Volume 3, No. 10 October 2024 - (2203-2216)
p-ISSN 2980-4868 | e-ISSN 2980-4841
https://ajesh.ph/index.php/gp

Software Change Request in Software Development Project : Factors and Methods (Scoping Review Methods)

Dwi Cahya Prasetya[1*], Yudha Prambudia[2], Muhammad Almaududi Pulungan[3]
Universitas Telkom, Indonesia
Email : dwicahyapras@telkomuniversity.ac.id

ABSTRACT
Software development is dynamic by nature, often requiring changes in software requirements through Software Change Requests (SCRs). This research aims to identify and analyze the key factors that influence SCRs in software development projects and evaluate the methods used to effectively manage these changes. This research uses the Scoping Review Method to systematically review and synthesize existing literature on SCRs, aiming to provide a comprehensive understanding of current practices, challenges, and strategies for managing software change requests. The review includes an extensive search of academic databases to identify relevant studies that address SCR factors and management methods. Key factors that influence SCR include project size, complexity, stakeholder involvement, and clarity of initial requirements. Methods for managing SCR are categorized into formal approaches, such as structured change control processes and tools, and informal approaches, which rely on ad-hoc communication and collaboration. The results underscore the importance of a well-defined change management process to mitigate the risks associated with SCR. Effective SCR management can improve project outcomes by improving software quality, increasing stakeholder satisfaction, and minimizing project delays. This research has practical implications for software development professionals by providing insight into effective SCR management techniques. This research also identifies gaps in the literature, recommending future research on the impact of agile methodologies and the use of automated tools in facilitating change management. Understanding these factors and methods enables practitioners to better navigate the complexities of software change, ensuring successful project delivery and maintaining high software quality.

Keywords: Software Change Requests, Change Management, Scoping Review.

INTRODUCTION

Software and software-based products are becoming increasingly important in various aspects of the modern society (Kumar et al., 2021). The information technology industry is currently one of the most dynamic and rapidly growing industries in the flobal economy in industry 4.0 (Lasi et al., 2014). The software industry faces serious challenges in the competition, due to the constant emergence of small and medium-sized software companies that stand for

more established companies (Lopez-Arredondo et al., 2020). These companies must make a great effort to improve their competitiveness and make it even more efficient (Lopez-Arredondo et al., 2020). To strengthen these kind of companies, efficient practices need to be adapted to their size and business type and process need to be improved increasing the quality and productivity of their services (Huang et al., 2015).

In companies that design and develop complex software products, especially specialized ones, change requests are inevitable due to ongoing improvements and adjustments. Managing these software change requests (SCRs) presents various challenges, primarily due to the frequent and sometimes unpredictable nature of such requests. The ability of software-based products to evolve over time often leads to increased complexity in managing the lifecycle of a product. One major challenge is the sheer volume of change requests that may arise, especially as products reach maturity or enter maintenance stages. The longer a product is in development or maintenance, the more likely it is to encounter numerous requests, which can place immense pressure on resources, timelines, and project management (Kumar et al., 2021).

Furthermore, the classification and prioritization of SCRs is another significant challenge. Without a systematic approach, change requests can disrupt development schedules, delay deliveries, and increase costs. Determining which requests are critical and which can be delayed requires careful analysis and often involves trade-offs between customer satisfaction, budget constraints, and technical feasibility (Rouse, 2016). Additionally, managing the expectations of customers who request these changes while maintaining control over the product's scope and quality can be difficult. Poor management of SCRs can lead to project delays, reduced quality, or even project failure if changes are not properly integrated into the development cycle (Crosno et al., 2015).

Based on the above background, the purpose of this research is to classify the causes of SCR and propose a methodology to handle them efficiently, ensuring that software products can evolve in response to user requests while maintaining project integrity. So that the benefits in this research are to contribute to software development companies in understanding the factors that affect software change requests (SCR) and how to better manage these changes. This research is expected to assist companies in formulating more effective SCR management strategies, which not only improve the efficiency of the software development process but also minimize the negative impact on budget, schedule, and product quality.

RESEARCH METHOD

Research question is created based on the needs of the selected topic. The following are the research question in this research:

RQ (1): What factors that generate SCR?

RQ (2): What methods are there in literature that can be used to deal with SCR?

RQ (3): What is the most frequently use methods to deal with SCR?
RQ (4): What is the dominant factors that make SCR?
RQ (5): What point of view are used to approach SCR?

In order to answer this question and understand the challenges, we searched from extend academic literature as the source data, that include Scopus. The process can be seen in Figure 1. We quired the Scopus database in August 2023. The query string used for search engine was “Software Change Request" OR “Change Request” AND “Software Development”. This query formulation is guided by research question that have been selected. The query was entered into Scopus search engine to yield articles for review. The search results are then refined by limiting publication date to year start from 2013 – 2023 resulting in 84 articles from Scopus. As inclusion criteria, the articles must be a journal. The second step in sorting these articles is to check whether it is a journal or not, if the articles is a book or report it will not be included in this research. This result in final list containing 79 journals. The third step in sorting these journals is to check whether is a literature review or not. The selected journals consist of 66 journals. The final step in sorting of these journals is to check the problem of these journals. The problem it must be about software change request. This results in final list containing 18 journals. The selected journals will be reviewed to answer the research question that has been determined.

RESULT AND DISCUSSION

This section will discuss the result of the review of all the journals. The results is 19 journals that will be reviewed. Table 2 will show the results of the quality assessment for which data was used in this research.

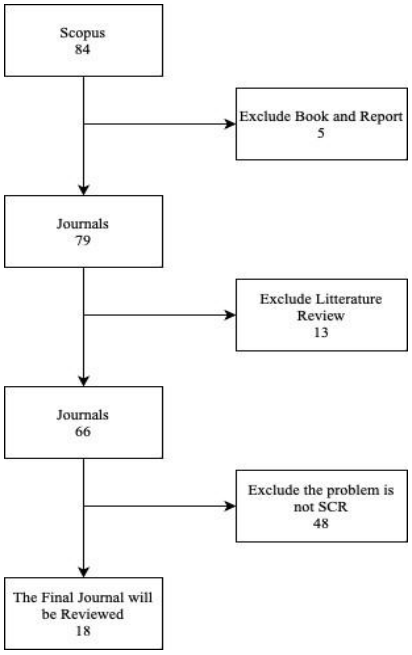

Figure 1. An Overview Literature Search Result

Table 2. Title Jurnal will be Reviewed

No	Writer	Title	Year	RQ1	RQ2	RQ3	RQ4	RQ5
1	John Anvik	Evaluating an Assistant for Creating Bug Report Assigment Recommenders	2016	√	√	√	√	√
2	Sufyan Basri, Roslina Ibrahim, Nazri Kama, & Saifuladli Ismail	A Change Impact Analysis Tools for Software Development Phase	2015	√	√	√	√	√
3	Haruya Iwasaki, Tsuyoshi Nakajima, Ryota Tsukamoto, Kazuko Takahashi & Shuichi Takumoto	A Software Impact Analysis Tool based on Change History Learning and its Evaluation	2022	√	√	√	√	√
4	Abeer Abdulaziz Alsanad, Azeddine Chikh, & Abdulrahman Mirza	Multilevel Ontology Framework for Improving Requirements Change Management in Global Software Development	2019	√	√	√	√	√
5	Mehran Halimi Asl & Nazri Kama	A change Impact Size Estimation Approach during the Software Development	2013	√	√	√	√	√
6	Morakot Choetkiertikul, Hoa Khanh Dam, Truyen Tran, Trang Pham, Chaiyong Ragkhitwetsagul, & Aditya Ghose	Automatically Recommending Components for Issue Reports Using Deep Learning	2021	√	√	√	√	√
7	Gyesik Oh & Yoo S Hong	Change Propagation Management by Active Batching	2017	√	√	√	√	√
8	Mariem Haoues, Asma Sellamia, & Hanêne Ben Abdallahb	Towards Functional Change Decision Support Based on COSMIC FSM Method	2019	√	√	√	√	√
9	Sidra Anwar	Decision Support System for Software Release Management	2019	√	√	√	√	√

No	Writer	Title	Year	RQ1	RQ2	RQ3	RQ4	RQ5
10	Hela Hakim, Asma Sellami, & Hanêne Ben Abdallahb	An in-Depth Requirements Change Evaluation Process using Functional and Structural Size Measures in the Context of Agile Software Development	2020	√	√	√	√	√
11	Phan Thi Than Huyen & Koichiro Ochimizu	An Inconsistency Management Support System for Collaborative Software Development	2014	√	√	√	√	√
12	Zaineb Sakhrawi, Asma Sellami, & Nadia Bouassida	Software Enhancement Effort Estimation using Machine Learning Regression Methods	2020	√	√	√	√	√
13	Sandeep Mitra	Using UML Modeling to Facilitate Three-Tier Architecture Projects in Software Engineering Courses	2014	√	√	√	√	√
14	Samir Omanovic & Emir Buza	Importance of Stable Velocity in Agile Maintenance	2013	√	√	√	√	√
15	Adnan Kraljić & Tarik Kraljić	Agile Software Engineering Practices and ERP: Is a sprint too fast for ERP Implementation?	2020	√	√	√	√	√
16	Kiana Rostami, Johannes Stammel, & Robert Heinrich	Architecture-based Assessment and Planning of Change Requests	2015	√	√	√	√	√
17	Kashif Asad & Dr. Mohd. Muqeem	Enhancing Requirements Change Request Categorization and Prioritizing in Agile Software Development Using Analytical	2023	√	√	√	√	√

No	Writer	Title	Year	RQ1	RQ2	RQ3	RQ4	RQ5
		Hierarchy Process (AHP)						
18	P Jalaja & T Adilakhsmi	Automated Change Impact Analysis Tool for Software Maintenance Phase	2023	√	√	√	√	√

The next stage will answer question from the research question (RQ) and discuss the results of the method and the dominant approach emerged from 2013-2023.

RQ1. What factors that generate SCR?

SCR can occur due to certain factors. These factors can occur due to external and internal factors of the project team or organization. The consequences of SCR is that costs can be increase and project completition time will increase (Butt & Jamal, 2017). When the time and cost become increase then it impacts a bad reputation of the software house on client and the company can lose its client and client is like an asset of the software house (Butt & Jamal, 2017). Based on the reviewed journal, changing requirements are frequently encountered in SCR. Since software systems must undergo changes throughout their lifecycle to reflect changes in their environment, such as requirements, technology, and usage profiles (Anvik, 2016).

Changes in requirements can occur due to the desired quality improvement / functionality requested by the client (Anvik, 2016); (Basri et al., 2015); (Alsanad et al., 2019); (Huyen & Ochimizu, 2014). Because software system development can continue evolving during the software system development stage. After being tested by the team, the results will be validated by the client. Customer performs the validation to check does implemented solution meets his requirements specified in the change request (Alsanad et al., 2019). Therefore, change requests will often emerge even though they have reached the validation stage. Other factors can be caused by ambiguity (Curcio et al., 2018), unclear requirements, the evolution of requirements to meet client needs (Haoues et al., 2019), fulfilling requirements (Oh & Hong, 2017), decision regarding change deployment (Sakhrawi et al., 2020), and software enhancement (Mitra, 2014).

RQ2. What methods are there in literature that can be used to deal with SCR?

There are severeal methods that can be used in dealing with SCR. Some of these methods are obtained from literature that has been summarized by the author. Here are some of these methods with the explanation: 1. Change Impact Analysis (CIA)

Change impact analysis is the method of predicting the impacts of the requirement change on the software elements (Haoues et al., 2019). Change Impact Analysis is involves analyzing the potential consequences of changes in software, specially focusing on the impact of changes on

different classes (Huyen & Ochimizu, 2014). The impacted software elements could be design artefacts such as packages and design classes, coding artefacts like components and classes, testing artefacts such as test cases and test reports, or any other software elements and documents (Haoues et al., 2019). Key feature of CIA is include identification of change, assessment of impact, risk analysis, dependencies and interrelationships, testing and validation, dicumentation and communication.

Change Impact Analysis is a critical process for managing changes within projects and systems. By thoroughly evaluating the potential impacts, risks, and dependencies associated with a proposed change, organizations can ensure smoother transitions, minimize negative effects, and achieve their desired outcomes more effectively. This systematic approach supports better planning, decision-making, and communication, ultimately contributing to the success of change initiatives.

1. Common Software Measurement International Consortium Functional Size Measurement (COSMIC FSM)

   COSMIC FSM method is used to measure the functional size of change requests, which provides a more realistic evaluation of the change request and aids in effort estimation (Mitra, 2014). COSMIC can be used to measure the size of a change request and estimate the effort required for its implementation (Oh & Hong, 2017). The COSMIC FSM method process includes three steps: Measurement strategy phase, mapping phase and Measurement phase.

   COSMIC FSM is a valuable tool for measuring the functional size of software, providing a standardized, objective, and flexible approach. It enhances project management by enabling accurate estimation, better resource allocation, and improved performance analysis. By focusing on functional user requirements and data movements, COSMIC FSM ensures that measurements are relevant and useful across different types of software projects, contributing to overall project success and effective communication among stakeholders.

2. Creation Assistant for Easy Assignment (CASEA)

   CASEA is a tool designed to enhance project management by leveraging the knowledge and skills of project members to create an assignment recommender system (Kaushik & Bhardwaj, 2017). Key Feature of CASEA is include skill mapping, knowledge base, assignment recommender, and collaboration enhancement. CASEA guides a project member through the assignment recommender creation process in four steps: Data Collection, Data Preparation, Recommender Training, and Recommender Evaluation (Kaushik & Bhardwaj, 2017).

   CASEA is a powerful tool for project management that leverages the knowledge and skills of team members to recommend optimal task assignments. By doing so, it can significantly improve the efficiency and effectiveness of project teams, leading to better project outcomes and enhanced team collaboration.

3. Multilevel Ontology Framework

Multilevel ontology framework is proposed to improve requirements change management (RCM) (Curcio et al., 2018). The framework aims to ensure the correctness of change requests by linking them to concepts from three ontologies: the requirement change ontology, the requirements engineering domain ontology, and the specific software application domain ontology (Curcio et al., 2018). In the context of Global Software Development (GSD), managing changes in requirements is a significant challenge due to the distributed nature of teams, diverse stakeholder needs, and varying levels of understanding. A Multilevel Ontology Framework can play a crucial role in improving requirements change management by providing a structured and coherent way to handle these complexities

Multilevel Ontology Framework significantly enhances requirements change management in Global Software Development by providing a structured and coherent approach to handling complexities. It ensures consistency, improves traceability, facilitates effective communication, and supports scalability and flexibility. By implementing such a framework, organizations can manage requirements changes more effectively, leading to better project outcomes and smoother collaboration among distributed teams.

4. Deepsoft-C

Issue reports are typically written in natural language, describing a request for implementing a new functionality, fixing a bug, or performing other project tasks (Akbar et al., 2021). DeepSoft-C takes as input the textual description (title and description) of a (new) issue and recommends a list of k components that are most relevant to the issue (e.g., components assignment and JavaScript) (Akbar et al., 2021). The architecture of DeepSoft-C is aims to recommend issue's components at the issue creation time where only the textual description is available.

5. Change Propagation Management by Active Batching

Change Propagation Management by Active Batching is a strategic approach to handling change in software engineering and system design (JALAJA & ADILAKSHMI, 2023). By grouping related changes and implementing them together, organizations can improve efficiency, ensure consistency, and mitigate risks associated with change propagation (JALAJA & ADILAKSHMI, 2023). This method supports agile practices by facilitating faster adaptation to evolving requirements and reducing the likelihood of disruptions during change implementation.

6. Decision Support System Release Management (DSSRM)

Decision Support System Release Management (DSSRM) provides support for management in analyzing and addressing the concerns related to resources, skills, cost, and schedule allocation (Sakhrawi et al., 2020). DSSRM also enables the management to independently validate the existence of problems, assess their impact, and evaluate the

potential benefits of proposed solutions (Sakhrawi et al., 2020). DSSRM is essential for organizations aiming to effectively manage the deployment of Decision Support Systems. By adopting structured release planning, rigorous testing, and proactive change management practices, organizations can ensure that DSS releases are successful, meet business objectives, and contribute to improved decision-making processes. DSSRM not only enhances the reliability and functionality of DSS but also supports organizational agility and responsiveness to evolving business needs.

7. The In-Depth Requirement Change Evaluation Process

   The In-Depth Requirement Change Evaluation Process based on the use of functional and structural size measurement methods. An in-depth requirement change evaluation process is crucial in software development and project management to ensure that proposed changes to project requirements are thoroughly assessed, validated, and implemented effectively. This process involves expressing user stories in terms of CFP (Cosmic Function Points) unit using the standard COSMIC FSM (Functional Size Measurement) Methods and in terms of CSM (Structural Size Measurement) units using the structural size measurement method.

8. Change Support Workflow Management System (CSWMS)

   Change Support Workflow Management System (CSWMS) is a specialized software system designed to facilitate and streamline the management of change requests within an organization. It provides a structured framework for handling and processing change requests, ensuring that they are evaluated, prioritized, and implemented effectively.

   CSWMS monitors the progress of Change Support Workflows (CSWs) and the ongoing changes in client workspaces to notify workers in advance of potential inconsistencies (Rostami et al., 2015). It provides workers with the contexts of changes, the changes causing the inconsistencies, and the CSWs associated with these changes (Rostami et al., 2015). Implementing a robust Change Support Workflow Management System helps organizations effectively manage change requests, mitigate risks, and maintain agility in responding to evolving business needs while ensuring the delivery of high-quality products and services.

9. UML Modeling

   UML modeling refers to use of the unified modeling language (UML) to visually represent and communicate the design and structure of a software system. UML modeling involves creating different types of diagrams, including the use case diagrams, class diagrams, sequence diagrams, state diagrams, and activity diagrams. These diagrams help to capture and communicate different perspectives of the system, such as Its functionality, structure, and dynamic behavior.

10. Agile Maintenance

    The method has been using is agile. The Agile development principles can be found in the manifesto for agile software development. Maintenance can be seen as an endless agile project whose product backlog is changing constantly – new user stories are added on the list and implemented user stories are removed from the list (Alsanad et al., 2019). In Agile maintenance, stable velocity serves as a cornerstone for effective planning, consistent delivery, and continuous improvement. It supports efficient resource management, enhances predictability for stakeholders, and fosters a culture of reliability and responsiveness. By maintaining a stable velocity, Agile teams can navigate the complexities of software maintenance more effectively, ensuring that they meet business objectives while delivering value to customers in a sustainable manner.

11. SAP Active Methodology

    SAP Activate Methodology, as described in the text, is structured approach used for managing complex projects, particularly for SAP system implementations. The texts highlight that SAP Activate Methodology is adaptable and integrates agile practices. It emphasizes a flexible, iterative, and incremental approach, allowing for early integration and adaptability to changing requirements.

12. Karlsruhe Architectural Maintainability Prediction (KAMP)

    The Karlsruhe Architectural Maintainability Prediction (KAMP) model represents a significant approach in software architecture research, focusing on proactive assessment and prediction of maintainability. Karlsruhe Architectural Maintainability Prediction (KAMP) to analyze the change propagation caused by a change request in a software system based on the architecture model (Anvik, 2016). Using context information annotated on the architecture KAMP enables project members to assess the effects of a change request on various technical and organizational artefacts and tasks during software life cycle (Anvik, 2016). By leveraging architectural metrics and predictive modeling, KAMP supports software architects and developers in making informed decisions to enhance the long-term maintainability of software systems. Its application contributes to improving software quality, reducing maintenance costs, and supporting the evolution of complex software architectures over time.

13. Analytical Hierarchy Process

    This paper presents a new framework for classifying agile software development change requests into small change request (SCRs) and large change requests (LCRs) and using the Analytic Hierarchy Process (AHP) to rank these requirements. The framework improves decision-making and resource and time allotment, improving project results and software quality. By integrating Analytic Hierarchy Process (AHP) into Agile software development practices, organizations can enhance the categorization and prioritization of requirements

change requests. AHP provides a structured and systematic approach to decision-making, ensuring that change requests are evaluated objectively based on criteria that align with business objectives and stakeholder priorities. This approach not only improves decision quality but also strengthens the agility and responsiveness of Agile teams in delivering value to stakeholders.

RQ3. What is the most frequently use methods to deal with SCR?

There are several methods that can be used to deal with SCR or CR. Several studies that have been conducted use various methods. However, out of the 18 reviewed studies the most methods frequently use method is “Change Impact Analysis”. Change Impact Analysis is use to assess the potential effects of change request in the software system (Anvik, 2016). Also change impact analysis is the method of predicting the impacts of the requirement change on the software elements. Change Impact Analysis involves analyzing the potential consequences of changes in software, specially focusing on the impact of changes on different classes (Iwasaki et al., 2022).

RQ4. What is the dominant factors that make SCR?

In several studies that have been conducted, the factor that frequently causes SCR is changes in user requirements / changing user needs. These requests arise from the client’s desire for specific development outcomes or their dissatisfaction with the software being developed. The occurrence of SCR or CR can happen repeatedly until the client feels satisfied with the achieved results. Additionally, we cannot predict the demands from clients, so we need to adapt accordingly.

RQ5. What point of view are used to approach SCR?

There are several point of view that can be used to approach SCR. Table 4 represents some of the approaches used in dealing with SCR. In certain cases, multiple approaches are combined to address SCR. However, the approach that is frequently employed to tackle these issues is agile management. This is because agile management possesses the capability to adapt to unpredictable changes in any phase of a project.

Table 5. List of Point of View

No	Writer	Year	RQ5
1	John Anvik	2016	Issue management life cycle, Text analysis
2	Sufyan Basri, Roslina Ibrahim, Nazri Kama, & Saifuladli Ismail	2015	Software development, Change impact analysis
3	Haruya Iwasaki, Tsuyoshi Nakajima, Ryota Tsukamoto, Kazuko Takahashi & Shuichi Takumoto	2022	Change impact analysis
4	Abeer Abdulaziz Alsanad, Azeddine Chikh, & Abdulrahman Mirza	2019	Change Management, Requirement Engineering
5	Mehran Halimi Asl & Nazri Kama	2013	Software change management, change impact analysis

No	Writer	Year	RQ5
6	Morakot Choetkiertikul, Hoa Khanh Dam, Truyen Tran, Trang Pham, Chaiyong Ragkhitwetsagul, & Aditya Ghose	2021	Software engineering analytics, Component recommendation, Data mining, Predictive model
7	Gyesik Oh & Yoo S Hong	2017	Project Management, Change Propagation Management, Active Batching
8	Mariem Haoues, Asma Sellamia, & Hanêne Ben Abdallahb	2019	Requirement management, Functional Size Measurement, Change Impact Analysis, Evaluation Estimation Models
9	Sidra Anwar	2019	Software Release Management, DSSRM
10	Hela Hakim, Asma Sellami, & Hanêne Ben Abdallahb	2020	Functional Size Measurement, Agile, indepth requirement change evaluation process, structural size measurement
11	Phan Thi Than Huyen & Koichiro Ochimizu	2014	Inconsistency awareness, change process, change support workflow (CSW)
12	Zaineb Sakhrawi, Asma Sellami, & Nadia Bouassida	2020	Functional Size Measurement, machine learning, software enhancement
13	Sandeep Mitra	2014	UML modeling, mapping techniques,
14	Samir Omanovic & Emir Buza	2013	Agile software development, change management process, customer behavior change,
15	Adnan Kraljić & Tarik Kraljić	2020	Enterprise Resource Planning (ERP), Agile
16	Kiana Rostami, Johannes Stammel, & Robert Heinrich	2015	Software architecture, maintainability, change propagation, change impact analysis
17	Kashif Asad & Dr. Mohd. Muqeem	2023	Agile software development, change impact analysis, AHP, Requirement management
18	P Jalaja & T Adilakhsmi	2023	Change impact analysis, static analysis, data structure

CONCLUSION

This research provides an overview of the factors and methods for handling SCR in software development projects. A review was conducted, comprising 84 studies, which resulted in the selection of 18 studies obtained from the Scopus database, covering the period from 2013 to 2023. These studies serve as references for scoping reviews. Based on the review, it is evident that SCRs are mostly triggered by changes in user requirements or evolving user needs. As the software project progresses, changes will continue to emerge, and customers or users will request these changes to achieve optimal products or software.

When addressing SCR, several methodologies are referenced in the 18 selected studies. Among the 14 methodologies/models discovered for solving the problem of SCR / CR, Change Impact Analysis (CIA) stands out as the most frequently used method. CIA offers the benefit of

predicting the impact of requirement changes. Which is SCR / CR characteristic is cannot be easily predicted, this model can be helpful in reducing the impact caused by SCR / CR.

REFERENCES

Akbar, M. A., Shameem, M., Khan, A. A., Nadeem, M., Alsanad, A., & Gumaei, A. (2021). A fuzzy analytical hierarchy process to prioritize the success factors of requirement change management in global software development. *Journal of Software: Evolution and Process*, *33*(2), e2292.

Alsanad, A. A., Chikh, A., & Mirza, A. (2019). Multilevel ontology framework for improving requirements change management in global software development. *IEEE Access*, *7*, 71804–71812.

Anvik, J. (2016). *Evaluating an assistant for creating bug report assignment recommenders*.

Basri, S., Kama, N., Ibrahim, R., & Ismail, S. A. (2015). A Change Impact Analysis Tool for Software Development Phase. *International Journal of Software Engineering and Its Applications*, *9*(9), 245–256.

Butt, S. A., & Jamal, Tjp. (2017). Frequent change request from user to handle cost on project in agile model. *Proc. of Asia Pacific Journal of Multidisciplinary Research*, *5*(2), 26–42.

Crosno, J. L., Dahlstrom, R., & Manolis, C. (2015). Comply or defy? An empirical investigation of change requests in buyer-supplier relationships. *Journal of Business & Industrial Marketing*, *30*(5), 688–699.

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

Haoues, M., Sellami, A., & Ben-Abdallah, H. (2019). Towards functional change decision support based on COSMIC FSM method. *Information and Software Technology*, *110*, 78–91.

Huang, S. Y., Lee, C.-H., Chiu, A.-A., & Yen, D. C. (2015). How business process reengineering affects information technology investment and employee performance under different performance measurement. *Information Systems Frontiers*, *17*, 1133–1144.

Huyen, P. T. T., & Ochimizu, K. (2014). An Inconsistency Management Support System for Collaborative Software Development. *IEICE TRANSACTIONS on Information and Systems*, *97*(1), 22–33.

Iwasaki, H., Nakajima, T., Tsukamoto, R., Takahashi, K., & Tokumoto, S. (2022). A software impact analysis tool based on change history learning and its evaluation. *Proceedings of the 44th International Conference on Software Engineering: Software Engineering in Practice*, 11–12.

Jalaja, P., & Adilakshmi, T. (2023). Automated Change Impact Analysis Tool For Software Maintenance Phase. *Journal of Theoretical and Applied Information Technology*, *101*(7).

Kaushik, S., & Bhardwaj, A. (2017). *Compressing Effort and Schedule in ERP Implementations with Iterative Methodologies: Few Industrial Cases*.

Kumar, M., Vaidya, O. S., & Srivastava, R. K. (2021). Impact of task priority on software supply chain: a simulation approach. *South Asian Journal of Business Studies*, *10*(3), 326–341.

Lasi, H., Fettke, P., Kemper, H. G., Feld, T., & Hoffman, M. (2014). Industrie 4.0. Busin. *Inform. Syst. Eng*, *4*, 239–242.

Lopez-Arredondo, L. P., Perez, C. B., Villavicencio-Navarro, J., Mercado, K. E., Encinas, M., & Inzunza-Mejia, P. (2020). Reengineering of the software development process in a technology services company. *Business Process Management Journal*, *26*(2), 655–674.

Mitra, S. (2014). Using UML modeling to facilitate three-tier architecture projects in software engineering courses. *ACM Transactions on Computing Education (TOCE)*, *14*(3), 1–31.

Oh, G., & Hong, Y. S. (2017). Change propagation management by active batching. *DS 87-4 Proceedings of the 21st International Conference on Engineering Design (ICED 17) Vol 4: Design Methods and Tools, Vancouver, Canada, 21-25.08. 2017*, 613–622.

Rostami, K., Stammel, J., Heinrich, R., & Reussner, R. (2015). Architecture-based assessment and planning of change requests. *Proceedings of the 11th International ACM SIGSOFT Conference on Quality of Software Architectures*, 21–30.

Rouse, W. B. (2016). *Universities as complex enterprises: How academia works, why it works these ways, and where the university enterprise is headed*. John Wiley & Sons.

Sakhrawi, Z., Sellami, A., & Bouassida, N. (2020). Software enhancement effort estimation using machine learning regression methods. *International Journal of Computer Information Systems and Industrial Management Applications*, *12*, 12.