

Requirements Re-usability in Global Software Development: A Systematic Mapping Study

Syeda Sumbul Hossain[1(✉)], Yeasir Arafat[1], Tanvir Amin[2], and Touhid Bhuiyan[1]

[1] Daffodil International University, Dhaka, Bangladesh
{syeda.swe,yeasir35-1501}@diu.edu.bd, t.bhuiyan@daffodilvarsity.edu.bd
[2] Infolytx, Dhaka, Bangladesh
tanviramin@ymail.com

Abstract. In global software development, requirements re-usability is a common practice which ultimately helps to maintain project quality and reduce both development time and cost. However, when a large-scale project is distributed, there are some critical factors needed to be maintained and managed for reusing requirements and it is considered a challenging job to interrelate the requirements between two identical projects. In this study, we have pointed out 48 challenges faced and 43 mitigation techniques used when implementing requirements re-usability in global software development projects among distributed teams. The challenges distributed teams frequently encounter can be divided into three considering issues as Communication, Coordination and Control of distributed teams in global software development. The results from this study can be used to plan development strategies while reusing requirements in distributed manners.

Keywords: Requirement re-usability · Agile · Distributed software development · Large-scale · Global Software Development (GSD) · Systematic mapping study

1 Introduction

Software engineering is a knowledge heightened work and this knowledge makes the software industry decline their cost in both helping co-located and distributed projects [2]. Close to these, practitioners in like manner get different sorts of methodology to constrain the cost of distributed projects. Regardless, distributed development as frequently as conceivable experience customary issues, for instance, communication, coordination, and control. Notwithstanding, software organization can be benefited by blending both of the advantages of agile and distributed software development, for instance, visit up close and personal interest, powerful correspondence, and client joint effort [57,58,63].

© Springer Nature Switzerland AG 2020
O. Gervasi et al. (Eds.): ICCSA 2020, LNCS 12252, pp. 960–974, 2020.
https://doi.org/10.1007/978-3-030-58811-3_68

In software development, the concept of re-usability helps the practitioners to reduce both cost and time. Requirement re-usability deals with efficient reuse of requirements and different software artifacts which help to minimize development cost and time [29]. Reliability of any product can also be increased by successfully applying the knowledge of previous projects. For reusing any requirements or software artifacts, some factors like cost of reuse, usefulness and quality of the reusable components need to be considered [18]. It is encouraged in [65] for agile development to use of reusable artifacts (e.g. design frameworks, patterns) where development costs will be reduced and the quality will be increased. Requirements reuse in a distributed system is used to increase system productivity, reliability, quality, to decrease system development sprint and to maintain consistency between two identical systems. In fact, development teams reuse requirements to reduce bug or error, and to maintain efficiency in a system. Hence, when it is a large-scale and distributed system, there are some critical factors are needed to be maintained and managed.

In a prior work [29], we have used survey study to pointed out requirement re-usability challenges and mitigation approaches in global software development. Using the semi-structured interview technique, 14 challenges and 10 mitigation approaches were splitted into three categories in respect of communication, coordination, and control. From that we are motivated to identify the challenges and mitigation techniques are stated in literature. We were driven by the following research question:

– *What are the requirements re-usability challenges and mitigation techniques in distributed large-scale agile projects reported in the up-to-the-minute?*

The remaining sections of this paper is structured as follows. Related works are presented at Sect. 2. In Sect. 3, the overall research methodology is described which is followed by results and analysis in Sect. 4. Discussion and conclusion are furnished in Sect. 5 and Sect. 6 respectively.

2 Related Work

Most of the studies have been claimed that requirement re-usability is a challenging job and practitioners face lots of challenges to reuse requirements in distributed and large-scale agile software development projects [11,14,28,61]. Different literature also stated that, software organizations face lots of problem if any changes occur in customer requirements or in technology. These changes result increase of project budget, slowing down the productivity, customers' dissatisfaction and lacking in professional behaviour among development team [5,24,51,60]. Successfully adaptations of reusable artifacts is not also fitted in agile development due to poor documentation and management of knowledge, experience, decisions and poor architecture [10,24]. In a study [1], it is shown a technique how to refactor software codes to reduce time and accelerate development.

A survey on requirement reuse conducted by Chernak, Y. [9], defined that, some participants of the survey were explaining some contradictory factors for instance maintenance cost of the project, resemblance of applications, existing requirements' quality and structure, and the level of abstraction at the time of reusing requirements. Those challenges arise at the time of distributed agile software development alongside with the customers' dissatisfaction while reusing requirements from any preceding similar projects [27]. Moreover, in different literature authors reported that, reuse of requirements is challenging due to many unknown requirements, external and internal forces, chances to miss important requirements, incomplete understanding, inconsistency or fault at the moment of person to person communication, sharing resources or pair programming, psychological impediments and knowledge reuse [6,51,70].

To improve product characteristics through requirement reuse, some well-known mitigation approaches adopted by the practitioners to overcome these challenges. For example, *Software Product Line Engineering (SPLE)* approach use to reduce time-to-market at reusing software components for a similar project, and that also reduce reuse overhead ratio or ratio of the reused asset integration effort to total development assets [9,47,48]. However, Wehrwein, B et al.[69] also sorted out some efficient re-usability approaches such as reusing design pattern, software architecture for component based, aspect oriented or service oriented software development, application framework, legacy system wrapping, Commercial off-the-shelf (COTS) integration, program libraries. These approaches help to increase productivity and effectiveness of development team, and reduce operational cost.

3 Research Methodology

3.1 Systematic Mapping Study

In this research, we have used systematic mapping study [37] as a research methodology to identify the current status of evidence reported in our research area and find out the literature ratio in related field. Systematic mapping studies reduce biasness of research, and also mitigate time and cost. To perform a systematic mapping study, we adopted following steps mentioned in [30,54]. Figure 1 represents the overall research methodology we have followed in this work.

Step 1: Define Research Questions (Research Scope). The research question is mentioned as one of the key factors of conducting systematic mapping study in [8]. This is essential to point out the related works and evidence according to research goal. For conducting this research, we stipulated our research question (RQ).

Step 2: Formulate and Execute Search Strings (All Studies). The search strings were drawn up according to intervention and scope area. Search string is used to identify primary studies from different digital sources or databases. For

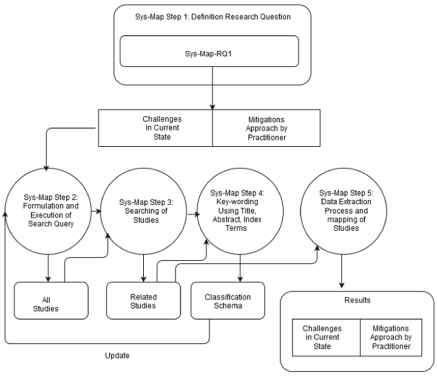

Fig. 1. Systematic mapping study

performing this step, first we have defined our search strategy to specify terms to be searched, searching process and resources where to search. Then we have formulate the search strings and execute those in different sources.

- Search Strategy
  - Searching terms were recognized by going through titles, abstracts and keywords of different studies.
  - Relevant papers were considered by forward and backward snowballing.
  - Lexicons are used to generate synonyms of keywords.
  - Boolean operators are used to formulate search strings.
- Search Strings
  - ("Requirements re-usability" OR "reusable requirements") AND ("Distributed" OR "dispersed") AND ("Agile" OR "scrum" OR "XP") AND ("Large-scale") AND ("Challenges" OR "Obstacles")
  - ("Requirements re-usability" OR "reusable requirements") AND ("Distributed" OR "dispersed") AND ("Agile" OR "scrum" OR "XP") AND ("Large-scale") AND ("Migitations" OR "Resolution")
- Scientific Databases
  We have chosen five different scientific data sources for executing the searching and set the time range between 1995 and 2020.
  - IEEE Xplore[1]
  - Association for Computing Machinery (ACM)[2]
  - Scopus[3]
  - Science Direct[4]
  - Springer[5]

[1] https://ieeexplore.ieee.org/Xplore/home.jsp.
[2] https://www.acm.org/.
[3] https://www.scopus.com/.
[4] https://www.sciencedirect.com/.
[5] https://www.springer.com/gp.

Step 3: Screen of Papers for Inclusion and Exclusion (Relevant Papers). Inclusion and exclusion criteria helps to include or exclude the studies according to relevant studies those relating with research area and question [52,54]. After applying inclusion and exclusion criterion at the end, we have finalized 52 papers out of 6832 papers in this research (See in Fig. 2). The following steps were followed to perform screening relevant papers.

- At first, we have picked only the papers which were written in English and only the peer-reviewed ones.
- For avoiding duplicate papers, we have used Mendeley[6]. It is one of mostly used online reference management systems.
- The researcher go through the abstracts and titles to identify relevant studies. In some cases, introduction and conclusion section are also considered for inclusion and exclusion process while abstracts and titles are not clear enough.
- We have excluded all the papers of which we have found no digital copies.

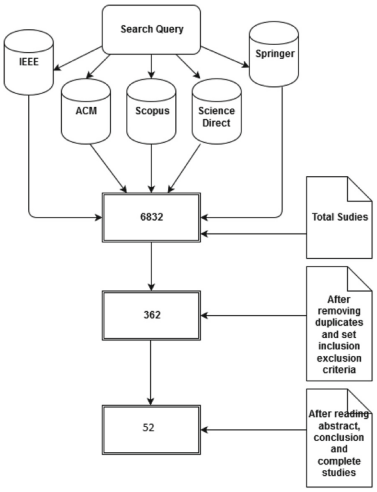

Fig. 2. Study results

Step 4: Develop Classification Scheme (Key-wording). Key-wording helps to achieve aim and objective in a targeted point [54]. We have used this step to get an overall high level idea of any relevant studies. Then we have categorized the relevant studies as per the nature and contribution.

[6] https://www.mendeley.com/.

Step 5: Extract Data and Map the Studies (Systematic Map). After development of classification scheme, all relevant studies are classified according to the classification scheme. Graph is used to generate report or visualized of mapping studies. After reading all the studies we identified challenges and mitigation approaches from different perspective and listed them. From listed findings, we removed the duplicate data and managed them by tables. Graphs and figures were used to distribute studies in different areas and sorted out publications year ratio.

3.2 Validity Threats

In this research, we have encountered different types of validity threats such as construct validity threat, internal validity threat, external validity threats and reliability as per the prior work [29]. We checked these threats throughout this research work. We have done systematic mapping study based on our research question and we formed search strings to find out the related studies from peer-reviewed database. Search strings were frequently reviewed and modified from the early phases of our research. Two researchers have done the searching query and inclusion-exclusion process in parallel, which minimized the risk of bias and lowered the internal validity threats. The presence of external validity threats is high as bias might be present in many literature. Two researchers work for finding all possible challenges and mitigation techniques, and then combined relevant and common ideas to achieve the final result. Our final result has been reviewed for several times. If any researchers search the queries according to our search string (within our searching time periods), will find the same data. This ensures the reliability of our research results.

4 Results and Analysis

In this section, we have reported the results of the systematic mapping study and also mentioned the analysis of studies in terms of the time distribution and categorizations based on our research theme.

4.1 Study Distribution

According to findings from the selected studies, we have observed, selected studies are distributed in the different sub-areas of requirement re-usability and most of the papers are published in requirements reuse and management area (See in Fig. 3).

In the below Fig. 4, we have represented the publications by year and we have seen that most of the papers published in between 2011 and 2020.

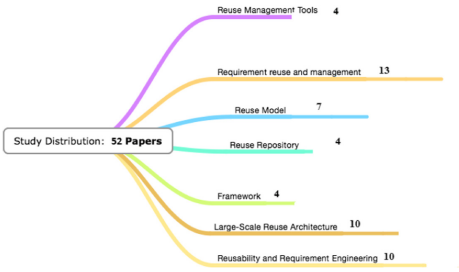

Fig. 3. Study distribution in different areas

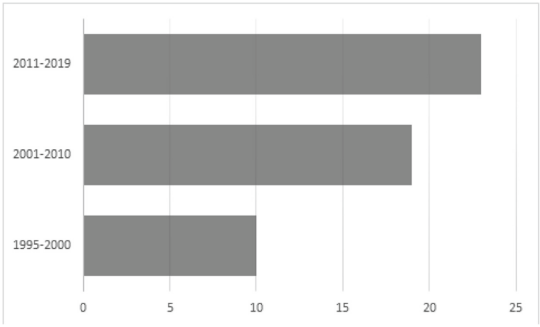

Fig. 4. Studies distribution over the publication years

4.2 Requirement Re-usability: Challenges

In Global Software Development (GSD), the development team confront some challenges like communication, coordination and control in managing the project as a consequence of distributed or dispersed nature of team structure. In this study, we have disjoint the findings of mapping studies into three grouping such as communication, coordination and control as per our prior work [29]. In Table 1 we have represent the challenges reported in the literature by categorize into three areas namely *Communication*, *Coordination*, and *Control*. As a whole, we have pointed out requirement re-usability challenges confronted by the practitioners in distributed large-scale agile projects. As we splitted the challenges into three sub-categories, we have observed that, distributed team struggling with control factor during requirement re-usability (See in Fig. 5, maximum challenges found in Control section).

4.3 Requirement Re-usability: Mitigations

We have also identified the mitigation techniques applied by practitioners that is found in literature in Table 2. Figure 6 depicts the number of mitigation techniques applied by the practitioners. Rather than the *communication* factor it

Table 1. Requirement re-usability challenges

Communication	Coordination	Control
Might miss important requirements [51]	Not sufficiently formalized [51]	Freezes the scope [51]
Incomplete understanding, Inconsistency or faulty between Team [5,51]	External and internal forces Often cause rapid changes Or growth of requirements [31,34,39,51]	Many unknowns Unclear vision of overall goal [6,51]
Communication gaps [6,16,17,34]	Tough job to pair programming and share resources [5] [62]	Trace-ability problem [3,27,34]
Low understanding the roles of others [6,9,12,17]	Dependency problem [3,4]	Organizational impediments [5,15]
Incompatible environments and conflicting assumptions [17]	Different identification [4]	Economic impediments[5]
Knowledge transfer [17]	Poor standard for reuse resources [4]	Administrative impediments[5]
Distributed access [12]	Trust issues [5]	Political impediments [5]
Reuse and corporate strategy [15]	Unstructured documentation [5,9]	Psychological impediments [5,17]
Distribution and collaboration [15]	Unclear requirements [6,9,17]	Continuous inflow of requirements [6]
Conflicts of interest [68]	User interface [12]	Low motivation to contribute at requested work [6]
	Incomplete requirements [9]	Overlapping requirements processes [6]
	Measurement and experimentation [15]	Keeping SRS updated [6]
	Common terminology [19]	Conflict between two system axioms [59]
	Volatile requirements [34]	Focus on process [15]
	Low architectural repository [1,36]	Lack of business analysis phase [27]
	Misunderstanding about reuse benefits [49]	Lack of traceability [27]
		Low coupling, high cohesion [32]
		Uncertain estimates [34]
		Architectural changes and code changes dependency [55]
		Informal model [44]
		Measurement quality of reusable requirement [50]
		Identify granularity [50]

seems that practitioners applying more mitigation techniques for both *coordination* and *control* factors during re-using requirement in a distributed large-scale agile project.

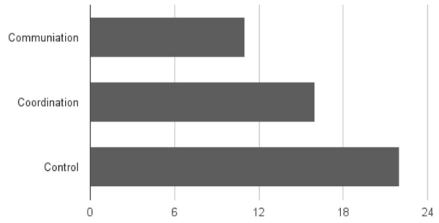

Fig. 5. Requirement re-usability challenges

Table 2. Mitigations techniques

Communication	Coordination	Control
Producer consumer model [51]	Test Driven Development [3,19,31,32,51]	Work flow control [17,51]
Cultural analysis [17]	Tailoring of transformation assets [3,12,13,31]	Platform Domain Analysis [3,5,12,13,20,27,32,33,59,64]
Face-to-face meeting [17]	Transformation composition [3,32]	Empirical knowledge analysis [4,7,17,67]
Local workshops [17]	Formal modeling approaches [7,15,19,25,32,39,44,51,59]	Theoretical knowledge analysis [3,4,17,41,47]
Distributed simulation approach [53]	Integrated knowledge utilization [4,5,12,55]	Schedule periodic check-ups[17]
	Semantic classification of experience element [12]	Use case driven approach [3,9,12,13]
	Packaging of analysis results [12,13,31,33,66]	Requirements interaction management [22,32,44,59,66]
	Shared repository for reusable requirements [10,12,13,17,32,36,68]	Multiple representations of experience elements [12,59]
	Reuse metrics [14,15,19,21,32,41,45,62]	Asset management units for project-related data [3,12]
	High-level reference model coordination [23,27]	Maintaining a project history [12,13]
	Evolutionary domain model [26]	Role Management [12]
	Pattern based software reuse [27,35]	Knowledge Based Tools [16,22,31,38,68]
	Architecture-centric software process [27,49]	Consistency Checker Tools [20,32]
	Requirement reasoning model [5,40]	Software Product Line Approach [4,15,44,46,64]
	Meta model to reengineered process [39,64,66,69]	Identifying commonalities and variations [39,64,66,68]
	Polymorphic reuse mechanism [42]	NFR or quality catalog [43]
	Incremental development approach [43]	Bottom-up Approach [64]
	Object oriented programming concept [56,60,64]	Prior in analysis phase [66]
	Variation points [68]	
	Linkage between product family [31]	

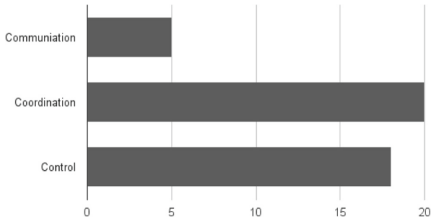

Fig. 6. Mitigations techniques

5 Discussion

Requirements re-usability is considered as an important technology [28], speeds up and aids in software development and maintenance processes. Other benefits of reusing requirements are increasing quality, reliability, efficiency, performance and productivity (faster, better, cheaper), minimizing developments and maintenance costs and time, reducing risk of project failures, reducing stress on technical people, and satisfying customers' needs [5,9,15,17,32,62] [14,44,60,64]. In a study, we have found that, one third errors of software occur from design and code, whereas, 40 to 60% of software errors occur from requirements errors [7].

Requirements reusing in agile software development, when it is distributed, needs supports like documentation, knowledge and risk management policies, experiences sharing, decisions, or architectural information [11,14,28]. In [13], re-using artifacts in software and product development domain is mentioned as a well-known concept. Project cost savings between 10 to 35%, in considered as well-planned reuse strategies in software development domain [14].

To build large and complex software as more reliable, less expensive and ready in internet time, reusing software artifacts can provide better way than traditional software engineering methods [15,39]. A literature through an experience mentioned that by reusing requirements defects reduced by 15%, and productivity increased by 57% [19]. Though requirements re-usability do not well fit in agile software development, however, modification and customization may be possible to adapt reusable artifacts successfully [60].

The motivation of the RQ was to find out the requirement re-usability challenges and mitigation techniques in distributed agile projects applied by the practitioners which are listed in the current-state-of-the-art. In response to RQ, through a systematic mapping study we have identified 48 challenges faced by the practitioners reported in different literature along with 43 mitigation techniques applied by the practitioners in large-scale distributed agile projects. We have divided literature findings into three categories which GSD teams frequently encounter. Identified challenges required to take into account during requirement re-usability in distributed large-scale agile projects which will help to reduce both time and cost. Test Driven Development (TDD), formal modeling, knowledge sharing and transfer, shared repository, Object Oriented Programming (OOP)

concepts, linkage between product family and so forth help team to mitigate stated challenges.

6 Conclusions

In software development, requirements re-usability helps the practitioners in reducing development time and cost. Requirement re-usability is re-using of requirements and other artifacts which helps to minimize development cost, time and increase reliability by applying knowledge effectively from previously learned lesson. Sustainable software development can be achieved by applying the concept of reusing requirements. In this research, we have find out requirement re-usability challenges and mitigation techniques reported in different literature. In future, it would be interesting to map the current findings with industrial practice by applying either case studies or survey from large-scale agile distributed projects.

References

1. Adorf, C.S., Ramasubramani, V., Anderson, J.A., Glotzer, S.C.: How to professionally develop reusable scientific software-and when not to. Comput. Sci. Eng. **21**(2), 66–79 (2018)
2. Aurum, A., Jeffery, R., Wohlin, C., Handzic, M.: Managing Software Engineering Knowledge. Springer, Heidelberg (2013). https://doi.org/10.1007/978-3-662-05129-0
3. Basso, F.P., Pillat, R.M., Oliveira, T.C., Becker, L.B.: Supporting large scale model transformation reuse. In: ACM SIGPLAN Notices, vol. 49, pp. 169–178. ACM (2013)
4. Berzisa, S., Grabis, J.: Knowledge reuse in configuration of project management information systems: a change management case study. In: 2011 15th IEEE International Conference on Intelligent Engineering Systems (INES), pp. 51–56. IEEE (2011)
5. Biddle, R., Martin, A., Noble, J.: No name: just notes on software reuse. ACM SIGPLAN Not. **38**(12), 76–96 (2003)
6. Bjarnason, E., Wnuk, K., Regnell, B.: Requirements are slipping through the gapsóa case study on causes & effects of communication gaps in large-scale software development. In: 2011 19th IEEE International Requirements Engineering Conference (RE), pp. 37–46. IEEE (2011)
7. Brown, S., Felsing, M.: Reuse based requirements development & management (RBRDM) (2005)
8. Budgen, D., Turner, M., Brereton, P., Kitchenham, B.: Using mapping studies in software engineering. In: Proceedings of PPIG, vol. 8, pp. 195–204 (2008)
9. Chernak, Y.: Requirements reuse: the state of the practice. In: 2012 IEEE International Conference on Software Science, Technology and Engineering (SWSTE), pp. 46–53. IEEE (2012)
10. Daneva, M., Van Der Veen, E., Amrit, C., Ghaisas, S., Sikkel, K., Kumar, R., Ajmeri, N., Ramteerthkar, U., Wieringa, R.: Agile requirements prioritization in large-scale outsourced system projects: an empirical study. J. Syst. Softw. **86**(5), 1333–1353 (2013)

11. Decker, B., et al.: A framework for agile reuse in software engineering using wiki technology. Wissensmanagement **2005**, 411–414 (2005)
12. Feldmann, R.L., Geppert, B., Mahnke, W., Ritter, N., Rossler, F.: An ORDBMS-based reuse repository supporting the quality improvement paradigm-exemplified by the SDL-pattern approach. In: 34th International Conference on Technology of Object-Oriented Languages and Systems, TOOLS 34. Proceedings, pp. 125–136. IEEE (2000)
13. Feldmann, R.L., Geppert, B., Rossler, F.: An integrating approach for developing distributed software systems-combining formal methods, software reuse, and the experience base. In: Fifth IEEE International Conference on Engineering of Complex Computer Systems, ICECCS 1999, pp. 54–63. IEEE (1999)
14. Fortune, J., Valerdi, R.: Considerations for successful reuse in systems engineering. AIAA Space **2008** (2008)
15. Frakes, W.B., Kang, K.: Software reuse research: status and future. IEEE Trans. Software Eng. **31**(7), 529–536 (2005)
16. Carrillo de Gea, J.M., Nicolás, J., Fernández-Alemán, J.L., Toval, A.: Automated support for reuse-based requirements engineering in global software engineering. J. Softw. Evol. Process **29**(8), e1873 (2017)
17. de Gea, J.M.C., Nicolás, J., Alemán, J.L.F., Toval, A., Vizcaíno, A., Ebert, C.: Reusing requirements in global software engineering. In: Maalej, W., Thurimella, A. (eds.) Managing Requirements Knowledge, pp. 171–197. Springer, Heidelberg (2013). https://doi.org/10.1007/978-3-642-34419-0_8
18. Gill, N.S.: Reusability issues in component-based development. ACM SIGSOFT Softw. Eng. Not. **28**(4), 4 (2003)
19. Goldin, L., Matalon-Beck, M., Lapid-Maoz, J.: Reuse of requirements reduces time to market. In: 2010 IEEE International Conference on Software Science, Technology and Engineering (SWSTE), pp. 55–60. IEEE (2010)
20. Gomaa, H., Farrukh, G.: A software engineering environment for configuring distributed applications from reusable software architectures. In: Eighth IEEE International Workshop on Software Technology and Engineering Practice, Proceedings, [incorporating Computer Aided Software Engineering], pp. 312–325. IEEE (1997)
21. Gomaa, H., Farrukh, G.A.: An approach for generating executable distributed applications from reusable software architectures. In: Second IEEE International Conference on Engineering of Complex Computer Systems, Proceedings, pp. 442–449. IEEE (1996)
22. Gomaa, H., Farrukh, G.A.: Automated configuration of distributed applications from reusable software architectures. In: 12th IEEE International Conference Automated Software Engineering, Proceedings, pp. 193–200. IEEE (1997)
23. Gómez-Hérnandez, M.A., Asensio-Pérez, J.I., Gómez-Sánchez, E., Bote-Lorenzo, M.L., Dimitriadis, Y.A.: A high-level reference model for reusable object-level coordination support in groupware applications. In: 15th EUROMICRO International Conference on Parallel, Distributed and Network-Based Processing, PDP 2007, pp. 315–324. IEEE (2007)
24. Gunasekaran, A.: Agile manufacturing: enablers and an implementation framework. Int. J. Prod. Res. **36**(5), 1223–1247 (1998)
25. Hill, J.H., Schmidt, D.C., Edmondson, J.R., Gokhale, A.S.: Tools for continuously evaluating distributed system qualities. Software IEEE **27**(4), 65–71 (2010)
26. Hofmeister, C., Nord, R., Soni, D.: Global analysis: moving from software requirements specification to structural views of the software architecture. IEE Proc. Softw. **152**(4), 187–197 (2005)

27. Hong, W.: Architecture-centric software process for pattern based software reuse. In: WRI World Congress on Software Engineering, WCSE 2009, vol. 4, pp. 95–99. IEEE (2009)
28. Hoover, C.L., Khosla, P.K.: An analytical approach to change for the design of reusable real-time software. In: Second Workshop on Object-Oriented Real-Time Dependable Systems, 1996, Proceedings of WORDS 1996, pp. 144–151. IEEE (1996)
29. Hossain, S.S.: Challenges and mitigation strategies in reusing requirements in large-scale distributed agile software development: a survey result. In: Arai, K., Bhatia, R., Kapoor, S. (eds.) CompCom 2019. AISC, vol. 998, pp. 920–935. Springer, Cham (2019). https://doi.org/10.1007/978-3-030-22868-2_63
30. Hossain, S.S., Jubayer, S.A.M., Rahman, S., Bhuiyan, T., Rawshan, L., Islam, S.: Customer feedback prioritization technique: a case study on lean startup. In: Misra, S., Gervasi, O., Murgante, B., Stankova, E., Korkhov, V., Torre, C., Rocha, A.M.A.C., Taniar, D., Apduhan, B.O., Tarantino, E. (eds.) ICCSA 2019. LNCS, vol. 11623, pp. 70–81. Springer, Cham (2019). https://doi.org/10.1007/978-3-030-24308-1_6
31. Irshad, M.: Measuring cost avoidance through software reuse. Ph.D. thesis, Masterís thesis, Blekinge Institute of Technology, school of computing, Sweden, MSE-2010: 38 (2011)
32. Kakarontzas, G., Stamelos, I.: Component recycling for agile methods. In: 2010 Seventh International Conference on the Quality of Information and Communications Technology (QUATIC), pp. 397–402. IEEE (2010)
33. Kamalraj, R., Geetha, B., Singaravel, G.: Reducing efforts on software project management using software package reusability. In: IEEE International Advance Computing Conference, IACC 2009, pp. 1624–1627. IEEE (2009)
34. Karlsson, L., Dahlstedt, Å.G., Regnell, B., och Dag, J.N., Persson, A.: Requirements engineering challenges in market-driven software development-an interview study with practitioners. Inf. Softw. Technol. **49**(6), 588–604 (2007)
35. Kavitha, C., Shidha, M., Jino, P.: Reusability and agile pattern mining. Int. J. Comput. Appl. **68**(2) (2013)
36. Khan, M.Z., Khan, M.: Enhancing software reusability through value based software repository. Int. J. Softw. Eng. Its Appl. **8**(11) (2014)
37. Kitchenham, B.: Procedures for performing systematic reviews. Keele, UK, Keele University **33**(2004), 1–26 (2004)
38. Knauss, E., Liebel, G., Schneider, K., Horkoff, J., Kasauli, R.: Quality requirements in agile as a knowledge management problem: More than just-in-time. In: 2017 IEEE 25th International Requirements Engineering Conference Workshops (REW), pp. 427–430. IEEE (2017)
39. Kovacevic, J., et al.: Survey of the state-of the-art in requirements engineering for software product line and model-driven requirements engineering. Technical report, Technical Report D 1.1, Ample-project (2007)
40. Liang, P., Avgeriou, P., Clerc, V.: Requirements reasoning for distributed requirements analysis using semantic wiki. In: Fourth IEEE International Conference on Global Software Engineering, ICGSE 2009, pp. 388–393. IEEE (2009)
41. Liang, X.: Research on the distributive knowledge management system of logistic enterprise integrating context management. In: 2009 WRI World Congress on Computer Science and Information Engineering, vol. 5, pp. 204–210. IEEE (2009)
42. Liu, L., Zicari, R., Hursch, W., Lieberherr, K.J.: The role of polymorphic reuse mechanisms in schema evolution in an object-oriented database. IEEE Trans. Knowl. Data Eng. **9**(1), 50–67 (1997)

43. López, C., Cysneiros, L.M., Astudillo, H.: NDR ontology: sharing and reusing NFR and design rationale knowledge. In: First International Workshop on Managing Requirements Knowledge, MARK 2008, pp. 1–10. IEEE (2008)
44. López, O., Laguna, M.A., Peñalvo, F.J.G.: Metamodeling for requirements reuse. In: WER, pp. 76–90 (2002)
45. Lucredio, D., de Almeida, E.S., Fortes, R.P.: An investigation on the impact of MDE on software reuse. In: 2012 Sixth Brazilian Symposium on Software Components Architectures and Reuse (SBCARS), pp. 101–110. IEEE (2012)
46. Mannion, M., Kaindl, H.: Product line requirements reuse based on variability management. In: 2012 19th Asia-Pacific Software Engineering Conference (APSEC), vol. 2, pp. 148–149. IEEE (2012)
47. Martinez, M.A., Toval, A.: COTSRE: a components selection method based on requirements engineering. In: Seventh International Conference on Composition-Based Software Systems, ICCBSS 2008, pp. 220–223. IEEE (2008)
48. Mili, A., Mili, R., Mittermeir, R.T.: A survey of software reuse libraries. Ann. Softw. Eng. **5**, 349–414 (1998)
49. Mohagheghi, P.: The impact of software reuse and incremental development on the quality of large systems. Ph.D. thesis, Norwegian University of Science and Technology (2004)
50. Motahari-Nezhad, H.R., Li, J., Stephenson, B., Graupner, S., Singhal, S.: Solution reuse for service composition and integration. In: 2009 World Conference on Services-I, pp. 400–407. IEEE (2009)
51. Mugridge, R.: Managing agile project requirements with storytest-driven development. Software IEEE **25**(1), 68–75 (2008)
52. Mujtaba, S., Petersen, K., Feldt, R., Mattsson, M.: Software product line variability: a systematic mapping study. Blekinge Institute of Technology, School of Engineering (2008)
53. Pan, K., Turner, S.J., Cai, W., Li, Z.: An efficient sort-based DDM matching algorithm for HLA applications with a large spatial environment. In: 21st International Workshop on Principles of Advanced and Distributed Simulation, PADS 2007, pp. 70–82. IEEE (2007)
54. Petersen, K., Feldt, R., Mujtaba, S., Mattsson, M.: Systematic mapping studies in software engineering. In: 12th International Conference on Evaluation and Assessment in Software Engineering, vol. 17. SN (2008)
55. Leite, J.C.S.P., Yu, Y., Liu, L., Yu, E.S.K., Mylopoulos, J.: Quality-based software reuse. In: Pastor, O., Falcão e Cunha, J. (eds.) CAiSE 2005. LNCS, vol. 3520, pp. 535–550. Springer, Heidelberg (2005). https://doi.org/10.1007/11431855_37
56. Przybyłek, A.: Systems evolution and software reuse in object-oriented programming and aspect-oriented programming. In: Bishop, J., Vallecillo, A. (eds.) TOOLS 2011. LNCS, vol. 6705, pp. 163–178. Springer, Heidelberg (2011). https://doi.org/10.1007/978-3-642-21952-8_13
57. Ramesh, B., Cao, L., Mohan, K., Xu, P.: Can distributed software development be agile? Commun. ACM **49**(10), 41–46 (2006)
58. Sahay, S., Nicholson, B., Krishna, S.: Global IT Outsourcing: Software Development Across Borders. Cambridge University Press (2003)
59. Shehata, M., Eberlein, A.: Requirements interaction detection using semi-formal methods. In: 10th IEEE International Conference and Workshop on the Engineering of Computer-Based Systems, Proceedings, pp. 224–232. IEEE (2003)
60. Singh, S., Chana, I.: Enabling reusability in agile software development. arXiv preprint arXiv:1210.2506 (2012)

61. Souag, A., Mazo, R., Salinesi, C., Comyn-Wattiau, I.: Reusable knowledge in security requirements engineering: a systematic mapping study. Requirements Eng. **21**(2), 251–283 (2016)
62. Suri, P., Garg, N.: Software reuse metrics: measuring component independence and its applicability in software reuse. Int. J. Comput. Sci. Netw. Secur. **9**(5), 237–248 (2009)
63. Sutherland, J., Viktorov, A., Blount, J., Puntikov, N.: Distributed scrum: agile project management with outsourced development teams. In: 40th Annual Hawaii International Conference on System Sciences, HICSS 2007, p. 274a. IEEE (2007)
64. Thakur, S., Singh, H.: FDRD: feature driven reuse development process model. In: 2014 International Conference on Advanced Communication Control and Computing Technologies (ICACCCT), pp. 1593–1598. IEEE (2014)
65. Turk, D., France, R., Rumpe, B.: Assumptions underlying agile software development processes. arXiv preprint arXiv:1409.6610 (2014)
66. Villegas, O.L., Laguna, M.A.: Requirements reuse for software development. In: RE 01 Doctoral Workshop. In 5th-IEEE International Symposium on Requirements Engineering, Toronto, Canada, pp. 27–31. Citeseer (2001)
67. Wahid, S., Smith, J.L., Berry, B., Chewar, C.M., McCrickard, D.S.: Visualization of design knowledge component relationships to facilitate reuse. In: Proceedings of the 2004 IEEE International Conference on Information Reuse and Integration, IRI 2004, pp. 414–419. IEEE (2004)
68. Waldmann, B., Jones, P.: Feature-oriented requirements satisfy needs for reuse and systems view. In: 17th IEEE International Requirements Engineering Conference, RE 2009, pp. 329–334. IEEE (2009)
69. Wehrwein, B., Matos, G., Tanikella, R.: A requirements-driven and collaborative decision support approach for evaluating the viability of candidate implementation technologies. In: 12th International IEEE Enterprise Distributed Object Computing Conference, EDOC 2008, pp. 293–299. IEEE (2008)
70. Xiang, F., Ling, G., Jin, Y.: User-driven GIS software reuse solution based on SOA and web2. 0 concept. In: 2nd IEEE International Conference on Computer Science and Information Technology, ICCSIT 2009, pp. 5–9. IEEE (2009)