.

15. Migration of data

There are cases where applications in the industry are not implemented from scratch. Developers enhance or rewrite the data from existing applications. Migration of data from existing data source is also a challenge. Teams should be careful in data migration process because it is not an easy task. Usually the new application has better business rules, and there are high expectations on quality of data. If one can improve the quality of data, there is a lot of scope for attracting businesses towards the application.

16. Security

Last but not the least; security is a major concern that should not be neglected during the software development. With the ever-increasing issues around security software developers must ensure that proper security standards are enabled as part of the application. Cloud Computing in particular poses as huge threat to software and hardware deployed and hosted in the cloud. Service Level Agreements and the standards must be followed before software implementation.

Other Factors and Standard development models

- In software development, planning and development is required. As we already know, software development is subject to budget and schedule constraints set by a particular organization. There are a few other distinctions that any team should be aware before developing any software project.
    - **The product is intangible:** By simply looking at the software artefact that is under construction, managers or developers cannot understand the progress.
    - **Each project is unique:** Ongoing projects are usually different in some ways from previous projects. Even managers who have good experience may find it difficult to anticipate problems. Though there are many factors involved, if software development follows the Iron triangle which involves understanding of scope, scheduling, and resource management, experts depict that it leads to better execution.

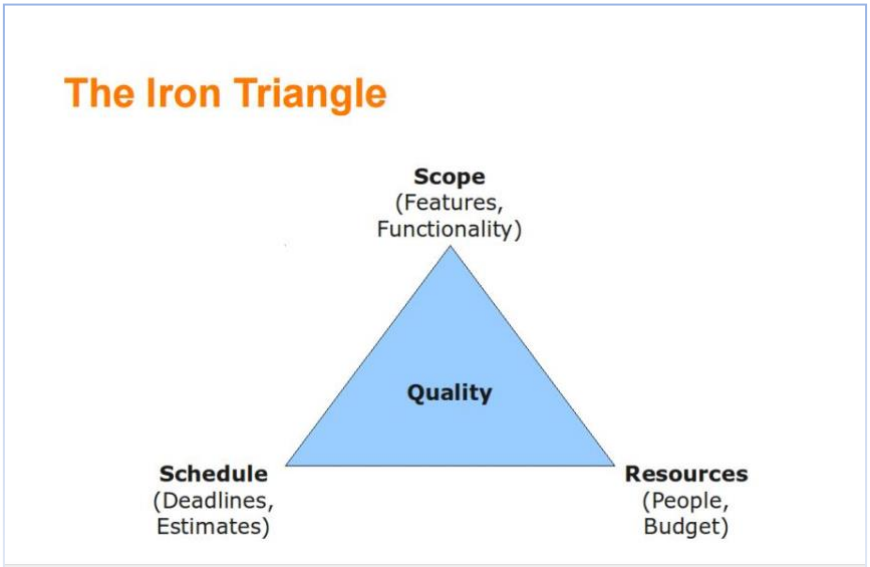

Figure 2: The Iron Triangle to maintain quality of software

- In addition, any development process can be measured using the industry standard known as Capability Maturity Model (CMM). This model describes a five level evolutionary path of organized and more matured process.
- Project Plan Supplement is a planning strategy used in software development. In this method, every stage of the project has its own planning document and strategy that helps in project reviews and budget and resource management.

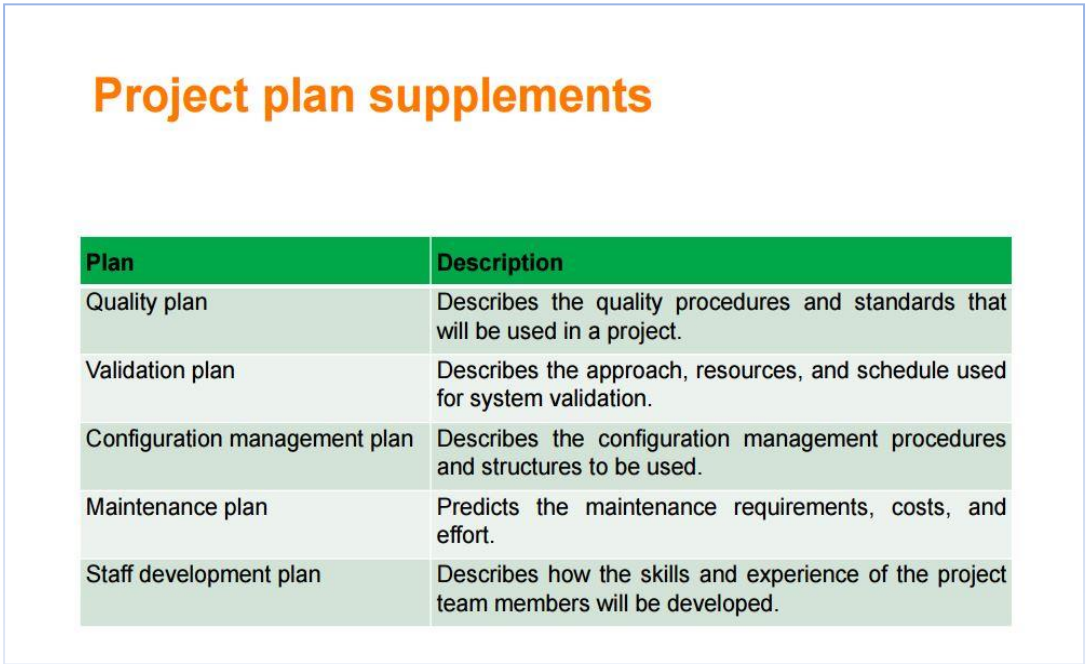

Figure 3: Project Plan Supplements

- **Agile methodology:** Final method to discuss in this segment is the agile method. Agile method has created a new wave in technology world, as this approach is totally reliable and different. Agile methodology has iterative approach towards software development and it is delivered to customers in increments because the decision on what to include in an increment depends on progress and on customer's priorities. When the customer priorities change regularly, this method accommodates a flexible plan to absorb these changes. This is the reason why it is not a plan-driven approach.

If any software development team implements the effective practices described above, then they could be well on the way in achieving a higher maturity level and successful project output. The above techniques help in managing a project given that all the requirements are certain to a degree. Yet another challenge posed to project management is when there are one or more factors of risk or uncertainty in the project.

Project management under uncertainty

A project risk, which occurs suddenly, is an uncertain factor that can considerably affect achievable performance in a project. For such circumstances, management must handle the situation responsibly by making appropriate decisions. The basic expertise of any management is the ability to make quick and significant decisions. The decisions made by management under uncertainty are immutable and it affects the resources. Outcomes of these decisions often fail and authorities judge these decisions.

In the initial stages of a software project, the manager would not find any kind of ambiguities or problems. They encounter surprises in the final aspects of the project when external events abnormally influence the delivery of the project. A simple uncertainty situation is represented in the graph below.

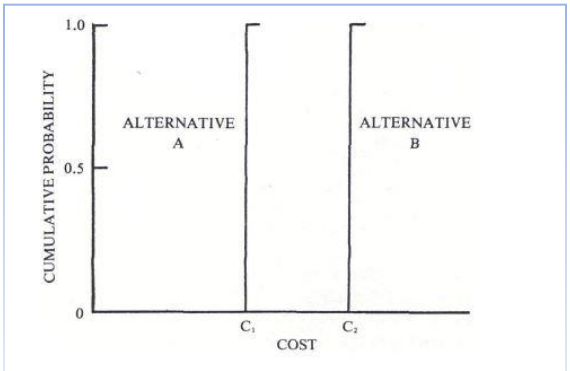

Figure 4: Uncertainty Explanation

In the above graph, executive management is asked to choose between two alternatives whose costs are presented as C_1 and C_2. In the project, if the objective is lowering the expenses, the management will choose 'Alternative A'. In other high budget scenarios, it is difficult to take a decision easily even if outcomes are certain.

Here is a simple illustration of other emerging factors in uncertainty management.

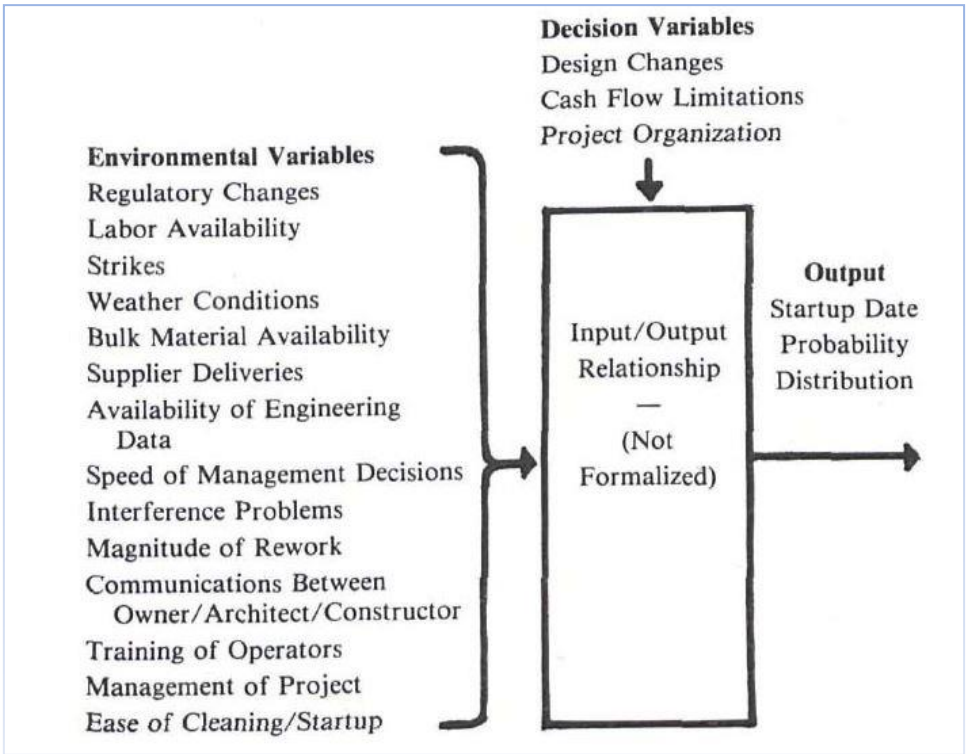

Figure 5: Subjective assessment

The above graph shows the possible uncertain conditions in software development and is also known as subjective assessment. This diagram helps in providing formalized methodology for quantifying uncertainty. This procedure allows an expert and a beginner in the same project to discuss the environmental factors and decision variables that can affect the outcome of the project.

Future Trends in software development

Technology trends always change very rapidly and so does software development. Constantly evolving tools and techniques for software development helps create more projects, more companies, and more products. According to *Digital Market Reports*, trends in software development lead to growth in the Information Technology sector.

- **Agile:** Usages of Agile will continue to steadily grow in next few years and it helps development teams to fail-quickly but at the same time make corrections quickly. In this way, it becomes a lot easier to push out multiple iterations of a product, rather than focusing on a single 'big bang' launch.
- **Continuous delivery** (CD) will also continue to play an important role. Though this method is less familiar than agile method, CD continuously focuses on early feedbacks so that corrections can be made earlier in the development phase.
- **Version control** acts as central pivot in future trends. Instead of just being a tool that software developers use to manage code, version control tools like bitbucket will be used across the industry for more complex source code management. Non-development teams is can use version controlling as well to maintain all assets correlated with project including diagrams, flowcharts, configuration files, web content etc.
- **Distributed Computing -** Sequential code construction leads to slower processes and workflows. As current libraries with in languages do not handle parallelism well, map reduce methods that are faster and run in a distributed fashion will be implemented in the code which helps in parallel execution of program across multiple servers also known in the big data world as data nodes. This will start to change the way people

think about programming from being a sequential flow-chart fashion to being parallel & data-driven.

- **Graphical programming** is another trend that has already penetrated the market and will continue to do so in the software development field. Graphical programming is when a program exists in visual format also known as visual programming language. It lets the user develop programs by manipulating program elements graphically rather than by specifying then textually. Visual drag and drop tools like Pentaho are already popular and used by ETL teams. Even the more complex fields like machine learning and artificial intelligence have seen tools like H20 and they will continue to behold an evolution of drag and drop tools for easier business executive friendly data modelling environment.

Conclusion

Software industry is undoubtedly a competitive diligence. Software development teams need to use every resource they possess, and should be able to acquire additional skills and knowledge from different sources. Development teams must adopt various models and improvements based on the project type. This document strongly focuses on the techniques that must be followed for successful management and planning when developing a software product. This paper clearly discusses some of the popular practices, methodologies, and models that must be used to create and deliver an efficient software product.

Bibliography

1. agilemanifesto.org. (2001). Manifesto for Agile Software Development. Available via DIALOG. http://agilemanifesto.org/

2. Nissenbaum, Helen. 2010. software In Context: Technology, software, and Integrity of Social Life. Stanford University Press.

3. Nayan B Ruperlia. May 3 2010. ACM SIGSOFT Software Engineering Notes, Volume 35, Issue

3 https://dl.acm.org/purchase.cfm?id=1764814&CFID=757176877&CFTOKEN=45364778

4. Kimi Fowder**,** Developing and Managing Embedded Systems and Products: Methods, Techniques
https://books.google.com/books?id=PDvLAwAAQBAJ&pg=PA125&lpg=PA125&dq=citation+for+software+development+and+management&source=bl&ots=930Zy9DONG&sig=BTVpF4W5BKpLO5PQbe2ElGkxsjE&hl=en&sa=X&ved=0ahUKEwiixsaNt9DTAhVp7IMKHexpABYQ6AEIbDAF#v=onepage&q=citation%20for%20software%20development%20and%20management&f=false

5. Tom smith, June 13 2015, Issues affecting software Development today, https://dzone.com/articles/issues-affecting-software-development-today

6. Traylor, R. C., Stinson, R. C., Madsen, J. L., Bell, R. S., & Brown, K. R. (1984). Project management under uncertainty. *Project Management Journal, 15*(1), 66–75.

7. IEEE, Date Added to IEEE *Xplore***:** 23 March 2017,INSPEC Accession Number**:** 16773190, http://ieeexplore.ieee.org/document/7883326/

Software Engineering Companion (ICSE-C), IEEE/ACM International Conference on

8. Advanshetsen, 22/October/2013

http://unifaceinfo.com/the-6-most-common-problems-in-software-development/