REQUIREMENTS ENGINEERING FRAMEWORK USING CONTEXTUALIZATION, PROFILING, AND MODELLING

Arunkumar Khannur[1], Manjunatha Hiremath[2]

[1]Computer Science Department, Christ University,
Hosur Rd, Bhavani Nagar, S.G. Palya, Bengaluru, Karnataka 560029. India.
[2] Hosur Rd, Bhavani Nagar, S.G. Palya, Bengaluru, Karnataka 560029. India.

ABSTRACT

The proposed Requirements Engineering Framework (REF) is novel and efficacious in producing quality Requirements Specifications. The framework is based on the principles of design thinking that include empathy; system visualization to understand the system, its components, and the pattern of their interactions; and observing the possible behavior under different contexts and situations. The proposed framework includes requirement elicitation, requirement understanding, defining requirement goals, requirement profiling, requirement contextualization, and requirements modeling. All these are carried out by keeping ISO 25000 standards for software engineering at the core. The proposed REF will be impactful in implementing modern-day projects which are disruptive and becoming more and more context-aware systems because of machine/ deep learning, data engineering to deal with structured and unstructured data, sensor technologies, interactive technologies, 5G technologies, and control-related technologies. The framework also will address the limitations of CMM-based or agile-based approaches.

KEYWORDS

Requirements Modelling, System Visualization, Contextualization, Quality Profiling, Risk Profiling, Quality Engineering, ISO 25000.

1. INTRODUCTION

Requirements Specification (RS) is the first phase which is the basis for developing all subsequent phases in the software development life cycle (SDLC) [1]. During the RS phase, activities that include collection, corroboration, representation, and specification of functional and non-functional requirements [4] are carried out and an RS document is produced which is a collection of all the requirements. According to IEEE, a requirement captured in the RS document is (1) A condition or capability needed by a user to solve a problem or achieve an objective. (2) A condition or capability that must be met or possessed by a system or system component to satisfy a contract, standard, specification, or other formally imposed document. (3) A documented representation of a condition or capability as in (1) or (2). [2] The requirements in the RS can be 'must have' (required) and 'good to have' (expectations). Also, RS shall be reviewed, approved, and configuration controlled, and shall serve as the formal agreement with the development team to deliver. Also, since RS is a Configuration Item (CI) any requirement to change RS mandates adherence to effective configuration management and change management.
Requirements Engineering (RE) is used to produce RS. RE and its activities have seen a continuous transformation to deal with ever-emerging ground realities and trends. In earlier

David C. Wyld et al. (Eds): SOEN, SIPP, PDTCA, ITE, CCSIT, NLPCL, DaKM, BIGML, AISC -2023
pp. 53-68, 2023. CS & IT - CSCP 2023 DOI: 10.5121/csit.2023.131205

years, the focus of software was delivering functionalities with number crunching and predominantly to users of an organization. Users were forced to learn and use the system. Computational devices used were monolithic with a central processing unit connected with many dumb machines to crunch structured data. The next stage of the Information Technology (IT) revolution has brought content, communication, connectivity, and computational powers together and started providing applications that were thought of as impossible. At his juncture, the focus of requirements was not only functionality but also, non-functionality. However, building applications continue to be using appropriate algorithms and structured data. However, systems had capabilities to serve diversified users by using multilayered internet-based solutions across the world. Then arrived, starting in 2010, the world of disruptions with the emergence of sensor technologies, interactive technologies, 5G technologies, data science and Machine Learning (ML) related advancements, automation, and tools that have created disruptions everywhere with altogether new and revolutionary innovations and applications. As the world has seen such a major shift in the nature and type of applications, the development of such applications also requires new ways and means of developing RE as well. This paper is proposing a novel requirement engineering framework (REF) as an effort to deal with the situation created by these disruptive technologies in developing software and applications. The proposed REF uses requirement elicitation, requirement understanding, defining requirement goals, requirement profiling, requirement contextualization, and requirements modeling. An attempt is made to do requirements modeling by using ISO 25000-based software and quality engineering principles. The proposed REF considers different viewpoints [5], use the RE process, and produces an RS document.

2. LITERATURE REVIEW

Software development till the 1960s was majorly developing only the code by an individual. NATO Software Engineering Conference 1968 [38] was organized to identify the challenges of developing large-scale software systems and to address them. This conformance brought in the much-required impetus on how to develop software. It laid the foundation of modern software engineering principles and introduced the idea of phase-wise development of software, that is Software Development Life Cycle (SDLC). The phase-wise development approach laid the groundwork for modern software engineering, providing a structured approach to software development. It typically includes Requirements Specification, Design and architecture, Implementation, Testing, Deployment, and Maintenance. This approach involves breaking down the software development process into distinct phases. Each phase has specific goals, deliverables, and exit criteria that must be met before the team can move on to the next phase. The purpose of phase-wise development is to ensure that each phase is completed before moving on to the next one. This helps to ensure that project risks are identified and addressed early in the development cycle and that the project stays on track. As can be seen, the Requirements Specification (RS) is the initial phase in the SDLC. It is efficacious in producing quality artifacts at the end of every phase and the SDLC. This heralded a series of changes in the way software is built including RE which is the basis of developing RS.

RE is a discipline with activities, techniques, methods, and tools to produce RS. The RE process involves eliciting, analyzing, specifying, validating, and managing the requirements of software systems. The discipline is continuously evolving to address the challenges of the ever-changing technology landscape. Following are some landmark and transformative stages of this evolution:
Waterfall Model: The requirements elicitation process was typically a one-time effort in the waterfall model that often led to a disconnect between the user's needs and the delivered software. [39]

Iterative Model: In the iterative model, requirements engineering is an ongoing process that changes based on the feedback received from the users. [40]

Agile Methodology: In the agile methodology, requirements are continuously elicited and prioritized based on their business value. [41]
DevOps: In DevOps, requirements engineering is integrated into the development process, and the feedback loop is closed by gathering feedback from end-users. [42]

Overall, the evolution of requirements engineering has focused on ensuring that the requirements result in software that meets the user's needs and has business value.

All these efforts though have improved RE and RS, but there persist perennial challenges regarding the impact of these efforts on the quality of software. Surveys and studies, experiences in projects, phase-end and end-of-project reviews and related metrics, and testing reports indicate that requirements continue to be major contributors to defects across all phases in SDLC and final software. [10] Poor quality RS causes the number of defects in each of the artifacts of subsequent phases of SDLC finally resulting in poor quality and unstable software. This results in user unhappiness, customer dissatisfaction, increased maintenance costs, increased cost of no quality, invite lawsuits, and failure to get repeat business. Studies performed at GTE, TRW, and IBM are stating that requirements errors creeping into the requirements phase are extremely expensive to repair. There is a multitude of cost increases in the subsequent phases as we progress in SDLC. These studies further brought to the notice that the major reasons and causes which are coming in the way of delivering quality software, on time and budget concluded that the top three reasons were lack of user input, incomplete requirements, and specifications, and changing requirements and specifications [10] and the causes for the failures in projects are inability to understand and capture requirements effectively, and weak engineering discipline in managing requirements.

In 1993, the first major international event on Requirements Engineering (International Symposium) was held where the software development fraternity came together to deliberate on improving the quality of requirements. Distilling this focus into compelling, enduring, and novel requirements engineering (RE) became a true challenge. Ever since there have been continuous improvements to bring in significant maturity in RE that are further strengthened by SEI-CMU's CMM framework which helped in achieving organizational maturity by institutionalizing Generic Goals (GG) and Generic Practices (GP), Specific Goals (SG) and Specific Practices (SP) in Process Areas (PAs) by continuously improving the capability of people, methods, and tools and technologies thereby improving software processes and also, monitoring through appropriate measurements. [11] Meanwhile, improved practices in Software Engineering evolved because of a shift of focus from Quality Control to Quality Assurance to Quality Engineering.

The ISO/IEC 9126 and the ISO/IEC 14598 made a fundamental difference in perceiving product quality and product evaluation. ISO/IEC 9126 standard defines a quality model for software product evaluation. ISO/IEC 14598 standard defines the process for software product evaluation. [16] These two standards majorly contributed to the creation of the series of standards, viz., ISO/IEC 25000, also known as SQuaRE (System and Software Quality Requirements and Evaluation) which has the goal of creating a framework for the development and evaluation of software product quality. It added a new dimension to the quality of systems and software by defining three kinds of quality of software products: quality in the use of a software product, external quality of a software product, and internal quality of a software product. It derives characteristics and sub-characteristics of a software product in terms of external quality and internal quality and integrates external quality and internal quality into software product quality. [17]

Attempts are made to reduce defects, by moving the representation of RS from textual to visual formats. The announcement, acceptance, and usage of the Unified Language Model (UML) played a very crucial role. The introduction of analysis and modeling like Use Cases, State Transition Diagrams, and User Stories accelerated the improvement. [44, 45, 46] Agile, lean, [47] and model-based approaches [48, 49] have added strength in visualizing and capturing requirements by involving customers, users, and stakeholders. The emergence of knowledge management and benchmarking have further strengthened software engineering practices including requirements engineering. The incorporation of design thinking [50], system thinking, induction and abductive thinking, group creativity, brainstorming, mind mapping, and SCAMPER have a very high impact on requirements engineering.
All these attempts resulted in defining requirements, characteristics of good requirements, RE process, classification of requirements, representational methods, and RS. However, defects of requirements and surrounding issues, problems, and causes of weak requirements engineering practices are yet to be addressed satisfactorily.

3. NEED FOR GOOD REQUIREMENTS ENGINEERING FRAMEWORK

Quality of software, SDLC and its phases, and related artifacts evolve based on the quality of requirements. So, there is a need to use RE practices to collect, corroborate, represent, and specify requirements and document requirements. However, the RS phase is continuing to be the weakest of all SDLC phases. Subsequent phases in SDLC depend and evolve based on RS and there will be cascading negative impact of weakness of requirements engineering and management practices on them. Hence there is a need for Requirements Engineering Framework (REF) that provides guidelines on how requirements should be defined which will play a very crucial role in arriving at quality RS.

To produce quality requirements many attempts were made in the past leading to significant improvements in RE and resulting RS. As the world is moving deep into pervasive computing, there is still a lot of scope for further improvement and also, there is a need to arrive at a more evolved and superior requirements engineering framework. This paper understands this need and proposes Requirements Engineering Framework to produce quality RS. The framework uses profiling, contextualization, and modeling concepts and also, quality engineering principles.
Irrespective of continuous improvement in requirements engineering (RE) practices to produce quality RS by using a process-centric CMM-based process maturity model or people-centric agile approach to engage stakeholders or using visual modeling techniques, many challenges persist. In addition to this, RE is not aligned with the present-day digital world with pervasive computing needs that are characterized by disruptive technologies, user-centricity, automation, and continuous deployment.

4. EFFORTS IN THE PAST TOWARDS IMPROVING REQUIREMENTS

A lot of effort has gone in the past towards addressing these needs, also, addressing reasons and causes that affect the quality of requirements through understanding and capturing requirements effectively, and to define effective RS discipline in managing requirements. These efforts focused on arriving at quality RS through the appropriate use of a systematic approach to RE by using REF.

These RE approaches either are process-centric based on frameworks like the CMM process maturity model [11] or are agile. These approaches though have advanced the practices of RE and RM have many limitations and issues that include- higher defect containment in phase-end artifacts and final software; not being aligned to the present-day digital world with pervasive

computing that is characterized by disruptive technologies, user-centricity, automation, and continuous deployment; failure to effectively understand and align to requirements environment; lacking in understanding user and product; and predominantly use analytical problem solving by using principles of functional decomposition and stepwise refinement. One can envisage many RE risks [12] and challenges [13, 14] in large-scale agile and scaled agile system development.

Surveys [15] and findings indicated that the major root causes for all these limitations and issues are mainly- weak requirements engineering and management practices, poor understanding of the requirements environment, lacking understanding of user requirements, changing requirements, and weak traceability.

RS is a formal document that represents and specifies conditions or capabilities needed by a user to solve a problem or to achieve an objective that must be met or possessed by a system or system component to satisfy the contract, standard, specification, or formally imposed documents. These requirements can be classified as - Requirements of Customers/Users that represent the structure of the system, and Requirements of Management that act as the driver of competitive advantage. These requirements shall be engineered and managed by using Requirements Engineering and Requirements Management processes [3]. Requirements are documented in the past predominantly in textual form. Nowadays, there is an increasing adoption of visual notational representation and also, enabling quality by having requirement representations in the form of different models language such as data, behavioral, or functional models [7, 8].

Irrespective of all these developments, at the core, requirements shall specify what are the goals of the system, what functionality the system shall provide, what data the system shall handle, how well the system shall perform (non-functional characteristics), what are the implementation and design constraints, what are the external systems with whom the system shall interface with, and what are the internal sub-systems and components the system under consideration shall integrate with. To accommodate all these, RE shall consider different viewpoints [5], consider the requirements environment, choose base models, use the RE process, and have requirements governance in place to produce an RS document.

Different viewpoints to be considered include requirements environment, base models, software development life cycle, and requirements governance.

The requirements Environment is an enabling factor that contains the infrastructure for requirements and comprises of organizational requirements environment and the product's technical environment. It defines an approach, framework, standards, and repository.

Base Models include representational methods, requirements basis artifacts or work products, and requirements models. These can be in the form of textual representation as requirements specification documents and/or diagrammatic representation like use cases, user stories, and state transition diagrams).

RE Process consists of five main tasks, namely, requirements extraction or elicitation, requirements analysis and design, requirements specification, requirements validation, and requirements management.

Requirements Governance focuses on the evaluation of requirements, configuration and change management of requirements, defect prevention and management, and requirements-related metrics [26].

The proposed REF attempts to deal with perennial software requirements-related problems and issues to address reasons and causes related to weak requirements, and also to control defects and develop quality software.

5. PROPOSED REQUIREMENTS ENGINEERING FRAMEWORK

Requirements shall possess characteristics and sub-characteristics of good requirements; align to the context in which software shall be made operational and requirements goals; fulfill user, customer, and stakeholder needs; and be specified and presented in such a way that different roles in the development team make the unambiguous meaning out of them. This helps software development teams and roles to carry out their work with ease and effective transformation with forward and backward traceability to arrive at inputs and quality phase-end artifacts of each phase and final software.

RE is a systematic iterative process of applying scientific principles, methods, and techniques for Identifying Needs (Elicitation); Analyzing and Prioritizing Needs (Analysis); Developing, communicating, and documenting the identified and analyzed needs (Specification); and Ensuring quality requirements (Quality Engineering). While doing so appropriate measures shall be taken to address key issues in RE and RM. RE that include opacity; volatility; requirements gold plating; informal representation and notation of requirements; and lack of knowledge in software engineers about requirements engineering notations, techniques, methods, and tools. RM issues include weak change and configuration management; higher defect containment in phase-end artifacts and final software; and weak traceability.

Our proposed RE framework (See Fig.1 Proposed Requirements Engineering Framework) understands all these issues and tries to address them. The proposed framework is at the macro level and in the future work shall be done to arrive at micro-level activities by defining required processes, tasks, and guidelines for implementing them. Also, piloting and refining shall be done.

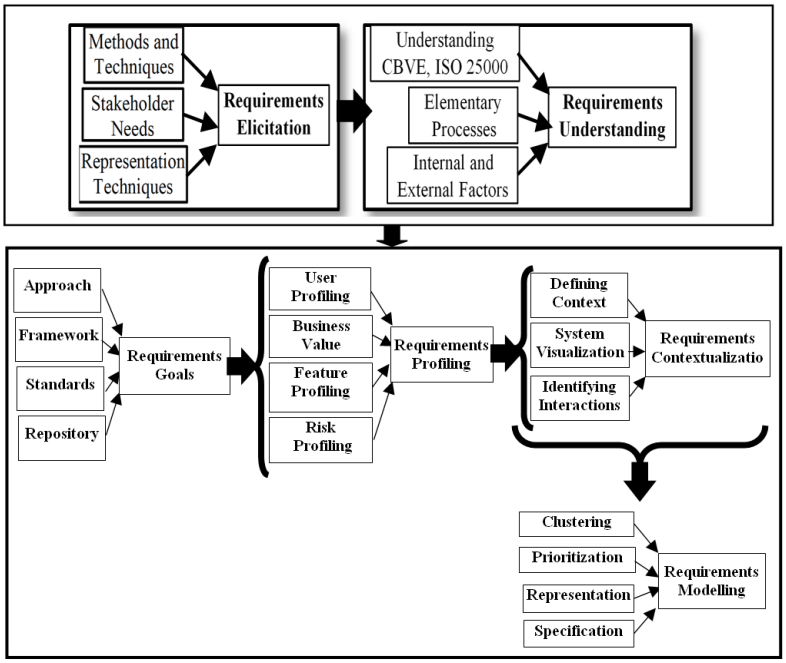

Fig.1 Proposed Requirements Engineering Framework

The proposed REF includes tasks, viz., identifying needs (Requirement Elicitation); understanding the captured needs (Requirement Understanding); defining Requirement Goals (RGs); devising requirements for a specific profile (Requirement Profiling); understanding, aligning, and operating in a particular context (Requirement Contextualization); constructing requirements models shall be developed to capture knowledge about the structure and also, behavior (Requirements Modelling). They become the basis for the remaining phases that include Transformation and Continuous Deployment. All these are carried out by using quality engineering principles and they together shall fulfill Requirements and Goals.

6. TASKS IN REF

In implementing REF, strategy shall be worked out towards mobilizing stakeholders, engaging stakeholders, and using Contextualize-Build-Validate-Evolve (CBVE) product element [18] to develop an adaptive requirement model is very important. In this endeavor, the practices of lean philosophy, design thinking, learning by experience (LXD), and quality engineering make a greater impact.

6.1. Requirements Elicitation

Requirements Elicitation identifies users and customers and selects appropriate requirement-related techniques as input; performs tasks that include capturing stakeholder mission, technical and non-technical needs/ features/wish list, corroborating requirements and related risk, choosing appropriate representation techniques of requirements, and automation needs. It produces the goals and corroborates requirements to arrive at a first draft of candidate requirements, and defines constraints and acceptance criteria with stakeholders.

Identifying stakeholders that include all types of users and customers is very crucial. Also, enough care shall be given in choosing appropriate requirement-capturing techniques like using a questionnaire, interviewing, procuring and studying existing documentation, forms, and databases; study and site visits; observation of the work environment; using techniques like brainstorming; storyboarding; use cases; and user stories.

As a next step, by using appropriate requirement-capturing techniques stakeholder engagement takes place, and requirements are captured from each of the stakeholders. The requirements so captured will be in the form of different documents and include requirements, related risks, and automation needs.

Then follows the task of corroborating requirements and related risks. Here all the requirement-related documents, risks, and automation needs are combined to arrive at the first draft of the mission, candidate technical and non-technical needs/ features/wish list, risks, automation, constraints, and acceptance criteria with stakeholders. Also, appropriate representation techniques for the requirements are finalized.

6.2. Requirements Understanding

Requirements Understanding involves splitting requirements into Elementary Processes (EP); defining characteristics and sub-characteristics of each EP using CBVE and ISO/IEC 25000 standards [16, 26]; grouping of requirements; analysis and prioritization; studying and considering internal and external environment factors that influence software development are done.

Requirements analysis and prioritization include: evaluating the quality of the requirements; prioritizing requirements; feasibility analysis; synthesis and validation of user requirements; determining quality attributes; and identifying risks.
Internal factors include sub-systems and components of a system, technology to be used, automation, build and release strategy, and related risks. External factors include knowledge about competing products, users, development platform, hardware and software the system shall integrate with, deployment platform, and related risks. These internal factors and external factors shall be aligned to generate value. In our future work, this will be undertaken.

6.3. Defining Requirements Goals (RG)

Defining RG involves many activities, namely, ideas shall be expressed, understanding macro-level intentions of the system from the perspectives of stakeholders shall be done, and crude information shall be refined to arrive at concrete requirements, and objectives of the system under consideration shall be specified. [19] As a result of this exercise, This exercise shall define the purpose of the proposed system; outline of external behavior; and mission needs of the customer and user. This is followed by decomposing them into precise, unambiguous requirements; and presenting them in an understandable way to relevant stakeholders. Thus, in defining RG, it is crucial to consider different dimensions, viz., RE Environment, RE Process, RE Quality, and RE Governance, and then state the Requirements and Goals.

According to Klaus Pohl, Günter Böckle, and Frank van der Linden [9] deriving goals shall be done first before arriving at RG. They note that in defining RG, it is important to consider factors that include domain, common domain requirements, and variability in domain requirements. After considering these factors domain requirements engineering shall be used to arrive at common domain requirements of a software product line in a specific domain.

6.3.1. RE Environment

The RE environment contains the infrastructure for RE. It comprises organizational RE enabling elements and consists of RE Approach, RE Framework, RE Standards, and RE Repository.

6.3.2. RE Process

RE process is implemented by mobilizing and engaging stakeholders and by using Contextualize-Build-Validate-Evolve (CBVE) product element. This exercise shall propose a viable solution by using the concepts of lean philosophy, design thinking, LXD, and quality engineering. As a result of this exercise, an adaptive model that represents the requirements

6.3.3. RE Quality

Quality output is determined by quality input. To ensure this, the RE process is defined to build capability and maturity by developing and using skilled and competent people, appropriate methods and techniques, and appropriate automation are ensured. Also, in-process and process-end reviews, peer reviews, and inspections are carried out to locate and remove defects by using quality engineering (QE) practices to produce quality RS.

6.3.4. RE Governance

RE governance tasks include requirements evaluation; traceability, configuration, and change management; defect correction, prevention, and management; and requirements metrics. By using these RE dimensions, in our REF, RGs shall be derived.

6.4. Requirements Profiling

Profiling is involving the selection and representation of relevant information related to a specific objective. To perform Requirements Profiling, it is required to understand the stakeholders' must-have requirements and good-to-have expectations and then derive Profiles of Users, Business Value, Features, and Risk and use them as input to define the needs of users, customers, and stakeholders in terms of characteristics and related sub-characteristics.

6.4.1. User Profiling

User Profiling involves identifying different types of users; and each user's operational conditions, mode of their operation, enabling environment, and usability profiling by understanding the users, each user's view, the specific environment under which the software product will be used, and the context of its use.

6.4.2. Business Value Profiling

Business Value Profiling involves understanding business viability, product benchmarking, competitive analysis, and value proposition.

6.4.3. Feature Profiling

A feature is an abstract requirement. It can be used to define an end-users visible characteristics of a system. [20] Feature models are abstract representations of the high-level requirements of architecture and are the basis for technical feasibility. Feature Profiling considers technical requirements and defines their Functional and Non-functional characteristics and sub-characteristics.

6.4.4. Risk Profiling

Risk Profiling is done for identified features. Here different types of risks associated with each feature are identified to develop strategies to address them. Risk Profiling of features involves creating a table by listing all features and then for each feature Identifying Risk Sources, Getting Risk Ideas, and Identify-Analyze-and-Prioritizing Risks. Identifying Risk Sources consists of studying, analyzing, and thinking about any possible entities or events which may have risks that would prevent software to behave as required and expected. Some Risk Sources are requirements related, to software and system that need to be developed, software/system-related operation and maintenance, and the environment under which software shall work. Risk Ideas consider product elements and project environmental factors to use them as the basis to identify risks. Product elements include structural, functional, temporal, data, platform, and operation related. Project environmental factors include information, customer, technology, tools and equipment, deliverables, team, logistics, budget, deliverables, and schedule. Identify-Analyze-and-Prioritize Risks include listing identified risks in the risk matrix table; calculating the Risk Exposure Factor (REF) of each risk by allocating value for probability of occurrence, timeliness of occurrence, impact of the risk, and multiplying all of them. Following this risks with higher REF are considered as priority ones. [21]

Requirements Profiling is being done by using Profiles of User, Business Value, Feature, and Risk as inputs. By using Requirements Profiling as the basis, context profiling shall be derived. This is done by retrieving contextual information, then performing a continuous evaluation of sensed/ inferred contextual information, and as a final step carrying out context adaptation through adaptation mechanisms. More details on this are available in [22].

6.5. Requirements Contextualization

Lehman's laws of software evolution [23] state that the system must be continually adapted else it becomes progressively less satisfactory in use. Hence need arises to maintain user satisfaction over the system's lifetime by continually increasing the functional capability of systems. This demands that the software that will be built shall be self-adaptive and context-aware to work in a particular environment; and shall possess self-adaptability to adjust, align, and respond to the context.

Contextualization involves the tasks, viz., defining the context, visualizing the system with Context-awareness, and understanding the ability to interact in the system.

6.5.1. Defining Context

The value of software to users and organizations arises from its actual behavior in a particular context. To achieve effective interaction between a user and an application the system shall be sensitive and responsive to the context [24] - any information that can be used to characterize the situation of an entity (i.e. whether a person, place, or object) that are considered relevant to the interaction. The software shall exhibit acceptable behavior in a particular context by adapting to that system so that it can be functional, reliable, efficient, usable, operable, relevant, maintainable, sustainable, and valuable.

6.5.2. Visualizing the System with Context-awareness

Visualizing the system requires the capability of context awareness which can be achieved through environment awareness, situation awareness, self-awareness, object awareness, and incident awareness. Advances in sensors, actuators, dynamic control, Artificial Intelligence, machine learning, and data science are helping in visualizing the system and practically implementing the same.

6.5.3. Understanding the Ability of Interactions in the System

Self-adaptive software are having the ability to sense the environment, understand the context, and build self-awareness and under these abilities sense and respond by modifying their behavior and/or structure. [25]

Sensing the environment can be implemented by using appropriate sensors to receive external signals about an object or an event, filtering and classifying these signals to reduce noise, and building logical clusters of signals.

Understanding the context can be achieved through awareness at different levels and building a facility of interactions in the system so that interactions between users and the system takes place to produce required functional and non-functional characteristics. The facility of interactions in the system requires different abilities like capturing explicit acts of communications from users than making communication effective by using the context to interpret, then having the ability to respond to these explicit acts, and then establishing interoperability dynamically by exhibiting co-existence, and then by bringing in collaboration among computers, content, communication, and connectivity.

6.6. Requirements Modelling

Models are abstract entities that are used as the basis to evolve software by carrying out model transformations to ensure software adaptation for both functional and non-functional goals. [27, 28, 29, 30]. In requirements modeling (RM), with the help of a process requirements are evolved by involving cross-functional and self-organizing teams to capture the exact needs of the stakeholders, represent them so that the software development team can understand them, and build the next phases in software development effectively.

In the proposed REF, a contextualization-based approach is used in RM to build represent requirements in the form of quality profile with the help of ISO 25000 [33, 34, 35, 36] software engineering standard. This approach helps in the development of quality software by considering different 'abilities' by considering different dimensions to respond to criteria like how effectively the software runs in different environments; how well functionalities are built; how easily it can be deployed on various hardware, operating systems, and configurations; how easy it is to use; how reliable it is; how efficiently it uses resources; how well it complies with external standards and conform with internal standards; and so on.

In this contextualization-based approach for RM, steps used to build a model of requirements include clustering, prioritization, representation, and specification. This helps us in developing the software with no ambiguity, increased clarity, superior understanding, and improved focus to get the best out of software development. [26]

6.6.1. Clustering

Clustering based on contextualization involves identifying relevant contexts under which the system shall exhibit its characteristics and sub-characteristics shall be identified and listed. Here, activities that will be carried out will include: building the list of contexts and defining applicable characteristics and sub-characteristics for each context based on the ISO 25010 standard that identified eight main characteristics of software quality, each with several sub-characteristics is used [33, 34, 35, 36], as shown in Table 1.

Table 1. ISO 25010 defined Eight Characteristics and Sub-characteristics

Characteristics	**Sub-characteristic**
Functional Suitability	Functional Completeness, Functional Correctness, Functional Appropriateness
Performance Efficiency	Time Behavior, Resource Utilization, Capacity
Compatibility	Co-existence, Interoperability
Usability	Appropriateness recognizability, Learnability, Operability, User error protection, User interface aesthetics, Accessibility
Reliability	Maturity, Availability, Fault Tolerance, Recoverability
Security	Confidentiality, Integrity, Non-repudiation, Accountability, Authenticity
Maintainability	Modularity, Reusability, Analysability, Modifiability, Testability
Portability	Adaptability, Installability, Replaceability

6.6.2. Prioritization

During prioritization, by involving stakeholders and by understanding their requirements and expectations from the system, consider each context and assign priorities to each sub-characteristic under each of the eight characteristics as High (H), Moderate (M), and Low (L).

This helps in understanding the priorities, and grouping contexts to be considered under each iteration both for development and testing.

6.6.3. Representation

Representation involves ordering and sequencing of contexts which will be provided for the development team for realization and testing team for verification and validation.

6.6.4. Specification

Finally, in the specification, context cases will be developed. Each context case will specify Context Case Id, Context Case Name, Description, Environment, Involved Sub-systems, Enablers, Inhibitors, Interactions and Outputs, Performance, and Observations.

The Context Cases so specified shall be understood by different roles in the development team. This understanding will help to capture the knowledge about the structural dimension and behavior dimension of the planned software. The structural dimension is also referred to as the static dimension. It involves 'what' elements constitute the system and its relationships. The behavior dimension which is also referred to as the dynamic dimension involves 'how' these systems interact to satisfy the requirements goals of the proposed software and provisions to obtain its functionality or behavior, and also non-functional characteristics, namely, functionality, reliability, efficiency, usability, portability, and maintainability. [16, 26] This can be achieved by building a model that captures syntactical knowledge and semantical knowledge, and represents it in an abstract form. This is achieved by embarking on continuously taking decisions with proper rationale on choosing relevant parts of a problem and solution, ignoring irrelevant parts of a problem and solution; and providing a diagrammatic representation of the solution by organizing knowledge by guidelines expressing idioms of usage, and using specific diagrammatic notations to depict knowledge (syntax) for communication.

Using models, developers, and others understand, interact with, configure, and modify the runtime behavior of software [31]. The software design community introduced feature models as an abstract representation to express the high-level requirements of architecture. In requirements engineering, there is a need to move down from high-level abstraction to detailing. This is done by using context cases that describe the system usage, interactions of customers and users with the system, and the responses of the system as a set of concrete sequences of actions. To convert the understanding so developed into solutions, solutions for each context case shall be proposed at service, functional, logical cluster, and implementation levels with different levels of abstraction for different types of stakeholders. Service level requires users with no functional and technical knowledge. The functional level requires functional-level domain knowledge. The logical Cluster level requires architects. And implementation level requires designers and programmers. Based on the level of solutions, the appropriate type of users shall be defined and shall be engaged in solutions.

In summary, the design and implementation of the adaptive system evolving out of these models are having the ability to adapt their behavior and have the in-built quality to withstand unpredictable, uncertain, and continuously changing execution environments [30]. These models will be very robust when context-awareness, adaptive features, and domain-specific modeling language are used while building these models that are dynamic and include [32] abstractions of runtime phenomena, automate runtime adaptation, analyze running software systems, controller with a specific technique [30], and feedback loops [32] Quality of models is achieved by having some characteristics like modularity, contextual alignment, clarity, completeness, unambiguity, and verifiability.

6.7. Quality Engineering

Verification ensures that we are doing things right through a process and confirms that the defined requirements specification work product is right. In this regard, there is a need to carry out RE by using a process. Also, there is a need to verify whether the RS could capture all necessary needs of users, customers, and stakeholders, and shall have quality attributes that are listed below.

- Complete: Requirements shall be captured in such a way that nothing is missing (no "To Be Determined"s)
- Consistent: Requirements shall be captured and documented in such a way that each requirement should not conflict with other requirements
- Correct: Requirements shall accurately state a customer's need
- Understandable: Requirements shall be captured and documented in such a way that every stakeholder shall get the same meaning when they read it.
- Unambiguous: Requirements shall have only one possible meaning
- Feasible: Requirements can be implemented within known constraints
- Modifiable: Requirements shall be easily changed, with history, when necessary
- Necessary: Requirements shall document something customers need
- Prioritized: Requirements shall be ranked as to the importance of inclusion in the product
- Verifiable: Requirements shall be documented so that they can be reviewed, modified (if necessary), and baselined.
- Testable: Requirements shall be documented in such a way that by using them tests can be devised to demonstrate correct implementation
- Traceable: Requirements shall be linked to system requirements, and designs, code, and tests

6.8. Requirements Governance

Governance to ensure appropriate requirements traceability, requirements evaluation, defect correction, prevention and management, defining configuration and change management, and requirements metrics.

7. CONCLUSION AND FUTURE WORK

Requirements Engineering and Frameworks have evolved and shown tremendous improvement since 1993. CMM, Agile, and Model-based software development well supported by processes, the creativity of people, and group thinking has taken practices of requirements engineering to the next level. However, there exists a gap between the speed at which technology is evolving and Requirements Engineering. Current practices require a facelift to align them to the technologies like sensors, actuators, mobile apps, smart applications, and self-adaptive systems. Also, there is a need to move away from 'zero defect' software product release criteria to continuous deployment with managing risks on the move with continuous release of new versions of software products with new features and mitigated risks. The current work is considering all these and proposed a novel requirement engineering framework which is the basis for more research work and also, implementation guidelines in the coming days.

ACKNOWLEDGMENTS

The authors would like to thank Dr. Joseph Varghese, Mathematics Department; Dr. Ashok Immanuel V, Head of Computer Science Department; Dr. Chandra J, Research Coordinator; Computer Science Department; Dr. Prabu P, Computer Science Department of Christ Deemed to be University, Bengaluru, Karnataka, India (www.christuniversity.in) for their timely advise, support, and guidance.

REFERENCES

[1] H. H. Khan, T. Society of Digital Information, and W. Communication, "Factors generating risks during requirements engineering process in global software development environment," Int. J. Digit. Inf. Wireless Commun., Vol. 4, No. 1, pp. 63-78, 2014.

[2] [IEEE-830] Institute of Electrical and Electronic Engineers, IEEE Recommended Practice for Software Requirements Specification, IEEE Std 830-1998, Institute of Electrical and Electronic Engineers, New York, 1998.

[3] Pandey D, Suman U, Ramani A.K.. "An Effective Requirement Engineering Process Model for Software Development and Requirements Management", Intl. Conf. on Advances in Recent Technologies in Communication and Computing.

[4] P. Jalote, An Integrated Approach to Software Engineering, 3rd edition, Narosa Publishing house, India, 2005.

[5] G. Kotonya and I. Sommerville, "Viewpoints for requirements definition," Software Engineering. J., Vol. 7, No. 6, pp. 375-387, 1992.

[6] Mohommad Azeem Akbar et. al, "A systematic study to improve requirements engineering process in the domain of global software development,' IEEE Access, March 2020

[7] A.M. Davis; Software Requirements: Objects, Functions, & States, 2nd edition, Prentice Hall, Englewood Cliffs, New Jersey, 1993.

[8] R.J. Wieringa; Requirements Engineering: Frameworks for Understanding, Wiley, New York, 1996.

[9] Klaus Pohl, Günter Böckle, Frank van der Linden, Software Product Line Engineering: Foundations, Principles, and Techniques, Springer-Verlag Berlin Heidelberg 2005

[10] S. Arun Kumar, T. Arunkumar, "Study the impact of requirements management characteristics in global software development project: An Ontology-based Approach", International Journal of Software Engineering & Applications (IJSEA), Vol. 2. No. 4. Oct 2011

[11] CMMI Product Team. CMMI for Development, Version 1.3. CMU/SEI-2010-TR-033. Software Engineering Institute, Carnegie Mellon University. 2010. http://resources.sei.cmu.edu/library/asset-view.cfm?AssetID=9661.

[12] Ramesh B., Cao L., Baskerville R. Agile requirements engineering practices and challenges: an empirical study, Inf. Syst. J., 20 (5) (2010), pp. 449-480, 10.1111/j.1365-2575.2007.00259.x

[13] Rashidah Kasauli, Eric Knauss, Jennifer Horkoff, Grischa Liebel, Francisco Gomes de Oliveira Neto, Requirements engineering challenges and practices in large-scale agile system development, Journal of Systems and Software, Volume 172,2021,110851, ISSN 0164-1212, https://doi.org/10.1016/j.jss.2020.110851.

[14] (https://www.sciencedirect.com/science/article/pii/S0164121220302417)

[15] Inayat I., Salim S.S., Marczak S., Daneva M., Shamshirband S. A systematic literature review on agile requirements engineering practices and challenges, Comput. Hum. Behav., 51 (2015), pp. 915-929, 10.1016/j.chb.2014.10.046

[16] Clancy, T. (1995). The Standish group report. Chaos report.

[17] The ISO/IEC 25000 series of standards, https://iso25000.com/index.php/en/iso-25000-standards

[18] Koh, S., & Whang, J. (2016). A critical review on ISO/IEC 25000 square model. In Proceedings of the 15th International Conference on IT Applications and Management: Mobility, Culture, and Tourism in the Digitalized World,(ITAM15) (pp. 42-52).

[19] Khannur, A. "Context Driven Software Development." International Conference On Systems Engineering. Springer, Cham, 2020.

[20] Van Lamsweerde; "Goal-Oriented Requirements Engineering: A Guided Tour", Proceedings of the 5th International Symposium on Requirements Engineering, Toronto, August 2001, pp.24 9–263.

[21] K. Kang, S. Cohen, J.A. Hess, W.E. Novak, and S.A. Peterson; Feature-Oriented Domain Analysis (FODA) Feasibility Study, Technical Report, Software Engineering Institute, Carnegie–Mellon University, 1990.
[22] Ronald P. Higuera and Yacov Y. Haimes, Technical Report - Software Risk Management, CMU/SEI-96-TR-012, ESC-TR-96-012, June 1996
[23] Sindico, A., Vincenzo Grassi, V.: Model Driven Development of Context-Aware Software Systems, COP '09: International Workshop on Context-Oriented Programming, July 2009 Article No.: 7, Pages 1–5 (2009)
[24] Lehman, M. M., L. A. Belady L.A.: Program Evolution: Processes of Software Change. USA: Academic Press Professional, Inc. (1985)
[25] Dey, A. K., Abbowd, G. D.: Towards a Better Understanding of Context-Awareness, Proceedings of the Workshop on the What, Who, Where and How of Context-Awareness, CHI2000 Conf. on Human Factors in Computer System, New York, NY: ACM Press (2000)
[26] Javier Cáma, Rogério de Lemos, Carlo Ghezzi, Antonia Lopes.: Assurances for Self-Adaptive Systems: Principles, Models, and Techniques. January 16, 2013, Springer (2013)
[27] Khannur, Arunkumar. Structured Software Testing: The Discipline of Discovering. Partridge Publishing India, 2014.
[28] Mukerji, J., Miller, J.: Model Driven Architecture,
[29] http://www.omg.org/cgibin/doc?ormsc/2001-07-01, July 2001 (2001)
[30] Atkinson. C., Kühne, T.: Model-Driven Development: A Metamodelling Foundation," IEEE Software, Vol. 20, No. 5, pp. 36-41,2003 (2003)
[31] Frankel D., "Model Driven Architecture: Applying MDA to Enterprise Computing," OMG Press, 2003 (2003)
[32] Filieri, A., Maggio, M.: Control Strategies for Self-Adaptive Software Systems, ACM Transactions on Autonomous and Adaptive Systems, Feb. 2017. (2017)
[33] Chulani, S., Williams, C., Yaeli, A.: Software Development Governance and its Concerns, SDG'08, ACM, 2008. (2008)
[34] Vogel, T.: Model-Driven Engineering of Self-Adaptive Software, UCT CS Colloquium, South Africa, 2015 (2015)
[35] https://iso25000.com/index.php/en/iso-25000-standards/iso-25010
[36] ISO/IEC 25010:2011 Systems and software engineering -- Systems and software Quality Requirements and Evaluation (SQuaRE) -- System and software quality models.
[37] Boehm, B. W. (2006). A view of 20th and 21st-century software engineering. Annals of software engineering, 1(1-4), 1-24.
[38] Pressman, R. S. (2014). Software engineering: a practitioner's approach. McGraw-Hill Education.
[39] Report on NATO Software Engineering Conference (1968), Garmisch, Germany, 7th to 11th October 1968. Editors: Peter Naur and Brian Randell
[40] Bennington, Herbert D. (1 October 1983). "Production of Large Computer Programs" (PDF). IEEE Annals of the History of Computing. IEEE Educational Activities Department. 5 (4): 350–361.
[41] Hans-Petter Halvorsen, Software Development A Practical Approach, ISBN: 978-82-691106-0-9 Publisher Identifier: 978-82-691106, 2020.
[42] "7 Steps of Agile System Analysis Process." http://www.targetprocess.com/blog/2006/06/7-steps-of-agile-system-analysis.html. N.p., n.d. Web. 12 Nov. 2013.
[43] Jabbari, Ramtin; bin Ali, Nauman; Petersen, Kai; Tanveer, Binish (May 2016). "What is DevOps?: A Systematic Mapping Study on Definitions and Practices". Proceedings of the 2016 Scientific Workshop. Association for Computing Machinery.
[44] Booch, G., Jacobson, I., Rumbaugh, J. 2004. The Unified Modeling Language Reference Manual, second edition. Addison-Wesley Professional.
[45] Cockburn, A. 2001. Writing Effective Use Cases. Addison-Wesley Professional.Google Scholar
[46] Cohn, M. 2004. User Stories Applied. Addison-Wesley Professional.
[47] Kupiainen, E., Mäntylä, M. V., & Itkonen, J. (2015). Using Metrics in Agile and Lean Software Development – A systematic literature review of industrial studies. Information and Software Technology, 62, 143-163. https://doi.org/10.1016/j.infsof.2015.02.005
[48] Madni, A. M., & Sievers, M. (2018). Model-based systems engineering: Motivation, current status, and research opportunities. Systems Engineering, 21(3), 172-190. https://doi.org/10.1002/sys.21438

[49] Pereira, J. C., & Russo, R. D. F. (2018). Design Thinking Integrated in Agile Software Development: A Systematic Literature Review. Procedia Computer Science, 138, 775-782. https://doi.org/10.1016/j.procs.2018.10.101

AUTHORS

Arunkumar Khannur is working as Chief Strategy Officer and Professor of Practice at CMR University, Bengaluru. He is having MSc, and MTech. to his credit. At present, he is pursuing his studies as Ph.D. Scholar in Computer Science Department, Christ Deemed to be University, Bengaluru. (www.christuniversity.in).

Dr. Manjunatha Hiremath is a Researcher, Guide, Mentor, and Assistant Professor in Computer Science Department, at Christ, Deemed to be University, Bengaluru (www.christuniversity.in). He is having MSc, MPhil, and PhD. to his credit. His areas of focus are Computer Vision, Image and Video Processing, Pattern Recognition, and Biometrics.