IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received April 17, 2022, accepted May 19, 2022, date of publication May 23, 2022, date of current version June 1, 2022.

Digital Object Identifier 10.1109/ACCESS.2022.3177659

Integration of Software Architecture in Requirements Elicitation for Rapid Software Development

MARYAM GILLANI[1], HAFIZ ADNAN NIAZ[1], AND ATA ULLAH[2]
[1]School of Computer Science, University College Dublin (UCD), Dublin 4, D04 V1W8 Ireland
[2]Department of Computer Science, National University of Modern Languages (NUML), Islamabad 44000, Pakistan

Corresponding author: Hafiz Adnan Niaz (hafiz.niaz@ucdconnect.ie)

This work was supported by University College Dublin, Ireland.

ABSTRACT Software Architecture describes system components and their connections. Requirement elicitation catering the perspective of software architecture is quite challenging and relatively less explored research area for the rapid software development. It has gain growing interest due to reusability of existing modules with less cost and quick developmental time. Software architecture in the context of requirement engineering is an abstraction of software system performing a particular task with the help of group of executable architectural components. In this paper, systematic literature review is adapted as a methodology to explore software architectural elements that provides better performance and simplicity in requirement engineering. We analyzed, reviewed and listed the strategies, tools & techniques along with state-of-the-art mechanisms, pros and cons and application areas. Architectural components that are already implemented in the requirement elicitation process for effective software architectural design are briefly analyzed. Purpose of the paper is to explore and discuss the elements that make software architecture more integral and flexible for traceability of requirements. Another purpose is to identify relation between the software requirements and architecture along with exploring the components to bridge gap between requirements and architecture by critically evaluating industrially and academically proposed methods, tools and frameworks. We also highlighted the open research challenges of Software architecture in requirement elicitation for better software development. In the later section, a resource bank is created acting as a valuable model that encompasses targeted relevant groups, sub-groups with latest software architecture tools & techniques, methods and framework sources to facilitate effective requirement engineering.

INDEX TERMS Software requirement engineering, software architecture, software development, software engineering, requirements elicitation.

I. INTRODUCTION

Software requirements outline the purpose of the development and design. It serves as the foundation of software intended to develop [1]. Requirements are defined in the beginning and act as a developmental milestones to accomplish successful executable software components [2]. Software requirement engineering is a systematic approach that is significantly developed in the course of the most recent decade [3]–[5]. Software architecture can be viewed as an organization of a system that comprehensively includes components interactions, operational environments, design principles, software functionalities, and often covers future evolutionary software perspective [6]–[9].

The associate editor coordinating the review of this manuscript and approving it for publication was Claudia Raibulet.

Software architecture differs from software design. Software architecture specifically targets structure of a system while software design merely deals with implementation details of the system (e.g. code level design) [10]. On the other hand, software architecture and design are different from software system architecture as it deals with the placements of software components [11]. Contrary to these three terms, ''Architecture Centric Requirement Elicitation'' (ACRE) covers architectural perspective of software-intensive components based on requirement engineering processes, practices, modelling, and designing in order to automate the requirement elicitation process [12].

We have considered ''Architecture Centric Requirement Elicitation'' (ACRE) as a central point of discussion in order to categorically analyse and critically investigate its impression on requirements engineering process for rapid software

development. ACRE can be considered as a broader term that establishes the convenient path to bridge architectural constraints with requirement engineering or requirement elicitation more precisely [13]. In simpler context, ACRE ensures and maintains the balance between software architecture and requirement elicitation operations and functionalities to facilitate rapid software development [14]. It additionally reviews the connections among different software components established during requirement elicitation processes.

Difference among software design, software architecture, software requirements engineering and ACRE is illustrated with detailed process flow in Figure 1. Requirements are subjected to prompt the architecture and facilitate the process of elicitation within scope, structure and context [15]. Accomplishing requirement specification from architectural models can give the requirements engineers a head start even before utilizing customary methods like workshops and situation based elicitation [16]. ACRE can be time and cost effective with better and quick software developmental cycles. It can precisely provide smooth and progressive flow of events with better accountability and traceability factors [17], [18].

In the light of the existing state of requirement engineering technicalities, the goal of this Systematic Literature Review (SLR) is to distinguish the most recent researches where ACRE approach has been utilized for the advancement of software development. The distinguished researches are grouped according to our research questions defined below;

RQ1: How software architecture and requirements elicitation are connected to each other? What are their state-of-the-art tools & techniques, pros and cons, operational applicability, current challenges and domain analysis?

RQ2: Is it possible to accommodate requirements in software architecture based on recent and latest software architecture frameworks, tools and techniques?

RQ3: What are research, industrial and academic gaps between software architecture and requirements based on current architectural trends and adopted measures for rapid software development?

RQ4: How to bridge the research gaps using Architecture Centric Requirement Elicitation approach for critically analyzing the balance between software architecture and requirement elicitation operations and functionalities?

RQ5: Interpreting the requirement elicitation processes in the context of their tools, techniques and relativity with architecture centric approaches.

The above mentioned Research Questions (RQs) are discussed in detail in the later sections of the paper where primary research contributions of this SLR are as follows;

1) A Systematic Literature Review (SLR) is performed for thorough and in-depth examination of the recent schemes, tools and platforms application on software requirement, software architecture and architecture centric requirement elicitation. We have identify various gaps for further research in this area.
2) We highlighted 60+ useful tools, schemes and patterns to justify proposed research questions. Relevant categorical analysis, preferable application environment, constraints, strengths and weaknesses to better guide researchers about industry practices, academic lapses and loopholes among components/modules of rapid software developments are highlighted.
3) A comprehensive Dendrogram is designed to underline research gaps and challenges for architecture centric requirements elicitation with appropriate discussion in order to grasp attention of researchers for real-time hindrances. Therefore, this opens the way of further investigations to resolve the identified problems.
4) Above mentioned RQs served as a motivation for this research in investigating the architectural concerns of requirement engineering for rapid value-oriented software development. A considerable gap between requirement engineering, software design and architecture is reduced keeping in view of the architecture of the system.
5) Above all, a Taxonomical Resource Bank is featured to group together recent and peer-reviewed valuable sources under proposed RQs with the classification of Groups, Sub-Groups and Functionalities to delve deeper. This resource bank is an efficient yet quick model to reach targeted category of concern under scope of study while saving time and eradicating the need of extensive resource engines surfing.
6) To the best of our knowledge, this is the first preliminary study that collectively investigate software design, software architecture, software requirement engineering and software requirement elicitation and architecturally centric requirement elicitation all together. Secondly, this is first perspective cohort study presenting association and interconnections among above mentioned fields through relevant tools, techniques and frameworks summative evaluation contemporaneously.

Rest of the paper is organized as: Section II discusses the research methodology for SLR where each sub task is also explored. Section III gives answer to RQ1, Section IV and V to RQ2 and RQ3 with detailed Dendrogram, Section VI covers RQ4 and Section VII analytically discusses RQ5 with detailed categorical Table of relevant tools and techniques followed by a resource bank model. Each section is supported through various sub-headings and tables. Section VIII concludes the paper and Section IX discovers the possible future work.

II. RESEARCH METHODOLOGY

We adopted the research method proposed by Petersen *et al.* [19] and used the suggested template for describing SLR approach. SLR is defined as a type of reviewing technique that facilitates repeatable analytical methods in order to critically appraise research in order to answer set of clearly formulated questions [20].

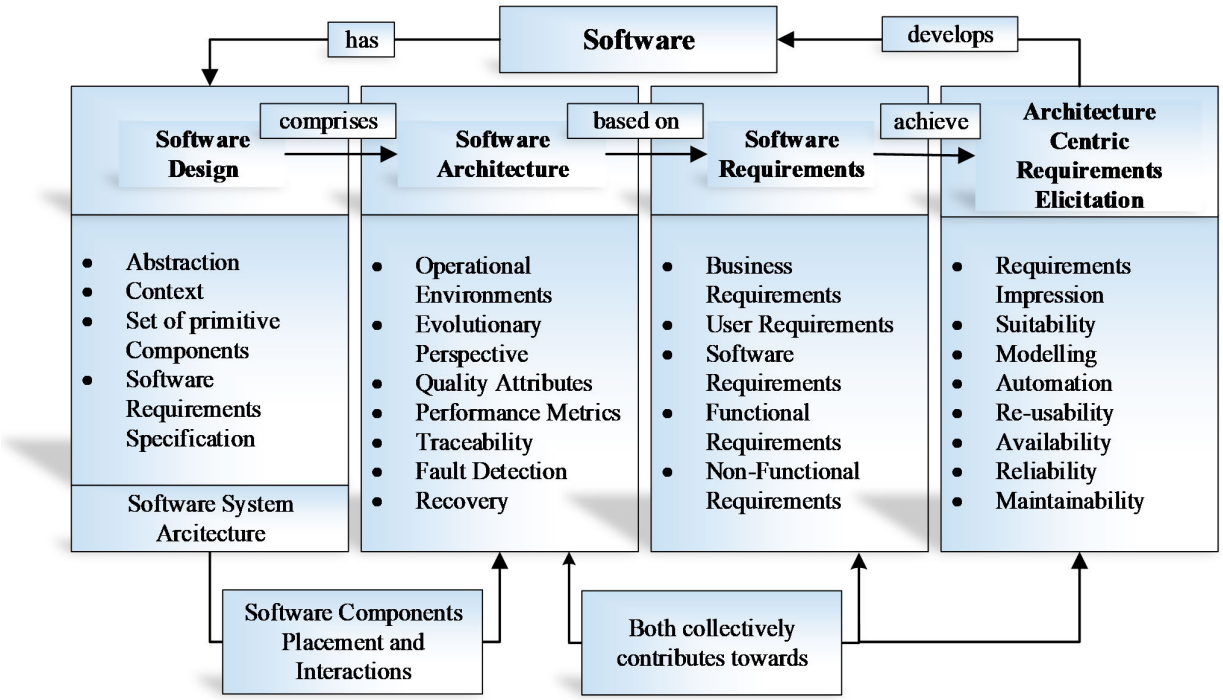


FIGURE 1. Search process based on inclusion-exclusion criteria.

A. REVIEW PROTOCOL DEVELOPMENT

We have developed a review research protocol by following the standards of SLR proposed in [19]. As per the standards of the review protocol, the search process is comprised of the following stages: 1) Development of Review Protocol, 2) Criteria for Selection and Rejection 3) Search Strategy, and 4) Data Extraction and Synthesis. The base questions have already been defined in the Introduction section. The remaining content is defined in the sub-sequent sections.

B. SELECTION AND REJECTION CRITERIA

A specific selection and rejection criterion is defined for the research process. This criteria in different categories is described as follows;

1) SUBJECT RELEVANCE

The research papers that are relevant to the subject are selected for the proposed study of interest.

2) LATEST SCHEMES

The research papers carrying the tools and techniques from 2009 till today are only included to satisfy recent area of study.

3) PUBLISHERS

The publishers of the referenced/ used papers are IEEE, ELSEVIER, SPRINGER, ACM and Taylor & Francis (peer-reviewed journals). No other papers from different engines are considered for highlighting Tools, schemes and patterns.

4) CRUCIAL EFFECTS

The selected papers must have the positive effect on the area of research and study. The papers selected for the topic ACRE must be affecting it productively. Papers that are not affecting the study are excluded.

5) RESULT ORIENTED

The selected papers should have properly concluded results via case studies so that those results would help in concluding better results.

6) REPETITION

All the papers having the similar content, is not included in order to avoid time wastage and knowledge replication.

7) INCLUSION AND EXCLUSION CRITERIA

are applied in order to find relevance among selected articles and ACRE. This process is completed by considering the title relevance filter and abstract study of the selected papers. We have considered various query strings, i.e. ''Architecture centric requirements'', ''tools for architectural requirements'', ''requirements impact on architecture'', ''requirement gathering tools'', ''requirements related to architecture'', ''architecture and requirement specs'', architecture and requirement relatedness'', ''architecture Significant requirements''. All other papers that were found mismatched as a result of the query words are excluded. AND/OR operations are applied to keywords. AND/J is for

AND/Journal. Detailed summary is given in Table 1 that describes the total count in response to query words.

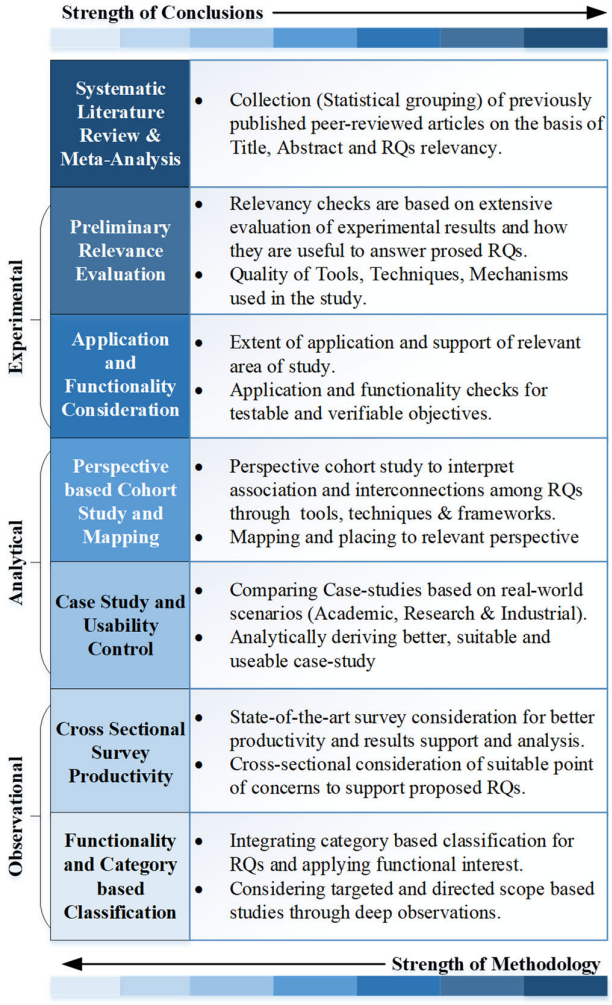


FIGURE 2. Data collection evidence about studies and Conclusions of research methodology.

In Figure 2, data collection evidence about selected studies and the conclusions of research methodology is highlighted. It shows, experimental, analytical and observational parameters based on 7 categories with suitable explanation to justify the methodological phases. Each of these mentioned categories assisted in selecting the quality oriented studies to answer proposed research question through systematic pattern. It starts from systematic literature review and Meta analysis and proceed towards narrower category i.e. functionality and category based classification. Key indicates that conclusions are drawn from bottom up approach and Methodology is refined and strengthen via Top to bottom approach.

C. SEARCH PROCESS

After the selection and rejection process, total of 61 Tools, Techniques, Schemes, and Patterns are selected. Figure 3 illustrates the search process with the number of articles that are rejected on the basis of title, abstract, general study and detailed study. It also presents the number of selected scheme after performing inclusion/exclusion criteria as per specifications of SLR. As shown in Figure 3, IEEE, Elsevier, ACM and Taylor & Francis provided plenty of research material in reference to our query words.

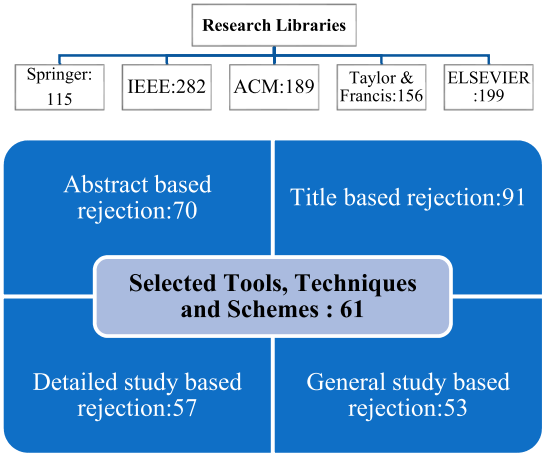


FIGURE 3. Search process based on inclusion-exclusion criteria.

Since all research articles could not have reviewed manually, so short-listing is based on title and abstracts. After a short-listing, a detailed study of the paper is done in order to get a detailed analysis for each research question. Summary of each paper was made as per relevance to the questions that helped in compiling the final results. A selected number of papers, i.e. 61 indicate those specific papers that are presenting Tools/Techniques with reference to our research title. However, many other research articles are utilised and used to make our proposed research valuable.

D. DATA EXTRACTION, SYNTHESIS AND QUALITY EVALUATION

As per Table 2, Title, year, concept, publisher, research Problem, proposed solution, contribution, and future works for extracting related schemes with detailed analysis are considered in data extraction, synthesis, and quality evaluation. During data synthesis and quality evaluation following conditions are met: i) identification for architecture-centric techniques, ii) identification of tools for requirements and architecture mapping, iii) identifying research gaps, iv) identifying research trends related to requirement gathering.

E. EVALUATION CRITERION OF THE QUALITY OF SELECTED STUDIES

Evaluation criterion of the quality of selected studies is given in Figure 4. Selected studies are identified, interpreted and than evaluated based on degree of relevance to draw suitable data. Outer loops complete performance analysis. Another inner loop evaluates research questions being answered in a selected studies based on adapted methodology's strengths and data contribution to desired context in order to ensure quality of selected studies.

TABLE 1. Summary of total count.

Sr. #	Search term	IEEE		Elsevier		ACM		Taylor & Francis	
		Count	Filter	Count	Filter	Count	Filter	Count	Filter
1	Architecture centric requirements	366	OR	77	OR	274	AND	78,655	OR
2	Tools for Architectural Requirements	135	OR	117	OR	438	AND	19,327	OR
3	Requirements impact on architecture	662	OR	525	OR	103	AND	56,524	OR
4	Architecture based requirements	7,785	OR	541	OR	609	AND	108,223	OR
5	Requirement gathering tools	109	OR	37	OR	366	AND	119,864	OR
6	Requirements related to architecture	9	OR	26	OR	43	AND	23,172	OR
7	Architecture and requirement specs	1005	OR	86	OR	293	AND/J	35,581	OR
8	Architecture and requirement relatedness	184	OR	5	OR	2	OR	87,514	OR
9	Architecture Significant requirements	933	OR	385	OR	161	AND/ J	85,486	OR

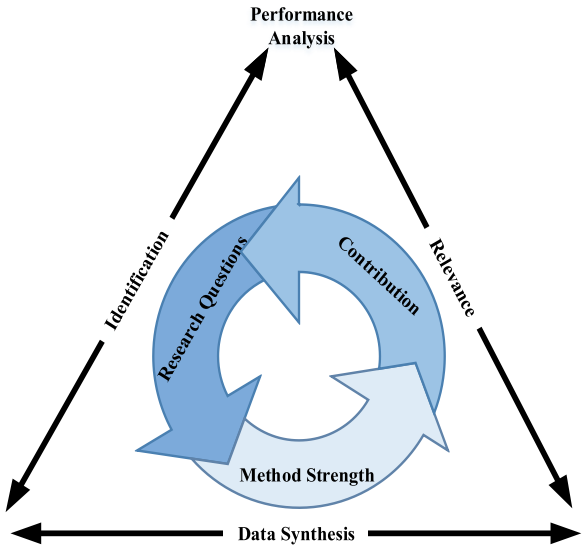


FIGURE 4. Evaluation criterion of the quality of selected studies.

TABLE 2. Parameters for data extraction and synthesis.

Category	Description
Data Extraction	
Title	Paper title relevance with research questions.
Year	Year of publication from 2009-2022
Concept	Concept synchronisation with intended research questions.
Publisher	Peer-reviewed and mature conferences
Research Problem	Problem statements synchronisation.
Proposed Solution	Related proposed solution to the problem addressed.
Contribution	Main contribution of the paper in the field of research.
Future Works	Holes or un-discovered solution that will be proposed by the researchers
Data Synthesis	
Identifying the architecture-centric techniques	Techniques that are used in software development, from requirement gathering to the construction of software
Identify the tools used for requirements and architecture mapping	Tools available to manipulate the process of mapping between requirements and architecture
Identifying the gaps in the related research	Identify the gaps and problems in the recent research that needs to be clarified
Identifying the research trends for requirement gathering	What are the recent trends, styles, and approaches for requirement elicitation?

III. RELEVANCE OF ARCHITECTURE AND REQUIREMENTS (RQ1)

This section covers relevance of software architecture and requirements elicitation on the basis of various state-of-the-art tools & techniques, their pros and cons, operational applicability, future setbacks and domain challenges as follows.

A. ARCHITECTURE TYPES & SIGNIFICANCE

Service-Oriented Architecture (SOA) is defined as a model with organized capabilities that can be used for providing solutions for defined domain of interest [21]. It is partially related to information system architecture [22]. Through SOA, knowledge sharing and information reusing, the relationship among requirements and architecture can be visualised conveniently [23]. SOA is achieved through consistent R&D (Research and development). R&D provides a better model-based platform for defining a relationship between architecture and requirements. WSDL (Web Description Services Language) is used for web-based requirements [24].

Knowledge-Sharing Architecture (KSA) complements relationship among strategies, results and theoretical and empirical contributions to deduce the better working collaboration of knowledge and design architecture [25]. KSA digs down deeper to provide better affiliation between requirements, design and architectural connections.([26]. KSA's strategic efforts helps to attain underlying objectives driven by qualitative and quantitative measures. Requirements elicitation on the basis of both theoretical and empirical facilitates inter-components/modules relationship formation with testable and uncovered association. KSA help in learning about alliance environment. On the other hand, SOA application is better operational at abstract level. It helps in shedding light on practical aspects requirements, specification, requirement validation and requirement management [27].

B. IMPACT OF REQUIREMENTS DESIGN ON SOFTWARE ARCHITECTURE

Diversity poses great challenge to software requirement engineers in identifying actual interaction of each segments,

suitable elicitation techniques and RE process capable of handling complexities of software architecture [28]. Requirements design influences directly in terms of meeting necessary conditions of project success [29]. The high interrelation between requirements design and software architecture for software artefacts reduces complexity in system design and leads to subjective decisions judgment. Controlled architecture intervention not only facilitates better adaptability, but also guides in smooth requirements elicitation phases.

Tools and technological software modules that are product of requirements designed from systematic software architecture demonstrates feasibility, better lifecycle and less developmental time [30]. Methods and approaches of architecture-centric design are targeted through New Product Development (NPD). It emphasises client-services centred architecture through component and testing based repetitive interpretative cycles [31].

High-level knowledge gained through architecture centered system designs establishes testable software components comprised of plug and play principle. Moreover, this factor also encourages reusability element for related state of software events and components. These are mostly repetitive cycles that holds traceable bonding among architecture and requirements [32]. Less repetition and iterative approaches are best suited for defining architecture and requirements where complexity of system is declared average overall [33].

C. ARCHITECTURAL MODELLING WITH BUSINESS GOALS & TOGA

Architecture modelling stands for reusable generic framework for generating executable software platforms in the context of background, specification, and classification demands for accomplishing specified business goal [34]. A very desirable factor in business goals especially in software market place is the ability of designed components to be applicable (good fit) for future business projects (clients). This factor is well-achievable through software architectural integration [28]. In the context of software architecture modelling, a series of models can be established based on categories of enterprise domain that may come handy for future developments thus reducing time, complexity, and resource allocation [35].

Explicit modelling of abstract requirements makes bridging of requirements components positioned rightly [22]. These requirements elaborate on association with architecture. Enterprise architecture rebuilds the modelling technique on the basis of business, application and technology layers [36]. The Open Group Architecture Framework (TOGAF's) is a proactive best practicing framework in developing enterprise architecture for any organization [37]. Architecture Development Method (ADM) is requirements centric management applied to enterprise architecture for multi-stage cycles [31]. ADM defines architecture and requirements relativity during architecture construction and modelling while giving practical benefits [38], [39].

D. SIMILARITY COMPUTATIONS AND ARCHITECTURAL INTEGRATION

Content Based Requirements (CBR) are based on similar requirements that can be used as proxies to retrieve similar software [40]. However, when a new contextual requirement is proposed by a stakeholder, CBR gets exploited in terms of previous affiliations [41]. This factor can be fixed through integrating requirements similarity, software similarity, and software architectural components. These three components are collectively called as similarity computation [42]. This similarity computation not only triggers three-tier application process verification to apply CBR but also integrates tracing and categorization.

Universal and Inclusive design (UID) uses a design approach for including and excluding requirements in locating appropriate/inappropriate designs for the user-centred process [43]. Good requirement design enables better help in communication design for the requirement elicitation process in design management [32]. During UID, two clusters of requirements are generated, i.e. user-centred requirements and process-centred requirements. These requirements are then utilised for establishing requirements and architecture connections together.

E. MULTI-MODEL SOFTWARE ARCHITECTURE APPROACH

Multi-model is an ability of architecture to accommodate multi-perspective (platforms) to assure expected results. However, multi-model catering multi-dimensions are tedious to deign, maintain, and incorporate. Their complex nature holds great potential in assisting gross level software development. Physical, virtual prototyping, targeted requirement models, design models, visual flow models, specifications, integration of architecture, multi-model requirements and architecture bridging all are the factors that are considered to design it [44], [32]. Designing process seems to be complex, but once generated can do wonders in term if it's greater flexibility and multi-information fusion [45]. This concept is newer in rapid software development and holds great potential for further advancement to join architecture and requirements relativity [46]–[49].

CAD tool allows the incorporation of multi-exchange models for architecturally defined requirements. Static translation and dynamic integration requires domain analysis for requirement elicitation and optimisation [33]. Different levels of details and integration illustrate the multi-exchange method by the degree of concurrence [23]. The degree of concurrence further defines specialised tasks to effectively elaborate requirement construction along with analysis and designing of requirement specifying process [22].

CAD tools are efficient in creating 3D visualisation of requirements and architecture relatedness. Simulation tools unlock various deadlocks for user-centred simulation [47]. Capability loss simulation and simulation tool kit are specific in re-defining and improvising conceptual approach that is given as empathic model [22]. Stimulation is the user-centred

design tool for architecturally significant requirements to make end-user requirements involvement possible to bridge and design requirements. The simulation tools enable requirements based on capability-centred product interaction to illustrate aarchitecture and requirements relatedness [50].

IV. ACCOMMODATING REQUIREMENTS INTO SOFTWARE ARCHITECTURE (RQ2)

This Section verifies the possibility of accommodating requirements in architecture with discussion on application and operational extent, traceability elements and characteristics of rapid software development integrating architectural perspective. This sections also discusses the limitations of ACRs and factors to overcome it to avoid future obstacles.

A. CHARACTERIZATION OF ARCHITECTURALLY SIGNIFICANT REQUIREMENTS

Characterisation of architecturally significant requirements has a great impact on the rapid software development lifecycle. Integration of architecture requirements is accomplished through identifying requirements based on their impact and critical factor [51]. Architecturally significant requirements are also identified on the basis of quality attributes, constraints, or application environment. Mapping of requirements in software design is attained through maintaining consistency between architecture and requirements. Architecture based requirements are written systematically in a document illustrating the requirements in the form of different architectural views to make it worth considering for future references.

The requirements and concerns of different stakeholders are captured in the 4+1 view model of architecture [52]. The stakeholders whose requirements are captured in the views could be architects, project managers, system engineers, developers etc. Tools and techniques available identify requirements that have an impact on the architecture automatically. They also classify requirements and recommend alternative solutions to adjust requirements in the architecture in the best possible way. The proposed architecture evaluation and application check is based on the Architecture Trade-off Analysis Method (ATAM). ATAM is scenario based technique that covers in-depth quality attributes analysis while prioritizing operational and functional workflows [53], [54].

B. APPLICATION AND OPERATIONAL COMPETENCIES

Accommodating requirements into software architecture raises many concerns in terms of applicability and operational competencies [55]. Secondly, how far these tow distinguished fields can manage simultaneously is another potential concern. To address these possibilities, Architecturally Significant Requirements (ASR) came into existence. ASRs are bidirectional operational i.e. requirements to design the architecture and architectural framework with a set of executable skills for practical re-usage.

ASRs are documented using architectural views [56]. Architectural views act as a tool that helps in knowing the decisions and the rationale behind the system's architecture based on the quality of requirements [57]. Initially architectural views are documented throughout the lifecycle of the project, to draw architecturally significant requirements. Different stakeholders have different quality requirements that are scattered in architectural views and covers reusability levels [58].

Requirements that are done through architectural views enhance the quality of software architecture documents with a keen focus on system architects [59]. It also provides two products i.e. operational software and a framework for further use that ultimately saves time, resources energy, and getting into the development loop unnecessarily.

C. TRACEABLE ARCHITECTURAL CONCERNS

Architecture centric requirement models are subjected to train on group of parameters such as performance check, experimental verification, quality validation, explainability and extent of robustness. These all parameters are specifically designed for making traceable factors controlled and recognizable at any stage of rapid software development [60]. However, every software traceability varies as per the stakeholder demands, but the architecture design choices gives the liberty to cater majority of them with various architectural tactics and authentication ratios (also known as end-product correct recognition rate) [61].

Secondly, traceability between design and architectural centric requirements of different stakeholders can be possible through Decision Centric Traceability (DCT) [62]. DCT covers all the important software engineering activities and planned steps while including validation of requirements through architecture as well. Whether functional or non-functional, a particular requirement can be addressed through DCT specific layered approach that helps in simplifying the whole process of the software lifecycle [63]. Incorporation of traceable architectural concerns with requirements elicitation adds extra layer of scrutiny to reflect better complexity of typical software artefact traceability.

D. LIMITATIONS OF ARCHITECTURALLY SIGNIFICANT REQUIREMENT

Despite being into solving many requirement elicitation concerns and setback, architecture centric requirements posses various limitations that sometimes leads to blind end for rapid software development [64]. The factor that are non-accommodating are discussed in detailed under research gaps and challenges section. However, notifying and analysing requirements that impact the architecture early in the software development life cycle can save a lot of resources [65].

Skills are required by the designer to capture functional requirements for architectural decisions before-hand. Identification and accommodation of functional requirements in the architecture from the requirement document can be automated through many tools and techniques to avoid blind ends. Thus making this issue easily sorted out through technical

expertise. For example, to eradicate these factor, one such tool is ArcheR [66]. This tool predicts non-suitability and inapplicability of various models to save potential setbacks and unexpected results. These kind of tools demands high level expertise with better precision to command better checks [67]–[70].

V. GAPS BETWEEN SOFTWARE ARCHITECTURE AND REQUIREMENTS (RQ3)

Requirement engineering, software architecture, and software design are three different processes of the Software Development Life Cycle (SDLC). Conventionally, requirements are gathered in the early phase, while software architecture is considered in the later phases of development. Although these processes are connected to each other, still certain gaps and lapses exist between them.

This Section explores an abstract view of research, industrial and academic gaps between software architecture, design and requirements based on current architectural trends and adopted measures for software development gaps that exist between software architecture and requirements.

A. IDENTIFICATION OF RESEARCH GAP

Identification of research gaps are tedious and intricate step when it comes to software requirements elicitation and software architecture. Whenever a problem arises, it is catered and fixed through possible solution to meet deadlines and develop software timely [71]. However, to mark an unusual activity as research entity/research gap is rarely done that leaves the situation same and problematic for future handling [72], [73]. Lack of research and open ended statements databases in a given software production houses makes things out of sight for prolonged period and sometimes not even considered as research gap.

Figure 5, illustrates in-depth view of research gaps and challenges. Ability to solve a problem and ability to locate a problem that needs out of the box context to resolve are two different things. Requirement engineering and software architecture are fundamental to each other for the success of a software artefact. Identification of research gaps is vital in identifying and establishing traceability between functional and architectural requirements [74].

Research gaps mainly exist between functional and architectural requirements linkages [75]. Secondly, the vague connections between problem and solution domain are critical because software architecture is a fundamental part of a software solution. To overcome research gaps, a better strategy is to treat both requirements and architecture at the simultaneously.

Conventionally and in legacy systems, Software requirement, software design and software architecture are supposed to be treated differently. Separating these domains proves to be problematic because of unanimously bringing hurdles that other domain overtakes. Working on these domains side by side facilitates a smoother process for research gaps identification [65], [76].

B. ADDRESSING QUALITY REQUIREMENT

Quality of a software product is an intransigence factor. For executable software components, quality comes first and foremost. Addressing quality requirements has always been a challenge for the requirement engineering process and software designing phase [77]. To focus functional requirements of the system, a better designing approach integrating architectural perspective is required along with suitable skills, tools, and techniques.

Quality attributes are the backbone for the success of the software artefacts. Traditionally, quality attributes like usability, reliability and other quality requirements are considered later stages of development that itself leads to late identification of flaws [78]. Therefore, software architecture is considered as favourable choice as it focuses on non-functional quality attributes along with functional requirements to make things smoother [78].

C. IDENTIFYING ARCHITECTURALLY SIGNIFICANT REQUIREMENTS

Architecturally Significant Requirements (ASRs) are requirements that determine the shape of the software architecture and help in taking important decisions [79]. ASRs are bidirectional operational (software architecture software design) requirements to design the software architecture and documented using architectural views [80]. The identification of these requirements is a major challenge for software requirement engineers.

During the elicitation, stakeholders and requirement engineers lack the ability of identifying these requirements at first place. Lack of ability covers factors like poor domain knowledge, lack of skill expertise and sometimes software automation tools ignorance [81]. Furthermore, lack of proper evidence (non-traceability) for identifying and characterising these requirements results in improper design and poorly designed architecture of the system [79].

D. FRAMING BUSINESS REQUIREMENTS TO BUSINESS MODELLING

Modelling a business-oriented system is already a challenging task in reference to software architecture [82]. Few business processes cannot be modelled by using traditional software architecture frameworks and semantics [3], [83]. Advanced studies and methodologies are required for framing business-oriented requirements combined with process modelling [84].

Business requirements are difficult due to different global priorities, varying cultural norms, changed business strategies (region, sector, and domain) and profit expectations [85]. Business factors are never globally alike that makes them difficult to synchronize at single platform to avail standard services [86], [87]. Framing business requirements to business modelling gives promising research platforms to researchers to explore and contribute in the area of study.

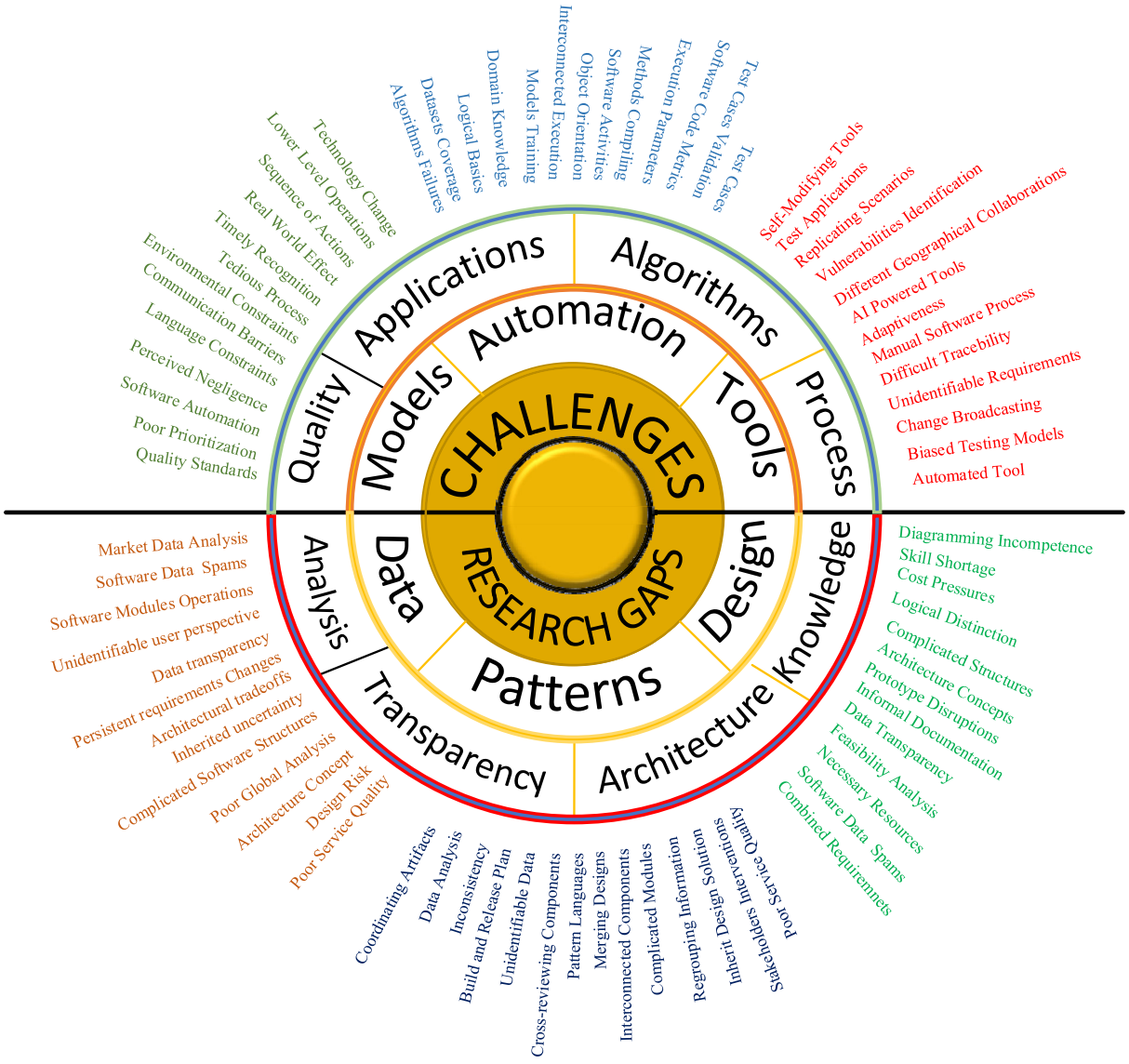


FIGURE 5. Dendrogram for research gaps and challenges.

E. TRADE-OFFS CONSIDERATION

According to IIIE (Industrial Information Integration Engineering) [88], certain trade-offs are required at the design level for mapping requirements to architecture, but most people lack knowledge of trade-offs that are essential for this purpose [89], [90]. Trade-offs can be defined as an compromise occurred between two desirable, but incompatible features. In requirement elicitation and software design, objected and relevant tradeoffs are required to avoid unforeseeable situation [91].

In software design, correctness of system behavior while keeping appropriate balance among various quality features is challenging task to accomplish [92], [93]. Software systems with higher uncertainty levels and environment is even more complex feature. Design principles and existing tools and techniques are still limited because they don not comply and link with design decision for quality requirements satisfaction.

Moreover, amount of information utilized by a human-designer is also limited and thus difficult to differentiate actual and right needed solution out of human knowledge pool. A computational tool or scheme to assist in tradeoffs management can bring ease to the process. Secondly, enabling the discussion platforms to design, select, and deselect tradeoffs among specified software development teams can temporary contribute to facilitate comprehension of tradeoffs [90], [94].

The requirement engineering process has two key sub-processes: i) Requirement elicitation and development ii) Requirement management [89]. Trade-offs that requires traceability poses difficult situation [95], [89]. Since new requirements are continuously introduced,

trade-offs traceability is difficult, along with mapping of requirements to architecture produces a major challenge [95].

VI. BRIDGING THE RESEARCH GAP USING ACRE (RQ4)

In this Section, different approaches regarding architecture-centric requirements are written and explained that is helpful in answering how we can bridge the architecture and requirements gap. Proposed frameworks for improving the requirement and architecture in software engineering are also discussed. Mapping requirements to architecture pros and cons along with state-of-the-art suggestions are part of this section.

A. BRIDGING REQUIREMENT ELICITATION AND SOFTWARE ARCHITECTURE

Complex and software intensive systems with high operational demands for environment and continuous availability are critical concern to bridge for requirement elicitation along with software architecture simultaneously. It is not always possible to link perfectly both areas for research and practice [96]. Bridging requirements into software architecture demands perfect alignment of steps and application procedures. Both process can be expected executable simultaneously to address sustainability and traceability for better framework and pattern. Bridging both approaches and principal aim to provide foundation and roadmap of intermediate themes and embedded conceptual construct with reusable properties [97].

Bridging requirement elicitation and software architecture emerging, clear and systematic design approaches that gives rise to high-quality, sustainable, maintained and estimated framework for better and rapid software development [98]. In developing software systems, understand-ability and acknowledgement of requirements and architecture is a prominent factor. To attain this factor, traceability between requirements artefacts and software artefacts needs to be monitored through well-designed techniques. There are various techniques available that are working on the classification of requirements into functional problems and architectural problems with the help of knowledge assisted approaches. One such technique is the Twin Peaks model [99]–[101]. Twin Peaks model is based on the classification of architecture into sub-contexts in order to bridge requirements and architecture more precisely [102].

B. BRIDGING PROS AND CONS

In bridging requirements elicitation and architectural research and industrial gaps, there are plenty of problems encountered by application and operational techniques [103]. There are always some good factors and some drawback associated with integrating requirements and architecture. In a generic adaptability approach, good factors are considered with prime importance and valued more then cons while deciding right approach to use [104]. In requirement engineering phases, there is nothing such as either a right choice or a wrong choice. In integrating two different processes together to have better results, approaches are decided based on lesser odds factor associated with adapting any approach.

While bridging pros and cons, the factors of right and wrong (but fixable) factors are mostly needed and expected [105]. The controlled situation even if with odd (but controllable) factors makes bridging suitable. To retain few odds factor along with positive components are somehow sign of a good choices. This is due to the reason unnecessary connections among requirement and architecture viewpoints may arises in the bridging process while looking for a perfect fit. These unnecessary connections not only make the system complex but also makes hidden sub-links [106].

Sub-links are difficult to eradicate due to high dependency, hidden nature, and interchangeable information during various steps. These sub-links can even cause failures in high-level systems domains and still remain unrecognizable. Bridging requirements and architecture requires high-level expertise knowledge that is sometimes impossible to identify in the early stages of software development [107]. This factor makes these bridging techniques less applicable to apply for software systems. In other words, less expertise makes easy solution hard to implement [2]. A proper domain knowledge is needed the most for robust, testable, and traceable bridging.

C. SYNCHRONIZATION OF EXPECTED SYSTEM BEHAVIOUR

''What may comes next'' is challenging yet success determining factor to keep track of right moves [108]. Anything that occurs unexpectedly raises many concerns i.e. failure in desired performance, ripple effects, inconsistent operations and sometimes whole project failure. Synchronization among expected outcomes and actual results of system behaviour is useful factor in bridging requirement elicitation and software architecture [109]. It helps in keeping track and makes factors traceable. Moreover, it also facilitates controlled environment for bridging and guarantees recognition and performance estimation.

D. BRIDGING AND QUALITY PERSPECTIVE

A system architect and requirement engineer uses quality requirements to design the architecture of a system. System's final design is also an essential perspective achieved through expected quality requirements [110]. Architectural frameworks requires expert skill set for practical usage. However, software quality requirements incorporated into software architectural block enables better accuracy, time performance, robustness, and explain-ability for all the executable components under consideration [111]. Considering quality perspective takes less identification and training time that gives rise to better alternate option if software product turns out differently. Due to supreme importance of quality attributes, it is alternatively named as reusable architectural building blocks. Thus, bridging requirements into architecture is seems nearly impossible without quality perspective [112].

VII. ACRE BASED REQUIREMENT ELICITATION PROCESSES AND TOOLS (RQ5)

In this Section, Architecture centric requirement is identified through state-of-the-art tools, techniques, and schemes. There are many tools and techniques for requirements and architecture mapping. During literature evaluation, various tools, methods, processes, and frameworks, along with few languages and techniques are identified.

A. TOOLS AND TECHNIQUES ANALYSIS

In ACRE based tools and techniques, the most effective is ADL along with a set of a defined rule set from PL-Aspectual ACME and PL-AOV graph [113]. Architecture Trade-off Analysis Model (ATAM) was developed to identify Architecturally Significant Work Items (ASWI) and different constraints on the current system, and its requirements that are architecturally significant [114]. Integration of LISA in the Eclipse IDE architecture-related activities along with Architecturally Significant Requirements (ASRs) and Architecture Design Decisions (ADDs) are described as part of the architecture model. In other words, the architecture description contains solution structures as well as architectural knowledge that serve as the basis for solution structures.

Strategy for Transition between requirements models and Architectural Models based on Architectural Patterns (STREAM-AP) also has a great impact as it considers the non-functional requirements [115]. To make sure that all requirements are completely described by the architecture AADL (Architecture Analysis and Design Language) software architecture shows the collection of schemes that is answered in RQ 4. In Table 3, there are four major approaches, including PL-Aspectual ACME and PL-AOV graph [113], use case and feature modelling [116], data mining [66] and Backlog modification, respectively.

Figure 6 illustrates detailed a quick Model for Taxonomical resources bank starting from the main heading i.e. ACRE followed by relevant groups and sub-groups with suitable resources that provides a relevant bulk of sources under a specified category. Moreover, six frameworks include methods tools and techniques for bridging requirements and architecture cited as [84], [95], [116]–[119].

We have identified the four sub-categories as follows; 1) heuristics; 2) exact searching techniques; 3) hybrid methods with the combination of both heuristics and exact searching techniques; 4) new strategy to deal with the problem. After observing the selected studies, heuristics is considered the most frequently used method in response to the selected problems. Due to the complex relationships among requirements, it is costly enough to find the solution that is optimal for requirements as well as for architecture that completely covers those requirements. Therefore, the heuristics is considered suitable for uncertainty and difficulty at the same time to find the best solution. The detailed percentages and their factors are discussed in Figure 8.

Certain studies use the hybrid methods, for instance, Durdik *et al.* [120] used a combination of quality attributes. According to the authors, not every decision about architecture deduced from architectural solutions are available and not every requirement is architecture relevant. Therefore, the semi-automated proposal and integration of solutions can improve overall system quality targeting sustainability, design speed, and assures that the design decisions are validated before implementation. These methods depicted a capability to resolve real selection of requirements and architecture mapping problems.

As per our opinion, the combination of new techniques is recommended to improve the performance and to find the best solution. The studies that adopt the exact method has main focus in programming oriented solution, and as far as the category of new methods are concerned for problem solving, like STREAM method by Silva, Lucena *et al.* [115] and its new variations shows the results about the problem type formulation that can be considered in the selection of software requirements, mentioning three types of objectives: studies with solo objective, multi-objectives and hybrid studies. The detailed study shows that the solo-objective design is frequently used to address the problem in software requirement selection till its conversion to architecture.

However, the paper Goknil *et al.* [121], was published, with the aim of generating and validating the architectural traces by the use of relations of requirements or architecture verification. Authors proposed that architecture is nearer to software engineering in real world as compared to the solo-objective engineering process. It gives opportunity to unite multiple, conflicting objectives, for instance meeting approximately all customers requirements but with minimal total effort in the whole process. Other applications of multi-objective approaches pay attention on several quality attributes, such as reliability, performance and scalability, besides less costs and risks via initial evaluation of decision [120].

Watching and dissecting the selected papers during systematic review, the first plan utilized mostly is the solo-objective and, after a few experimentation, the use of multiple objective definitions is utilized, going for more exact subsequent results. In the current state, different software engineering issues arises. For example, the study [122] had to describe the role of requirement change in which they introduced a model using Twin Peaks and instead of verifying the results for their model they used the expert opinion to justify their model which is not sufficient. Figure 7 is categorizing Tools based on hybrid (light blue), supervised (green) and un-supervised (blue) methods. Ratio of supervised methods are in majority, un-supervised in the middle while hybrid are in less.

Targeted approach for analyzing architecturally equipped design with respect to software process is crucial factor to determine. Various tools and methods are used for bridging requirements and architecture. Advancements are made for the implications and practical concern regarding the methodologies. Although, theoretical concerns are located, but no quantitative analysis is made to have statistical view of purposed and deduced results. The measurable analysis is not

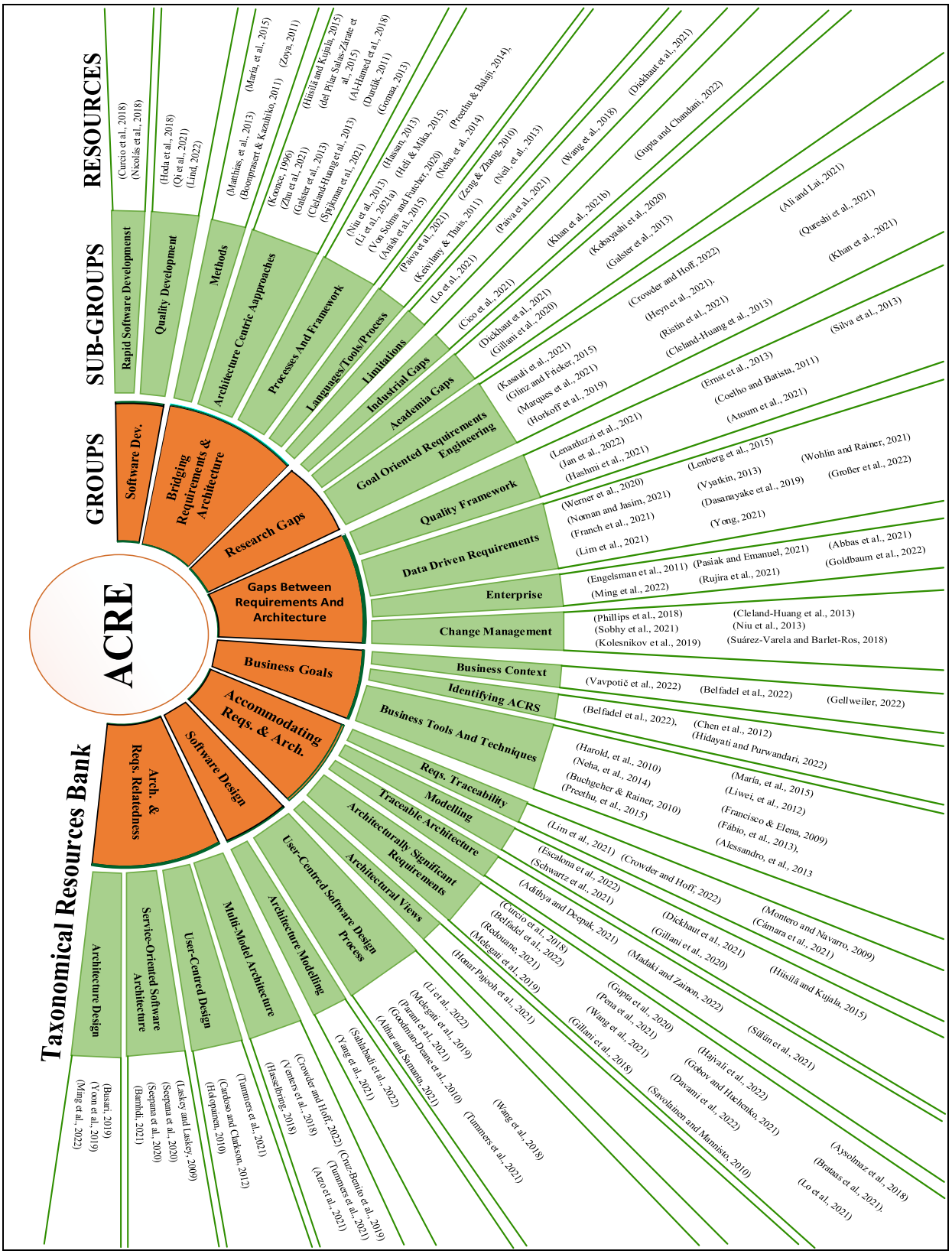


FIGURE 6. Taxonomical resource bank for integrating software architecture in requirements elicitation.

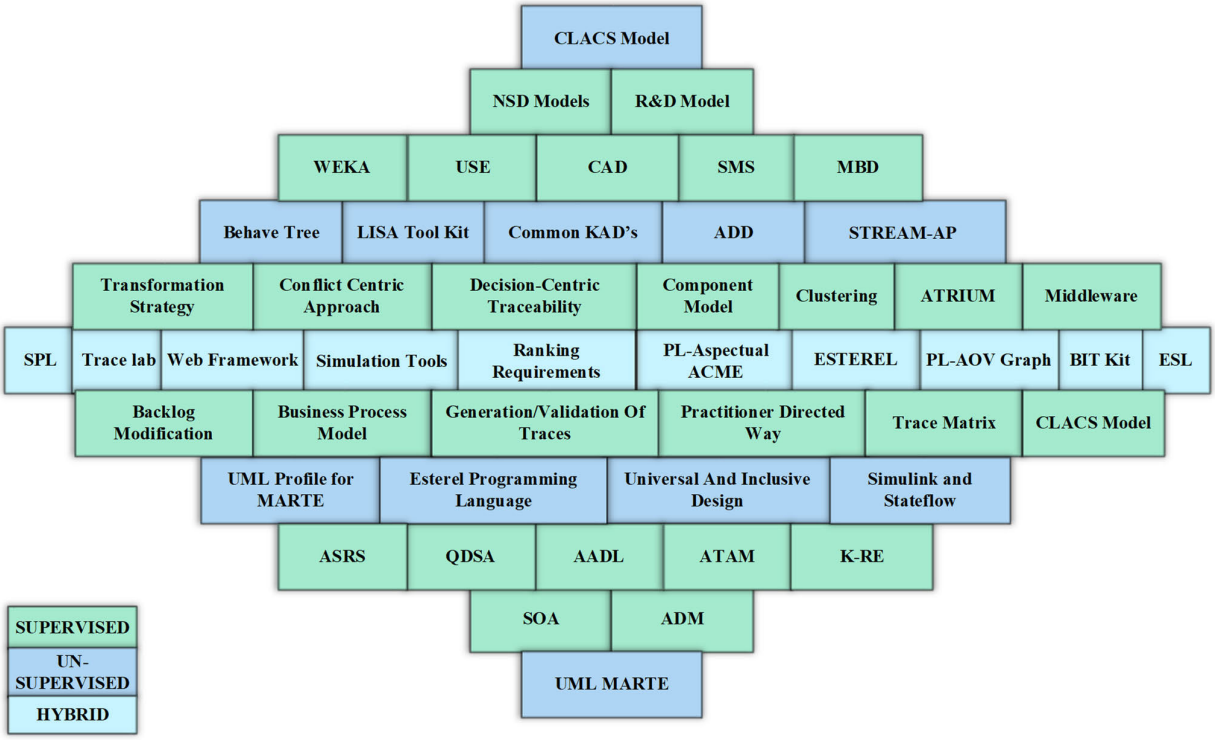


FIGURE 7. Tools, techniques, models, and frameworks categorization.

only contributory enough to step forward for correctness of methodologies, but it is also suitable enough for clear acceptance and rejection in opting/selecting best techniques and tools for establishment the architecture centric requirement elicitation.

Critical and lacking aspect of requirement analysis is its inability to produce future targeted tools. There are many case where conducted studies are not sufficient to produce relevant and most accurate methodologies for the architecture centric requirement elicitation. Secondly, requirements are subjected to have huge variance that sometime results in difficulty to attain desired certainty and output results. In some cases, processes are replicated, duplicated, and missed in such a way that is difficult to eradicate out. Targeted tools and techniques for future work is possible when required concerning conditions are declared suitable for the further research applications.

Most of the techniques that are related to architecture centric requirement engineering are methodology centred. These techniques focus on derivation of output rather than functionalities, traceability and adopted approaches. Focus in requirement analysis is subjected for diversity to seek better results and excellence to have accurately specified methodology.

The problem formulation from software requirements is a critical task. Solo-objective design is beneficial in problem formulation and addresses software requirement selection iteratively until its conversion to architecture. However, Goknil *et al.* [121] targeted the aim of generating and validating traces by the use of relations of requirements and architecture verification through a multi-objective approach. The multi-objective approach is better in operational software engineering in the real world as compared to the solo-objective. Moreover, a solution taken from a multi-objective approach gives various possibilities for quality attributes such as reliability, performance and scalability along with reduced cost and risks [120].

A targeted approach for analysing architecturally equipped design with respect to the requirement process is a crucial factor to determine. Various tools and methods are used for bridging requirements and architecture. Advancements are made for the implications and practical concerns regarding methodologies. Although theoretical concerns are located, but no quantitative analysis is made to have a statistical view of purposed and deduced results. The measurable analysis is not only contributory enough to step forward to keep a check on the correctness of methodologies, but it is also suitable enough for clear acceptance and rejection in opting best techniques and tools for the establishment of ACRE.

The critical and lacking aspect of requirement analysis is its inability to produce future targeted tools. It is somehow replicated, duplicated and missed in such a way that is difficult to eradicate out. Targeted tools and techniques for future work is possible when the current concerned conditions are declared suitable for further research applications.

TABLE 3. Comparative analysis of tools and techniques.

Tool/Technique	Computational /Time Complexity	Pros/Cons	Description/Reason	Expertise Level	User-friendly
Ranking Requirements [123]	Linear	It facilitates better prioritization of requirements, but can be problematic if right choices for a specific requirement is not designated.	Linear time complexity if the orders are already computed	Low	Yes
Clustering Method [124]	Linear	It efficiently work with group entities based on attributes, but creates dependency among various requirements that is against object oriented approach.	For a data-set with m objects, each with n attributes, the k clustering algorithm o (i*k*m*n). For large data-sets where k << m & n << m, the complexity is approximately o(m).	Low	Yes
PL-Aspectual ACME [113]	Linear	An aspect oriented architectural description language, but requires high technical expertise and skill set.	Can be modelled using conventional ADL description.	Medium	No
SPL [116]	Linear/ logarithmic	It is better to cater structured requirements, but can be difficult to handle for other unstructured features.	Linear in case of SPL-queues and SPL-arrays whereas quadratic in case of SPL hash table.	Medium	No
Common KAD's [125]	High	It provides better support for industrial design processes, but lower level of constraints and scope are not a good fit for it.	As it works on peer to peer structure.	High	Yes
UML Profile for MARTE [126]	Medium	It is particularly suitable for modelling and analysis for real-time and embedded systems, but being a domain specific makes it usage limited .	Complexity depends on the complexity of the UML model used.	High	Yes
Business Process Model [84]	Low	Its area of expertise is related to business and enterprise, but remains an abstract and depends on intended use only.	Very close to real-world, less technical terms.	Low	Yes
R&D Model [89]	Linear	It facilitates research and development that goes well for complex scenarios and developments, but it is very time consuming.	The model makes a linear, one-way outlook between consecutive stages.	Low	Yes
PL-AOVgraph [66]	High	An aspect oriented architectural description language, but it is semi-automatic strategy composed of relationship of requirements.	Include both variability and requirement information. Hence difficult to manage.	High	No
Quality-Driven Self-Adaption [127]	High	It helps in incorporating requirements and architecture level to attain quality rich product, but quality parameter differs globally thus challenging its performance.	Continuous adaption of standards to meet quality.	Low	Yes
Behave Tree [128]	High	Its predicts well the expected output along with actual results, but often leads to false predictions based on data requirements.	The use of natural language and high scalability causes complexity.	High	No
ATRIUM [78]	Low	It works better for business intelligence designing, but does not fit well for lower complexity products.	Covers abstraction levels and functional requirements only.	High	Yes
Trace lab [129]	High	It is integrated software solution for trace forward analysis, but only applicable to larger projects with higher traceability requirements.	Traceability is based on architectural decisions.	High	No
ATAM [114]	Medium	It is a method of architecture evaluation, but has limited capacity to evaluate requirements along with it.	High in case of excessive cots usage else, it is low.	Medium	No
STREAM-AP [115]	High	Efficiently work to reduce gaps between requirements and architecture, but mainly emphasize on functional requirements.	The process includes refinement of NFR.	High	No
Conflict Centric Approach [52]	High	It is suitable for eradicating conflicts through systematic approach, but uses more time along with other processes.	Conflict handling is a quite challenging task.	Low	Yes

TABLE 3. *(Continued.)* **Comparative analysis of tools and techniques.**

Component Model [130]	High	It works to satisfy mechanism and methods for the composition of components, but mainly emphasize on separation of functionalities.	Trying to bridge the gap between requirements and design.	Low	Yes
Practitioner Directed Way [89]	Low	A customize tool, easily adaptable to users requirements, but raises concerns over precision.	Only considering the qualitative approach and parameters to meet quality requirements.	High	Yes
Trace Matrix [129]	High	A mathematical model with better accuracy and directed results, not suitable for lower level design projects.	Difficulty in dealing with dependencies between requirements.	High	No
ADD [100]	Medium	A requirements traceability iterative matrix, but often leads to extensive scrutiny for simpler requirements.	High in the case when so many iterations else medium.	High	No
ASRS [79]	Low	It is cost-effective, simpler automated tool for structuring requirements, but not really suitable for non-structured context.	This characterises the architecturally significant requirements. If no common features between requirements, then complexity is high.	Low	Yes
WEKA [66]	Low	A latest, machine learning software, mostly suitable to modern and complex design solution, but only applicable to larder datasets (requirements).	Provide the simple interface and quick processing of results.	High	No
LISA Toolkit [131]	High	It is for regression testing and interactive analysis, but does not suits wells for Object oriented cases.	Solution structures but also the architectural knowledge.	High	No
K-RE [117]	Low	It helps in breaking down software components interactively, but only fits well to larger software components.	Division of architecture into sub-categories.	High	No
Simulink and Stateflow [118]	High	Sequential decision making model, but retains dynamic changes and controls oriented solutions.	Deals with dynamic systems, so the operations are also complex.	Low	No
Generation/Validation Of Traces [121]	High	It automatically traces requirements and architecture components, but only applicable for larger industrial projects.	Deals with complex sets of input, e.g. Trace, architecture and requirement model.	Medium	Yes
BitKit [132]	Low	It is a platform for requirements engineering and capability models, but must be incorporated in early prototyping.	It is a prototyping tool and includes the simple business processes.	High	Yes
Decision-Centric Traceability [133]	Low	All links are traced for architectural models based on platforms and constraints. But, holds stakeholder factor which adds time and cost to bring quality.	Deals with the system and abstract level of requirements.	High	Yes
Backlog Modification [120]	Low	A tool to support better backlog management, but again cost and time are effected.	Deals with requirements that are irrelevant to architecture.	High	No
SOA [50]	Medium	It is service oriented architecture that supports service orientation, but only applied at software design level.	Complex when taking too many services from outside.	Low	Yes
Transformation Strategy [134]	Low	A digital approach for business strategy (plan of action), but only limited larger enterprises.	Automatic transformation strategy from requirements to architecture is used.	Low	Yes
ESTEREL Programming Language [135]	Medium	A language mainly used for complex reactive systems, but it is synchronous	Deals with the real world.	High	No
USE [136]	Low	It supports unified modelling, but has worth in small scale projects.	Includes verification of critical properties.	Low	No
CLACS Model [137]	Medium	It is specifically designed for quality retaining factors, but can take more time if issue resides in later stage.	Deals with architecture constrain.	High	No
ESL Methodology [138]	How	It is methodology for task based project development, but applicable to larger enterprises.	Deals with multi-threaded processes.	High	No
Middleware [139]	High	It facilitates common specification for data rich applications. But, mostly suited for OS platforms.	Deals with distributed and real-time systems.	Low	No

TABLE 3. *(Continued.)* **Comparative analysis of tools and techniques.**

SMS [140]	Low	It supports goal oriented requirements engineering to produce systematic mapping study. But, low end requirements often neglected.	Analyse the requirement and architecture relations progressively and systematically.	High	No
Simulation Tools [23]	High	It is a set of tools for elements based structural software development but only caters rapid action development.	Deals with user-centred deadlock requirements.	Low	Yes
NSD Models [31]	Medium	Computation model for new service development, but particularly for role of service.	Includes component and testing based cycles.	High	No
Universal And Inclusive Design (UID) [32]	Medium	It applied universal design principles for greater accessibility, but limited to principles and instructional design.	Selects the proper design approach that is well suited to design requirements.	High	No
CAD Tool [33]	High	It is design and automation tools helpful for prototyping, but requires higher level of expertise and more technical insight.	Static translation and dynamic integration requires cad tool domain analysis.	Low	No
ADM [22]	High	It is specific architectural design method, but facilitates design level features only.	Deals with critical architectural centric requirements.	High	No
PL-AOV Graph [113]	High	It is used for goals modelling and composition, but needs to be performed by experts and with higher level knowledge.	Aspect oriented requirements engineering and modelling language.	High	No
MBD [118]	Linear	It is model-based software development that caters rapid point of actions with better time management. It is resource taking along with adding extra cost.	Model based design is mathematical and visual complexity control method specifically for automation and simulation.	High	No
Web Framework [119]	Linear	It is design to support easy software and application development, but only experienced developers can use it properly.	Set of tools for software developers to build and manage applications development.	Low	Yes
AADL [128]	Linear	It is modelling language that supports early and iterative analysis, but preferable for performance critical properties.	Modelling language that facilitates early and iteratively system's architecture relevant to its design and performance.	High	No

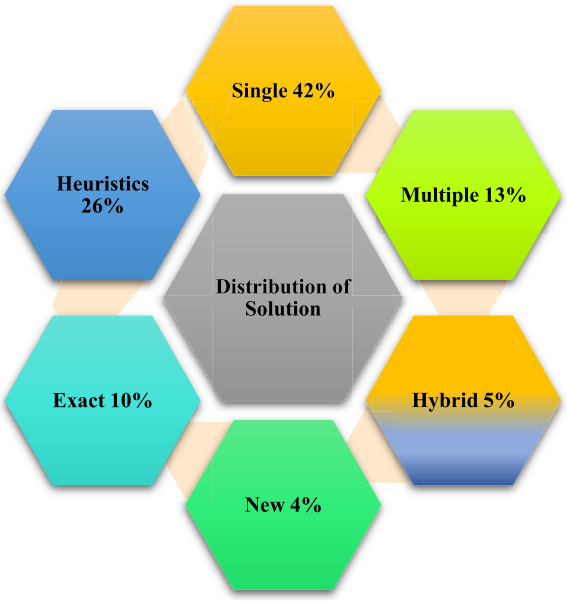


FIGURE 8. Percentage distribution of solutions availability to support ACRE gaps and challenges.

In the light of Figure 8 and Table 3, there are only 5% hybrid tools and techniques that can support architecture centric requirements elicitation. However, tools with single objective holds major percentage ration i.e. 42%. This factor indicates that multi-objectives specification are only 13% catered while covering 5% hybrid approaches. Only 4% recent and modern tools are encountered that are coping up with recent lapses. The overall solution availability percentages are not yet fulfilling and demands researcher interest to meet current needs as referred in Figure 8. Moreover, Single stands for those techniques that are based on one algorithm/method and Multi stands for multiple algorithmic applications in requirement elicitation. The solution deduced from multi-objective approach not a single solution, but a solution set with multiple possibilities and states to be decided by the person who makes the decision. Hybrid stands for complex tools and techniques that are single at a certain point and gets multi-objective when required. Hybrid tools are dire need of the current scenarios and market needs in order to facilitate better. The great benefit of this is the opportunity to unite multiple, conflicting objectives, for instance

meet approximately all customers' and requirements engineer needs with minimal total effort in the whole process.

Table 3, holds detailed yet targeted aspects of tools and techniques with pros and cons, short meaningful description and computation complexity, user-friendliness along with expertise level. User-friendliness determines that whether a tool and technique is easy to adopt by a requirement engineer while interacting with layman clients. Expertise level covers the level of skill set required by a requirement engineer to adapt a method for a specified purpose. Description is given for reasonable short purpose supported by pros and cons. Pros and cons present a clear picture of the advantages of adapting a method and a possible drawback in order to guide requirements engineer better and ahead of time.

VIII. CONCLUSION

Concluding all, we have studied about requirements and architecture integration. How they are relevant to each other, and whether is it possible to accommodate requirements within architecture. We covered research gaps between requirements and architecture along with potential challenges. In this article, considerable amount of tools and techniques are mentioned and thoroughly discussed that can bridge gaps effectively. Moreover, a detailed comparison and analysis of these tools and framework is also provided. We have covered very extensive taxonomical resource bank that encapsulates relevant group of areas with recent and relevant resources for better reader experience.

This paper is a clear proof that the foundation of a software development process is the identification of the relationship between the software architecture and the requirements since the software architecture's quality is dependent on how well the system design has accommodated the requirements. Modelling of requirements through frameworks can be done among various architectures in order to trace requirements in software design. Moreover, tools and techniques (Table 3) can help in determining the evolving requirements by considering the software architecture only. Architecture centric requirement engineering approaches are vital in mentioning the importance of requirements traceability in software design. It helps in balancing the requirements in architecture during any phase for software life cycle.

The method that has been used to conduct this research is based on the fundamentals of systematic literature review (SLR). In this paper, we grouped recently proposed different tools, techniques, frameworks, guidelines and languages to accommodate requirements in architecture. A software engineer can choose anyone of mentioned tools and framework based on the requirements and domain. By analysing all the papers in this SLR, we have observed that requirements that are gathered keeping in view of the architecture of the system helps in reducing the gaps between the field of requirement engineering, software design and architecture. The study conducted has concluded the usefulness and applicability of all the latest tools and techniques through an extensive analysis that can be viewed in the papers referred.

FUTURE WORK

The effort required in this research has demonstrated that there could be many dimensions in which our work can be taken forward. In future, proposed analysis, tools, and techniques can be extended by adding more tools/frameworks and giving valuable insights about the importance of the connection between the requirements and architecture. Another future perspective is to work on reducing the paradigm shift between software architecture and requirement engineering. Moreover, it would be worth comparing the architecture with the rest of the software development processes by considering software architecture i.e. pillar of the software system.

REFERENCES

[1] S. Lim, A. Henriksson, and J. Zdravkovic, "Data-driven requirements elicitation: A systematic literature review," *Social Netw. Comput. Sci.*, vol. 2, no. 1, pp. 1–35, Feb. 2021.
[2] J. A. Crowder and C. W. Hoff, "Introduction to multidisciplinary requirement engineering (MDRE)," in *Requirements Engineering: Laying a Firm Foundation*. Cham, Switzerland: Springer, 2022, pp. 1–9, doi: 10.1007/978-3-030-91077-8.
[3] A. Belfadel, J. Laval, C. Bonner Cherifi, and N. Moalla, "Requirements engineering and enterprise architecture-based software discovery and reuse," *Innov. Syst. Softw. Eng.*, vol. 18, pp. 39–60, Jan. 2022.
[4] X. Franch, A. Henriksson, J. Ralyte, and J. Zdravkovic, "Data-driven agile requirements elicitation through the lenses of situational method engineering," in *Proc. IEEE 29th Int. Requirements Eng. Conf. (RE)*, Sep. 2021, pp. 402–407.
[5] A. Redouane, "Towards goal-oriented software requirements elicitation," in *Proc. IEEE Int. Conf. Syst., Man, Cybern. (SMC)*, Oct. 2021, pp. 596–599.
[6] M. Sahlabadi, R. C. Muniyandi, Z. Shukur, and F. Qamar, "Lightweight software architecture evaluation for industry: A comprehensive review," *Sensors*, vol. 22, no. 3, p. 1252, Feb. 2022.
[7] T. Yang, Z. Jiang, Y. Shang, and M. Norouzi, "Systematic review on next-generation web-based software architecture clustering models," *Comput. Commun.*, vol. 167, pp. 63–74, Feb. 2021.
[8] C. C. Venters, R. Capilla, S. Betz, B. Penzenstadler, T. Crick, S. Crouch, E. Y. Nakagawa, C. Becker, and C. Carrillo, "Software sustainability: Research and practice from a software architecture viewpoint," *J. Syst. Softw.*, vol. 138, pp. 174–188, Apr. 2018.
[9] W. Hasselbring, "Software architecture: Past, present, future," in *The Essence of Software Engineering*. Cham, Switzerland: Springer, 2018, pp. 169–184.
[10] J. Cruz-Benito, F. J. García-Peñalvo, and R. Therón, "Analyzing the software architectures supporting HCI/HMI processes through a systematic review of the literature," *Telematics Informat.*, vol. 38, pp. 118–132, May 2019.
[11] X. Li, S. Moreschini, Z. Zhang, and D. Taibi, "Exploring factors and metrics to select open source software components for integration: An empirical study," *J. Syst. Softw.*, vol. 188, Jun. 2022, Art. no. 111255.
[12] S. A. Busari, "Modelling and analysing software requirements and architecture decisions under uncertainty," Univ. College London, London, U.K., Tech. Rep., 2019. [Online]. Available: https://discovery.ucl.ac.uk/id/eprint/10067421
[13] J. Melegati, A. Goldman, F. Kon, and X. Wang, "A model of requirements engineering in software startups," *Inf. Softw. Technol.*, vol. 109, pp. 92–107, May 2019.
[14] T. Wagemann, R. Tavakoli Kolagari, and K. Schmid, "Exploring automotive stakeholder requirements for architecture optimization support," in *Proc. IEEE Int. Conf. Softw. Archit. Companion (ICSA-C)*, Mar. 2019, pp. 37–44.
[15] K. Curcio, T. Navarro, A. Malucelli, and S. Reinehr, "Requirements engineering: A systematic mapping study in agile software development," *J. Syst. Softw.*, vol. 139, pp. 32–50, May 2018.

[16] J. Horkoff, F. B. Aydemir, E. Cardoso, T. Li, A. Maté, E. Paja, M. Salnitri, L. Piras, J. Mylopoulos, and P. Giorgini, "Goal-oriented requirements engineering: An extended systematic mapping study," *Requirements Eng.*, vol. 24, no. 2, pp. 133–160, Jun. 2019.

[17] J. Nicolás, J. M. C. De Gea, B. Nicolás, J. L. Fernández-Alemán, and A. Toval, "On the risks and safeguards for requirements engineering in global software development: Systematic literature review and quantitative assessment," *IEEE Access*, vol. 6, pp. 59628–59656, 2018.

[18] V. Gupta, J. M. Fernandez-Crehuet, T. Hanne, and R. Telesko, "Requirements engineering in software startups: A systematic mapping study," *Appl. Sci.*, vol. 10, no. 17, p. 6125, Sep. 2020.

[19] K. Petersen, S. Vakkalanka, and L. Kuzniarz, "Guidelines for conducting systematic mapping studies in software engineering: An update," *Inf. Softw. Technol.*, vol. 64, pp. 1–18, Aug. 2015.

[20] M. Borrego, M. J. Foster, and J. E. Froyd, "Systematic literature reviews in engineering education and other developing interdisciplinary fields," *J. Eng. Educ.*, vol. 103, no. 1, pp. 45–76, Jan. 2014.

[21] K. B. Laskey and K. Laskey, "Service oriented architecture," *Wiley Interdiscipl. Rev., Comput. Statist.*, vol. 1, no. 1, pp. 101–105, 2009.

[22] W. Engelsman, D. Quartel, H. Jonkers, and M. van Sinderen, "Extending enterprise architecture modelling with business goals and requirements," *Enterprise Inf. Syst.*, vol. 5, no. 1, pp. 9–36, Feb. 2011, doi: 10.1080/17517575.2010.491871.

[23] C. Cardoso and P. J. Clarkson, "Simulation in user-centred design: Helping designers to empathise with atypical users," *J. Eng. Des.*, vol. 23, no. 1, pp. 1–22, Jan. 2012.

[24] S. Yoon, J. Um, S.-H. Suh, I. Stroud, and J.-S. Yoon, "Smart factory information service bus (SIBUS) for manufacturing application: Requirement, architecture and implementation," *J. Intell. Manuf.*, vol. 30, no. 1, pp. 363–382, Jan. 2019.

[25] C. Seepana, A. Paulraj, and F. A. Huq, "The architecture of coopetition: Strategic intent, ambidextrous managers, and knowledge sharing," *Ind. Marketing Manage.*, vol. 91, pp. 100–113, Nov. 2020, doi: 10.1016/j.indmarman.2020.08.012.

[26] M. Purba, E. Ermatita, A. Abdiansah, V. Ayumi, H. Noprisson, and A. Ratnasari, "A systematic literature review of knowledge sharing practices in academic institutions," in *Proc. Int. Conf. Informat., Multimedia, Cyber Inf. Syst. (ICIMCIS*, Oct. 2021, pp. 337–342.

[27] A. Bamhdi, "Requirements capture and comparative analysis of open source versus proprietary service oriented architecture," *Comput. Standards Interfaces*, vol. 74, Feb. 2021, Art. no. 103468.

[28] Z. Ming, A. B. Nellippallil, R. Wang, and J. K. Allen, "Requirements and architecture of the decision support platform for design engineering 4.0," in *Architecting a Knowledge-Based Platform for Design Engineering 4.0*. Cham, Switzerland: Springer, 2022, pp. 1–22, doi: 10.1007/978-3-030-90521-7.

[29] H. Nordal and I. El-Thalji, "Modeling a predictive maintenance management architecture to meet industry 4.0 requirements: A case study," *Syst. Eng.*, vol. 24, no. 1, pp. 34–50, Jan. 2021.

[30] A. Parant, F. Gellot, P. Alexandre, and V. Carré-Ménétrier, "Model-based engineering for designing cyber-physical systems control architecture and improving adaptability from requirements," in *Proc. 11th Workshop Service Oriented, Holonic Multi-Agent Manuf. Syst. Ind. Future (SOHOMA)*, 2021. [Online]. Available: https://hal.archives-ouvertes.fr/hal-03427348

[31] M. Holopainen, "Exploring service design in the context of architecture," *Service Industries J.*, vol. 30, no. 4, pp. 597–608, Apr. 2010, doi: 10.1080/02642060903067563.

[32] J. Goodman-Deane, P. Langdon, and J. Clarkson, "Key influences on the user-centred design process," *J. Eng. Des.*, vol. 21, nos. 2–3, pp. 345–373, Jun. 2010.

[33] D. A. Koonce, "Manufacturing systems engineering and design: An intelligent, multi-model, integration architecture," *Int. J. Comput. Integr. Manuf.*, vol. 9, no. 6, pp. 443–453, Jan. 1996, doi: 10.1080/095119296131418.

[34] Y. Zhu, Q. Guo, H. Yin, K. Liang, and S. S. Yau, "Blockchain-based software architecture development for service requirements with smart contracts," *Computer*, vol. 54, no. 12, pp. 72–80, Dec. 2021, doi: 10.1109/MC.2021.3091379.

[35] M. L. Castro Pena, A. Carballal, N. Rodríguez-Fernández, I. Santos, and J. Romero, "Artificial intelligence applied to conceptual design. A review of its use in architecture," *Autom. Construction*, vol. 124, Apr. 2021, Art. no. 103550.

[36] R. Hoda, N. Salleh, and J. Grundy, "The rise and evolution of agile software development," *IEEE Softw.*, vol. 35, no. 5, pp. 58–63, Sep. 2018.

[37] M. L. Pasiak and A. W. Rahardjo Emanuel, "Enterprise architecture planning (EAP) using TOGAF-ADM at fuel supplier," in *Proc. 13th Int. Conf. Inf. Commun. Technol. Syst. (ICTS)*, Oct. 2021, pp. 73–77.

[38] J. Qi, X. Ma, L. Li, and F. Wang, "Multi-stage TOGAF architecture development method adaption in small-and-medium enterprises—A case study in a start-up logistics service company," in *Proc. 33rd Chin. Control Decis. Conf. (CCDC)*, May 2021, pp. 4106–4112.

[39] T. Rujira, P. Nilsook, and P. Wannapiroon, "Vocational education digital enterprise architecture framework (VEDEAF)," in *Proc. 9th Int. Conf. Inf. Educ. Technol. (ICIET)*, Mar. 2021, pp. 63–67.

[40] E. Lind, "The impact that the quality of requirements can have on the work and well-being of practitioners in software development.: An interview study," Dept. Softw. Eng., Blekinge Inst. Technol., Karlskrona, Sweden, Tech. Rep., 2022. [Online]. Available: http://bth.diva-portal.org/smash/record.jsf?pid=diva2%3A1637596&dswid=9800

[41] J. Marques, S. Yelisetty, and L. Barros, "Requirements engineering in aircraft systems, hardware, software, and database development," in *Requirements Engineering for Safety-Critical Systems*. Gistrup, Denmark: River Publishers Series in Software Engineering, 2021, p. 85.

[42] M. Abbas, A. Ferrari, A. Shatnawi, E. P. Enoiu, and M. Saadatmand, "Is requirements similarity a good proxy for software similarity? An empirical investigation in industry," in *Proc. Int. Work. Conf. Requirements Eng., Found. Softw. Quality*. Essen, Germany: Springer, 2021, pp. 3–18, doi: 10.1007/978-3-030-73128-1.

[43] R. R. Althar and D. Samanta, "The realist approach for evaluation of computational intelligence in software engineering," *Innov. Syst. Softw. Eng.*, vol. 17, no. 1, pp. 17–27, Mar. 2021.

[44] J. Tummers, A. Kassahun, and B. Tekinerdogan, "Reference architecture design for farm management information systems: A multi-case study approach," *Precis. Agricult.*, vol. 22, no. 1, pp. 22–50, Feb. 2021.

[45] Q. Li, R. Gravina, C. Ma, W. Zang, Y. Li, and G. Fortino, "A collaborative BSN-enabled architecture for multi-user activity recognition," in *Data Science and Internet of Things*. Cham, Switzerland: Springer, 2021, pp. 103–119, doi: 10.1007/978-3-030-67197-6.

[46] S. T. Arzo, R. Bassoli, F. Granelli, and F. H. P. Fitzek, "Multi-agent based autonomic network management architecture," *IEEE Trans. Netw. Service Manage.*, vol. 18, no. 3, pp. 3595–3618, Sep. 2021.

[47] H. M. Noman and M. N. Jasim, "A proposed linear multi-controller architecture to improve the performance of software defined networks," in *Proc. J. Phys., Conf.*, vol. 1773, 2021, no. 1, Art. no. 012008.

[48] H. Honar Pajooh, M. Rashid, F. Alam, and S. Demidenko, "Multi-layer blockchain-based security architecture for Internet of Things," *Sensors*, vol. 21, no. 3, p. 772, Jan. 2021.

[49] Y. Wang, G. Zhu, J. Shi, Y. Huang, and X. Guo, "OSAI: A component-based open software architecture for modern industrial control systems," *Arabian J. Sci. Eng.*, vol. 47, pp. 3805–3819, Sep. 2021.

[50] Y.-Y. Cheng and H.-H. Lai, "Constructing an interoperable, design-centric, service-oriented and knowledge-sharing architecture," *Int. J. Comput. Integr. Manuf.*, vol. 24, no. 12, pp. 1075–1094, Dec. 2011.

[51] M. Gillani, A. Ullah, and H. A. Niaz, "Survey of requirement management techniques for safety critical systems," in *Proc. 12th Int. Conf. Math., Actuarial Sci., Comput. Sci. Statist. (MACS)*, Nov. 2018, pp. 1–5.

[52] J. Savolain and T. Mannisto, "Conflict-centric software architectural views: Exposing trade-offs in quality requirements," *IEEE Softw.*, vol. 27, no. 6, pp. 33–37, Nov. 2010.

[53] F. Davami, S. Adabi, A. Rezaee, and A. M. Rahmani, "Fog-based architecture for scheduling multiple workflows with high availability requirement," *Computing*, vol. 104, no. 1, pp. 169–208, Jan. 2022.

[54] M. Hajvali, S. Adabi, A. Rezaee, and M. Hosseinzadeh, "Software architecture for IoT-based health-care systems with cloud/fog service model," *Cluster Comput.*, vol. 25, no. 1, pp. 91–118, Feb. 2022.

[55] Z. Wang, C.-H. Chen, P. Zheng, X. Li, and L. P. Khoo, "A graph-based context-aware requirement elicitation approach in smart product-service systems," *Int. J. Prod. Res.*, vol. 59, no. 2, pp. 635–651, 2021.

[56] D. Gobov and I. Huchenko, "Software requirements elicitation techniques selection method for the project scope management," in *Proc. ITPM*, 2021, pp. 1–10.

[57] B. Aysolmaz, H. Leopold, H. A. Reijers, and O. Demirörs, "A semi-automated approach for generating natural language requirements documents based on business process models," *Inf. Softw. Technol.*, vol. 93, pp. 14–29, Jan. 2018.

[58] H. Femmer and A. Vogelsang, "Requirements quality is quality in use," *IEEE Softw.*, vol. 36, no. 3, pp. 83–91, Mar. 2018.

[59] G. Brataas, A. Martini, G. K. Hanssen, and G. Ræder, "Agile elicitation of scalability requirements for open systems: A case study," *J. Syst. Softw.*, vol. 182, Dec. 2021, Art. no. 111064.

[60] V. Adithya and G. Deepak, "OntoReq: An ontology focused collective knowledge approach for requirement traceability modelling," in *Proc. Eur., Asian, Middle Eastern, North Afr. Conf. Manage. Inf. Syst.* Istanbul, Turkey: Springer, 2021, pp. 358–370, doi: 10.1007/978-3-030-77246-8.
[61] A. A. Madaki and W. M. N. W. Zainon, "A review on tools and techniques for visualizing software requirement traceability," in *Proc. 11th Int. Conf. Robot., Vis., Signal Process. Power Appl.* Singapore: Springer, 2022, pp. 39–44, doi: 10.1007/978-981-16-8129-5.
[62] M.-J. Escalona, N. Koch, and L. Garcia-Borgoñon, "Lean requirements traceability automation enabled by model-driven engineering," *PeerJ Comput. Sci.*, vol. 8, p. e817, Jan. 2022.
[63] E. Sülün, E. Tüzün, and U. Doğrusöz, "RSTrace+: Reviewer suggestion using software artifact traceability graphs," *Inf. Softw. Technol.*, vol. 130, Feb. 2021, Art. no. 106455.
[64] S. Kit Lo, Q. Lu, L. Zhu, H.-Y. Paik, X. Xu, and C. Wang, "Architectural patterns for the design of federated learning systems," 2021, *arXiv:2101.02373*.
[65] D. M. B. Paiva, A. P. Freire, and R. P. de Mattos Fortes, "Accessibility and software engineering processes: A systematic literature review," *J. Syst. Softw.*, vol. 171, Jan. 2021, Art. no. 110819.
[66] P. R. Anish, B. Balasubramaniam, J. Cleland-Huang, R. Wieringa, M. Daneva, and S. Ghaisas, "Identifying architecturally significant functional requirements," in *Proc. IEEE/ACM 5th Int. Workshop Twin Peaks Requirements Archit.*, May 2015, pp. 3–8.
[67] Y. Wang, C. Zhang, and F. Wang, "What do we know about the tools of detecting design patterns?" in *Proc. IEEE Int. Conf. Prog. Informat. Comput. (PIC)*, Dec. 2018, pp. 379–387.
[68] R. X. Schwartz, A. Monge Roffarello, L. De Russis, and P. Apostolellis, "Reducing risk in digital self-control tools: Design patterns and prototype," in *Proc. Extended Abstr. CHI Conf. Hum. Factors Comput. Syst.*, May 2021, pp. 1–7.
[69] E. Dickhaut, M. Li, A. Janson, and J. M. Leimeister, "Developing lawful technologies–A revelatory case study on design patterns," Dept. IT Manage., Shidler College Bus. HICSS, Tech. Rep., 2021. [Online]. Available: https://hicss.hawaii.edu/
[70] V. Lenarduzzi, F. Lomio, S. Moreschini, D. Taibi, and D. A. Tamburri, "Software quality for AI: Where we are now?" in *Proc. Int. Conf. Softw. Quality*. Vienna, Austria: Springer, 2021, pp. 43–53, doi: 10.1007/978-3-030-65854-0.
[71] K. Großer, V. Riediger, and J. Jürjens, "Requirements document relations," *Softw. Syst. Model.*, pp. 1–37, Jan. 2022.
[72] C. Werner, Z. S. Li, N. Ernst, and D. Damian, "The lack of shared understanding of non-functional requirements in continuous software engineering: Accidental or essential?" in *Proc. IEEE 28th Int. Requirements Eng. Conf. (RE)*, Aug. 2020, pp. 90–101.
[73] M. Glinz and S. A. Fricker, "On shared understanding in software engineering: An essay," *Comput. Sci.-Res. Develop.*, vol. 30, nos. 3–4, pp. 363–376, Aug. 2015.
[74] R. Kasauli, E. Knauss, J. Horkoff, G. Liebel, and F. G. de Oliveira Neto, "Requirements engineering challenges and practices in large-scale agile system development," *J. Syst. Softw.*, vol. 172, Feb. 2021, Art. no. 110851.
[75] O. Cico, L. Jaccheri, A. Nguyen-Duc, and H. Zhang, "Exploring the intersection between software industry and software engineering education—A systematic mapping of software engineering trends," *J. Syst. Softw.*, vol. 172, Feb. 2021, Art. no. 110736.
[76] S. Von Solms and L. A. Futcher, "Adaption of a secure software development methodology for secure engineering design," *IEEE Access*, vol. 8, pp. 125630–125637, 2020.
[77] M. Gillani, H. A. Niaz, and A. Ullah, "Multi-cyclic requirement engineering for educational and industrial models in software development," in *Proc. IEEE 23rd Int. Multitopic Conf. (INMIC)*, Nov. 2020, pp. 1–6.
[78] F. Montero and E. Navarro, "ATRIUM: Software architecture driven by requirements," in *Proc. 14th IEEE Int. Conf. Eng. Complex Comput. Syst.*, Jun. 2009, pp. 230–239.
[79] L. Chen, M. A. Babar, and B. Nuseibeh, "Characterizing architecturally significant requirements," *IEEE Softw.*, vol. 30, no. 2, pp. 38–45, 2012.
[80] A. Hidayati and B. Purwandari, "Software engineer competencies in global software development: An Indonesian perspective," *Tehnički Vjesnik*, vol. 29, no. 2, pp. 683–691, 2022.
[81] A. Jan, A. Abbas, and N. Ahmad, "Monitoring and controlling software project scope using agile EVM," in *Evolving Software Processes: Trends and Future Directions*. Hoboken, NJ, USA: Wiley, 2022, pp. 89–121.
[82] D. Vavpotič, D. Kalibatiene, O. Vasilecas, and T. Hovelja, "Identifying key characteristics of business rules that affect software project success," *Appl. Sci.*, vol. 12, no. 2, p. 762, Jan. 2022.
[83] C. Gellweiler, "IT architects and IT-business alignment: A theoretical review," *Proc. Comput. Sci.*, vol. 196, pp. 13–20, Jan. 2022.
[84] H. Hiisila and M. Kujala, "Combining process modeling and requirements engineering: An experience report," in *Proc. IEEE 17th Conf. Bus. Informat.*, Jul. 2015, pp. 242–249.
[85] M. N. A. Khan, A. M. Mirza, S. U. Rehman, R. A. Wagan, and I. Saleem, "Addressing communication, coordination and cultural issues in global software development projects," *EMITTER Int. J. Eng. Technol.*, vol. 9, no. 1, pp. 13–30, 2021.
[86] C. Gupta and P. Chandani, "SERIES: A software risk estimator tool support for requirement risk assessment," in *Research Anthology on Agile Software, Software Development, and Testing*. Hershey, PA, USA: IGI Global, 2022, pp. 1139–1153.
[87] S. Goldbaum, A. Mihaly, T. Ellison, E. T. Barr, and M. Marron, "High assurance software for financial regulation and business platforms," in *Proc. Int. Conf. Verification, Model Checking, Abstract Interpretation*. Philadelphia, PA, USA: Springer, 2022, pp. 108–126, doi: 10.1007/978-3-030-94583-1.
[88] N. Li, L. Zhao, C. Bao, G. Gong, X. Song, and C. Tian, "A real-time information integration framework for multidisciplinary coupling of complex aircrafts: An application of IIIE," *J. Ind. Inf. Integr.*, vol. 22, Jun. 2021, Art. no. 100203.
[89] N. Niu, L. D. Xu, J.-R. C. Cheng, and Z. Niu, "Analysis of architecturally significant requirements for enterprise systems," *IEEE Syst. J.*, vol. 8, no. 3, pp. 850–857, Sep. 2014.
[90] S. Kolesnikov, N. Siegmund, C. Kästner, A. Grebhahn, and S. Apel, "Tradeoffs in modeling performance of highly configurable software systems," *Softw. Syst. Model.*, vol. 18, no. 3, pp. 2265–2283, Jun. 2019.
[91] J. Cámara, M. Silva, D. Garlan, and B. Schmerl, "Explaining architectural design tradeoff spaces: A machine learning approach," in *Proc. Eur. Conf. Softw. Archit.* Springer, 2021, pp. 49–65.
[92] D. Sobhy, R. Bahsoon, L. Minku, and R. Kazman, "Evaluation of software architectures under uncertainty: A systematic literature review," *ACM Trans. Softw. Eng. Methodol.*, vol. 30, no. 4, pp. 1–50, 2021.
[93] D. M. Phillips, T. A. Mazzuchi, and S. Sarkani, "An architecture, system engineering, and acquisition approach for space system software resiliency," *Inf. Softw. Technol.*, vol. 94, pp. 150–164, Feb. 2018, doi: 10.1016/j.infsof.2017.10.006.
[94] J. Suárez-Varela and P. Barlet-Ros, "Flow monitoring in software-defined networks: Finding the accuracy/performance tradeoffs," *Comput. Netw.*, vol. 135, pp. 289–301, Apr. 2018.
[95] J. Cleland-Huang, M. Mirakhorli, A. Czauderna, and M. Wieloch, "Decision-centric traceability of architectural concerns," in *Proc. 7th Int. Workshop Traceability Emerg. Forms Softw. Eng. (TEFSE)*, May 2013, pp. 5–11.
[96] T. Spijkman, S. Brinkkemper, F. Dalpiaz, A.-F. Hemmer, and R. van de Bospoort, "Specification of requirements and software architecture for the customisation of enterprise software: A multi-case study based on the RE4SA model," in *Proc. IEEE 27th Int. Requirements Eng. Conf. Workshops (REW)*, Sep. 2019, pp. 64–73, doi: 10.1109/REW.2019.00015.
[97] M. Ristin, D. F. Edvardsen, and H. W. van de Venn, "RASAECO: Requirements analysis of software for the AECO industry," in *Proc. IEEE 29th Int. Requirements Eng. Conf. (RE)*, Sep. 2021, pp. 280–290, doi: 10.1109/RE51729.2021.00032.
[98] D. Yong, "Design and practice of software architecture in agile development," in *Proc. J. Phys., Conf.*, Jan. 2021, vol. 2074, no. 1, Art. no. 012008, doi: 10.1088/1742-6596/2074/1/012008.
[99] T. Spijkman, S. Molenaar, F. Dalpiaz, and S. Brinkkemper, "Alignment and granularity of requirements and architecture in agile development: A functional perspective," *Inf. Softw. Technol.*, vol. 133, May 2021, Art. no. 106535, doi: 10.1016/j.infsof.2021.106535.
[100] J. Cleland-Huang, R. S. Hanmer, S. Supakkul, and M. Mirakhorli, "The twin peaks of requirements and architecture," *IEEE Softw.*, vol. 30, no. 2, pp. 24–29, Mar. 2013, doi: 10.1109/MS.2013.39.
[101] M. Galster, M. Mirakhorli, J. Cleland-Huang, J. E. Burge, X. Franch, R. Roshandel, and P. Avgeriou, "Views on software engineering from the twin peaks of requirements and architecture," *ACM SIGSOFT Softw. Eng. Notes*, vol. 38, no. 5, pp. 40–42, Aug. 2013, doi: 10.1145/2507288.2507323.
[102] S. Dasanayake, S. Aaramaa, J. Markkula, and M. Oivo, "Impact of requirements volatility on software architecture: How do software teams keep up with ever-changing requirements?" *J. Softw., Evol. Process*, vol. 31, no. 6, p. e2160, 2019, doi: 10.1002/smr.2160.

[103] P. Lenberg, R. Feldt, and L. G. Wallgren, "Human factors related challenges in software engineering–An industrial perspective," in *Proc. IEEE/ACM 8th Int. Workshop Cooperat. Hum. Aspects Softw. Eng.*, May 2015, pp. 43–49.

[104] V. Vyatkin, "Software engineering in industrial automation: State-of-the-art review," *IEEE Trans. Ind. Informat.*, vol. 9, no. 3, pp. 1234–1249, Aug. 2013.

[105] C. Wohlin and A. Rainer, "Challenges and recommendations to publishing and using credible evidence in software engineering," *Inf. Softw. Technol.*, vol. 134, Jun. 2021, Art. no. 106555.

[106] E.-M. Arvanitou, A. Ampatzoglou, A. Chatzigeorgiou, and J. C. Carver, "Software engineering practices for scientific software development: A systematic mapping study," *J. Syst. Softw.*, vol. 172, Feb. 2021, Art. no. 110848.

[107] H.-M. Heyn, E. Knauss, A. P. Muhammad, O. Eriksson, J. Linder, P. Subbiah, S. K. Pradhan, and S. Tungal, "Requirement engineering challenges for AI-intense systems development," in *Proc. IEEE/ACM 1st Workshop AI Eng.-Softw. Eng. AI (WAIN)*, May 2021, pp. 89–96.

[108] N. Ali and R. Lai, "Global software development: A review OF its practices," *Malaysian J. Comput. Sci.*, vol. 34, no. 1, pp. 82–129, 2021.

[109] S. Qureshi, S. U. R. Khan, and J. Iqbal, "A study on mitigating the communication and coordination challenges during requirements change management in global software development," *IEEE Access*, vol. 9, pp. 88217–88242, 2021.

[110] H. U. Khan, M. Niazi, M. El-Attar, N. Ikram, S. U. Khan, and A. Q. Gill, "Empirical investigation of critical requirements engineering practices for global software development," *IEEE Access*, vol. 9, pp. 93593–93613, 2021.

[111] S. D. Hashmi, K. Shahzad, and M. Izhar, "Proposing total quality management as a buffer between global software development challenges and project success," *TQM J.*, Sep. 2021, doi: 10.1108/TQM-08-2020-0192.

[112] I. Atoum, M. K. Baklizi, I. Alsmadi, A. A. Otoom, T. Alhersh, J. Ababneh, J. Almalki, and S. M. Alshahrani, "Challenges of software requirements quality assurance and validation: A systematic literature review," *IEEE Access*, vol. 9, pp. 137613–137634, 2021.

[113] K. Coelho and T. Batista, "From requirements to architecture for software product lines," in *Proc. 9th Work. IEEE/IFIP Conf. Softw. Archit.*, Jun. 2011, pp. 282–289.

[114] N. A. Ernst, I. Ozkaya, R. L. Nord, J. Delange, S. Bellomo, and I. Gorton, "Understanding the role of constraints on architecturally significant requirements," in *Proc. 3rd Int. Workshop Twin Peaks Requirements Archit. (TwinPeaks)*, Jul. 2013, pp. 9–14.

[115] F. Silva, M. Lucena, and L. Lucena, "STREAM-AP: A process to systematize architectural patterns choice based on NFR," in *Proc. 3rd Int. Workshop Twin Peaks Requirements Archit. (TwinPeaks)*, Jul. 2013, pp. 27–34.

[116] H. Gomaa, "Evolving software requirements and architectures using software product line concepts," in *Proc. 2nd Int. Workshop Twin Peaks Requirements Archit. (TwinPeaks)*, May 2013, pp. 24–28.

[117] P. R. Anish and B. Balasubramaniam, "A knowledge-assisted framework to bridge functional and architecturally significant requirements," in *Proc. 4th Int. Workshop Twin Peaks Requirements Archit. (TwinPeaks)*, 2014, pp. 14–17.

[118] N. Rungta, O. Tkachuk, S. Person, and J. Biatek, "Helping system engineers bridge the peaks," presented at the 4th Int. Workshop Twin Peaks Requirements Archit., Hyderabad, India, 2014, doi: 10.1145/2593861.2593863.

[119] M. D. P. Salas-Zárate, G. Alor-Hernández, R. Valencia-García, L. Rodríguez-Mazahua, A. Rodríguez-González, and J. L. López Cuadrado, "Analyzing best practices on web development frameworks: The lift approach," *Sci. Comput. Program.*, vol. 102, pp. 1–19, May 2015.

[120] Z. Durdik, "An architecture-centric approach for goal-driven requirements elicitation," in *Proc. 19th ACM SIGSOFT Symp. 13th Eur. Conf. Found. Softw. Eng. (SIGSOFT/FSE)*, 2011, pp. 384–387.

[121] A. Goknil, I. Kurtev, and K. V. D. Berg, "Tool support for generation and validation of traces between requirements and architecture," presented at the 6th ECMFA Traceability Workshop, Paris, France, 2010, doi: 10.1145/1814392.1814398.

[122] F. Al-Hamed, S. Al-Doweesh, R. Al-Omar, W. Al-Doweesh, and A. Najjar, "Service oriented software engineering (SOSE): A survey and gap analysis," in *Proc. 21st Saudi Comput. Soc. Nat. Comput. Conf. (NCC)*, Apr. 2018, pp. 1–6.

[123] M. Galster and A. Eberlein, "Facilitating software architecting by ranking requirements based on their impact on the architecture process," in *Proc. 18th IEEE Int. Conf. Workshops Eng. Comput.-Based Syst.*, Apr. 2011, pp. 232–240.

[124] M. Galster, A. Eberlein, and L. Jiang, "Structuring software requirements for architecture design," in *Proc. 20th IEEE Int. Conf. Workshops Eng. Comput. Based Syst. (ECBS)*, Apr. 2013, pp. 119–128, doi: 10.1109/ECBS.2013.14.

[125] B. Surakratanasakul and K. Hamamoto, "CommonKADS's knowledge model using UML architectural view and extension mechanism," in *Proc. 7th Int. Conf. Adv. Inf. Manage. Service (ICIPM)*, Nov./Dec. 2011, pp. 59–63.

[126] A. Alebrahim, D. Hatebur, and M. Heisel, "A method to derive software architectures from quality requirements," in *Proc. 18th Asia–Pacific Softw. Eng. Conf.*, Dec. 2011, pp. 322–330.

[127] L. Shen, X. Peng, and W. Zhao, "Quality-driven self-adaptation: Bridging the gap between requirements and runtime architecture by design decision," in *Proc. IEEE 36th Annu. Comput. Softw. Appl. Conf.*, Jul. 2012, pp. 185–194.

[128] Z. Yao-Wen and Z. Mao-Lin, "A approach about translating from requirement model to AADL software architecture," in *Proc. Int. Conf. Inf., Netw. Autom. (ICINA)*, Oct. 2010, p. 275.

[129] J. Cleland-Huang, "Thinking about quoins: Strategic traceability of architecturally significant requirements," *IEEE Softw.*, vol. 30, no. 5, pp. 16–18, Sep. 2013, doi: 10.1109/MS.2013.117.

[130] W. Hong, "Architecture-centric software process for pattern based software reuse," in *Proc. WRI World Congr. Softw. Eng.*, vol. 4, May 2009, pp. 95–99, doi: 10.1109/WCSE.2009.188.

[131] G. Buchgeher and R. Weinreich, "Tool demonstration: A toolkit for architecture-centric software development," in *Proc. 8th Int. Conf. Princ. Pract. Program. Java*, vol. 2010, pp. 158–161.

[132] H. Ossher, R. Bellamy, I. Simmonds, and D. Amid, "Flexible modeling tools for pre-requirements analysis: Conceptual architecture and research challenges," presented at the ACM Int. Conf. Object Oriented Program. Syst. Lang. Appl., Reno/Tahoe, NV, USA, 2010, doi: 10.1145/1869459.1869529.

[133] M. Mirakhorli, "Tracing architecturally significant requirements: A decision-centric approach," in *Proc. 33rd Int. Conf. Softw. Eng.*, 2011, pp. 1126–1127.

[134] J. Davies, D. Milward, C.-W. Wang, and J. Welch, "Formal model-driven engineering of critical information systems," *Sci. Comput. Program.*, vol. 103, pp. 88–113, Jun. 2015.

[135] G. Berry and G. Gonthier, "The esterel synchronous programming language: Design, semantics, implementation," *Sci. Comput. Program.*, vol. 19, no. 2, pp. 87–152, Nov. 1992.

[136] M. Gogolla, F. Büttner, and M. Richters, "USE: A UML-based specification environment for validating UML and OCL," *Sci. Comput. Program.*, vol. 69, nos. 1–3, pp. 27–34, Dec. 2007.

[137] C. Tibermacine, S. Sadou, M. T. Ton That, and C. Dony, "Software architecture constraint reuse-by-composition," *Future Gener. Comput. Syst.*, vol. 61, pp. 37–53, Aug. 2016.

[138] A. Cilardo, L. Gallo, and N. Mazzocca, "Design space exploration for high-level synthesis of multi-threaded applications," *J. Syst. Archit.*, vol. 59, no. 10, pp. 1171–1183, Nov. 2013.

[139] M. Sojka, P. Píša, D. Faggioli, T. Cucinotta, F. Checconi, Z. Hanzálek, and G. Lipari, "Modular software architecture for flexible reservation mechanisms on heterogeneous resources," *J. Syst. Archit.*, vol. 57, no. 4, pp. 366–382, Apr. 2011.

[140] C. Yang, P. Liang, and P. Avgeriou, "A systematic mapping study on the combination of software architecture and agile development," *J. Syst. Softw.*, vol. 111, pp. 157–184, Jan. 2016.

[141] W. Brace and V. Cheutet, "A framework to support requirements analysis in engineering design," *J. Eng. Design*, vol. 23, no. 12, pp. 876–904, Dec. 2012.

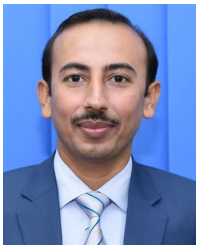
[142] T. Kobayashi, K. Arai, T. Imai, S. Tanimoto, H. Sato, A. Kanai, T. Miyazaki, and A. Tsujino, "Service-oriented software design model for communication robot," in *Proc. IEEE Int. Conf. Service Oriented Syst. Eng. (SOSE)*, Aug. 2020, pp. 31–39.

[143] E. Berrio-Charry, J. Vergara-Vargas, and H. Umana-Acosta, "A component-based evolution model for service-based software architectures," in *Proc. IEEE 11th Int. Conf. Softw. Eng. Service Sci. (ICSESS)*, Oct. 2020, pp. 111–115.

[144] L. Ben Othmane and M. Lamm, "Mindset for software architecture students," in *Proc. IEEE 43rd Annu. Comput. Softw. Appl. Conf. (COMPSAC)*, Jul. 2019, pp. 306–311.
[145] Y. Cai, L. Xiao, R. Kazman, R. Mo, and Q. Feng, "Design rule spaces: A new model for representing and analyzing software architecture," *IEEE Trans. Softw. Eng.*, vol. 45, no. 7, pp. 657–682, Jul. 2019.
[146] J. Y. Bang, Y. Brun, and N. Medvidovic, "Collaborative-design conflicts: Costs and solutions," *IEEE Softw.*, vol. 35, no. 6, pp. 25–31, Nov. 2018.
[147] G. Srivastava, Y. More, and J. Sam, "Effort estimation model for an enterprise software upgrade," in *Proc. Int. Conf. Emerg. Technol. (INCET)*, Jun. 2020, pp. 1–6.
[148] A. Aldea, M.-E. Iacob, A. Wombacher, M. Hiralal, and T. Franck, "Enterprise architecture 4.0–A vision, an approach and software tool support," in *Proc. IEEE 22nd Int. Enterprise Distrib. Object Comput. Conf. (EDOC)*, 2018, pp. 1–10.
[149] T. Farooqui, T. Rana, and F. Jafari, "Impact of human-centered design process (HCDP) on software development process," in *Proc. 2nd Int. Conf. Commun., Comput. Digit. Syst. (C-CODE)*, Mar. 2019, pp. 110–114.
[150] I. Signoretti, S. Marczak, L. Salerno, A. D. Lara, and R. Bastos, "Boosting agile by using user-centered design and lean startup: A case study of the adoption of the combined approach in software development," in *Proc. ACM/IEEE Int. Symp. Empirical Softw. Eng. Meas. (ESEM)*, Sep. 2019, pp. 1–6.
[151] M. D. Dzulfiqar, D. Khairani, and L. K. Wardhani, "The development of university website using user centered design method with ISO 9126 standard," in *Proc. 6th Int. Conf. Cyber IT Service Manage. (CITSM)*, Aug. 2018, pp. 1–4.
[152] E. Wulandari, V. Effendy, and G. A. Ary Wisudiawan, "Modeling user interface of first-aid application game using user centered design (UCD) method," in *Proc. 6th Int. Conf. Inf. Commun. Technol. (ICoICT)*, May 2018, pp. 354–359.
[153] M. Mukelabai, B. Behringer, M. Fey, J. Palz, J. Krüger, and T. Berger, "Multi-view editing of software product lines with PEoPL," in *Proc. 40th Int. Conf. Softw. Eng., Companion*, May 2018, pp. 81–84.
[154] B. Paech and K. Schneider, "How do users talk about software? Searching for common ground," in *Proc. 1st Workshop Ethics Requirements Eng. Res. Pract. (REthics)*, Aug. 2020, pp. 11–14.
[155] R. Chatterjee, A. Ahmed, and P. R. Anish, "Identification and classification of architecturally significant functional requirements," in *Proc. IEEE 7th Int. Workshop Artif. Intell. for Requirements Eng. (AIRE)*, Sep. 2020, pp. 9–17.
[156] M. Waterman, "Agility, risk, and uncertainty, part 2: How risk impacts agile architecture," *IEEE Softw.*, vol. 35, no. 3, pp. 18–19, May 2018.
[157] G. Abbas, M. Imran, Y. Hafeez, A. Kiani, A. Ali, and T. Jabbar, "Improving software architecture design decision by selecting set of solutions," in *Proc. 3rd Int. Conf. Comput., Math. Eng. Technol. (iCoMET)*, Jan. 2020, pp. 1–7.
[158] A. Akhtar, Y. H. Motla, H. Aslam, and M. Jamal, "Role of requirement change in software architecture using twin peaks model," in *Proc. IEEE 5th Int. Conf. Softw. Eng. Service Sci.*, Jun. 2014, pp. 174–177.
[159] S. Ghaisas, "Traceability for a knowledge-driven software engineering," in *Proc. IEEE/ACM 10th Int. Symp. Softw. Syst. Traceability (SST)*, May 2019, p. 1.
[160] I. Rubasinghe, D. Meedeniya, and I. Perera, "Traceability management with impact analysis in DevOps based software development," in *Proc. Int. Conf. Adv. Comput., Commun. Informat. (ICACCI)*, Sep. 2018, pp. 1956–1962.
[161] T. Durschmid, E. Kang, and D. Garlan, "Trade-off-oriented development: Making quality attribute trade-offs first-class," in *Proc. IEEE/ACM 41st Int. Conf. Softw. Eng., New Ideas Emerg. Results (ICSE-NIER)*, May 2019, pp. 109–112.
[162] A. A. A. Saeed and S.-W. Lee, "Non-functional requirements trade-off in self-adaptive systems," in *Proc. 4th Int. Workshop Requirements Eng. Self-Adapt., Collaborative, Cyber Phys. Syst. (RESACS)*, Aug. 2018, pp. 9–15.
[163] E.-M. Arvanitou, A. Ampatzoglou, N. Nikolaidis, A.-A. Tzintzira, A. Ampatzoglou, and A. Chatzigeorgiou, "Investigating trade-offs between portability, performance and maintainability in exascale systems," in *Proc. 46th Euromicro Conf. Softw. Eng. Adv. Appl. (SEAA)*, Aug. 2020, pp. 59–63.
[164] R. Edwards and N. Bencomo, "DeSiRE: Further understanding nuances of degrees of satisfaction of non-functional requirements trade-off," presented at the 13th Int. Conf. Softw. Eng. Adapt. Self-Manag. Syst., Gothenburg, Sweden, 2018, doi: 10.1145/3194133.3194142.

MARYAM GILLANI received the B.S. degree in software engineering and computer sciences from the Forman Christian College (FCCU), Lahore, and the M.S. degree in computer software engineering from the NUST College of Electrical and Mechanical Engineering (CEME), Islamabad, Pakistan. She is currently pursuing the Ph.D. degree in machine learning and artificial neural networks with University College Dublin (UCD), Ireland. She worked as a Lecturer for two years in a public sector university and worked on data collection and communication protocols for VANETs, intelligent transport systems, and rapid software development.

HAFIZ ADNAN NIAZ received the B.S. degree in computer systems engineering from UCET-IUB and the M.S. degree in computer engineering from the NUST College of Electrical and Mechanical Engineering (CEME), Islamabad, Pakistan. He is currently pursuing the Ph.D. degree with University College Dublin (UCD), Ireland, and working on optimizing energy of communication for high-performance heterogeneous computing. He has two years of industrial experience as a System Engineer in which he got hands-on experience on hardware and software domains of various networking areas. He worked as a Lecturer for two years and explored research areas that include data collection and communication protocols for VANETs along with image processing and rapid software development.



ATA ULLAH received the B.S. and M.S. degrees in computer science from COMSATS University Islamabad (CUI), Islamabad, Pakistan, in 2005 and 2007, respectively, and the Ph.D. degree in computer science from IIUI, Pakistan, in 2016. From November 2017 to 2018, he was with the University of Science and Technology Beijing, China. In 2008, he joined the National University of Modern Languages (NUML), Islamabad, where he is currently working as an Associate Professor. He was awarded ICT funding for the development of various projects. He has published 60 papers in ISI indexed impact factor journals and international conferences. His research interests include WSN, the IoT, health-services, the IoV, NGN, VoIP, and their security solutions. He is a reviewer and a guest editor of journals and conference publications.