Global Journal of Computer Science and Technology: C
Software & Data Engineering
Volume 17 Issue 3 Version 1.0 Year 2017
Type: Double Blind Peer Reviewed International Research Journal
Publisher: Global Journals Inc. (USA)
Online ISSN: 0975-4172 & Print ISSN: 0975-4350

# Software Development Top Models, Risks Control and Effect on Product Quality

By Ajayi W ., Adekunle, Y.A., Awodele, O., Akinsanya, A.O., Eze, M.O. & Ebiesuwa Seun
*Babcock University*

*Abstract-* In recent time, considerable efforts have been made to improve the quality of software development process and subsequently the end product. One of such efforts is finding a way to avoid or prevent risks in the overall process; and where or when it is not possible to prevent, risk alleviation readily comes handy.

Several problem solving methods such as six thinking hat, risk table, and riskit analysis graph (RAG) applied along with generic models such as spiral, waterfall, prototyping and extreme programming have been used in the past to prevent risk and enhances both delivery time and product quality.

However, some gaps were identified in the earlier works done in this area and in the generic models designed for evaluating and controlling risks prompting the development of modern ones.

Hence, this work tries to investigate different types of risks and risk management models, leaning on the gaps in research; it attempts to create a framework for better risk prediction and alleviation with the aim of enhancing delivery time and product quality.

*GJCST-C Classification: K.6.3*

SOFTWAREDEVELOPMENTTOPMODELSRISKSCONTROLANDEFFECTONPRODUCTQUALITY

*Strictly as per the compliance and regulations of:*

# Software Development Top Models, Risks Control and Effect on Product Quality

Ajayi W [α]., Adekunle, Y.A [σ]., Awodele, O [ρ]., Akinsanya, A.O [Ѡ]., Eze, M.O[¥]. & Ebiesuwa Seun[§]

*Abstract-* In recent time, considerable efforts have been made to improve the quality of software development process and subsequently the end product. One of such efforts is finding a way to avoid or prevent risks in the overall process; and where or when it is not possible to prevent, risk alleviation readily comes handy.

Several problem solving methods such as six thinking hat, risk table, and riskit analysis graph (RAG) applied along with generic models such as spiral, waterfall, prototyping and extreme programming have been used in the past to prevent risk and enhances both delivery time and product quality.

However, some gaps were identified in the earlier works done in this area and in the generic models designed for evaluating and controlling risks prompting the development of modern ones.

Hence, this work tries to investigate different types of risks and risk management models, leaning on the gaps in research; it attempts to create a framework for better risk prediction and alleviation with the aim of enhancing delivery time and product quality. To enhance good understanding and reading of the work, it has been structured into different sections. It concludes on some recommendations for future research in this paradigm.

## I. Introduction

In our world today, virtually everything around us depends on software. Our businesses, banking sector, educational system, our phones, home gadgets, even our cars and houses have been made smart and are being controlled by software (Chappell, 2012). Based on this reality, it simply means without quality software most business, basic home appliances and security, even modern civilization could fall apart.

To attain quality in software development, a range of possible factors such as the process that births the software, the choice of models used, formation and motivation of the teams involved in the development, handling of risks and risk areas all must come to play.

As would be explained later, amongst these factors, the choice of process models vis-à-vis how risks is handled are some of the major determinant of quality and quick delivery of software and these two are inevitable entities in the developmental process (Poth and Sunyaev, 2013).

Office of Government Commerce- OGC (2013) defined risk as an uncertainty or set of events that if allowed to occur, will have adverse or negative effect on the software development process or the quality of the end product. Risk is not limited by the location or site of the software project, the time spent planning or the sophistication of the resources invested into the development process, it could happen anywhere and at anytime during the software development life cycle (SDLC).

Some examples of where improper management of risks has led to either delay in delivery, poor quality or total failure of projects include: Canada's payroll system which was proposed to make accounting management easier but failed probably due to coding error or some other unforeseen factors, and this happened after spending whooping $50M.

Again, National Aeronautics and Space Administration – NASA (1986) reported that for thirty two (32) months, space shuttle could not launch into space due to an unforeseen circumstances leading to the death of the crew of "challenger" on Jan 28, 1986.
The popular "Y2K problem" in the late 1990s was caused as a result of ignorance about the sufficiency of using just the last two digits to represent the year (Aggarwal and Singh, 2007).

These few aforementioned are just some examples of notable projects that have either failed or did not complete as scheduled due to poor risk control procedure and bad planning.

Here in this work, an attempt would be made to create a model for better risk prediction and alleviation with an aim to enhance delivery time and product quality. Since this work tries to address software risks and its prevention, it is deemed fit to introduce its major concepts.

### a) *Major software risk Concepts*

Based on OGC (2013) and the work of Chappell (2012), the following are some of the major concepts associated with software risks and the systematic identification, evolution and prioritization of risk events and their likely consequences.

1. *Software Risk Identification:* the concept of risk identification falls into a futuristic category; it is a prediction of the unpleasant events that may occur along the developmental process.
2. *Software Risk Analysis:* understanding the nature of the risk, likelihood of occurrence, and the degree of impact. Impact level may be set from beginning from range 0 to 5, or from low to medium and high.

Author α σ ρ Ѡ ¥ §: Babcock University. e-mail: seunebi@gmail.com

3. *Software Risk Planning:* this is usually based on the information gathered from analysis, one can then come up with strategic actions and implement them in order to avoid risk
4. *Software Risk Monitoring:* ensuring that the risk does not occur and looking out for signals that indicate occurrence.

i. *Aim and Objectives*

The aim of this work is to examine the possibility of improving software quality through better control of risk.

*The basic objectives are to:*

1. Show that proper risk control will enhance fast delivery of software project objectives.
2. Show that quick identification of risk and risk areas of software development process will reduce the risk of the overall developmental project
3. Identify the basic parameter that must work together to attain quality product (software).
4. Analyze previous risk management models and existing works to establish gap or new trend in this paradigm.

b) *Problem Statement*

It is very imperative to state first that like every sector; software development process too is characterized by different types of challenges.

Earlier works studied in this paradigm show that in most cases, success rates of software projects have been found to be lower than expectation; and inability to easily identify and control risk have been identified as a major factor contributing to the failure rate.

Again, nowadays software is a major player in our daily life. Almost all our daily activities, our gadgets, cars, house security, depend on it, hence there are needs to design and develop software with utmost caution. It is believed that quality can only get better if risk is handled well because it has a direct effect on the quality of the software produced at the end of the whole process.

Thus, the main goal of this work is to review existing risk management techniques models along some traditional software models and related works in areas of software quality. After this, then come up with research gaps and ideas on how to develop a more meticulous model that will overcome the limitations in existing models and help enhance quick delivery and better quality.

c) *Methodology*

The methodology adopted in developing this work includes:

1. Literature search and analysis.
2. Model adaptation (from generic ones).

## II. Literature Review

Of late, the study of risk in software development has attracted great interest. To an extent, one could look at it as just mere interest which started as an attempt to test the strength of technology or computer science in handling just about anything possible; but more likely, the study of risk tends more to the quest to attain "better quality" in software and software developmental process. Hence to confirm either of the assertions, in this section, we try to evaluate some previous works done in this paradigm vis-à-vis design, problem solving techniques and models. However before proceeding, it is very pertinent to look into the categorized and other intrinsic risks (as seen in literature).

a) *Categories of Risks*

As analysed in OGC(2012), software project risks and other Information Technology related projects risks can be categorized into the following major areas.

i. *Technical Risk:* These categories of risks identify potential design, implementation, interface, verification and maintenance problems. If not handled and managed very well, this category of risk may threaten the quality and timeliness of the software to be produced.
ii. The second category is the development risk. This risk according to OGC(2012), involves inadequate planning, wrongly developed product features, interfaces which are not user oriented and failure of real life testing.
iii. *Business Risk:* The third category is the business risk. Further classifications of this risk are:

- *Market risk:* okay but no one really wants it
- *Strategy risk:* okay but no longer fits into the clients strategy
- *Sales risk:* okay but sales force can-not sell
- *Management risk:* losing the support of senior management due to a change
- *Budget risk:* okay but lost budgetary or personnel commitment.

Furthermore, analysis and deductions made from the work of Ghayyur and Khan(2010) used along with Kaur, Kaur and Kaur (2014) on *"Study of Different Risk Management Model and Risk Knowledge acquisition with WEKA"* revealed some other intrinsic risks that may occur or hinder the success of software development and the processes associated with it.

*These amongst others include:*

"Personnel Hiring and Shortfalls, Poorly trained project team members (personnel risk), Unrealistic Schedules and Budgets, Developing the Wrong Functions and Properties, Developing the Wrong User Interface, Gold-Plating, User Platform Incompatibility, Continuing Stream of Requirements Changes, Shortfalls in Externally Furnished. Components, Shortfalls in Externally Performed Tasks, Real-Time Performance, Shortfalls, Straining Computer-Science Capabilities, Case Tools under Performance, Unrealistic nature of temporary project plan, Loss of project *data,* development risk, Facility and equipment Machine etc*"*

Aside what is identified as direct risk which may delay, hinder success or cause total failure of software projects, sometimes software project may also fail as a result of the following.

a. Customer Involvement – for example in prototyping.
b. Using wrong process model.
c. Non consideration of risk.
d. Repetition of Task – e.g in the Risks management of Spiral model.

Having done with the different categories of risks possible in the software project development, the following sections enumerate the different methods that have been used in one way or the other to solve problems or (and) in handling risks.

### b) Overview of Some Existing Methods for Solving Problems and Handling Risks

Leveson (2013) shows that several methods have been developed in the past to predict, avoid or alleviate risks in the software development process. Some of these methods include:

(a) Use of risk table/log using RMMM (risk mitigation, management and monitoring). (b) Brainstorming. (c) Six thinking hat. (d) Risk analysis graph (RAG). (e) Risk matrix. (g) The Rich picture. (h) Use of financial models.

*Other methods used for identifying risk include:*

i. Check-listing: listing risks from past project.
ii. Interviews and Surveys: ask the right questions.
iii. SWOT Analysis: of products and methods.
iv. Direct Observations.

### c) The Risk Table

A risk table or risk rating table is a tool for assessing the likelihood and consequences of risk (Worksafe, 2014). Although there are different opinions on what should constitute the headings of the risk table, It appears that the constituent of the headings is subjective (based on the environment being assessed). However, generally based on Williams (2004) on risk management and some other earlier works in this area, headings of a risk table template should at least comprise of risk category, rank, risk-item, probability of risk occurrence, last ranking and action taken. Other views and addition that exist in this area tend to prefer the use of risk matrix or in some cases use both table and matrix.

A major point to note here is that to get better result while trying to get inputs for the table, it is better to consider an equally fit problem solving method for the purpose. For instance, to generate the Risk table, brain storming seems a perfect tool in enhancing the input for the table. Else, capturing all that needs to be captured may be a little challenging. To exemplify this, some inputs were generated and presented as table 1 below.

*Please note* that the input figures and other parameters were generated during a class session with some undergraduate software engineering students through brainstorming and other available data.

*Table 1:* Showing risk inputs generated from the use of brainstorming technique and other available data (from the client requirement /requirement engineering) for an action platform

| Risk item | Risk category | Components likely to be affected | Probability | Impact level (if allowed to happen) | RMMM (Risk monitoring, mgt &mitigation |
|---|---|---|---|---|---|
| Team member | Human resources | Schedule/cost/over head | 10% | 3 | Team members must have clear knowledge of project |
| Poor estimate and planning | Project team and finance | Schedule, cost and performance | 15% | 2 | Correct budget estimation |
| Project data | Equipment/tech | Schedule,cost,personnel | 20% | 4 | Backup of files, duplicate duties, |
| Cyber threats | Technical | Cost/data | 10% | 4 | Build-in/Ensure proper security |
| Theft/AZrm robbery | Project/technical | Physical systems and others/cost | 2% | 5 | Hire guard, burglary, Install security gadgets |

*The cyclic management approach of William (2004).*

Essentially, the work of Williams (2004) which was one of the earlier works done in the area of risk in the early 2000 used the educational sector as a case study. The approach sees risk management as cyclic events which involve monitoring, identification, analysis, prioritization, planning and mitigation, all of which stands on communication. The work presents an in-depth analysis of risk management, and also provided an insight to inputs for the risk table that are not readily available. For example, the work explains that if numerical values were attached for the probability of a risk happening, (say in percentage) and impact is given in (monetary terms), the risk exposure can then be calculated. According to their work, the risk exposure is given by:

$$\text{Risk Exposure (RE)} = P \times C$$

Where: P is the probability of occurrence for a risk and C is the impact of the loss to the product should the risk occur.

However, less was done to compare what would have been the result if a different model is chosen instead of agile method which was used in the scenario; this could also be improved on.

### d) *The Rich Picture*

The rich picture is a requirement gathering and knowledge elicitation tool which uses cartoon-like and somehow inexperienced pictures, diagram and symbols to aid quick thinking and depict ideas about a situation (Berg and Pooley, 2013). Going by Better Evaluation-BE (2016) analysis, it is a mind map which helps to open discussion, and then later lead to shared understanding of a situation. Though to use this method, one needs to first identify the issue that needs to be addressed, and then develop an unstructured narrative of the scenario of the challenge.

In their work, Bell and Morse (2010) used rich picture to harness solutions to problems from team members mind expressed through their different drawing. According to them, in using this method, two major rules have to be followed.

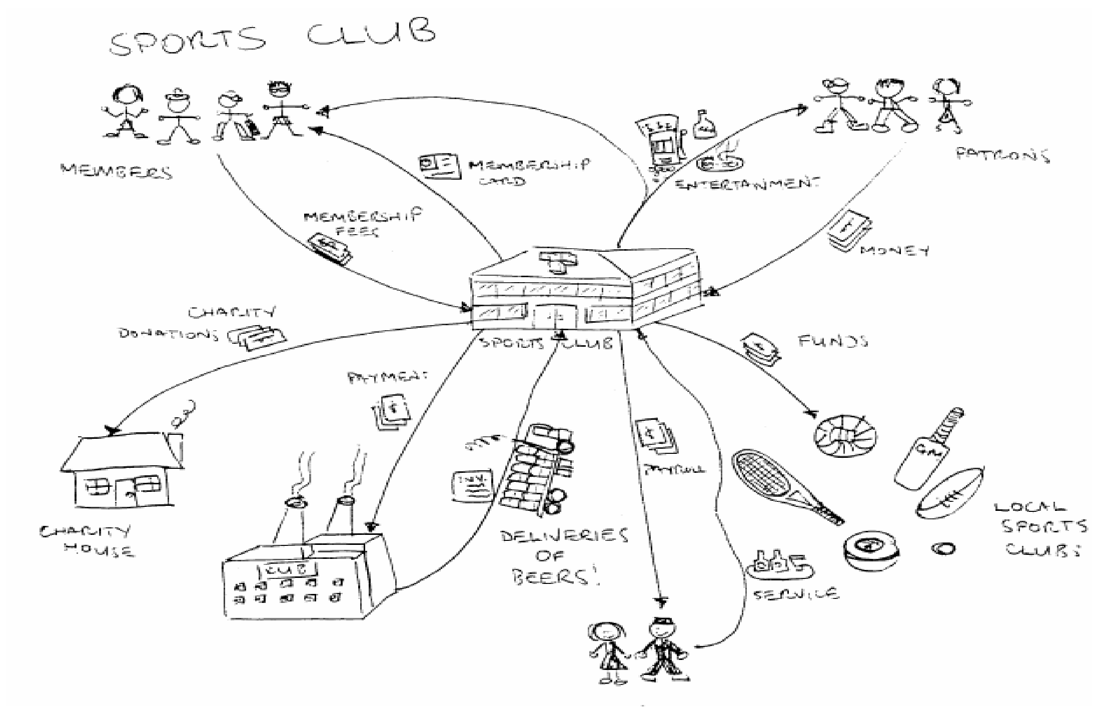
The drawings have to be visible to all team members at all times so it is clear to all what decisions have been made as to the components and linkages within the system being considered. Secondly, text should be limited or avoided totally because diagrams are much easier to appreciate visually.

Generally, the rich picture belongs to the category of soft system methodology (SSM) which is used for gathering information about complex or "hard knot" situation. As shown in fig 1a and fig 1b below, the end point should be a picture of the problem situation ; a very detailed and rich one which can be put together and analyzed within the time frame.

Though Bell and Morse (2010) work depicts rich picture in clear terms and richness in solving the set goal of their work, it however did not present much on the drawback or weaknesses of the model.

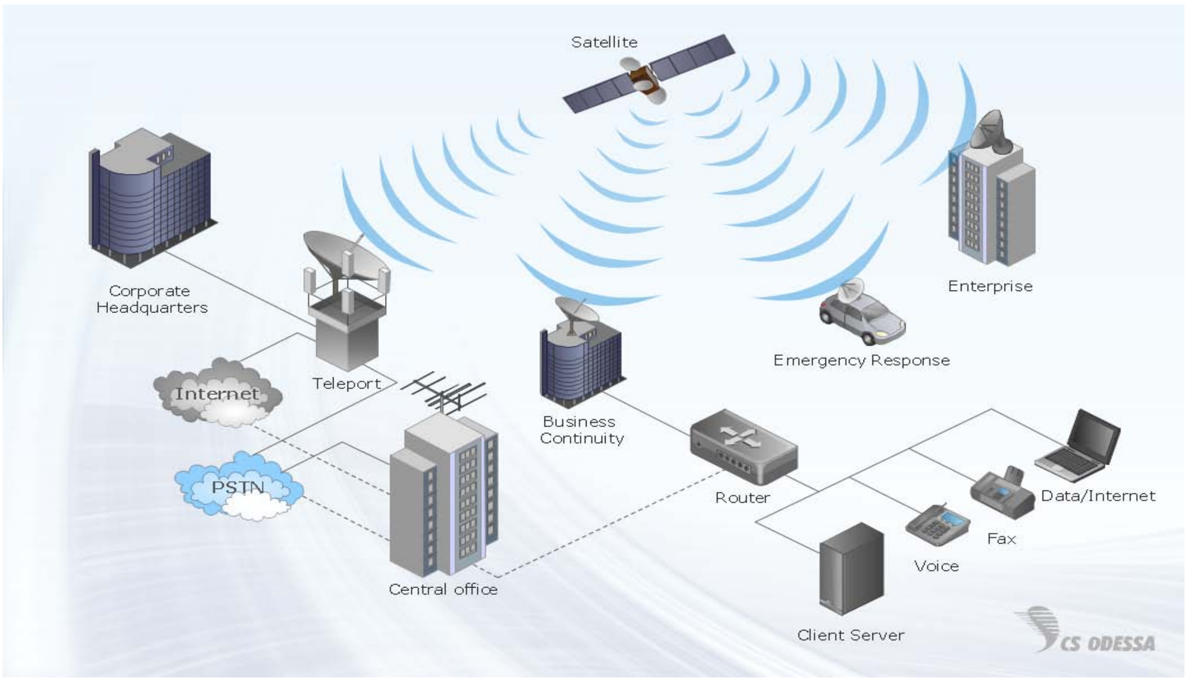
As seen in Pedell and Vetere (2005) and some other works of earlier researchers of the technique, in order to understand the pictures in its true form, the initial sketches might also need to be detailed which may lead to waste of project time. Although to some Information Technology project managers, this may seem like few minutes wasted, but when compared to the execution time of other techniques, this means a lot! And this constitutes a major gap compared to other methods for addressing risk.

Again, the rich picture does not take care of issues of laziness and team members who cannot create or interpret pictorial representations. In most cases, another form of algorithm may be needed for pictorial interpretation.



source: (Horan, 2000)

*Fig. 1a:* Showing rich picture drawn with free hand



Source: http://www.conceptdraw.com

*Fig. 1b:* Showing another example of rich picture

*e) Brainstorming*

Brainstorming is a fast and easy way to generate original ideas for problem solving and innovation (Unicef, 2015). Based on this author, it can be done alone or in a group. However, before the brainstorming exercise, some grand rules must be set for participant. Amongst others, some of these rules include, originality of ideas, no criticism, and the exercise must be done within a time frame.

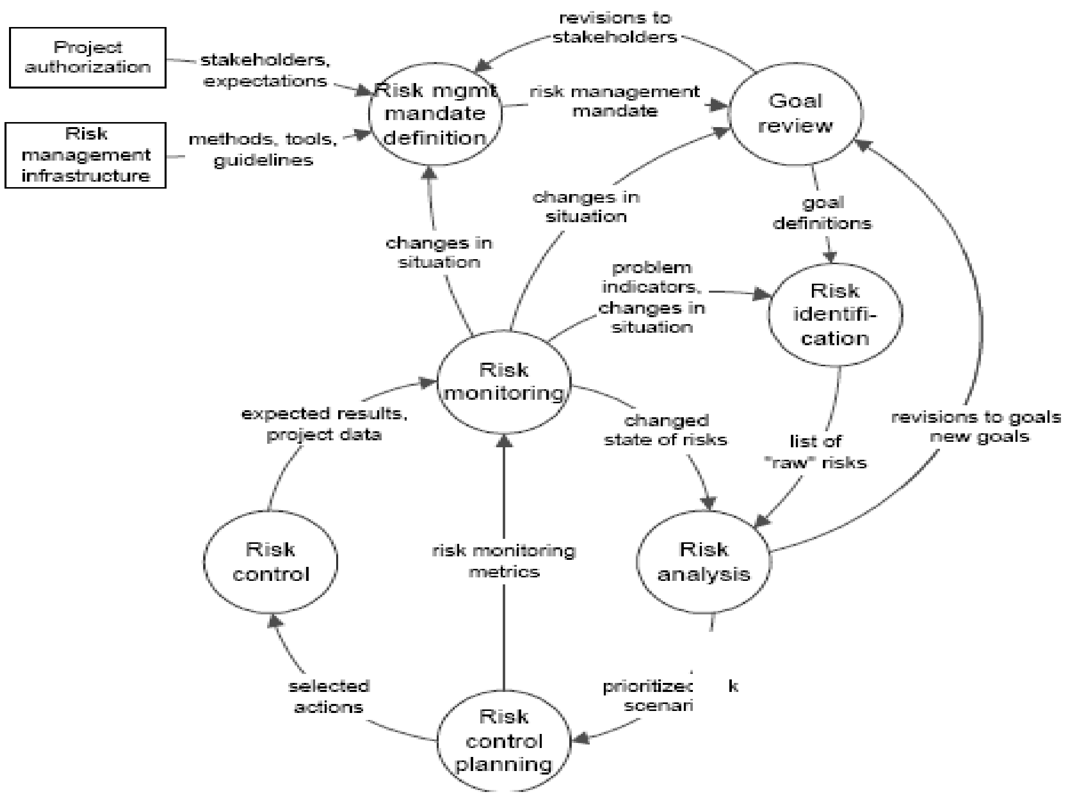
In Naser and AlMutairi(2015) brainstorming technique was implemented to find its effect in improving the problem solving skills for a set of male students in Kuwait. The result tends to be positive as envisaged from the beginning. However, the authors view and usage of this method is too narrow or simply biased along gender line.

Females' capacity to offer solutions and advice has enjoyed lots of advancements with good result in recent times (Forbe, 2014) and (Claremont, 2012). Hence, restricting females to the confines of household limit opportunities and it's a waste of potential for ideas. Again, the author did not analyse the risks embedded in using the approach.

Generally, brainstorming ought to be used for divergent thinking and must be used as such. It is an important strategy in provoking creativity and solving problems in virtually every field. The technique must be applied in a controlled team meeting, restricted to one point per person at a time and judging others is not allowed. Through the technique, lots of ideas about risk and difficult issues can be generated.

*f) The Risk Analysis Graph (RAG)*

The RAG is an acronym for Riskit Analysis Graph. It is one of the oldest Model or methods of analysing and managing risks. Several works have been done to analysed the RAG. The work of Freimut et.al.(2001) sees Riskit technique as a broad risk management process that is rooted on sound theoretical principles designed to have sufficiently low overhead and complication so that it can be deployed in a real-time limited software development project.



Source : (Freimut et.al 2001)

*Fig. 2:* Showing RAG.

Based on this author, the model allows the totality of risks captured in the developmental process and the project as a whole to be broken down into components such as factors, events, outcomes of an event, reactions, and effects on overall goals. By doing this, the impact of any risk can be explicitly considered by building up the scenario that encapsulates it.

Furthermore, it allows visual yet more formal documentation of risks and risk areas (enhances communication)

*Major limitations noted from this model are in the following areas:*

1. Risks prioritization during risk analysis is based on their probability and loss.

2. Documentation effort may be too high.

Literatures consulted for this study show that each of these risk control methods comes with basic strength as well as weakness.

For example the Capacity Maturity Model Integration- CMMI strength could be an advantage when used along with RAG since the CMMI is well grounded in documentation (Coffin and Lane, 2009).

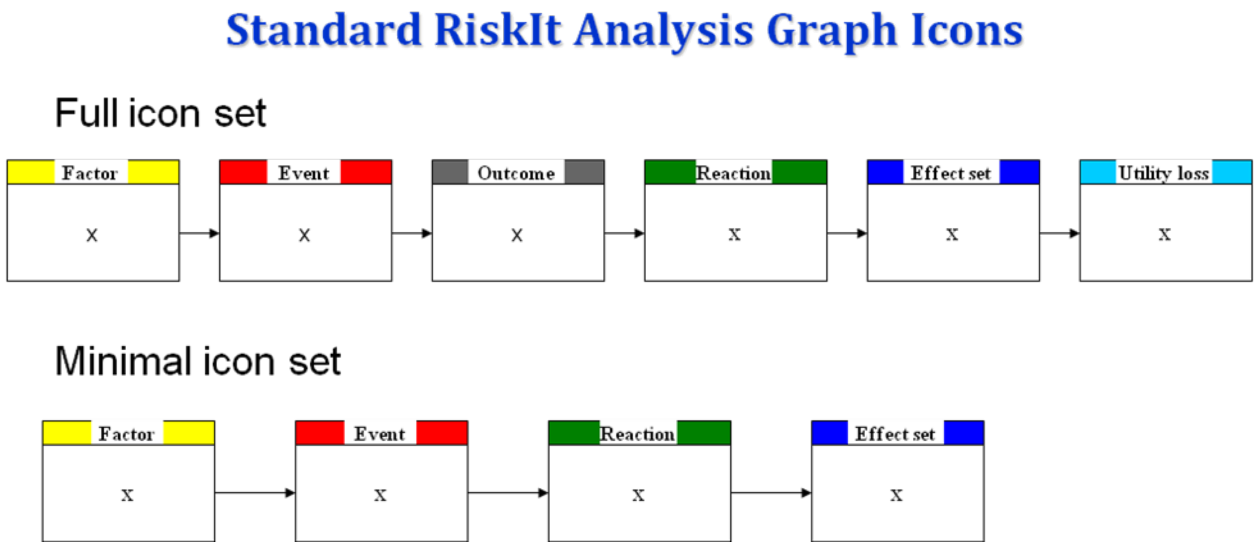


*Fig. 3:* Showing standard riskit analysis graph icons

*g) Software Process Models and Risk*

A software process is a planned set of activities which are considered necessary to develop a software system while a software process model is as an abstract representation of a process which presents a description of the process from some particular point of view (Sommerville, 2011). Software process model presents a description of a process from some particular perspective as:

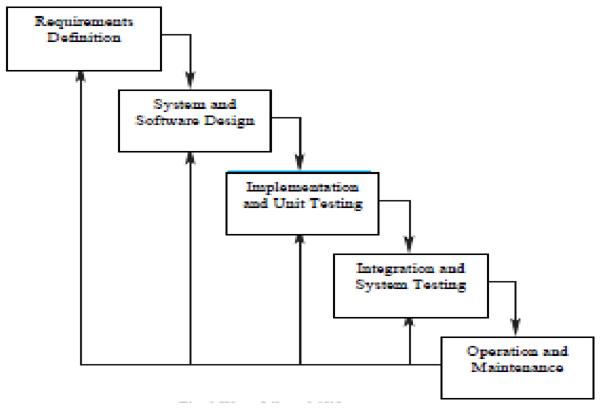
1. Specification.
2. Design.
3. Validation.
4. Evolution.

Several or different process models could be employed for the development of software (Ali Munassar and Govardhan, 2010). Based on this author and deductions from the works of SEI CMMI (2014) and Moniruzzaman and Hossain (2013) these process models which have been used in the past for software development involve the following major process.

1. The waterfall model
2. The spiral model
3. The V- Model
4. Prototyping
5. Extreme programming
6. Capacity Maturity Model Integration
7. Agile

Ali Munassar and Govardhan(2010) work was an extensive comparison work on the major but different models of software engineering. Basically, their work presents the five of the development models namely, waterfall, Iteration, V-shaped, spiral and Extreme programming. Based on the review of some existing work, their study was able to analyse the advantages and disadvantages of the different models, and make comparison amongst them to show the defects. However, this work was just a" literary comparison" no empirical or practical study was done to establish their claims.

We can say based on their work and other literatures, that the models do have their strengths, weaknesses and limitations. While the waterfall model (fig 4) may be used in small or medium projects low overhead and less attention to risk, the spiral model may not be suitable for small projects but has an inherent plan for risk. Hence, for the purpose of this work, our attention shall be on the spiral model. The choice of the spiral model was due to the original tenacity built into it for risk prevention.


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

*Source: adapted from (Ali Munassar and Govardhan, 2010)*

*Fig. 4*

*h) The Spiral Model*

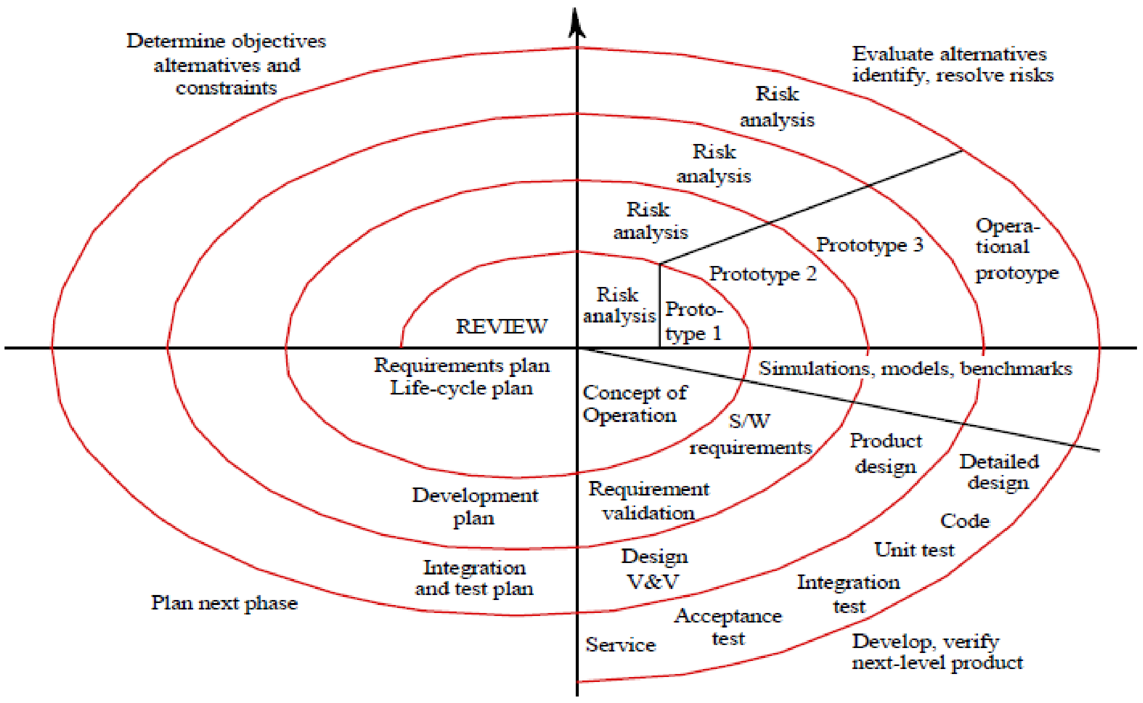
Under normal circumstances, a process model covers the entire lifetime of a product (Sommerville, 2011). Hence, a major risk that can emanate during software development is wrong choice of model. However, once the model is chosen right, the risk is already alleviated to a certain level. A generic software process model with such perception that risk may occur is the spiral model (Ali Munassar and Govardhan, 2010). Software risks were introduced for the first time in the Spiral model by Mr. Berry Boehm (Boehm, 1988; and Khan & Ghayyur, 2010) The spiral model as shown in fig 5 below, operates in loops with all the stages(or loops) of the spiral designed with at least an aspect of the requirement engineering which also include the verification and validation (known as V&V) and a perception of risk.

The development processes are represented as a spiral rather than as a sequence of activities with backtracking. Each loop in the spiral corresponds to a phase in the developmental process. Unlike other models such as the waterfall model, phases such as specification or design in spiral model are not fixed. The different loops of the spiral are chosen based on what is required and risks are explicitly addressed at every loop as they are encountered throughout the process.

*Advantages of Using the Spiral model.*

Based on the works of Sommerville(2011) and Ali Munassar and Govardhan (2010) amongst others, the following are the advantages of the spiral model.

1. *It is realistic:* the model accurately reflects the iterative nature of software development on projects with unclear requirements
2. *It is flexible:* it combines the advantages of the waterfall model and some evolutionary methods
3. It is a comprehensive model which decreases risk along the loop
4. It provides good visibility for the project



*Source: (Sommerville, 2011)*

*Fig. 5:* Showing the spiral model.

*Disadvantages*

1. It requires great technical expertise in risk analysis and risk management to function well.
2. Model is not so widely used because it is poorly understood by nontechnical management.
3. It involves high administrative overhead because of competent professional management involvement.
4. It may not work well for small project.

*i) Review of Related Works*

This section showcases previous works done in this area of study (using some other methods) to enhance the quality of software.

The first to consider in this group is Hossain, Kashem and Sultana(2013) work on "Enhancing Software Quality Using Agile Techniques"; their work depicts agile as a capable technique for ensuring good quality in software through measuring the "traditional quality factors" against how they are handled using agile technique. The work began by first Identifying the software quality factors (SQF) and Quality Assurance (QA), then went ahead to describe the agile techniques with special reference to software quality evaluation with agile technique. It however, did not analyse agile flavours, which may make the work a little too broad and difficult to know which one really helps in achieving quality. More on this will be discussed under the gap in research.

In another view by Vashisht, Lal and Sureshchandar (2016) on "Defect Prediction Framework Using Neural Networks for Software Enhancement Projects", they argue that though various approaches have been proposed in the past for effective and accurate prediction of software defects but most are not easily adopted in real life situations. Hence, their work aimed (majorly) at providing a more user-friendly, effective and acceptable framework which will help in predicting the defects in the phases across software enhancement projects. The work began with an analysis of the Software enhancement project life cycle, and then followed by the overview of the neural networks stressing their automatic learning ability over the traditional expert system. The design or proposed framework was later presented. The work is a clear approach to identifying defect and thereby enhancing the quality of the end product. The only set back here is not analyzing other methods such as fuzzy or other classification models to see if or not a neural network is better.

Poth and Sunyaev (2013) research an "Effective Quality Management: Risk- and Value-based Software Quality Management "by designing effective quality management (-EQM) to help software quality management (-SQM) to negotiate acceptable quality targets (based on standard quality factors) with all stakeholders - and to adjust them as the development progresses if need be. Based on their work, the main stakeholder parties are the end users or customers, the development team or department, and the operational management. Most often in software projects some stakeholders, like users or customers, do not personally participate in the quality assurance (-QA) planning process, and make only a review of the QA strategy and plan. In this case, in the first step, the SQM has to substitute for the missing stakeholders in the QA planning meetings. In the second step, the SQM has to legitimate the plan for the stakeholders to accept. The same happens if changes with the planned QA activities are required to react to unexpected occurrences which cause adjustments to the planning.

The authors went further to describe the stages of the IPDCA-cycle of EQM which guides the SQM during the product life cycle. Three different models – the V-model, the Scrum and Spice were presented and analysed in details. The "V-model example is based on the electric/electronic development of an engineering company, while the SCRUM (scrumalliance.org) example is based on the software for an airline's customer benefit program and the spice (ISO/IEC 15504) example is based on the electric/electronic product development organization of an automotive supplier". In all cases, the authors were able to establish its main aim. However their work did not link their findings with other notable metrics for quality.

## III. Gaps

After the analysis of the existing works both in the area of problem solving techniques and the closely related works the following were identified as major gaps in their works.

1. From the work of Hossain et.al(2013) agile strength and technique for enhancing quality were clearly outlined; but very little or nothing was mentioned on how agile handles risk when used in software development and how this could help in quality.

   Again the work treated agile technique as a broad topic and did not say much on its different flavours. Although all agile product must conform to agile manifesto but special attention to a particular one among the different flavours (which according to *Ferreira and Cohen, (2008)* include - "eXtreme Programming (XP), crystal methods, scrum, dynamic systems development methodology (DSDM), feature-driven development (FDD), and pragmatic Programming") would have made it easier to know the exact flavor with the strength in making the quality better.
2. Vashisht, Lal and Sureshchandar(2016) view of enhancing quality through defect prediction framework using neural networks-: The work was able to achieve the set objective. It however did not

Global Journal of Computer Science and Technology ( C ) Volume XVII Issue III Version I Year 2017

analyse other methods such as fuzzy or other classification models to see if or not they would have done better than the neural networks in the paradigm being considered.

The work is more like an extension of what they already have in use; it did not demonstrate that risk has a direct impact on quality, it rather infer it and the work did not link their findings with other notable metrics for quality.

Aside these gaps, most of the researchers have only dwelt purely on the generic models. Although they seems to have handled some (NOT ALL) of the identified risk one way or the other, but we don't know if or not other methods could have done it better. For instance, the risk analysis graph (presented as riskit) worked on by Freimut et.al.(2001), is very strong and unique in its approach to risk management and as stated earlier, it is rooted on sound theoretical foundations, helps in overhead reduction of cost and can be applied in real, time-constrained project. However, RAG as a method is a broad risk management process which may not be suitable for medium or small projects such that would be considered as the prototype later in this work

We believe to test their strength and forestall any problem along developmental process, some of the models or methods may have to be combined as hybrid to ensure smooth running e.g spiral and prototyping used vis-à-vis a problem solving method. Another aspect is combining the strength of agile for handling small project and that of the CMMI (though normally used in big projects) for documentation.

## IV. Conclusion

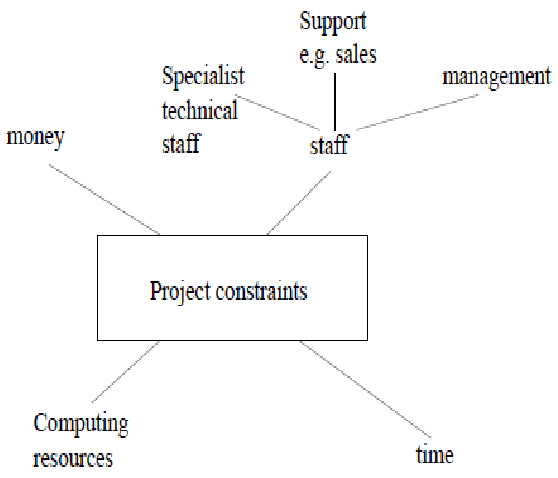
Software development takes a lot of planning, money, team work and energy. The interaction of these basic things called the constraints in Sommerville (2011) is shown in fig 6 below. However, it must be noted that no matter the amount of these factors put into it; it takes just one thing to go wrong for the whole process to go wrong and end up in lesser product quality. Conversely, it takes a combination *of at least three things* to have a quality product. These three things include: tools, technology and methods.

Moreover, after attaining the "initial or presumed quality", measuring it to confirm if actually it is the intended or proposed quality level is another major concern. Hence, some certain metric needs to be put in place to ascertain if or not the end product is qualitative. To this end, Chappell (2012) reports on how the quality of software product can be measured. Going by the report, the following basic and cogent parameters must be looked out for.

a) *Functionality* - this involves factors such as the performance, ease of learning and ease of use plus other things that have to do or fall under the functional requirements of the developed system.

b) *The process that births the software.*

c) *Structure:* this involves code efficiency, maintainability, security, testability, understandability etc.



*Source : (Sommerville, 2011)*

*Fig. 6:* Showing software project constraints.

Aside these, the system and other components must meet specified requirements by the client as stated by both parties in the memorandum of understanding (MOU). Again, the development must ensure that the system and other component meet client needs. By monitoring quality risks and product evolution over its life cycle, quality assurance team can make right choices and enhance the quality of product.

The concept of software risk is broad and generally risk abounds in virtually every aspects of software project development. The more we are able to predict them, the easier and smoother the process and the better the quality of software produced at the end of the developmental process.

a) *Future work*

In the future we intend to :

1. Improve on RAG (expand an aspect to capture aspects relating to data during system migration)
2. Do a comparative analysis of two software models - possibly two that were not already analysed here (using some basic factors) to test their suitability and possibly acceptance in software projects.
3. Apply the developed model in identifying and pre-empting risk that may occur in a particular software project area or task.
4. Implement and evaluate the efficiency level of the present models compared with proposed one.

b) *Further proposition on tools to employ in this work*

1. Set theory.
2. Fuzzy logic and;
3. Bayesian algorithm/nearest neighbor (to Hazard /risk) in this case we set conditions for a project entity (say the critical path).

## References Références Referencias

1. Aggarwal K.K and Y. Singh (2007). Software Engineering (3RD edition.), New Age.
2. Ali Munassar1 N. M. and Govardhan A. (2010) A Comparison Between Five Models Of Software Engineering, International Journal of Computer Science Issues, Vol. 7, Issue 5, September 2010.
3. Bell S. and Morse S.(2010). Rich Pictures: a means to explore the 'Sustainable Group Mind', The Open University's repository of research publications and other research outputs, accessed on 04/03/2017 from http://oro.open.ac.uk/24617/1/ISDRC_16_~_Bell_Morse_Rich_Pictures.pdf
4. Berg T. and Pooley R.(2013) Contemporary Iconography for Rich Picture Construction, Systems Research and Behavioral Science, wiley onlinedigital library.
5. Better Evaluation- BE (2016): Rich pictures, viewed on 27th mar, 2017 from: http://www.betterevaluation.org/en/evaluation-options/richpictures
6. Boehm, B.(1988) A Spiral Model for Software Development and Enhancement Computer, vol. 21, No. 5, May 1988, pp. 61-72.
7. Chappell D (2012) The Three aspects of software quality: Functional, Structural and Process, sponsored by Microsoft Corporation, viewed on 05/03/2017 from : http://www.davidchappell.com/writing/white_papers/The_Three_Aspects_of_Software_Quality_v1.0-Chappell.pdf
8. Coffin and Lane (2007) A Practical Guide To Seven Agile Methodologies, Part 1, viewed from http://www.devx.com/architect/Article/32761/1954
9. Forbes (2014) Who Makes A Better Leader: A Man Or A Woman? Accessed on 04/03/2017 from https://www.forbes.com/sites/sebastianbailey/2014/07/23/who-makes-a-better-leader-a-man-or-a woman.
10. Freimut B., Hartkopt S., Kaiser P., Kontio J and Kobitzsch W.(2001) An Industrial Case Study of Implementing Software Risk Management, ESEC/FSE-9: *Proceedings of the 8th European software engineering conference held jointly with 9th ACM SIGSOFT international symposium on Foundations of software engineering,* ACM , pp 277 – 287.
11. Horan P(2000) Using rich pictures in information system teaching, 1st International Conference on system thinking in management, 2000, pp 257 – 262.
12. Hossain A., Kashem A., and Sultana S.(2013) Enhancing Software Quality Using Agile Techniques, *IOSR Journal of Computer Engineering (IOSR-JCE)* Vol 10, Issue 2 (Mar. - Apr. 2013), PP 87-93.
13. Kaur K., Kaur A.,Kaur R. (2014) Study of Different Risk Management Model and Risk Knowledge acquisition with WEKA, International Journal of Engineering Research and General Science Volume 2, Issue 4.
14. Khan Q. and Ghayyur S. (2010) Software risks and mitigation in global software development, Journal of Theoretical and Applied Information Technology JATIT 2010.
15. Leveson N.G (2013) *Learning from the Past to Face the Risks of Today*, Communications of the ACM, Vol. 56 No. 6, Pages 38-42.
16. Merchant K.(2012) How Men And Women Differ: Gender Differences in Communication Styles, Influence Tactics, and Leadership Styles, accessed on 05/03/2017 scholarship.claremont.edu/cgi/ view content.cgi?article=1521&context=cmc_theses
17. Moniruzzaman A B M and Hossain S.A (2013) Comparative Study on Agile software development methodologies
18. Naser A. and AlMutairi M. (2015) Effect of Using Brainstorming Strategy in Developing Creative Problem Solving Skills among male Students in Kuwait: A Field Study on Saud Al-Kharji School in Kuwait City , Journal of Education and Practice, Vol 6 , Nos 3.
19. Office of Government Commerce (2012; 2013) Project in a Controlled environment (PRINCE 2), TSO, UK.
20. Oxford Advanced Learners dictionary (2016). Oxford Advanced Learners, Oxford University Press, accessed on 27/02/20167 http://www.oxfordlearnersdictionaries.com/definition/english/
21. Parnas D.L.(2011) The Risks of Stopping Too Soon, Communications of the ACM, Vol. 54 No. 6, Pages 31-33.
22. Pedell S. and Vetere F (2005) Visualizing use context with picture scenarios in the design process, *MobileHCI '05: Proceedings of the 7th international conference on Human computer interaction with mobile devices & services*, ACM, Salzburg, Austria, 271-274.
23. Poth A.and Sunyaev A. (2013). Effective Quality Management: Risk- and Value-based Software Quality Management, IEEE Software Publication, viewed from : http://www.isq.uni-koeln.de/fileadmin/wiso_fak/wi_isq/pdf/IEEE_Software_Sunyaev.pdf on 05/05/2017.
24. Reich B.H and Sauer C.(2010) *Roles of the External IT Project Manager*, Communications of the ACM, Vol. 53 No. 5, Pages 126-129.

25. SEI Software Engineering Institute (2004)*:* viewed from: http://www.sei.cmu.edu/cmmi/general/ Sommerville I. (2011). Software Engineering 9, Pearson, USA.
26. Unicef (2015) Brainstorming , Free-flowing creativity for problem-solving, accessed on 30th march 2017 from: https://www.unicef.org/knowledge-exchange/files/Brainstorming_production.pdf
27. Vashisht V., Lal M., and Sureshchandar G.S. (2016) Defect Prediction Framework Using Neural Networks for Software Enhancement Projects, British Journal of Mathematics & Computer Science, 16(5): 1-12, 2016, Article no.BJMCS.26337.
28. William L.(2004) Risk management, accessed on 29th March 2017 from: http://agile.csc.ncsu.edu/SEMaterials/RiskManagement.pdf
29. Worksafe (2014) Assess- risk rating table, accessed 29th March, 2017 from: http://www.worksafe.govt.nz/worksafe/toolshed/safe-use-of-machinery-toolkit/assess-risk-rating-table.
30. Wright D. (2011) *Should Privacy Impact Assessments Be Mandatory?* Communications of the ACM, Vol. 54 No. 8.