Transformation Requirement Pattern for Capturing Data-Intensive Applications Requirements

Renita Raymond[1] and Margret Anouncia Savarimuthu[2*]

[1]School of Computer Science and Engineering, Vellore Institute of Technology, Vellore, India.

[2*]School of Computer Science and Engineering, Vellore Institute of Technology, Vellore, India.
smargretanouncia@vit.ac.in

Received: August 06, 2022; Accepted: September 30, 2022; Published: November 30, 2022

Abstract

A business analyst typically encounters activities and features that they have previously worked on as they move from project to project. Interestingly, despite the fact that these tasks are identical, analysts typically begin to work on them from scratch. Recreating the wheel leads to fundamental problems like neglecting the flaws in requirement analysis. Therefore, the analyst must begin implementing the strategy of requirement reuse through requirements patterns to ensure higher quality requirements with far less ambiguous parts and in a brief time. The focus of building a requirement pattern for transformation requirements of software-intensive systems ensures that critical information is not overlooked. It also enhances the effectiveness of business analysts. Although there is a plethora of research on requirements patterns in the studies, the studies did not focus on the requirement patterns for software-intensive systems. This research paper focuses mainly on the interactions of data-intensive systems namely transition requirements and a Transformation Requirement Pattern (TFReqPat) template is generated to capture the transitions requirements. It also analyses the anatomy of TFReqPat which is a guideline for capturing the data-intensive applications transition requirements. As a case study, a requirement pattern catalog with a systematic example of requirements for a banking application is presented. Furthermore, an empirical investigation was conducted to study participant's perceptions of requirement patterns in general and in particular in order to evaluate the proposed TFReqPat. The statistical analysis reveals that the TFReqPat is more appropriate, efficient, and successful for software-intensive systems than the conventional reuse strategy.

Keywords: Transformation Requirement Pattern, Transittion Requirements, Requirement Reuse, Data-Intensive Systems, Requirement Engineering.

1 Introduction

The growing complexity of big data leads to frequent and unpredictable changes in businesses. The way products are developed and executed also changed since requirements are changed indefinitely to suit customers' needs [1]. Also, the percent of projects both business analysts and development teams work on in the industries have relatively similar functionalities, as outlined in the project scope. Reinventing the wheel for functionalities that are similar to different projects leads to numerous problems. At the

Journal of Internet Services and Information Security (JISIS), volume: 12, number: 4 (November), pp. 126-138
DOI: *10.58346/JISIS.2022.I4.009*

*Corresponding author: School of Computer Science and Engineering, Vellore Institute of Technology, Vellore, India.

most significant level of abstraction, reusing software artifacts increases productivity and reduces numerous problems caused by reinventing the wheel. An adequate, structured, and well-defined requirement design is required for data-intensive systems. Because of the progressive aspects of a data-intensive system, it is one of the most complicated issues to retain requirements up to date with the ongoing evolution [2, 3]. This study aims to look into requirement engineering activities concerning big data needs with the help of Requirement Patterns (RP), one of the proven reusable structures. One of the critical principles of Software Requirement Pattern (SRP) evolution is that numerous requirements are recurrent and pertain to a limited range of types while designing a system. It's a reasonable approximation of a requirement's specification. Business analysts may be steered away from an obvious way of specifying a requirement with the help of RP as it drives them to conceive about requirements alternatively, which is an essential thing in designing data-intensive systems. As a result, the requirements would be easier to comprehend for the development and testing teams. According to Stephen Withall [4], RP is a template for writing a specific sort of requirement. It outlines how to work on a specific requirement, how it should be formulated, and what aspects should be extensively worked on. It also provides references to alternative requirements that business analysts should consider. Based on the Requirement Pattern Template, this paper proposes a software requirement pattern TFReqPat framework for data-intensive application requirements, especially transition requirements otherwise called as transformation requirements. Transformation Requirements are defined as "functionalities that the system must exhibit to enable the enterprise's transformation from its contemporary state to a desired future form, but which will be obsolete once the transformation is accomplished." They are distinct from other sorts of requirements by their inherent transitory character and their inability to be created until both a current and emerging solution are defined." Regardless of how well the data-intensive systems are designed, stakeholders will be dissatisfied with the eventual result without considering transformation requirements. Therefore, this paper highlights the importance of transformation requirements and a requirement pattern template is exhibited as a case study, with a systematic illustration of requirements for a banking application in section 5.

There are several sorts of requirements that a project's stakeholders express. Even though most requirements are generally stated at the outset of an initiative or project, it is conceivable that new requirements may be identified throughout the project's later stages. In the case of data-intensive systems, requirements will almost certainly be raised continuously, although the majority of the significant requirements are stated early on. So, all diverse requirements must be grouped inside their respective categories to manage the requirements process efficiently. Understanding the various requirements becomes a natural and straightforward process when determining whether a requirement is and then specifying the type. Also, when eliciting requirements, analysts frequently emphasize functional and non-functional needs. Holding transformation requirements till while all other requirements have been met might potentially result in consequences. Not exploring TFReq (however transitory) in advance can lead to delays, uncertainties, and added expenses. “The transition of prior years’ data to the newer system to create evaluation reports” is an instance of a transformation requirement.

The major contribution of the proposed TFReqPat can be broken into the following items:

- It summarizes the revisions that must be made to people, processes, and technology per changes from the as-is to the to-be.
- It incorporates awareness-raising, learning, and training for the new technique workers must work, accounting for and highlighting the contrasts between the previous and current events.

- It specifies any reorganization, relocation, augmentation, or transformation of data and information away from the core structures and into new data homes.

The following is the rest of the paper: the related study on SRP is presented in Section 2. In Section 3, a framework of TFReqPat and guidelines to design TFReqPat is described; in Section 4, the impact of TFReqPat in data-intensive applications is described; in Section 5, a case study of a banking application is presented; in Section 6, the validation of TFReqPat and in Section 7 conclusion and future work are presented.

1 Related Works

This section describes the requirement pattern methodologies and its applicability in various domains. The success of the conceptual context is making SRP a prominent research subject in the Software Engineering (SE) community. RP is generic and so may be carried out and adapted to any field. Some professionals use SRP templates to focus on certain domains. In a specific scenario, others create a new catalog of requirements to tackle unique challenges. SRP may be leveraged even during RE's elicitation, analysis, validation, and specification phases [5].

The RP is primarily concerned with recording the gathered requirements to facilitate design and implementation. The main concern with adopting the RP is appropriate for the domain, rendering system design straightforward [6]. PABRE framework is a prominent requirement pattern framework that promotes requirement reuse via SRPs. For the construction of a requirement book based on the requirement pattern catalog, Renault [6, 7, and 8] created the PABRE approach, which is tailored to the specific requirements of IT enterprises. It was created based on the knowledge gained via practical experience to update the required pattern. Incorporated into the PABRE system is a meta-model that explains the concepts about a pattern, a technique for the elicitation and documentation process, a library of patterns, and a tool to assist in maintaining the catalog. The primary outcome of using PABRE is an SRS with requirements defined in natural language. The validation of the PABRE framework and the creation of a catalog for functional needs, on the other hand, are not taken into account still. A requirement pattern group is designed to reduce the recurrence of requirement patterns with similar characteristics. The terms "refers to" and "extends" describe the relations between requirement pattern groupings. Stephen [3] organized 37 patterns into eight categories as requirement groups based on their applicability like security, trust, reliability, etc.

One of the recognized criteria that are required in uncertain circumstances is trust. Hoffman et al. [9] created a standardized requirement template for specifying the trust needs of recommender systems. In this pattern, the antecedents of trust are discussed. Twenty SRPs were written in natural language and constructed on the sense of trust. This strategy allows the analyst to include trust requirements into system specifications, saving time and effort to compile a list of software requirements. However, because antecedents can often overlap, it was impossible to establish the source of the antecedent.

Not only are requirement patterns used to tackle trust concerns, but they are also utilized to overcome security issues that arise while designing products for various sectors. The time spent in the requirement elicitation step is substantially reduced while using the security requirement pattern. It decreases the probability of product failure, which is directly proportional to product quality. However, choosing the right pattern is the most challenging part of making this practical. Integration of security patterns in a system of patterns is a promising technique to address complicated security concerns. As a result, Christophe Feltus and Raul Mazo [10] suggested the COPERATE (Complicated sEcurity Requirements pAtTErns) paradigm for addressing complex security requirements, particularly in the sphere of cloud

computing. This architecture addresses a variety of security concerns, including availability, confidentiality, and privacy. However, this is only the first iteration of the framework's creation, and it is not yet complete and ready for market. Rocky Slavin et al. [11] suggested a new method that blends an inquiry-cycle methodology with feature diagram notation. The hierarchical structure cuts down on the time and effort required to choose a suitable security-related requirement pattern for a complicated system. In the context of social networking platforms, privacy and trust are critical concerns. As a result, the I* framework modeling language is used to represent the patterns of online social networks, preventing the omission of critical requirements. The recurrent features are discovered first in this model, followed by the requirement patterns that correlate to these features. This framework is used to model ideas during the requirement elicitation phase and explicitly define the elements included in RE for social networks [12].

The Software Development Life Cycle (SDLC) for Multicore systems varies from conventional sequential programming. Hardware characteristics limit software functions. To obtain the complete system specification, must include the requirements specification document's hardware features and limits. Multicore systems will be successful as a result of this. So, the requirement patterns for multicore applications' performance and scheduling have been discovered. The requirement pattern for concurrency and decomposition should be discovered in the future [13]. People are becoming increasingly reliant on mobile devices in their daily lives. As a result, ensuring user privacy has become a top priority for current mobile operating systems. As a consequence of their empirical investigation, Xiao Xuan et al. [14] proposed seven privacy-based requirement patterns. There are two stages to this strategy. The privacy needs for mobile operating systems were elicited in the first step. Later in the second phase, based on the requirement pattern template, privacy patterns related to the privacy requirements were established. Finally, a skilled and knowledgeable business analyst refines these privacy need patterns in mobile operating systems to improve their correctness. Scenario-based requirements engineering ensures the quality of requirements specifications by resolving the response synchronization of such systems. Markus et al. [15] proposed a requirement pattern catalog for requirements modeling language. The requirement pattern library combines and integrates requirement patterns from three well-known requirement pattern libraries, focusing on chronological succession, real-time, and safety. Because the three underlying requirement pattern catalogs were derived from real-world requirements specifications, the requirement pattern catalog is practice-oriented. Some researchers emphasize on requirement patterns based on specific sorts of requirements. For instance, Franch et al. [16, 17, and 18] presented a catalog of RP for functional, non – functional, and non – technical requirements. Betty et al. [19] identified RP for embedded systems. Even a small number of typically used requirements patterns, as per the author, can significantly facilitate novices in formulating reasonably comprehensive embedded systems. It also enables reuse in terms of usability and programming. The use of requirements patterns allows for more uniform development of system specifications, making them easier to comprehend and maintain. As software-intensive systems get more complicated, the size of their requirement specifications expands. To save time and not reinvent the wheel, RP comes in handy. In order to reuse, one must first understand what exists.

At the moment, requirements reuse occurs only at the most fundamental levels [20]. In 2019, Wasim et al. [21] listed out the requirement engineering challenges faced by the large-scale distributed systems. Few of the major challenges are a lack of a clear vision of the requirements early in the development cycle and an inability to focus on the larger picture, difficulties with the process of requirement refinement and agile lacks a significant amount of high-level requirements management. The reason for the lack of awareness of requirement patterns is due to the lack of knowledge about the requirement standards. According to Franch et al. [22], the amount of knowledge with RE-related standards is lesser

than one might anticipate. This lack of understanding may impair practitioners' ability to make knowledgeable judgments about their projects, such as whether to adopt a certain approach, adhere to a particular template, or use the appropriate language. The most significant issue in a fast-developing IT system is the capacity to immediately monitor the impact of software on the environment and to reconcile those insights to stakeholders' demands and aspirations [23]. In 2021, Tukur et al. [24] addresses the RE challenges on the perspective of academic and industry by conducting a study. Out of the 27 challenges listed out 5 challenges namely customer's lack of comprehension of system needs [25-27], adaptation and development of requirements [28-31], lack of sufficient domain knowledge and expertise by software engineers [32], inadequate communication link [33-34] and incomplete requirements [34-36] are the most frequent challenge experience by the practitioners. Data-intensive applications deal with many requirements within the same software application. Customers and end-users are increasingly trying to find the most up-to-date, quick, and efficient interfaces, applications to address their social constraints. Developers need not start from scratch; new products can be derived from family requirements with minimal revisions to address their social constraints with the help of the reuse technique. According to Palomares et al. [37], the notion of requirement reuse is to adapt requirements written for prior projects and reuse them in a new one. Requirement Reuse (RR) makes it possible to produce solutions systematically by minimizing the cost and frequency of errors. RP is one of the proven reuse approaches. RP represented in the template ensures a consistent structure for improving RR. They assist by outlining the information to be included and flagging the possible risks in using the pattern catalog. In Data – Intensive systems, transformation requirements play a significant role. It can be inferred that none of the background work carried out so far has concentrated on developing RP for the interactive needs (transformation requirements) of data-intensive systems, which is a pivotal point in designing such systems. As a result, the research's novelty is forming a Requirement Pattern Template for dynamic requirements, notably Transformation Requirements (TFReq). Also, TFReqPat addresses all the major requirement challenges faced by academic and industry experts mentioned by Tukur et al. [24] in the year 2021.

2 Software Requirement Pattern for Transformation Requirements – TFREQ Pat

RePa is an International Workshop on Requirement Patterns devised to offer a uniform framework for describing requirements [38, 39]. The anatomy of RP comprises pattern author, related patterns, applicability, solutions, and so on in a logically organized way [4]. In this section, the structure of TFReqPat is described. The structure of the TFReq pattern is customizable. A template for transformation requirements is proposed on the extension of Withall’s SRP template [4].

Table 1: TFReqPat Template

Section		Description
Basic Details	**Pattern Manifest - ation**	Pattern Version Number – TFReq V.1 Requirements Specification Language – English Customer Organization – Industries
	Belongs to Domain	Data-Intensive Applications
	Related Patterns	Fundamental and Information Requirement Pattern
	Anticipat - ed Frequen - cy	Smaller systems have minimal requirements, while sophisticated systems may have a dozen or more.
	Pattern Classific - ation	Affects database, yes
Applicability		By exchanging messages, the TFReqPat may specify the interactions, which implies that when one of the stakeholders provides input, the requirements transform into another.
Template		It serves as a platform for the formulation of transformation requirements. Initially, requirements are gathered and formalized based on the recommendations of the requirement and business analyst. TFReq are one of the types of interactive requirements identified. Prioritize the set of interactive requirements gathered that might impede the development of data-intensive systems. Finally, generate a comprehensive list of transformation requirements elucidating its functionalities. According to Geoffrey et al. [40], TFReq needs to support diversified compute-intensive, analytic processing, and machine learning techniques, batch and real-time analytic processing.
Examples		Banking Application, Health Care, Recommendation Systems, E-Commerce
Considerations for Development		Always comply with the standards of never changing the transformations (request, action, response) after the appropriate action and keep a record of the processing information. The features of real-world elements that are significant for the application being built and the functionality executing on them should be reflected in the software. The attributes of big data must be appropriate in requirements descriptors when eliciting behavioral instances of desirable system reactions, so solution design may be built to fulfill the specifications [41]. Issues such as data replication, scalability, inconsistency, should be acknowledged while operating with dynamic data streams. Unlike conventional software applications, developers require a wide range of professional knowledge, skills, and competencies [42]. Collaborative knowledge acquisition is needed. Find insights and information from massive data and incorporate it in organizational processes to develop new technological solutions [43]. Data must be well–formulated and aggregated. When processing massive amounts of data, response time must be ensured. It is essential to facilitate the activities of the processing and information units [44]. In order to fulfill quality requirements, designs must be consistent across software, data, and deployment architectures. Visual analytics can be applied for knowledge discovery. The developers must correlate the software requirements and corporate goals [45]. A clear distinction between the former and conventional projects should be made in various aspects [46]. Quality attributes should be connected with big data characteristics [47].
Considerations for Testing		Identify all of the significant variations that a complex and challenging transaction might have. Testers should generate a vast set of test cases to cover all possible outcomes. Transition times should be given special consideration. The recording of system actions offers relevant details about what transpired in the system. Create test scenarios to identify technical glitches caused by properly structured but incorrect data. The test data provided for testing should be actual and application-specific. Ensure data quality by checking the data semantics [48]. Write test cases to model the domain of data interactions based on the dependence rules [49].

To capture the transformation requirements for data-intensive applications in an effective and efficient manner, the guidelines in Table 1 should be consistently applied. The template is divided into eight segments, commencing with the core of the proposed pattern and concluding with recommendations for testers and developers, as well as some instances of how to capture the transformation requirements in the first place.

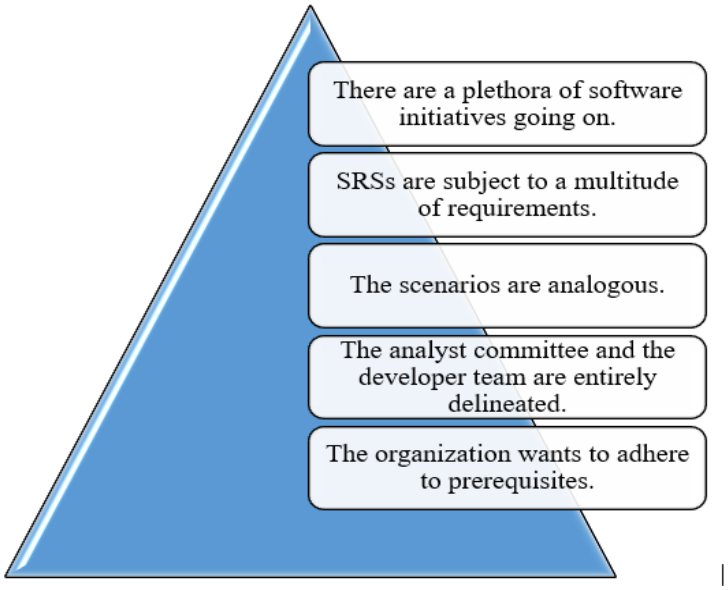

Figure 1: Applicability of TFReqPat

Figure 1 depicts the applicability of the proposed TFReqPat. When there is a myriad of software initiatives underway and SRSs are subject to a diverse set of requirements, the pattern can be adopted.

TFReq Pattern has a more significant impact in each requirement engineering phase: analysis, validation, and specification and is illustrated in Figure 2. Despite of how efficiently data-intensive systems are designed, users will be unsatisfied with the end result if transformation requirements are not addressed, as they have a significant effect on the real-world scenarios. Figure 3 illustrates the proposed pattern's strengths.

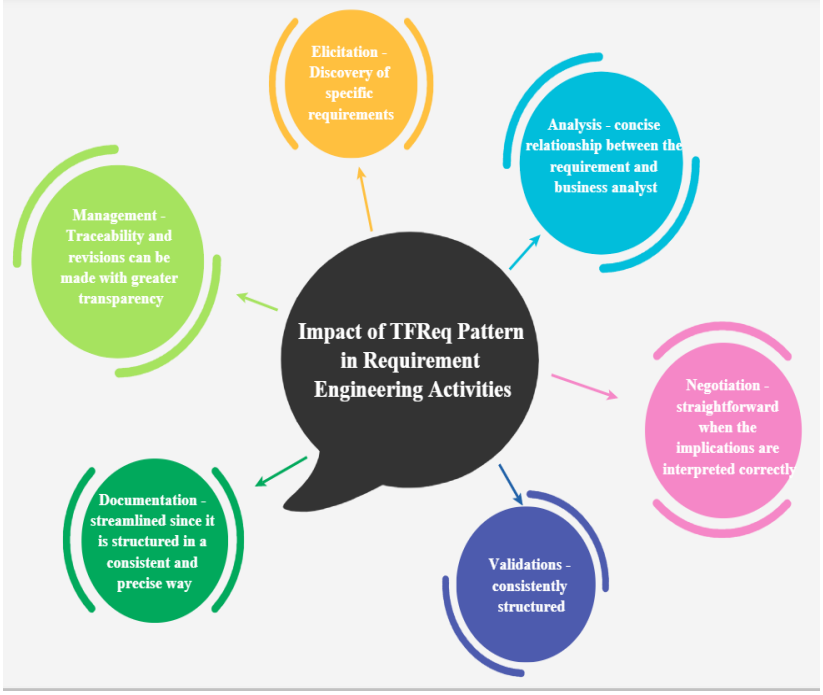

Figure 2: Impact of Pattern in RE Activities

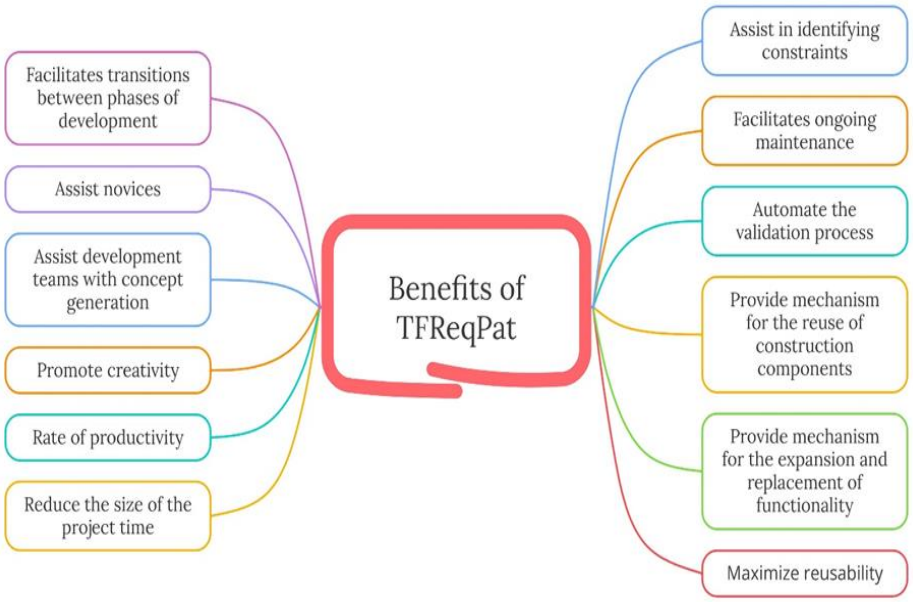

Figure 3: Potential Benefits of TFReqPat

3 Validation of TFREQ Pattern

An expert validates the TFReq pattern. This section presents the results of an expert survey conducted to validate a suggested structured requirement pattern for transformation needs (TFReq). Forty professionals in requirements engineering (RE), data-intensive systems, and application development participated in the survey and provided their perspectives on the TFReqPat framework. The survey's findings indicate that the TFReq technique is appropriate and capable of significantly improving requirements engineering efforts for data-intensive systems, particularly transformation requirements.

3.1. Research Objectives

- To determine the appropriateness and applicability of TFReqPat in data-intensive system requirements reuse.
- To assess the TFReq pattern's efficacy in reusing requirements in software-intensive systems.

3.2. Experimental Set – Up

The assessment instrument used for data collection is an online questionnaire created with Google Forms. The questionnaire was designed and divided into sections: Section A for analyst statistical profile, Section B for core validation queries, and Section C for miscellaneous validation questions. The questionnaire responses are quantified using a five-point Likert scale with the following values: "Strongly disagree = 1", "Disagree = 2", "Neutral = 3", "Agree = 4", and "Strongly agree = 5". The questionnaire consists of three main constructs: requirement reuse in data-intensive systems, TFReq pattern for data-intensive systems, and metamodeling of TFReq pattern. It also includes open-ended questions to elicit additional ideas, opinions, and guidelines for qualitative data analysis. The survey's quantitative results (based on close-ended questions) are evaluated using the statistical software tool SPSS Version 21.

3.3. Results

The questionnaire was subjected to a reliability test to determine the level of internal consistency using Cronbach's alpha measure. Cronbach Alpha is a popular technique for assessing reliability, particularly in recent years. According to Amirudin et al. [50], the correlation of variance and standard deviation can more accurately depict the Cronbach Alpha reliability coefficient than Range. The greater the Cronbach's Alpha score, the more reliable the instrument will be designed to quantify the construct. The value of Cronbach's alpha should be above 0.7 to be reliable [51]. The table 3 shows that the instrument is reliable as the value of α is 0.825. Table 4 gives the case processing summary of 40 respondents.

Table 3: Reliability Statistics

Cronbach's alpha	Cronbach's alpha based on standardized items	N of items
0.825	0.828	14

Table 4: Case Processing Summary

		N	%
Cases	Valid	40	100.0
	Excluded	0	0
	Total	40	100.0

Table 5: Descriptive Statistics for Questionnaire Items

	Valid	Missing	Mean	Median
Q1	40	0	4.03	4.5
Q2	40	0	4.14	4
Q3	40	0	4.01	4
Q4	40	0	4.72	4
Q5	40	0	3.89	4
Q6	40	0	4.16	4.5
Q7	40	0	4.68	4
Q8	40	0	4.32	4
Q9	40	0	4.29	4
Q10	40	0	4.5	4
Q11	40	0	4.58	4
Q12	40	0	3.67	4
Q13	40	0	4.42	4
Q14	40	0	4.65	4

The descriptive findings in table 5 pertain to the 14 questionnaire questions to which the 40 experts quantitatively replied using the five Likert scales ranging from strongly agree -5 to strongly disagree – 1. According to the Median scores in the table 5, the median value for all items varies between 4.00 and 5.00, equating to agree and strongly agree on the 5-Likert scale. It reveals that the 40 experts agreed on every single issue in the questionnaire. As a result, the TFReq Pattern framework is well-suited for use in the system design of data-intensive systems. Correspondingly, the TFReq Pattern paradigm may be significant for facilitating and accomplishing systematic requirement reuse in data-intensive systems. The questionnaire elicited responses from 40 experts like requirement analyst, business analyst, and project developer, revealing that 50% of professionals have more than ten years of experience in RE. It depicts that the experts' level of expertise instils trust in whatever viewpoint they present.

3.4. Threat to Validity

Internal and external validity are the primary threats to the experiment's validity. Internal validity is compromised since the conceptions utilized to design the questionnaire are not grounded on established

theories and tested models; thus, the instruments' validity may be compromised. However, the research makes use of previously published notions as characteristics that enable systematic requirement reuse. As a result, the questionnaire was created to elicit expert judgment about the validity of the TFReq Pattern framework. Additionally, the sample size of respondents may influence external validity regarding the instrument's reliability and the experiment's conclusion. Nonetheless, research indicates that even a small sample size of four can produce a satisfactory Cronbach's Alpha score [52]. The survey had a response rate of 40 and a total of 14 items, resulting in a Cronbach's Alpha value of 0.828, which is much more than the acceptable limit (.7).

4 Conclusion

The majority of projects on which the industry collaborates have similar features. Thus, rather of recreating the wheel, reusing requirements for projects using the requirement pattern approach minimizes budget overruns while maintaining the desired product quality. The proposed TFReqPat framework is one such framework that is more suitable, efficient, and effective than the usual reuse strategy for software-intensive systems. It contains recommendations for writing the requirements for all data-intensive systems' transformations. Cronbach's alpha of 0.825 demonstrates that the TFReqPat framework is capable of reliably generating suitable quality requirements for all data-intensive applications.

References

[1] Ya'u, B.I., Nordin, A., & Salleh, N. (2018). Meta-modeling constructs for requirements reuse (RR): software requirements patterns, variability and traceability. *Malaysian Journal of Computing (MJoC)*, *3*(2), 119-137.

[2] Wang, P., Tao, K., Gao, C., Ning, X., Gu, S., & Deng, B. (2017). Eliciting big data requirement from big data itself: A task-directed approach. *In IEEE 6th International Workshop on Software Mining (SoftwareMining)*, 17-23.

[3] Hummel, O., Eichelberger, H., Giloj, A., Werle, D., & Schmid, K. (2018). A collection of software engineering challenges for big data system development. *In IEEE 44th Euromicro Conference on Software Engineering and Advanced Applications (SEAA)*, 362-369.

[4] Withall, S. (2007). *Software requirement patterns*. Pearson Education.

[5] Srivastava, S. (2013). A repository of software requirement patterns for online examination system. *International Journal of Computer Science Issues (IJCSI)*, *10*(3), 247.

[6] Renault, S., Méndez-Bonilla, Ó., Franch, X., & Quer, C. (2009). PABRE: pattern-based requirements elicitation. *In IEEE Third International Conference on Research Challenges in Information Science*, 81-92.

[7] Franch, X., Palomares, C., Quer, C., Renault, S., & Lazzer, F.D. (2010). A metamodel for software requirement patterns. *In International Working Conference on Requirements Engineering: Foundation for Software Quality*, 85-90. Springer, Berlin, Heidelberg.

[8] Franch, X., Quer, C., Renault, S., Guerlain, C., & Palomares, C. (2013). Constructing and using software requirement patterns. In *Managing requirements knowledge*, Springer, Berlin, Heidelberg, 95-116.

[9] Hoffmann, A., Söllner, M., & Hoffmann, H. (2012). Twenty software requirement patterns to specify recommender systems that users will trust.

[10] Mazo, R., & Feltus, C. (2016). Framework for engineering complex security requirements patterns. *In IEEE 6th International Conference on IT Convergence and Security (ICITCS)*, 1-5.

[11] Slavin, R., Lehker, J.M., Niu, J., & Breaux, T.D. (2014). Managing security requirements patterns using feature diagram hierarchies. *In IEEE 22nd International Requirements Engineering Conference (RE)*, 193-202.

[12] Bouraga, S., Jureta, I., & Faulkner, S. (2014). Requirements engineering patterns for the modeling of online social networks features. *In IEEE 4th international workshop on requirements patterns (repa)*, 33-38.

[13] Mansoor, A., & Streitferdt, D. (2012). Requirement Patterns for Multicore Systems. *In IEEE 36th Annual Computer Software and Applications Conference*, 344-345.

[14] Xuan, X., Wang, Y., & Li, S. (2014). Privacy requirements patterns for mobile operating systems. In *2014 IEEE 4th International Workshop on Requirements Patterns (RePa)*, 39-42.

[15] Fockel, M., Holtmann, J., Koch, T., & Schmelter, D. (2018). Formal, Model-and Scenario-based Requirement Patterns. *In MODELSWARD*, 311-318.

[16] Palomares, C., Quer, C., Franch, X., Renault, S., & Guerlain, C. (2013). A catalogue of functional software requirement patterns for the domain of content management systems. *In Proceedings of the 28th annual acm symposium on applied computing*, 1260-1265.

[17] Renault, S., Méndez Bonilla, Ó., Franch Gutiérrez, J., & Quer, C. (2009). A Pattern-based Method for building Requirements Documents in Call-for-tender Processes. *International journal of computer science & applications*, *6*(5), 175-202.

[18] Palomares, C., Quer, C., Franch, X., Guerlain, C., & Renault, S. (2012). A catalogue of non-technical requirement patterns. *In Second IEEE International Workshop on Requirements Patterns (RePa)*, 1-6.

[19] Konrad, S., & Cheng, B.H. (2002). Requirements patterns for embedded systems. *In Proceedings IEEE Joint International Conference on Requirements Engineering*, 127-136.

[20] Kasauli, R., Knauss, E., Horkoff, J., Liebel, G., & de Oliveira Neto, F.G. (2021). Requirements engineering challenges and practices in large-scale agile system development. *Journal of Systems and Software*, *172*, 110851.

[21] Alsaqaf, W., Daneva, M., & Wieringa, R. (2019). Quality requirements challenges in the context of large-scale distributed agile: An empirical study. *Information and software technology*, *110*, 39-55.

[22] Franch, X., Glinz, M., Mendez, D., & Seyff, N. (2021). A study about the knowledge and use of requirements engineering standards in industry. *IEEE Transactions on Software Engineering*.

[23] Niu, N., Brinkkemper, S., Franch, X., Partanen, J., & Savolainen, J. (2018). Requirements engineering and continuous deployment. *IEEE software*, *35*(2), 86-90.

[24] Tukur, M., Umar, S., & Hassine, J. (2021). Requirement engineering challenges: A systematic mapping study on the academic and the industrial perspective. *Arabian Journal for Science and Engineering*, *46*(4), 3723-3748.

[25] Williams, I., & Yuan, X. (2019). Recommender Systems for Software Requirements Engineering: Current Research & Challenges. *SoutheastCon*, 1-6.

[26] Puarungroj, W., Boonsirisumpun, N., Phromkhot, S., & Puarungroj, N. (2018). Dealing with change in software development: a challenge for requirements engineering. *In IEEE 3rd Technology Innovation Management and Engineering Science International Conference (TIMES-iCON)*, 1-5.

[27] Nilsson, S., Sundin, E., & Lindahl, M. (2018). Integrated product service offerings–Challenges in setting requirements. *Journal of Cleaner Production*, *201*, 879-887.

[28] Hiisilä, H., Kauppinen, M., & Kujala, S. (2015). Challenges of the customer organization's requirements engineering process in the outsourced environment–a case study. *In International Working Conference on Requirements Engineering: Foundation for Software Quality*, 214-229. Springer, Cham.

[29] Besrour, S., Rahim, L.B.A., & Dominic, P.D.D. (2016). A quantitative study to identify critical requirement engineering challenges in the context of small and medium software enterprise.

In IEEE 3rd International Conference on Computer and Information Sciences (ICCOINS), 606-610.

[30] Schön, E.M., Winter, D., Escalona, M.J., & Thomaschewski, J. (2017). Key challenges in agile requirements engineering. *In International Conference on Agile Software Development*, 37-51. Springer, Cham.

[31] Bano Sahibzada, M., & Zowghi, D. (2012). Service oriented requirements engineering: practitioner's perspective. *In International Conference on Service-Oriented Computing*. Springer, Berlin, Heidelberg, 380-392.

[32] Shaheen, R., Ahsan, A., & Anwar, Z. (2018). Requirements management for market driven software products—key issues. *In IEEE International conference on computing, mathematics and engineering technologies (iCoMET)*, 1-6.

[33] Curcio, K., Navarro, T., Malucelli, A., & Reinehr, S. (2018). Requirements engineering: A systematic mapping study in agile software development. *Journal of Systems and Software*, *139*, 32-50.

[34] Rolland, K.H. (2015). 'Desperately 'seeking research on agile requirements in the context of large-scale agile projects. *In Scientific Workshop Proceedings of the XP2015*, 1-6.

[35] Preethi, P., & Asokan, R. (2019). An attempt to design improved and fool proof safe distribution of personal healthcare records for cloud computing. *Mobile Networks and Applications*, *24*(6), 1755-1762.

[36] Heikkilä, V.T., Damian, D., Lassenius, C., & Paasivaara, M. (2015). A mapping study on requirements engineering in agile software development. *In IEEE 41st Euromicro conference on software engineering and advanced applications*, 199-207.

[37] Preethi, P., & Asokan, R. (2021). Modelling LSUTE: PKE Schemes for Safeguarding Electronic Healthcare Records Over Cloud Communication Environment. *Wireless Personal Communications*, *117*(4), 2695-2711.

[38] Chung, L., Paech, B., Zhao, L., Liu, L., & Supakkul, S. (2012). Repa requirements pattern template. *In International Workshop on Requirements Patterns (RePa '12)*.

[39] Leite, J.C., Zhao, L., Kopczńska, S., Supakkul, S., & Chung, L. (2017). Report from the 6th International Workshop on Requirements Patterns (RePa'16). *ACM SIGSOFT Software Engineering Notes*, *42*(1), 32-33.

[40] Fox, G., & Chang, W. (2014). Big data use cases and requirements. *In 1st Big Data Interoperability Framework Workshop: Building Robust Big Data Ecosystem ISO/IEC JTC*, (1), 18-21.

[41] Madhavji, N.H., Miranskyy, A., & Kontogiannis, K. (2015). Big picture of big data software engineering: with example research challenges. *In IEEE/ACM 1st International Workshop on Big Data Software Engineering*, 11-14.

[42] Gurcan, F., & Cagiltay, N.E. (2019). Big data software engineering: Analysis of knowledge domains and skill sets using LDA-based topic modeling. *IEEE access*, *7*, 82541-82552.

[43] Lamba, H.S., & Dubey, S.K. (2015). Analysis of requirements for big data adoption to maximize IT business value. *In IEEE 4th International Conference on Reliability, Infocom Technologies and Optimization (ICRITO) (Trends and Future Directions)*, 1-6.

[44] Chen, C.P., & Zhang, C.Y. (2014). Data-intensive applications, challenges, techniques and technologies: A survey on Big Data. *Information sciences*, *275*, 314-347.

[45] Arruda, D. (2018). Requirements engineering in the context of big data applications. *ACM SIGSOFT Software Engineering Notes*, *43*(1), 1-6.

[46] Volk, M., Jamous, N., & Turowski, K. (2017). Ask the Right Questions: Requirements Engineering for the Execution of Big Data Projects. *In AMCIS*.

[47] Noorwali, I., Arruda, D., & Madhavji, N.H. (2016). Understanding quality requirements in the context of big data systems. *In Proceedings of the 2nd International Workshop on BIG Data Software Engineering*, 76-79.

[48] Felderer, M., Russo, B., & Auer, F. (2019). On testing data-intensive software systems. *In Security and Quality in Cyber-Physical Systems Engineering*, Springer, Cham, 129-148.

[49] Sen, S., Marijan, D., Ieva, C., Grime, A., & Sander, A. (2016). Modeling and verifying combinatorial interactions to test data intensive systems: Experience at the norwegian customs directorate. *IEEE Transactions on Reliability*, *66*(1), 3-16.

[50] Amirrudin, M., Nasution, K., & Supahar, S. (2021). Effect of variability on Cronbach alpha reliability in research practice. *Jurnal Matematika, Statistika dan Komputasi*, *17*(2), 223-230.

[51] Kongkaew, C., Scholfield, C.N., Supapaan, T., Mann, C., Mongkhon, P., & Chanunun, S. (2020). Impact of research-based learning on student knowledge and assessment in Pharmacoepidemiology: a one group pretest-posttest experimental study. *Thai Journal of Pharmaceutical Sciences (TJPS)*, *43*(4).

[52] Bujang, M.A., Omar, E.D., & Baharum, N.A. (2018). A review on sample size determination for Cronbach's alpha test: a simple guide for researchers. *The Malaysian journal of medical sciences: MJMS*, *25*(6), 85.

[53] Schmidbauer, T., & Wendzel, S. (2022). Detection Of Computational Intensive Reversible Covert Channels Based On Packet Runtime. *Journal of Wireless Mobile Networks, Ubiquitous Computing, and Dependable Applications, 13*(1), 137-166.