CHAPTER – 2

REVIEW OF LITERATURE

2.1 INTRODUCTION

Software Requirements Engineering (SRE) can be defined as the process by which the features of software systems acquired by the user are established. In general, software requirements engineering refers to the earlier concept of identification and analysis stages. It also differentiates between user requirements with respect to information systems. It comprises a set of interrelating activities that result in the production of the Software Requirement Specification Document (SRSD). The SRE involves the following steps:

- Requirement elicitation activities
- Requirement analysis activities
- Requirement specification activities
- System modeling
- Requirements validation
- Requirements management

Requirement Engineering provides the service that the customer needs from a software system. Requirement elicitation recognizes the items of information that detects the required characteristics of the software system. These characteristics identify the behavior of the system in response to user input. It requires collaboration between different groups also called stakeholders. The adequacy and characteristics of the identified requirements are established by the Requirement analysis process.

The requirement analysis can be done to reveal inadequacies in the elicited requirements that can prevent the software from meeting user satisfaction and cause difficulties later in the software development process. Requirement engineering is an iterative process and the practitioners develop the business requirements and baseline them. The business requirements are given as input into the development of the user-level requirements. The practitioners discover the gaps in the requirements and further refinement are carried out. The information gained from refining the business requirement is used for updating the user-level requirement. The business and the user-level requirements are fed into the definition of the product-level requirements. These
requirements are used as input for the software design and development process. It provides the best opportunity to consider all the stakeholders’ interest in context with one another. The stakeholders fall under one of the following three categories:

- Acquirers of the software product
- Suppliers of the software product
- Other stakeholders

The acquirer type stakeholders comprise of two groups. The first group includes customer who request, purchase, or pay for the software products in order to meet the business goal. The second group is end-users, who actually use the product directly or indirectly by receiving the reports, outputs, or other information generated by the product. The suppliers of the product involve individuals and teams, who are part of the organization that develops the software product. Stakeholders, except the acquirers and suppliers of the software product are considered as other stakeholders.

2.2 PREPROCESSING THE REQUIREMENT

Lal, et al (2013) identified the various factors of requirements preprocessing process. The actual wants and needs of the customers in an enterprise or organization was redefined. It includes various parameters, which were important in the requirements preprocessing process. Requirements are divided in several ways related to technical management.

- **Customer requirements**: These requirements define the expectation of the system in terms of objectives, environment, constraints and measures of effectiveness and suitability.
- **Functional requirements**: These requirements give the clear view about what has to be done by identifying the necessary task, action or activity. These are used as the top-level functions for functional analysis.
- **Non-Functional requirements**: These requirements specify the criteria that are used to judge the operation of a system, rather than specific behaviors.
- **Performance requirements**: During requirements analysis, performance requirements are interactively developed across all identified functions based on the system life cycle factors and characterized in terms of the degree of certainty.
• **Design requirements**: These requirements consist of the basic outline of the product to be built and a set of instruction to execute the code.

• **Derived requirements**: These are applied or transformed from the high-level requirement.

• **Allocated requirements**: These requirements are established by splitting or allocating high-level requirements into multiple low-level requirements.

The major concept identified during the requirement analysis was the pre-processing process. The preprocessing process detects unclear, incomplete and ambiguous or contradictory data. The organization and stakeholders were unable to detect the problem and their requirements. The detection process follows a series of steps. It becomes difficult to develop a system for the engineer by using the incomplete requirements from the stakeholders. The developers avoid the ambiguity during the development process. The developers analyze the functional and non-functional requirements, whether it was related to system and development process.

Virdi, et al (2012) developed an automated system that represented the web requirement engineering. This system was designed for the web applications where the user merges web page requirements with respect to validation. Requirement gathering is the main field of software development that provides the basic outline for development process. It involves all preprocessing and the post-processing process of the software development approach.

The business or the organization was usually represented by a website. The web application finds the regular change in the requirement given by the customer. These requirements were given based on the appearance of the website or web portal. These requirements require some regular updating, respective to the information that an owner wants to provide to the stakeholder. Feasibility was checked in the website with respect to the system support, web support, environment support, etc. Once the feasibility was defined, it does not require any change. The next phase verification was carried out by end users.

In the case of web application the verification phase cannot be done before the implementation of the code. During the verification phase, there was a chance of modifying the requirements by the user. Again the product will be verified till it attains
user satisfaction. The validation process was done by the developer and tester to block the user from performing any wrong task. Drechsler, et al (2013) addressed the preprocessing problem at the specification level. The automatic guideline checking and fixing and requirements classifications were depicted. These guidelines support requirement engineer in writing good requirements and also to a large audience globally.

The guidelines were used internally as an agreement between the employees of a company. Specifications of technical systems comprises of many requirements written in natural language. These requirements were typically expressed by means of one sentence and were classified into low-level and high-level requirements. The low-level requirements were easily translated into formal counterparts that refer to specific design and concrete values. High-level requirements require further inquiry of the context that was received from reading the specification and a direct translation was not possible.

The textual description was transformed into a formal model, which maps informal description into formal one. The dynamic behavior of formal models was verified by means of pre and post conditions of operations. It enables a descriptive representation of the behavior without giving a precise implementation. In pre-condition stage, the system operations can be called, whereas post-condition phase implements the effect of an operation on the system state. This condition was specified directly from the designer or determined by invariant elimination.

Budak, et al (2012) depicted a program system, for data-point preprocessing which involves data preparation, error filtering, data smoothing, data reduction and output file format generation. It concerns with the quality of digitization result that significantly complicate processing. The starting point of RE was a physical object, which can be an existing part or product handmade model. A complete programming module was implemented using MATLAB. The preparation of point-clouds comprise of noise and error that was filtered by reducing the number of points available in the programming module. This filtering process was called as the preprocessing.

Sultanov, et al (2010) depicted various swarm methods to support requirements traceability. This technique effectively addresses the requirement engineering problems. It also demonstrates the applicability of swarm intelligence to the requirement tracing problem. It propagates a Requirement Traceability Matrix (RTM) between textual
requirements artifacts level requirements to low level requirements. Requirements were more important when developing the mission or safety critical systems. The consequences of poor quality lead to loss of life or damage to the environment.

The Swarm intelligence was applied using swarm techniques, which was a property of non-centralized group of self-organized agents that jointly perform a desired work. A training was conducted for a collection of queries and resulting in the retrieved documents. It used a trained set and also a set of validations to reduce over fitting. It optimized a standard IR measure called Mean Average Precision (MAP). Requirements tracing was defined as the ability to describe the requirements in both forward and backward direction. This process was used for requirement tracing of natural language artifacts either manually or automatically. This comprises the following steps:

- Document parsing
- Candidate link generation
- Candidate link evaluation
- Traceability analysis

If a requirement document was traced to a set of use cases, document parsing extracts elements from the two artifacts, which includes uniquely, identified requirement elements and use case elements. Candidate link evaluation was carried out to ensure that the extracted elements were correct. Traceability analysis decides the high-level artifact, which has been satisfied by the lower-level artifact. Mansilla, et al (2013) illustrated a process model for requirement elicitation information mining projects. A process of requirements collected for information mining projects was implemented. These projects concentrate on conceptualization, business definition and information mining process identification.

A set of high level activities was performed as a part of the business understanding stage. This process breaks down the problem of requirement elicitation in information mining into several stages. The first two steps of the process comprise of two activities, namely, identification of project stakeholders and set the list of people. Software development projects were started by receiving the understanding of the business domain and rules that govern them. The understanding of business domains help to detect the requirements of the business and product level.
Business domain defines the product to be built by considering the context application. Models such as Context Diagram, Data Flow Diagrams and other relevant diagrams were used to graphically represent the business process. It was used as validation tools for the business process. A functional analyst collects data about inputs and outputs of the software products to be developed and how that information was transformed by the software system. Models for requirements elicitation and project management focus on the software product development.

Rehmen, et al (2013) deliberated requirements elicitation that facilitate to change the requirements from time to time. It was an integral part of the software development lifecycle and the successful software depends on précising its requirements. Requirement engineering was an approach in which RE activities incorporate the entire system. The requirement elicitation acts as foundation for all the subsequent software development work. All the key stakeholders were asked to follow the same requirement gathering process. During the requirement elicitation, the engineer selects the requirements based on available resources and time constraint.

These techniques involve questionnaires, surveys, interviews, task analysis, domain analysis and introspection. Requirement engineering was the most critical and complex process for the software development because of the diversification in the received requirements and due to the changes in the requirement. It was always difficult to develop an accurate requirement that remains stable in large and complex systems. The majority of the errors in the software functionality were directly linked to the mistakes done at the time of requirement gathering and elicitation phases.

RE processes were iterative that ensure the maximum user participation and also facilitate change in requirements from time to time. Bokhari, et al (2011) suggested a software requirement specification or quality attributes for requirement engineering. Requirement engineering starts with inception task that defines the scope and nature of the problem to be solved. Then the elicitation step helps the customer to define what was required. In elaboration the basic needs were modified and defined. The requirement engineering has serious issues due to the lack of stakeholder involvement. The lack of requirement management also leads to bad engineering. Requirement analysis and negotiation are processes during which the requirements were analyzed and modeled. Some techniques that can be used in the requirement analysis were described below:
• Unified Modeling Language (UML) is a collection of techniques for modeling software or systems.

• Specification and Description Language (SDL) is a relatively mature requirement description and definition language.

• Structured Analysis Structured Design (SASD) uses a set of different techniques such as functional decomposition technique, data-flow diagram and data dictionary.

• Goal-based techniques are used in the later stage of elicitation as well as effective requirement analysis techniques.

• Petri Nets is good at modeling state transitions and control flow in an asynchronous system where there were a lot of parallel and asynchronous events.

These software metrics identifies and measures the essential factors, which affect software development. The collection of requirements was a tedious phase in the software development specification. Requirement traceability metrics, requirement completeness metrics, requirements volatility metrics and size metrics were used to measure the requirements engineering phase of the software development life cycle. Collection of metrics was faster and more reliable with the use of automated requirement tools.

Sharma and Kushwaha (2010) suggested a requirement based complex measure on Software Requirement Specification (SRS) document. It was a robust method and encompasses all major parameters that were needed for finding the requirement complexity. These were comparable with code based and cognitive information based complexity measures. Requirement Based Complexity (RBC) follows code based measures that was computed on the basis of program by identifying a number of operators and operands and was aligned with the difficulty metrics. It was assigned with the control flow based graphical complexity measure. Finally the result of RBC was the similar kind of result that the Cognitive Based Complex measures were given and can be computed using software information.

Functional requirements define the fundamental actions, which takes place in the software by processing the inputs and generates the output. Figure 2.1 depicts the overall process in preprocessing. The measure was computationally simple and supports the developer and practitioner in evaluating the software complexity. Gupta, et al (2011)
illustrated a data warehouse design using requirements engineering. The object oriented system can be easily classified using the model based object oriented requirement engineering approach. It has the benefits of higher maintainability, reusability and productivity.

**Figure 2.1 Overall process involved in Requirement Preprocessing**

It provides the mechanism to transform the existing requirement document in text form to natural language. The domain knowledge was properly built and then effectively reused. It develops a uniform requirement model as opposed to the traditional approach with loose information scattered in different documents. It mainly concentrates on documenting the requirements rather than the entire requirement engineering process. The aspect oriented engineering, detects and manages the cross cutting in an effective way and helps in tangled requirement representation.

It was difficult to maintain the requirements and also ensure the consistency of stakeholder concerns with global requirements and constraints. A scenario based design method was useful to assess the functionalities that were required for the system. These scenarios can be used systematically to derive all the requirements. This module level scenario was divided into basic and composite. These modules were composed of more
composite modules and was used together to find the refined set of requirements. These were called the processed requirements on which the data warehouse was designed.

These models have the capability to grow as needed and help in reusability. It can be independently used to define the requirements accurately. Each module has the restructuring capability independently. The constraint level defines the constraints on the attributes and their values. It also provides a mechanism for ensuring that the data conforms to specified guidelines. This helps in ensuring that the data in the data warehouse conforms to the basic level of data consistency and correctness. The properties of attribute constraints generally specify the semantics conditions that were held in elements of a model at all times. Ali, et al (2011) developed information retrieval approaches for recovering traceability links. These approaches produce rank list of traceability links and are pruned using various methods. It also uses a traceability recovery approach to obtain a set of traceability links.

2.3 FEATURE SELECTION AND EVALUATION

Razali, et al (2011) constructed the effective selection process of stakeholders, who were identified through content analysis. Requirements elicitation was the most critical phase in software development as it captures the required functionality of a software system. The elicitation process was indeed resource intensive and involves a number of dedicated stakeholders. The effectiveness of the process was greatly influenced by the credibility and suitability of the stakeholders involved. The correct and complete requirements were achieved and they were collected from the right stakeholders.

Requirement elicitation was the important method and critical phase in requirement engineering. An ineffective elicitation process produces poor requirements, which leads to low system quality, extension of schedule and increase in budget. Selecting the appropriate stakeholders from the right subject at the right time was one of the major factors of the software process. Certain mechanisms were introduced to determine the stakeholder’s level of knowledge and interest.

Merugu, et al (2012) deliberated non-functional requirements and incorporates them in the software. Identification of non-functional requirement was important for successful development and deployment of the software product. The acceptance of the software product by the customer relies on the non-functional requirements, which were
incorporated in the software. As the complexity of software system increases nonfunctional requirements elicitation and analysis were increasingly difficult in software development. This analysis includes a variety of people in an organization and the stakeholder was used to refer any person. The application of this approach characterized the following:

- Identifying the non-functional requirements
- Reducing the time consumption and there was no loss of information
- Using the functional requirements along with non-functional requirements as validation criteria involved in SRS.

Figure 2.2 Overview of RSLingo approach: Usage at Project-level

Ferreria and Silva (2012) devised a RSLingo, a linguistic approach that permits to automatically extract the domain knowledge from textual requirement specifications. This approach improves the quality of requirement specification, which was based on two languages. Here, domain analysis was carried out for identifying all the relevant concepts, their relations and the software system manipulates them to provide the series capabilities. Verification was done for performing consistency checking on the extracted domain knowledge through inference and ambiguity resolution based on glossaries. Figure 2.2 illustrates the overview of the RSLingo approach at the project level. The transformation for automatically generating alternative requirement specifications such as diagrams, tables, reports or even template based phrases was done.
The term decoupling refers to the extraction or removal of data. Decoupling allows users to deal with the requirements and also enables a deeper understanding at a semantic level. The Decoupling between linguistic aspects and RE-specific concerns provides a flexible approach towards formal requirements specification and the validation of these artifacts was done by the business stakeholders. Kamalrudin, et al (2010) designed an approach for supporting quality improvement through semi-formal model. The requirements specifications were checked against the consistency, completeness and correctness to check high quality. The sequence of essential use case interaction was compared with interaction patterns. The sequence of abstract interactions and Essential Use Cases (EUC) components should be in the same order as the sequence of essential interactions in the textual requirements.

It detects the inconsistencies between the edited and unedited models due to the ordering of interaction between user and system needs. The name of EUC components was same as the abstract interaction or vice versa, and was implemented during a specific essential interaction. The abstract interaction elements and sequence of elements in EUC model needs to match a suitable template in the EUC pattern library. This technique permits us to detect the following:

- **Intra-model inconsistencies**: One or more unexpected abstract interaction or interactions out of expected sequence was extracted from the EUC model. Incompleteness–missing interactions that occurred in the extracted EUC model was compared to the generic template matched in the EUC pattern library.

- Incorrect requirements captured in the extracted EUC model that do not match a best practice template in the pattern library indicate incorrect textual requirements.

The sequence of EUC interactions was compared to common sequences or EUC patterns. The extracted EUC models abstract interactions were compared to an expected EUC pattern set of abstract interactions and their sequencing.

- **Sequencing of requirements elements**: It detects inconsistencies between models where one has been edited and the ordering of the interactions between user and the system needs to be made consistent.
• **Naming of requirement elements:** The name of EUC component was same as the abstract interaction. It also needs to match one of the abstract interactions in the EUC pattern library.

• **EUC interaction pattern matching:** The abstract interaction elements and sequence of elements in EUC model needs to match a suitable template in the EUC pattern library. Updating an abstract interaction to conform the matching components requires updating the equivalent in the textual natural language based on the match pattern. It also identifies the incomplete and incorrect requirements.

• **Consistency within models:** The abstract interaction sequence semi-formal notation has metadata models with constraints expressed. It allows the low-level validation of correctness and internal notation consistency. These models check for low-level intra-notation consistency, completeness and correctness.

• **Consistency between changing components:** All the three requirements representations, textual natural language scenario abstract interaction and EUC must be consistently updated if any elements in any of the models were modified by the requirement engineer.

Requirement specification was captured and the quality problems were analyzed when working with natural language requirements. Li, et al (2014) presented a modeling language for non-functional requirements. It views the Non-Functional Requirements (NFR) as requirements over quality, mapping a software-related domain to a quality space. The language was compositional and allows complex NFRs structures in several ways. It adopts an ontological interpretation of NFRs based on qualities in foundational ontology. It offers a compositional modeling language for capturing NFRs, where the subjects of involved qualities were identified using notation, resembling feature structures in linguistics. It identifies three meta-qualities for discussing the fulfillment of a requirement. It also deliberated a goal oriented requirements methodology for refining ambiguous, unsatisfiable or vague NFRs measurable ones.

Fricker (2010) depicted a view on stakeholder involvement in requirement engineering. The requirements value chains evolve when stakeholders enter or leave the software ecosystem. It groups the members and establishes the relationship.
Active integration includes an application for a given role and the passive involvement happens through personal interests, traits and relationship that make the person attractive for being integrated. If people had same goal, they attained group membership and pursued the same goals as other group members. They were able to interact with other members and shared a set of norms with each other. Group enrollment was active or passive and the constitution of groups influence, negotiation tactics and methods. It implemented and maintained a software ecosystem by managing stakeholder’s interest. It had introduced the concept of requirements value chain where requirements emerge and propagate with inter-stakeholder collaboration. The resulting view on a stakeholder management and requirements engineering addresses with negotiation principles. The Requirements value chain structure affected innovation, requirements engineering performance and software success. The distance feed-forward and feedback affected the overall alignment of stakeholders to collaborate.

Mahaux, et al (2013) presented a collaborative creativity framework that helped RE practitioners and researchers. RE practitioners gained knowledge of the important factors for their projects. The factors were organized in two phases, which pertain to the team and those pertaining to the individual. It also offered ideas for addressing each factor in practice arising from the authors. The framework of factors for collaborative creativity in RE can be helpful to both RE practitioners and researchers. The RE practitioners can be benefited from better knowledge of important factors for their projects.

RE practitioners formed a new understanding of their discipline, potentially reshaping the way they undertake RE efforts. RE requires creativity in a form where interactivity among stakeholders was particularly important. These factors enabled a systematic investigation of the collaboratively creative nature of RE. It guided practitioners when facilitating RE efforts and also provided researchers with idea on where to focus when developing methods and tools for RE.

Park, et al (2013) suggested the requirement extraction and a prioritization method individually from the customer or the user needs. The prioritization approaches analyzes the individual requirements. In this process, both requirements from the customer and user needs were extracted. Hybrid approach had different scope values to each requirement from the customer or the user needs. It combined both requirements and redefines the scope of hybrid approach. Based on the threshold values the use cases decided elimination.
Mulla, et al (2012) depicted the requirement elicitation based on stakeholder recommendation. This methodology was divided into four parts, namely:

- Select software project to identify the list of stakeholders and requirements
- Build social network, which helps to detect and prioritize stakeholders
- Develop a method that uses collaborative filtering to identify and prioritize requirements
- Develop a software tool that supports the above method and apply to the real projects by having practitioners in their projects.

Stakeholders were asked to recommend other stakeholders and a social network was built by considering stakeholders as nodes and their recommendations as links. A social network was a structure that comprises of actors and relations. Actors were discrete individuals, corporate, or collective social units. Requirement elicitation involved a wide range of people and these people included customers or clients who pay for the system. It elicited requirements from the stakeholders and the model also prioritized the requirements accurately and handled information overload by drawing stakeholders.

Lim, et al (2011) described a stakeholder analysis features in the stake source to support the identification and prioritization of requirements. Requirements were organized in a simple hierarchical structure where child nodes were more specific than parent nodes. Analyst and stakeholders provided specific requirements and their descriptions at different levels. It also detected the requirement by asking stakeholders to rate a given list of requirements and suggested other requirements.

Harrison, et al (2010) interpreted a stakeholder management approach and managed the stakeholders utility. An enterprise that manages stakeholders sought to detect and understand the welfare of the stakeholders.

A stakeholder refers to an individual, or collection of individuals or a firm. It also seeks to understand the stakeholders utility function was made up of two types of knowledge. Stakeholder utility function changes due to the number of factors including innovations, entry of new firms and social influence on trends.
Fabian, et al (2010) depicted a conceptual framework for security engineering with a strong focus on security elicitation and analysis. Multiple requirements obtained from the customers were interdependent and can interact with each other.

These interactions were positive, negative or exhibited conflict results during implementation. Analysis and management of requirement interactions were necessary and important part of requirement engineering. This conflict may arise due to a number of reasons at different stages of requirement engineering. The views of the stakeholders were generally inconsistent. They have different and contradicting requirements. Identification heuristics were used by the requirement analyst to study the existing security and privacy policies, requirement analysis and design documentation in order to identify both strategic and tactical goals related to the organizational assets. These goals were annotated, including information about stakeholders and responsibilities.

2.4 FEATURE CLASSIFICATION

Cruzes, et al (2011) reviewed a software engineering process in order to enhance the transparency of the software requirements. The software requirements concentrate on a well-defined question. This aimed to provide an answer by synthesizing the findings from a relatively narrow range of quality assessed. A fundamental distinction regarding objective of such reviews was whether to provide knowledge support or decision support.

Zhan, et al (2011) represented a requirement analysis based on the classification framework. Activity based instruction was recognized as an effective way to engage the students and improve the learning performance. Field notes of online learning activities were taken by researchers that were observed during the online learning activities. Activity records were written by instructors who organized online learning activities. The data sources included interview notes and records from instructors and student related to the online learning activities, which were called as a collection of data. Some basic information was extracted from the original data by using a sample recording template. The classification framework was set up and the data were collected from various sources, including online instruction, observing online platform and interviewing instructor.

Axial coding was the process of connecting related categories with their subcategories where coding occurs around the axis of category, linking categories by properties and dimensions. Axial coding includes the following applications:
- **Synchronous Discussion:** Online synchronous was done through real time software such as Net Meeting, video conference, and live streaming.

- **Asynchronous Discussion:** Online asynchronous discussion was carried out through posting text, pictures and short videos on course platform, online forums, etc.

- **Instructor-Shared Resources:** The instructor uploads coursework, reading materials and other related resources were provided guidance to assist students learning.

Argumentation theory has been the most frequently used activities in distance education. It was easy to organize and cooperate with the available instructional materials. However, learners usually require better online discussion interface, higher hardware quality and more frequent interaction than the existing systems.

Nayak, et al (2012) constructed a framework for identifying the advancement of defect identification. The delivery of error free software was important for the software development organization. Software defects were the basic reason for malfunctioning and software failure, which imposes a direct impact on software reliability. Initial requirement was the requirement collected at the earliest stage of the software development. After processing stage, the initially collected requirements were placed under several specific heads. The requirement scope was directly proportional to the customer’s objective and was responsible to manage statements from the user’s point of view.

The Decision table includes two quadrants of conditions; one was for requirement defect and other for Action strategy. The first quadrant statement implements one or more conditions for requirement defects. In the second quadrant of decision table condition entries was meant for the completing the condition statement.

Qin, et al (2012) illustrated a model for classification of customer requirements. The requirements were classified into five types, namely, binary, optional, parameter, description and explanation. Generally, the expert system in the customization defines the characters that distinguish the product type distinctly and establish a decision tree.

The analyzing function of products that share different functionalities was established. The candidate modules was combined effectively under the constraints
relative to the module models and attributes, by which some combination of invalidate module was avoided and hence the efficiency of product configuration was promoted.

Franch, et al (2013) deliberated a framework for reusing the requirement knowledge following a pattern-based approach. The specific pattern based approaches were used in a specific model or diagram. The goals were achieved by including the use cases in the system, which denotes the properties of the system.

Based upon the notion of variability proposed in domain engineering, the common requirements were necessary in any system of the domain. Refinement-oriented pattern based approaches was established to attain certain goals by using goal oriented modeling language. Requirement engineers were guided in the process of deciding the requirements that were necessary to implement a system to satisfy the certain goals. Template-oriented pattern based approaches was the ultimate goal of refinement-oriented based approaches, which was used to produce an SRS.

This pattern based approach does not follow any predefined structure. This approach was very basic and was enriched with some searching facilities. The additional information about the context was applied to guide the requirement engineer during the elicitation process.

Ramzan, et al (2011) introduced a requirement prioritization in software development. The requirement elicitation was done electronically and all stakeholders submitted their requirements, which were designed on web scripts. Stakeholders submit their requirements according to their own property. In the first tasks, the review was given by the experts for all the requirements. Requirement prioritization was based on fuzzy logic and was a multi-level approach. In this technique, stakeholders, experts define a fuzzy logic based system to perform separate prioritizations at three different levels.

Liao, et al (2013) demonstrated a requirement classification techniques for software development phase. The software requirement is acquired by customer demands and the range of software requirement was determined and managed on the basis of requirement elicitation. The management of software requirement scope was groundwork and the requirement elicitation was processed according to the priorities of the requirements. The process of requirement elicitation changed the requirement and project scope, which results in project failure. This was refined and perfected constantly during
the RE and the software sets the preliminary project scope in accordance with the software project feasibility.

Requirement management aimed at enhancing the demand management and reduced the requirement changes. The accuracy of the requirement was affected by the definition of the enterprise information project. Enterprise information was essentially a great leap of enterprise existing organization structure, management concept and management mode. The project had clear aim and developed the available function.

Liang, et al (2011) represented an integrated modeling approach for requirement classification. ModelicaML was an UML profile and a language extension for modelica. The primary aim of ModelicaML was to enable the graphical system modeling using the standardized UML notation together with the modeling and simulation power of modelica. ModelicaML defined different views on system models based on a subset of UML and reused some concepts from SysML. ModelicaML was designed to generate code from the graphical models. The ModelicaML profile was an extension of the UML metamodel, so it can be used in both UML and SysML.

The ModelicaML formalized the requirements for design verification and the verification scenarios were modeled that captures a specific course of actions, which stimulates the design model in order to cause a particular reaction. Verification scenarios were modeled that captured a specific course of actions, which stimulated the design model in order to cause a particular reaction.

Qin and Wei (2011) proposed a framework for requirement classification and analysed these requirements. The customer requirements of product included functionality, performance, appearance, and dimension of the product. The customer requirement was significant in the customer domain and included the essential characters of the product needed. The customer with higher product knowledge outlined the requirement precisely.

The customers communicated with the sales staff in technology traits and engineering characteristics. The requirement information was translated to value the module attribute variable directly, which was easy for mass customization. The binary type represented the additional attribute function or configurable object. Formally, there were two elements or values in the domain, namely 0 and 1 where 1 implies the required object and 0 denotes the objects that are not required. Optional type defined available
finite options where each of them was substituted for each other. But it represented different character or performance of the product. The parameter type was numeric, which describes the quantity variety of component and numerical attribute.

Naseer, et al (2013) developed a computer aided software engineering and improved the software development process. Computer aided software engineering tools facilitated the activities during software development process. As a result, increased productivity and improved quality was targeted. The tools varied in functionality from a single editing application, supporting the document requirements to a complex set of tools and also facilitated in requirement elicitation to code writing.

Requirement traceability was very important in requirement engineering process. Asghar and Umar (2010) depicted a customer off the shelf components (COTS) in requirement engineering. RE acted as foundation for any software and was one of the most important tasks. The entire software was supported by four pillars of requirement engineering processes. Functional and non-functional requirements were considered as bricks for software development process. Bugs in requirements were not detected during the development phase and poor requirement leads to modification in requirement specification.

It also acquired re-designing, re-implementing and re-testing the requirements. The model encapsulated overall challenges faced in requirement engineering rather than identifying them in any particular domain. A systematic understanding of requirement engineers to broader their vision and to detect upcoming problems and risks in requirement engineering were introduced. The goal of the requirement engineering process was to investigate the tasks which are to be performed in the boundaries and constraints in the software. Acquiring and comprehending the requirements for complex domains or critical systems always have great challenges for requirement engineers. As a result, requirement specification was vague, perplexing and ambiguous.

Jamaludin, et al (2011) proposed a Project-Base Learning from the elicitation process. RE was still unresolved because of the rapid changes in technologies. Project-based learning courses identified the suitable tools and expertise people with a guideline, which refers a tertiary culture. After identifying the requirement of the customer, the course was developed and the students were trained for the new technology. It comprised
of software requirement, software design, software development, and software testing and software maintenance. If RE did not capture a right problem or requirement from the user then the solution for software artifact was not accurate. It was used to control the change management and these changes affect the overall projects.

2.5 BAYESIAN NETWORK FORMATION

Bensi, et al (2013) described an efficient BN formulations for modeling the performance of general systems. Bayesian Network (BN) was a convenient tool for probabilistic modeling of system performance to update the reliability of the system. Standard BN structures defined the system node as a child of its constituent components or its minimum link that lead to converging structures. The formulation used an integer optimization algorithm to identify the most efficient BN structure. The BN was an ideal framework for the analysis of such systems, particularly when updating of the probabilistic model.

The BN was a graphical model consisting of nodes and directed links, which respectively represent random variables and their probabilistic dependencies. These variables represented the states of the components of a system, or their capacities and demands. It also provided a convenient means for modeling dependency between the component states. For such an application, efficient computations and near real-time inference in models of large systems were important. The considered systems were characterized in terms of their Minimum Link Sets (MLSs) and Minimum Cut Sets (MCSs).

These BNs for simple systems differ by defining a systematic approach to develop an efficient BN structure for modeling the reliability of complex system. It was characterized by a directed acyclic graph consisting of a set of nodes implementing random variables and a set of links denoting probabilistic dependencies. The potentials and consequently the efficiency of the inference algorithm depend on the ordering of node eliminations. However, there exist computationally optimal elimination sequences, all of which lead to the same domain set.
The domains corresponding to the optimal elimination sequences were called cliques. The total size of the potential associated with these cliques was considered as a good proxy. These proxies were used computing the performance of inference in BN and also used to evaluate the efficiency of the BN system formulations. Figure 2.3 clearly depicts the processes involved in requirement engineering. The total size of the potential associated with those cliques was a good proxy for the computational effort required for performing inference in the BN and was used to evaluate the efficiency of the BN system formulations.

Fenton, et al (2010) presented a quality modeling using Bayesian networks. In this, each of the random variables defines a statistical distribution and deals with a wide range of possible outcomes. The nodes in this model were indicated by numeric scales and BNs was not limited to any set of states, which was assigned as a probability handled by a BN. An observation reduces the marginal probability distribution of the observed variable to a unit probability of the observed state and zero otherwise. Dynamic Bayesian Network was
a temporal model denoting a dynamic system, which changes state usually over time. It also contains the variables relating to the extent to which the project was likely to produce accurate and clear requirement and specifications.

This approach was very flexible and does not insist on an organization using specific metrics and depends on the regularity. The Bayesian resource model was powerful and compared with traditional software cost models. The observation of resource model was entered anywhere to perform various types of trade-off analysis and risk assessment.

Wagner, et al (2010) developed a Bayesian networks model to assess and predict the software quality using activity based quality model. These quality models were described in a structured way, which represents the software’s quality. The quality and functionality of the software system were specified in order to control the activity based quality models. The two main uses of quality models in a software project were:

- It forms the basis for defining the quality requirements
- It was used to ensure the quality assurance techniques.

The requirement engineers commonly limit the well-known quality attributes (reliability, maintainability, etc) as defined in a quality model. The quality engineers always measure a certain metrics such as number of faults in the system, which was detected by inspection and testing. The information about the characteristics of a system contains important facts about the process, the team, the environment and their respective influence. The four elements of the meta-model were namely, entity, attribute, impact and activity.

An entity can be anything, which influence the software quality e.g. source code of a method or the involved testers. These entities were characterized by attributes such as structuredness or conformity. The combination of an entity and an attribute was called fact. The relationship between the facts and activities has an associated uncertainty, statistical methods, which were necessary. The two major reasons were:

- It cannot determine the exact relationship, but can derive an uncertain range
- The measured values can be uncertain, e.g., values from expert opinion.

The statistical method needs to be directly model the dependencies of different factors from the quality models. Bayesian networks i.e., Bayesian belief nets or belief
networks were a modeling technique for causal relationship based on Bayesian inference. It was represented as a directed acyclic graph with nodes for uncertain variables and edges for direct relationships between the variables. This graph, models all the relationship abstractly and the code complexity.

It influences the testing effort and the number of field failures of software. For each node or variable there was a corresponding node probability table. These tables define the relationship and the uncertainty of these variables. These variables have been usually discrete with a fixed number of states. For each state, it gives the probability of a variable in the table. The parent nodes influence the current node and define these probabilities and independence of the states of the parents. The process of building a Bayesian network comprises the identification of important variables that was modeled.

This was represented as nodes and the topology was constructed, which specifies the NPTs. Each of these steps was important and non-trivial in the identification of important variables. In the second step, the creation of the topology utilizes the assumption that the model builder decides on the dependence and independence of the identified variables. The problem of constructing the NPT was widely acknowledged in the literature. A major part of the problem involves defining the quantitative relationships between variables. Bayesian networks were used as an assessment and prediction model to derive the results systematically from an activity-based quality mode. The resulting Bayesian network contains three types of nodes, namely:

- Activity nodes that indicate activities from the quality model
- Fact nodes that denote facts from the quality model
- Indicator nodes that represents metrics for activities or facts.

Feng, et al (2012) devised a Bayesian network to control risks in software projects. This network provides an effective system method for software project risk analysis. A BN was a network of nodes connected by directed links with a probability function attached to each node. The network was a directed acyclic graph and there was no directed starting path or ending path at the same node. If a node does not have any parent, then the node contains a marginal probability table. If a node has parents or more links then the node contains a conditional probability table.
Each node represents a variable A with parent node indicating variables and was assigned to a conditional probability table. Once the BN analysis model was built then the probability of risk was predicted. This helps to take precautions and reduces risks in requirement process. The Bayesian network topology structures fit for the change of characteristics of risk factors in software projects. The Bayesian network topology has strong self-learning ability and is easily expandable in nature. This network analyzes the risk factors of software projects and was more suitable than other approaches.

Hu, et al (2012) illustrated an intelligent analysis model for outsourced software projects based on empirical data. Bayesian network suits for small and incomplete data sets and structural learning possibility. The visual modeling of cause-effect relationships help to identify risk sources to provide explicit knowledge for risk analysis. It provides probabilistic estimates and explicit treatment of uncertainty. The key to establish a Bayesian network based risk prediction model with good interpretability from data was introduced. Bayesian Network integrates the accurate probability distribution and prior knowledge of experts, which were adequate for uncertainty modeling.

It was relatively easy to build the visual modeling that helps to detect the source of risks. It incorporates structure constraints between risk factors to search for the best network. Constraints among risk dimensions was clear and can easily obtain literature support and industry approval. Bayesian belief network was one of the most theoretical models in the field of uncertain knowledge expressing and reasoning. Based on graph theory and probability theory, the network can establish the relationship between variables reasoning from incomplete, imprecise or uncertain information. This network structure learning algorithm was generally classified into two types:

- The search+score algorithm
- Dependence analysis based algorithm

The search+score algorithm aims at finding an optimal network structure. It measures the goodness of each explored structure in the space of available solutions. This algorithm uses a scoring function, which is often defined as a measure of fit between the network structure and the data. The optimal network structure is the highest degree of fit with the training data. The dependence-analysis algorithm based on quantitative validation
of mutual information is used for training the data. In the Chengs algorithm, dependence analysis was divided into three phases, namely:

- Drafting
- Thickening
- Thinning

In the drafting phase, if any pair of nodes was not independent within the specific condition set an arc is added between these two nodes. In the thinning phase, it carries out a conditional test for each arc that was received in the thickening stage. It also removes an arc if the linked pair of nodes was conditional independent for some given specific condition set.

Voinov, et al (2010) described a stakeholder engagement, collaboration, or participation shared learning or fact-finding. It was of various types and has emerged as a powerful tool that comprises of following advantages:

- Enhance the stakeholder knowledge and understanding of a system and its dynamics under various condition
- Identification and clarification the impacts of solutions to a given problem

Participatory modeling is a generic term that does not associate with any particular group. At the same time there were names that were quite closely connected to particular schools or researchers and serve as trademarks of these groups. Group Model Building was a method based on formal modeling, more on causal loop diagrams and similar visual tools. It involves a group of people, stakeholders in one or more sessions to build the conceptual model. A facilitator who has experience with the method helps the group to build the model. This usually stays neutral of the content and the modeling was considered as a process of building mutual understanding, defining terms, notions and sharing experiences.

This session starts with reading a concept learning history, or even an unsorted pile of interview facts and narratives. During and after the session the so called Learning History was extended and then prepared for the further implementation in decision making. Companion modeling was usually associated with a stakeholder process that includes a combination of agent based models and role playing games. The basic objective
of this modeling was to enhance the awareness of stakeholders of the variety of points of views and their consequences in terms of actions.

The Bayesian network represents patterns of probabilistic dependence or in other words a BN defines a relation between variables in terms of the conditional probabilities for each variable included in the network. It allows reasoning under the uncertainties that were associated with these probabilities. A BN model situation was characterized by inherent certainty and helps to understand the statistical inference.

Henrlon (2013) addressed practical issues in a Bayes Belief network and demonstrated the feasibility of the requirement engineering. Sensitivity analysis was done in order to discover the relative importance of different indicants. This was also useful during the construction of the network and quantification of influences to help identification of part of models and the parameters, which were critical.

Weber, et al (2012) deliberated the applications of Bayesian networks in risk analysis, dependability and maintenance. BN was particularly suitable for collecting and representing the knowledge on uncertain domains but also enable to perform probabilistic calculus and statistical analyses in an efficient manner. The main drawback of BN was that there was no specific semantic to guide the model development and to guarantee the model coherence. This models integrates various dimensions correlated with systems behavior in reliability, risk analysis and maintenance fields.

It concerns with the translation of the classical dependability model into a BN model. Srinivas, et al (2013) implemented a BN that strongly depends on the amount of expert information available. The goal of the system was to obtain a network that explicitly reveals as much information regarding conditional independency as possible. The network was built incrementally adding one at a time. The experts’ information and a greedy heuristic algorithm were used, which tries to keep the number of arcs added at each step. The network was built incrementally adding one at a time and the experts’ information and a greedy heuristic that tries to keep the number of arcs is added at each step. The probabilistic model predicts and answers queries about independencies in the domain.

Bencomo, et al (2010) depicted a requirement approach as runtime entities. Requirement reflection was important because software systems were self-managed and
adapted continuous changes. These approaches follow the non-traditional principles such as

- The role of explicit runtime representation of system requirement’s as an appropriate formalism for donating the systems with self-awareness capabilities.
- The subsequent need to maintain the relationship at runtime between goals and underlying system structures.
- A recognition that uncertainty was intrinsic to SASs and was managed at all stages of the life-cycle including runtime.

Peng, et al (2012) presented an algorithm to modify the probability distributions represented as BN. The modification was done by changing the conditional probability tables of the network while leaving the network structure intact. This algorithm was valuable in BN construction, merging and refinement when low-dimensional distributions need to be incorporated into the network.

Plant, et al (2011) demonstrated a BN using both simplified scenarios and then a real world predication example. Using the simplified scenarios and real-world prediction, the uncertainty in the input data was accurately transferred to uncertainty in the predictions.

The uncertainty in the boundary was computed only when the BN does not contain the uncertainties and forcing data. The BN reduces the dimensionality problem and estimates the uncertainty for all predictions. Fenz, et al (2012) illustrated an ontology based approach for constructing Bayesian networks. It was commonly used for determining the probability of events that were influenced by various variables. These probabilities encode degrees of belief about certain events and a dynamic knowledge was used to strengthen, update or weaken the assumptions.

2.6 CLUSTERING TECHNIQUES

Niu, et al (2012) suggested a clustering requirements artifacts from the ever-growing software project repositories. The automated cluster labeling techniques in information retrieval was implemented by automatic labeling of requirements clusters. In labeling, a cluster of documents and similarly a cluster of web pages or a cluster of
requirements an automatic method was used. This characterized the cluster items which were generated as cluster labeling. The three basic categories of cluster labeling were described as follows

- Cluster–internal labeling
- Differential cluster labeling
- Hybrid labeling

The cluster internal labeling selects the labels by considering the contents of the clusters. The differential cluster labeling labels a cluster by comparing the terms in one cluster with the term occurring in other clusters. The hybrid labeling propagates labels by integrating both intra and inter-cluster features. Khan, et al. (2012) depicted a component selection process that helps system analysts to model interdependencies. The component selection process was developed and comprises of three phases

- Goal-oriented specification
- Dependency analysis
- Cluster analysis

Different clustering techniques were used in the requirement analysis of traditional and component based systems. The clustering algorithm was used to assist the system analyst in selecting the suitable components for a Component Based System (CBS). This approach uses classes defined in a domain model. The classes with the highest number of relationships were grouped together into one cluster. This model uses a set of heuristic for further refining the clusters and subsequently selects a component for a CBS. The subsystems were re-organized to attain high cohesion and low coupling among subsystems of CBS.

The re-organization was done based on cohesion and coupling values by using a clustering algorithm. Various fuzzy clustering techniques were applied to the component selection. The goal model arranges a high-level goals and concrete-level goals using refinement relationships. It helps to detect the Concrete Level Goal (CLGs) that was used for component selection. The cluster based component selection process was shown in figure 2.4, which consists of three phases. The component selection process requires considering the semantic dependencies between CLGs. These dependencies were modeled
as a signed graph with every node representing a CLG and every edge implementing a semantic dependency.

![Cluster Based Component Selection Process Overview](image)

**Figure 2.4 Cluster Based Component Selection Process Overview**

Each edge was assigned a positive or negative weight to impose the nature and the strength of the corresponding dependency. The component selection process uses a signed graph clustering algorithm and a matching index to select the CBS components and establish multiple candidates for each clustered requirements of the CBS.

Arafeen, et al (2013) presented a test case prioritization clustering algorithm in order to improve the prioritization techniques. The requirements were clustered by using the textual similarity among the requirements. Textual similarity was studied in the field of text mining for clustering the documents and information retrieval. This process includes three tasks:

- Term extraction
- Term document matrix construction
- K-means clustering
Term extraction was considered for each requirement as a bag of words or terms. In this process, these words were extracted from each requirement and the words that were added have no meaning to the sentence. After rejecting these words, all distinct terms across all the requirements were identified and used in the subsequent tasks. Term-document matrix was created by receiving the distinct terms of the term extraction process. In this matrix, the row corresponds to the distinct terms across all the requirements.

The term document matrix lists the frequency of the word occurring in the corresponding requirements. It also lists the term frequency-inverse document frequency, which was used for retrieving the text. K-means clustering was suitable for document clustering and it permits to specify the number of clusters. K-means allocate each requirement to the clusters, which reduce the inter-cluster sum of squares. Figure 2.5 shows the steps involved in clustering the requirements and prioritization of clusters. The users are enabled to utilize the priority information while selecting the test cases from each cluster to obtain a complete set of reordering test cases by prioritizing the requirement.

Figure 2.5 Clustering of Requirements and Cluster Prioritization
The stakeholder requirements were classified as follows:

- Commit (C) - developer implements the given requirements
- Target (T) - Developers strive to represent the given requirements and does not guarantee
- No-Commit-Developers implement the given requirements if they have time.

In the original order of clusters, the clusters were visited one by one and the same process was repeated using a round robin method. The requirements based clustering approach incorporates traditional code analysis information, which improves the effectiveness of test case prioritization techniques.

Reddivari, et al (2012) proposed a cluster hierarchies for visual exploration tasks in requirement engineering. An automatic grouping of requirements into clusters was classified by an internal coherence and or an external isolation. These activities include feature identification, system modularization and automated tracing. The hierarchical visualization of requirement clusters was addressed by several traceability issues. This visualization was routinely augmented for textual requirements with summarization that aggregates large amounts of information into a single representation for shared understanding and quick absorption by stakeholders. Niu, et al (2012) proposed an approach for the information retrieval based automated tracing by assessing the cluster hypothesis. Clustering plays a major role in information retrieval.

On the document side, the searchable artifacts in tracing comprises of individual requirements, classes, test cases, etc. Such a collection tends to be significantly smaller than the document collection targeted in a typical search or online library search. The search space was reduced according to the needs of requirement tracing through document clustering. On the query side, a trace query was composed from the text of a requirement or other software artifact. The requirement was concise, primitive, and ambiguous.

The query side clustering was less useful in requirements tracing. The baseline process was commonly adopted in the state-of art tracing tools like RETRO and Poirot. The human analyst selects the requirements artifact to trace the IR algorithm and retrieves the traceability links by computing the similarity between the query and the software artifacts in the repository. Pruning aims to automatically discard the portion of the
retrieved results with a low density of correct links. The effort required to discard false positive become higher than the effort required to validate the correct links.

The clustering technique groups the correct and incorrect links into high quality and low quality clusters. Mullner (2011) developed a hierarchical agglomerative clustering, which perform more efficiently in the general purpose. The input data were the requirements given by pair wise similarities data points and the output was a stepwise dendrogram. The output of a hierarchical clustering procedure was called a dendrogram. A graphical representation was drawn from the data and the graphical representation loses information at the same time contain extra information, which was not contained in the data itself.

Apel and Beyer (2011) developed a model to assess the characteristics of feature cohesion in software product lines. Clustering, partitions a set of elements into subset according to certain properties. The selection of nodes on the dependency graphs of a software system considers the set of elements to be clustered and guide the clustering algorithm to divide this set according to the graph structure. Layout based clustering includes the distances between the elements in a two-dimensional space, in which related elements have close positions and unrelated elements have distant positions.

The mapping from the nodes to position was called layout and the advantage of clustering layout was that, it reveals a degree of relatedness, where the two elements can be very close in the same cluster or belong to two different clusters. Clustering criteria for layout based clustering was applied for software graphs and a feature has a higher cohesion than coupling if its elements were close to each other. Goknil, et al (2011) suggested a requirements traceability in the software engineering community.

Requirement traceability was the ability to relate the requirements back to stakeholders and forward to corresponding design artifacts, codes, and test cases. The relations between requirements influence a number of activities during software development, such as consistency checking and change management. The semantics of the relations were provided with formalized first-order logic. The application of requirements reasoning based on formal semantics resolves many of the deficiencies observed in other approaches. It provides an explicit structure to requirements document called Metamodel.
This metamodel includes mostly commonly found entities. Interference and consistency checking aims at deriving new relations based on given relations and determine the contradictions among relationships. These were used for supporting a form of logic programming based on facts and rules. The modeling process was forked into two activities, consistency checking and inference, where these activities were processed in parallel. The requirements model was updated with inferred relations. An inconsistent part of the model was determined and the consistency checking of requirement model enriches the set of requirements relation.

It allows the creation, storage and retrieval of requirements models and bridges the user interface layer with the data layer was done at modeling environment. Acher, et al (2011) presented an approach that distinguishes between domain variability and software variability, which explicitly breaks the variability spaces into two types. Feature modeling aims at implementing the common variable and features. Software Product Line (SPL) engineering usually promotes the systematic reuse through two complementary processes. It also improves the deployment cost and quality of application development. First, the reusable platform was modeled on the top of different products by domain engineering. Second, the application engineering derives an appropriate product from the SPL platform. This approach considers the interactions between specifications and implementation choices and the Feature Model (FMs) was interrelated with transformation rules.

In SPL, the specification of choices in the domain causes changes in the software platform and leads to better configurations of the platform. These inter-model constraints make dependencies and interaction between features by mapping the task specification to the software implementation. Veerappa, et al (2011) depicted a cost-value based requirements selection process. Multi-objective decision problem was ubiquitous in requirement engineering. This approach was used to apply the search-based techniques to propagate a set of non-dominated solutions formally known as Pareto front. It characterizes all solutions for which no other solution performs better on all objectives simultaneously.

The shape of Pareto front was analyzed and provides similar levels of goal attainment correspond to minor variants within a same design. The requirement engineering problems were taken into consideration with a multitude of stakeholder’s goals that were not directly comparable one to another. The idea of clustering solution in a
Pareto optimal set was used to understand the optimal solution in the system design. The design decision in such problem comprises in selecting optimal values for a small number of continuous variables. The identification of such groups of solution that share common design decisions were useful for the following reasons:

- It helps decision makers to understand large solution sets instead of inspecting a large number of individual solutions. A smaller number of groups of related solutions were focused on their attention based on the important characteristics of the group rather than the particularities of their individual solutions.
- This allows the decision to be made incrementally and considered to select one solution in a large set of individual solution decision makers decides for a group of solutions before selecting one solution within the group.
- Such groupings reveal areas of the generated Pareto front where significantly different requirement selections have similar levels of objective attainment.

Na, et al (2010) implemented an algorithm for clustering large sets of data. The computational complexity of the standard k-means algorithm was high owing to the need of reassigning the data points. A simple data structure was used to store some information in each iteration, which was further used in the next iteration. K-means was a numerical, unsupervised and non-deterministic iterative method. It was simple and fast used in many applications. The standard k-means algorithm calculates the distance from each other object to all the centers of k clusters.

It executes the iteration for each time and the compilation time was high for large databases. This algorithm sets two simple data structures and retains the labels of the cluster and the distance of all the data object to the nearest cluster was identified. If the computed distance was smaller or equal to the old center, then the data object stays within the cluster and was assigned to the previous iteration. Therefore, there was no need of computing the distance from the data object to the other k-clustering centers. The k-means algorithm needs pre-estimated number of clusters k, which was same to the standard k-means algorithm.

Ling, et al (2011) deliberated a scalable, distributed and adaptive routing algorithm. The algorithm retains the throughput and optimality of the original back pressure algorithm and reduces the memory complexity by reducing the number of queues
maintained at each node. Routers were grouped into clusters and associated with each cluster was one or more gateway routers that allow to transit into the clusters. Each router requires only being aware of the destinations within its cluster and the gateway routers associated with the other clusters. Loose source routing was used in the cluster based back pressure algorithm. In this, the source first chooses a gateway at the cluster containing the destination and then the packet was routed to the destination via gateway using the back-pressure.

2.7 RISK ASSESSMENT IN SOFTWARE REQUIREMENT ENGINEERING

Asnar, et al (2011) deliberated a goal driven risk assessment in requirement engineering. Risk analysis was traditionally considered as a critical activity for the whole software system’s lifecycle. Risks were identified by considering the technical aspects, namely, failure of the system, and unavailability of services. Here, the risks were analyzed with the stakeholder’s interests. According to goal oriented requirement engineering, analysis of stakeholder’s goals leads to alternative sets of functional requirements. Shifting risk analysis to the early phases of the software development process can give the advantage of considering risk mitigation.

Sharma and Kumar (2013) proposed an efficient risk analysis based risk priority in requirement engineering using modified goal risk model. The relations of goals were defined between multiple goals and events, which indicate the importance of a particular goal. The event may be considered as a risk according to the likelihood value. So the inter relation values of the goals and events gives the impact of the event on the particular goal. In order to analyze the risk in achieving some particular goals, a set of candidate solutions were generated. These candidate key were evaluated on the basis of an affinitive value of the goals according to their events. The risk affinitive value was calculated from the different set of risk parameters, which was set like high, medium and low. The risk parameters clearly evaluate the affinity of the event to a particular set of goals.

Khan, et al (2012) depicted a review of software risk assessment and estimation models. Risk identification produces lists of the projects specific risk items to compromise a project success. A typical risk identification technique includes examination of decision drivers, assumption analysis and checklist. Risk analysis assesses the loss probability and loss magnitude for each identified risk item and it access the compound risk in the risk
item interactions. This technique includes performance models, cost models, network analysis, etc. Risk monitoring involves tracking the projects progress towards resolving its risk items and taking corrective action appropriately.

Franqueira, et al (2011) introduced a risk based argument method for security in requirement engineering. This method uses public catalogs of security expertise to support the risk assessment and to guide the security argumentation. It differentiates between the risks that should be mitigated by the system context and the risks that should be mitigated by the system. It also analyzes the risks to be mitigated by the system and prioritize the risks found through arguments.

2.8 SUMMARY

Requirement specification provided by the stakeholder was analyzed and the priorities of the stakeholder roles were detected from an individual perspective. This activity led to the omission of stakeholder roles and was not able to differentiate the suitable stakeholders. Existing methodologies does not effectively handle the sparse requirement specification of stakeholders. An ideal method in requirement elicitation should identify and priorities the stakeholders from a global perspective. In the proposed method, preprocessed stakeholder requirement is considered as input. The feature evaluation, feature selection and fuzzy classification are done to predict the exact stakeholder’s requirement by using weka tool. The classified data are grouped together based on the relevancy of stakeholder’s requirement. The clustering technique is used to identify the homogeneous group of stakeholders. The Jaccard similarity mechanism is applied to predict the similarity between stakeholder requirements. In this mechanism, similarity between the stakeholder’s requirements is determined by analyzing the distance between the stakeholders’ requirement vector. A smaller distance implies a higher similarity between the requirements. The proposed techniques are discussed in the following chapters.