Contents lists available at Science-Gate

International Journal of Advanced and Applied Sciences

Journal homepage: http://www.science-gate.com/IJAAS.html

The key factors contribute to time pressure in software development projects: A review

Ruqaya Gilal *, Mazni Omar, Mawarny Md Rejab

School of Computing, Universiti Utara Malaysia, Kedah, Malaysia

ARTICLE INFO

Article history:
Received 22 May 2023
Received in revised form
22 September 2023
Accepted 4 October 2023

Keywords:
Time pressure
Software development
Software failure
Failure factors

ABSTRACT

The success of software development projects is often hindered by time pressure (TP), leading to decreased productivity, compromised quality, and increased risk of failure. To address this issue, it is crucial to understand the key factors contributing to TP in software development projects. In line with the study's objectives, the review methodology followed the Kitchenham and Charters criteria, and a search strategy encompassed four primary digital databases, namely IEEE, ACM Digital Library, Science Direct, and Springer, resulting in 4,500 relevant sources. After applying inclusion and exclusion criteria, a total of 128 papers were selected for analysis. This paper offers a comprehensive overview of the factors contributing to TP in software development. This study synthesizes the findings from multiple studies to guide practitioners in improving their project management approaches and highlights the significance of enhancing various aspects of the development process. The findings highlight the importance of improving project management, estimation techniques, knowledge, and skills to effectively manage TP. Additionally, managing requirements volatility, setting clear goals and objectives, and reducing distractions and interruptions emerge as crucial strategies for mitigating TP and enhancing project success. Furthermore, selecting software developers based on their personality traits is recommended to foster a work environment conducive to reduced TP and improved software development outcomes. By understanding and addressing these factors, software development teams can alleviate TP and increase the likelihood of successful software products. Implementing these recommendations can contribute to reduced TP, improved project outcomes, and enhanced overall success in software development.

1. Introduction

Software development has become an essential aspect of modern business and daily life, representing a growing field that is ubiquitous across small roadside shops and large industries. However, the rapid growth and increasing demand for software can sometimes create challenges for software developers. Failure means the loss of a value-creating project that has failed to meet its goals (Ibraigheeth and Fadzli, 2019; Shepherd et al., 2014). When a system part fails to perform the required function or performs the required function but not within specified limits, it is called a failure (Zhu, 2017). Therefore, software failure occurs when it does not meet user requirements within the client's budget and timeframe (Ibraigheeth and Fadzli, 2019). Therefore, delivering software on time is important, but it is equally vital to ensure it aligns with client requirements.

According to the Standish report of 2020 (Krasner, 2021), the success rate of software development projects remains lower compared to the rates of challenges and failures. The report indicates a success rate of 33%, with challenge and failure rates of 54% and 19% respectively. When software fails, it not only impacts the employees and software but it impacts the whole organization. It is difficult to quantify the worldwide budget for software failure. The failure rate in 2017 was 1.3 trillion billion USD. It is a very huge amount of software failure. Regarding software development projects, exploring the exact areas of divergence to improve performance and avoid the negative repercussions of opposing viewpoints is important. With the increasing demand for software

* Corresponding Author.
Email Address: ruqayagilal@hotmail.com (R. Gilal)
https://doi.org/10.21833/ijaas.2023.10.018
Corresponding author's ORCID profile:
https://orcid.org/0000-0001-7263-2291
2313-626X/© 2023 The Authors. Published by IASE.

applications, development teams have been pressured to deliver projects on time. The fast-paced nature of software development has created a challenging environment where time pressure (TP) is a constant factor. Therefore, TP is a well-known factor that can lead to software failure. TP refers to the condition where there is an insufficient amount of time available to fulfill the requirements associated with various tasks (Basten, 2017). Moreover, TP is a psychological condition that arises whenever there is insufficient time to complete a task (Speier-Pero, 2019). Studies have shown that TP can negatively impact the quality of software development, leading to issues such as bugs (Kuutila et al., 2017), burnout (Kuutila et al., 2020; Verner et al., 2008), and poor performance (Kuutila et al., 2020). For instance, a report by the Consortium for Information and Software Quality highlighted the staggering economic impact of 2.08 trillion USD attributed to low-quality software in the US during 2020, underscoring the significance of addressing TP and its associated software failures. Additionally, TP can lead to increased stress and burnout among software developers, which can further contribute to software failure

Despite the extensive literature on software failures, Raunak and Binkley (2017) argued that further research was required to highlight software engineers' issues and to investigate how to avoid software project failures. TP is a critical factor that can lead to decreased productivity, compromised quality, and increased risk of failure. Understanding the key factors contributing to TP in software development projects is crucial for practitioners to effectively manage and mitigate its effects. Therefore, through a systematic review of existing research, this study aims to identify the key factors that contribute to TP in software development projects and assess their impact on software failure, providing a comprehensive overview of these factors.

2. Literature review

From the start, TP has been recognized as a problem in SE (Kuutila et al., 2020). TP is a psychological state that occurs when there is insufficient time to complete a task (Speier-Pero, 2019). Moreover, the word "time pressure" refers to a lack of time to complete tasks (Basten, 2017; Gilal et al., 2019a). On the other hand, TP is "the degree to which employees believe they do not have enough time to complete their work tasks" (Ohly and Fritz, 2010). TP can be defined as a sense of urgency or constraint in completing a task or achieving a goal within a specified time frame.

When TP is generated, it can have numerous unfavorable effects on software development, ultimately resulting in software failure. The literature review shows how TP impacts employees' performance. According to the study of Kuutila et al. (2020), TP has many negative impacts, leading to software failure. According to the previous literature, TP is one reason for software developer's unhappiness and stress (Graziotin et al., 2017; Girardi et al., 2021). It creates depression among software developers; as a result, they are mentally disturbed, impacting their performance (Girardi et al. 2021). Moreover, TP increases the burden on employees, which in turn causes other problems (Basten, 2017). When the burden on employees increases, it impacts them psychologically and physiologically, which has negative effects on their personal lives. They struggle to balance their work and family time and cannot perform effectively, ultimately leading many employees to resign from their jobs (Basten, 2017). This is a main cause of burnout and it has a significant impact on the entire organization.

Furthermore, Mäntylä et al. (2014) claimed in their study that TP is an unavoidable situation, which encourages developers to take shortcuts. When an organization works under TP, employees take shortcuts to meet the deadlines, which makes the software quality worse. Furthermore, it is the reason for conflicts in the working environment. Whenever TP generates tension among the employees, they start arguing about who will do which task. Failing to match people correctly over time may create conflict and decrease cooperation. It impacts their performance and leads to software failure. Fig. 1 shows the negative impacts of TP on software development, leading to software failure.

Fig. 1 highlights the previous literature, which shows the negative impacts of TP. Kuutila et al. (2020) suggested that TP can lead to software failure in software development. Additionally, focusing on the impact of TP reveals that it can cause unhappiness and stress among software developers (Girardi et al., 2021), leading to depression and, ultimately, software failure (Kuutila et al., 2020). These factors are interconnected. TP can also increase the burden on software developers (Maruping et al. 2015), leading to burnout and further contributing to software failure (Verner et al., 2008). Furthermore, a lack of communication (Shah et al., 2014) and a lack of user involvement (Zahid et al., 2018) can also arise when there is a lack of time, leading to software failure.

Furthermore, the study suggests that TP can force software developers to take shortcuts to meet deadlines (Mäntylä et al., 2014), which negatively impacts the quality of the software and results in software failure (Lavallée and Robillard, 2015). Increased TP can also impact the work environment, leading to conflicts (Maruping et al., 2015) that affect the overall performance of employees and result in software failure. Overall, these studies highlight TP's negative impacts and its relationship with software failure.

3. Methodology

To achieve the main objective of this study, the systematic literature review followed the guidelines established by Kitchenham and Charters (2007), which were also used in a recent review by Kuutila et al. (2020). Kitchenham and Charters (2007) guidelines lie in their well-established reputation and widespread adoption in the field of software engineering research. Kitchenham and Charters (2007) guidelines provide a systematic and rigorous methodology for conducting literature reviews, ensuring a comprehensive and objective assessment of the existing body of knowledge. The study's foundation guidelines are as follows: research questions, inclusion and exclusion criteria, search sources and strategies, study selection, data extraction, and synthesis of results. These steps are common in various fields, including software engineering, and were also used in a review conducted by Thomas (2021). By adhering to the Kitchenham and Charters (2007) guidelines, we aimed to minimize biases and ensure the reliability and validity of our review process.

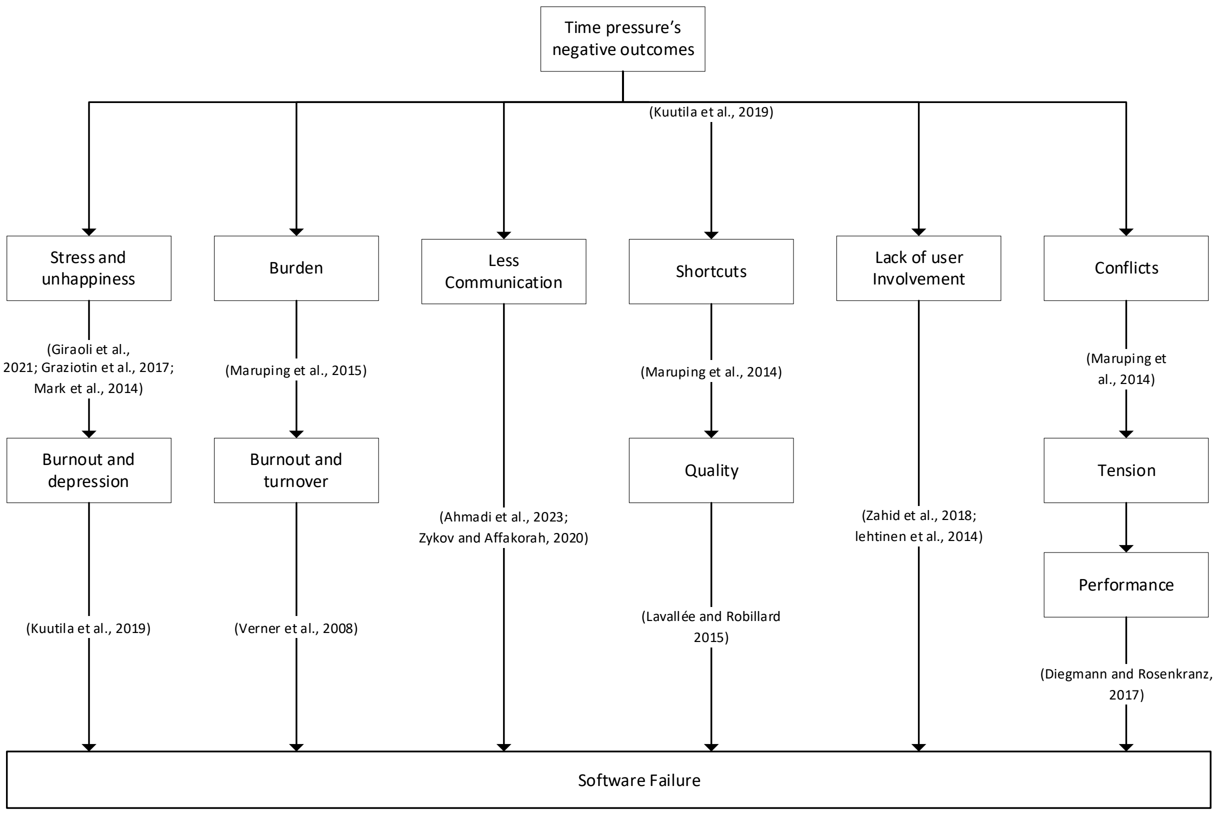

Fig. 1: Impacts of TP

3.1. Research questions

The research questions for this study were designed to contribute to the literature on software development by exploring software failure factors that cause TP in software development projects. This study has set the following question to be answered:

Q: What software failure factors contribute to TP in software development projects?

Based on previous research, the question aims to identify the common software failure factors. Its parts are specifically designed to clearly understand these factors and determine those that lead to TP.

3.2. Inclusion and exclusion criteria

The inclusion and exclusion criteria can help ensure that the studies included in the literature review are relevant to the research question and meet the necessary standards of academic rigor. Accordingly, the following criteria were generally considered for inclusion and exclusion after focusing on the dimensions.

- Inclusion criteria:

1. The study should focus on software development projects.
2. The study should investigate the factors that contribute to software failures.
3. The study should examine the impact of TP on software failure factors.
4. The study should be published in the English language.
5. The study should have been published in a peer-reviewed journal or conference proceedings.
6. The study should have been published between 2008 and 2023.
7. The study should be relevant to the research question.

- Exclusion criteria:

1. The study does not focus on software development projects.

2. The study does not investigate the factors that contribute to software failures.
3. The study does not examine the impact of TP on software failure factors.
4. The study is not published in the English language.
5. The study is not available online.
6. Incomplete or abstract works, keynote lectures, reports, dissertations, and books.
7. The study was published before 2008.
8. The study is not relevant to the research question.

3.3. Search sources and strategies

Two episodes were used to collect studies for the review: searching digital databases/libraries and searching references. Four primary digital databases were selected: IEEE, ACM Digital Library, Science Direct, and Springer. Only digital libraries were used, focusing on journal articles and conference papers from 2008 to December 2023; we did not narrowly limit our search to the time period as software failure and TP have been ongoing problems in software development. The decision to use the IEEE, ACM Digital Library, Science Direct, and Springer databases was based on several factors, including their relevance to the research topic, the quality of the content they provide, and the availability of resources. Overall, using these databases provides a wide range of scholarly literature that will enable us to draw robust conclusions supported by existing literature.

To conduct our search, we utilized a query that searched within the fields of title, abstract, and keywords for each database. We focused on the terms "time pressure," "software," and "software failure," as well as relevant synonyms for these terms. Our search pattern consisted of the following: ("deadline pressure" OR "time pressure" OR "time restriction*" OR "time limitation" OR "schedule pressure" OR "time constraint*") AND ("program" OR "software" OR "project" OR "information technology" OR "information system*") AND ("Program error" OR "program failure" OR "software burnout" OR "software crash"). We adjusted this pattern to match the syntax of each database. Keywords were finalized for the search, and filters were applied to explore the libraries. Once the studies were selected, the search from references was performed manually. The primary studies were selected based on the research objectives. IEEE returned 1175 papers, ACM Digital Library returned around 1090 publications, Science Direct returned 1015, and Springer returned 1220.

3.4. Study selection

The study selection process was divided into pre-selection and selection, following guidelines from Kitchenham and Charters (2007). The pre-selection process filtered studies based on titles, abstracts, and conclusions, resulting in 90% of returns being eliminated. Thirty-nine studies from IEEE, 29 from ACM, 35 from Science-Direct, and 25 from Springer were shortlisted. The selection was based on inclusion and exclusion criteria, with 128 pre-selected studies reviewed, resulting in 13 meetings the study's demands. A reference search of the 13 selected studies identified 116 related references, with four additional studies meeting the selection criteria, resulting in a total of 17 studies for the review. Fig. 2 shows the selection process.

3.5. Data extraction

The information obtained from this process was thoughtfully segregated into two distinct categories, each with its unique focus. The first category, known as "about basics," included crucial details such as the study titles, authors, publishers, and years of publication. These fundamental pieces of information are essential in comprehensively understanding the study.

On the other hand, the second category, "about contents," honed in on extracting data related to the more specific and nuanced aspects of the study. This category focused on the study objective, failure factors, populations, and study settings. By examining these factors, researchers can gain a deeper understanding of the context and underlying mechanisms of the research, which can be crucial in interpreting the data and drawing meaningful conclusions. Overall, the logical division of data into these two categories allows for a more comprehensive and insightful analysis of the study, enabling researchers better to understand the study's fundamental and specific aspects.

3.6. Synthesis of results

To obtain the answer to the first question, it was crucial to amalgamate the outcomes of the primary investigations. The factors of software failure were tabulated through a descriptive synthesis. Additionally, the resulting amalgamation is showcased in Table 1 and Fig. 3.

The description of the synthesis results was based on failure factors, as it was the study's main objective. It concluded some factors categorized below to identify the software failure factors contributing to the TP.

4. Discussion

This section highlights and discusses the collected results for the study topic. The studies received a comprehensive examination and analysis, as outlined in the Methodology section. The studies that were assessed looked at the underlying causes of software failure and highlighted how these factors contribute to TP to provide an answer to the research question. Some identified failure factors are outlined in Fig. 3.

1. Poor Project management (PM): this is an essential aspect of every software development

project. Poor management has been described as hindering project progress (Zykov and Attakorah, 2020; Zarndt, 2011). However, if the project is poorly planned, it faces difficulties and eventually fails. Poor project management has been identified as a factor in project failure (Ahmadi et al., 2023). Poor PM is known as a cause of TP.

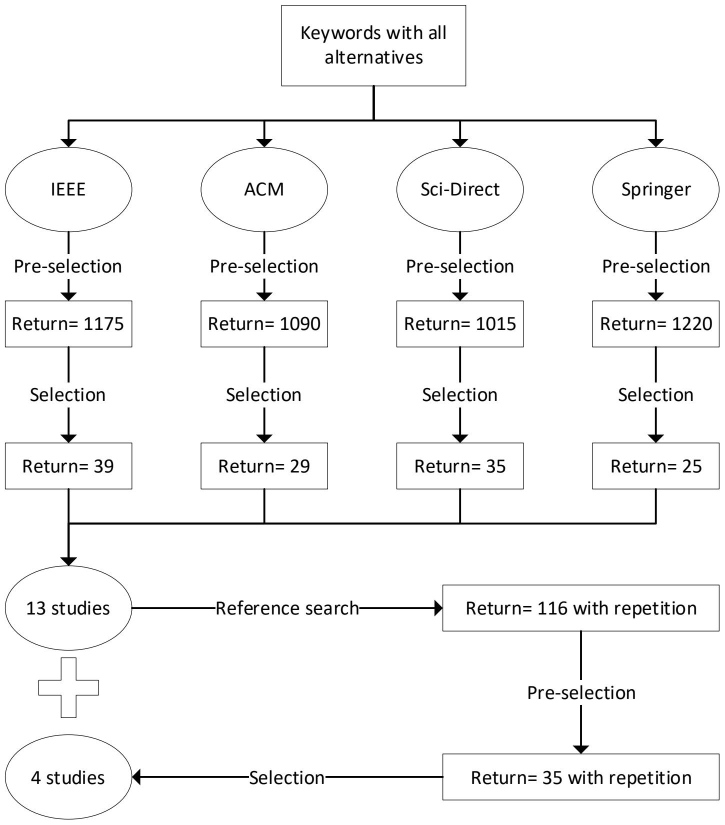

Fig. 2: The study selection process

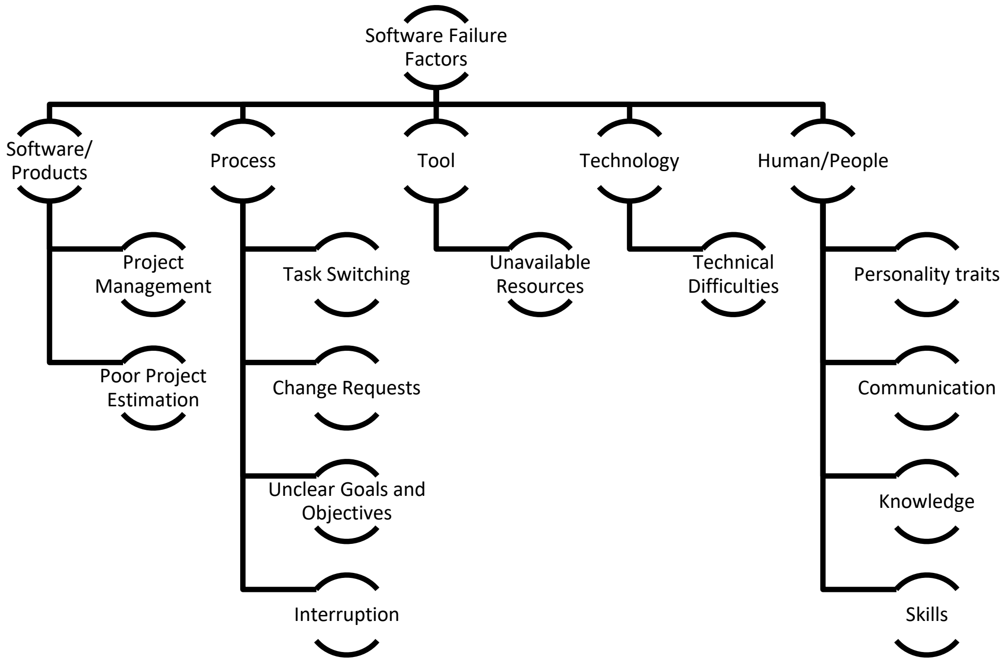

Fig. 3: Software failure factors

TP is induced when project managers underestimate the job, the extent to which workers believe they do not have enough time to complete their work or must work faster than normal (Ohly and Fritz, 2010). Poor PM can generate TP in several ways such as project management can result in a lack of clear goals, objectives, and timelines, leading to delays and missed deadlines (Ahmadi et al., 2023), which can create a sense of TP. Moreover, Poor project management can lead to poor communication and collaboration between team members, causing confusion and increasing TP as deadlines approach (Hassan et al., 2019; Ahmadi et al., 2023). In summary, poor project management can create an environment of uncertainty and disorganization, increasing the likelihood of missed deadlines and generating significant TP, which can negatively impact software quality and increase the risk of software failure (Ahmadi et al., 2023).

2. Poor estimation: as previously mentioned, 54 % of software projects were affected by budget and schedule overruns (Johnson, 2018). The project's success or failure is often determined by whether it is finished on schedule and within budget. When the estimate is incorrect, the conclusions are unreliable (Kuutila et al., 2020), and the project's efficiency suffers (Smith, 2014). Poor estimation of the time and resources required to complete a project can result in tight deadlines and missed targets, leading to TP. It also results in underestimating technical challenges that may arise during the development process. This can result in the need for additional time and resources, causing delays and adding to the overall TP. Poor estimation can also result in an incomplete risk assessment, leading to unexpected problems and delays, adding to the overall TP. In conclusion, a poor estimation can cause significant TP in software development, as developers and teams are forced to rush to meet deadlines and complete projects within tight timeframes. This can result in decreased software quality, increased risk of bugs and failures, and reduced overall satisfaction with the development process (Dullemond et al., 2011). According to Kuutila et al. (2020), poor estimation generates TP in software development.

3. Inappropriate skills or unskilled workers are the main factors that affect software projects (Gupta et al., 2019; Guillaume-Joseph and Wasek, 2015; Al-Ahmad et al., 2009). Since they have no prior experience dealing with tight deadlines, the results can be unsatisfactory (Verner et al., 2008). As a result, it is not about the number of employees to appoint to a job but rather about the employees' software development expertise and skills. Langer et al. (2014) concluded that the lack of knowledge negatively impacts software development's performance. Furthermore, when a software project is delayed due to inexperienced workers, it creates TP on the employees, and employees try to take shortcuts to meet deadlines (Kuutila et al., 2020), lowering the quality of the project and leading to software failure (Mäntylä et al., 2014). Individual skills or experience, according to Kuutila et al. (2020), could also minimize the negative effects of TP on individuals. Sometimes software projects have inexperienced project managers who cannot handle the tight schedule and complexities of projects and cannot grasp whether or not workers are familiar with the tools and techniques. Such types of situations impose a burden on employees then they have to deal with a heavy workload, and employees start taking shortcuts to reach the deadlines. Furthermore, in the CHAOS report in 2015, software was 60 % challenged and 23% failed because of unskilled people. Lack of appropriate skills or expertise required to complete tasks can result in decreased productivity, adding to the overall time to complete the project within tight deadlines. It can increase the risk of mistakes, leading to the need for additional time and resources to correct these errors, adding to the overall TP. When workers are unwilling to adapt to new technologies and processes can result in difficulty in problem-solving and increased TP as developers struggle to find solutions (Al-Ahmad et al., 2009).

4. Task switching, or switching attention between multiple tasks, can cause TP in software development in several ways. It can result in decreased productivity, as workers are forced to constantly shift their attention from one task to another, leading to decreased efficiency and increased TP to complete each task within tight deadlines (Sawyer and Southwick 2002; Kuutila et al., 2020; Chong et al., 2010). It also increases cognitive load, or the mental effort required to perform a task, making it more difficult to complete tasks efficiently (Byrne et al., 2015). Task switching can also lead to decreased quality, as workers are forced to work on multiple tasks simultaneously, leading to decreased attention to detail and increased risk of errors (Chong et al., 2010). In conclusion, task switching can significantly impact software development, decreasing productivity, quality, and overall efficiency. It can also lead to missed deadlines and an increased software failure risk.

5. Lack of knowledge can cause TP in software development in several ways. When developers lack knowledge about a particular technology or programming language, it can result in increased learning time, adding to the overall TP to complete the project within tight deadlines (Sharma and Spinellis, 2018). It can also make it more difficult to solve problems and resolve technical issues, leading to increased TP as developers struggle to find solutions. Lack of knowledge can also increase the risk of mistakes, leading to the need for additional time and resources to correct these errors, adding to the overall TP. In conclusion, lack of knowledge can have a significant impact on software development, causing increased TP, decreased efficiency, and increased risk of mistakes (Sharma and Spinellis, 2018; Langer et al., 2014; Lavallée and Robillard 2015; Martini et al., 2014).

6. Requirements volatility, or the changing nature of project requirements, can generate TP in software development in several ways: If requirements change frequently, it can result in increased development time, as developers are forced to constantly adjust their work to meet new requirements, adding to the overall TP. Requirements volatility can also make planning and scheduling the development process difficult, leading to increased pressure as developers struggle to meet deadlines (Lavallée and Robillard, 2015). It can also result in a decreased focus on quality, as developers are forced to prioritize meeting changing requirements over ensuring the quality of their work, leading to increased TP to complete the project within tight deadlines (Akbar et al., 2019;

Jayatilleke and Lai, 2018; Shafiq et al., 2018). Therefore, if requirements change frequently, it can increase the risk of mistakes, leading to the need for additional time and resources to correct these errors, adding to the overall TP. In conclusion, requirements volatility can have a significant impact on software development, increasing TP, decreasing quality, and increasing the risk of mistakes (Singh and Vyas 2012; Akbar et al., 2019; Jayatilleke and Lai, 2018; Shafiq et al., 2018).

7. Interruptions: such as phone calls, emails, and unscheduled meetings, can result in decreased productivity, as workers are forced to switch their attention from one task to another. This can result in decreased efficiency, increased cognitive load, and added TP to complete tasks within tight deadlines (Girardi et al., 2021). Additionally, frequent interruptions can lead to missed deadlines and an increased risk of mistakes, adding to the overall TP in software development.

8. Technical difficulties: can generate TP in software development projects in several ways: Technical difficulties can arise unexpectedly during the development process, such as unexpected bugs or compatibility issues, which can cause delays and increase the workload (Graziotin et al., 2017). If the development team lacks the technical expertise required to solve a technical issue, they may spend more time trying to resolve it, leading to increased TP. Suppose the technology used in the project is outdated or inadequate. In that case, it can cause technical difficulties and increase TP as the team tries to work around the limitations of the technology. Integrating different technologies or systems can also lead to technical difficulties, such as data compatibility issues or performance problems, increasing TP. Therefore, technical difficulties can cause significant disruptions in software development projects and generate TP by increasing the workload, causing delays, and requiring additional resources.

9. Unavailable resources can lead to TP in software development projects in several ways: If the development team is understaffed, it can lead to longer hours and increased workload, resulting in TP. If the development team lacks access to the necessary tools and equipment to complete the project, they may have to work around these limitations (Khan et al., 2019), which can cause delays and increase TP. The project relies on external resources, such as third-party software or hardware; if these resources become unavailable, it can cause delays and increase TP. Moreover, resource unavailability generates frustration in software developers, leading to software failure (Ford and Parnin, 2015; Girardi et al., 2020). When software developers cannot access the resources they need to complete their work, it can cause delays and increase the pressure to meet deadlines. This can lead to frustration and a decrease in job satisfaction, which can negatively impact the quality of the software being developed. In addition, the frustration caused by the unavailability of resources can lead to burnout and turnover among software developers, further exacerbating the problems of TP and increasing the risk of software failure (Maruping et al., 2015). When software developers are frustrated and stressed, they may make more mistakes, take shortcuts, or miss important details, which can negatively impact the quality and reliability of the software they are developing (Mäntylä et al., 2014). In conclusion, unavailable resources can create significant challenges in software development projects and generate TP by causing delays, increasing workload, and requiring additional resources.

10. Unclear goals and objectives can generate TP in software development projects in several ways: If the goals and objectives of the project are not clearly defined, it can result in confusion among the development team and lead to delays and a lack of progress (Ahmadi et al., 2023). Moreover, if the goals and objectives of the project change frequently, it can confuse and cause the development team to have to rework their efforts, leading to increased TP. The goals and objectives of the project, if not properly planned and documented, can result in misunderstandings and the need for additional work to correct mistakes, leading to increased TP. Hence, if the project's scope is not clearly defined, it can result in confusion among the development team and cause delays as they try to determine the project's scope. In conclusion, unclear goals and objectives can cause significant disruptions in software development projects and generate TP by causing confusion, leading to delays, rework, and misaligning expectations.

11. Personality traits, also known as soft skills, are an often overlooked factor in software development. However, personality is a combination of behavior, emotion, motivation, and thought patterns that define an individual (Gilal et al., 2019b). Personality traits can foster positive employee behavior, ultimately leading to successful project outcomes (Gilal et al., 2019c; Gilal et al., 2017a). However, research has shown that personality traits such as stress tolerance, flexibility, and motivation can significantly impact the success of a software development project (Gilal et al. 2019a; Gilal et al. 2018; Gilal et al., 2017b). In particular, how software developers interact with each other and handle stress can have a major effect on the outcome of a project. There are lots of studies done to understand the role of personality traits in software development that can help project managers make informed decisions about team composition and better manage risk (Gilal et al., 2017c; Gilal et al., 2019b; Gilal et al., 2016).

Personality traits can contribute to TP in software development in several ways. For example, individuals who struggle with stress management may have difficulty meeting deadlines and keeping pace with the project's demands. It can lead to burnout and decreased productivity, ultimately contributing to TP. Furthermore, poor interpersonal skills and conflicts between team members can also create distractions and slow the project's progress, leading to additional TP. By considering the impact of personality traits on software development, organizations can work to mitigate these risks and create a more positive and productive work environment. In Table 1, we categorize these factors into five different categories.

Table 1: Factors of software failure

Categories	References	Failure Factors	Effects of the factors
Software /product	(Zarndt, 2011; Ahmadi et al., 2023)	Poor project management	Lack of planning and organization Ineffective use of resources Poor communication and collaboration Unrealistic expectations
	(Kuutila et al., 2020; Johnson, 2018; Dullemond et al., 2011)	Poor estimation of time and effort	Underestimating the scope of a project Unforeseen technical challenges Inaccurate resource allocation Incomplete risk assessment
Process	(Kuutila et al., 2020; Sharma and Spinellis, 2018; Lavallée and Robillard, 2015; Martini et al., 2014; Ferreira et al., 2009).	Change request/ requirements volatility	Decreased productivity, efficiency, quality Increase complexity, development time, risk of mistakes Reduced ability to plan and schedule
	(Kuutila et al., 2020; Sawyer and Southwick, 2002; Chong et al., 2010)	Task switching	Decreased productivity, quality Increased cognitive load Missed deadlines
	(Ahmadi et al., 2023)	Unclear goals and objectives:	Confusion and decreased productivity Lack of direction Changing requirements Misaligned expectations Inadequate planning: Incomplete scope definition
	(Girardi et al., 2021)	Interruption	Decreased efficiency, productivity Increased cognitive load
Tools	(Girardi et al., 2021; Khan et al., 2019; Attarzadeh, 2008)	Unavailable resources	Decreased productivity Delays Limited workforce Inadequate tools and equipment Limited access to information Frustration
Technology	(Graziotin et al., 2017)	Technical difficulties	Decrees productivity Unforeseen technical issues Integration issues Increasing the workload causing delays Requiring additional resources.
Human/people	(Gilal et al., 2019c; Sardjono and Retnowardhani, 2019; Gilal et al., 2014)	Personality/ Human psychology	Difficulty meeting deadlines Poor interpersonal skills conflicts Lead to burnout
	(Ahmadi et al., 2023; Hassan et al., 2019; Sardjono and Retnowardhani, 2019)	Poor communication and collaboration:	Decreased productivity, increased risk of mistakes Tight deadlines.
	(Hassan et al., 2019; Gupta et al., 2019; Guillaume-Joseph and Wasek, 2015; Sharma and Spinellis 2018; Lavallée and Robillard, 2015; Martini et al., 2014)	Inappropriate skills or unskilled workers	Decreased productivity Increased risk of mistakes Difficulty in problem-solving Reduced ability to meet changing requirements
	(Sharma and Spinellis, 2018; Langer et al., 2014; Lavallée and Robillard, 2015; Martini et al., 2014)	Lack of knowledge	Increased learning time: Difficulty in problem-solving Decreased efficiency Increased risk of mistakes

In conclusion, the findings of this review shed light on the key outcomes derived from the synthesis of multiple studies. These outcomes provide valuable guidance for practitioners in addressing TP in software development projects.

The findings concluded that practitioners can adopt effective project management strategies, accurate estimation techniques, and efficient requirement management practices. Additionally, mitigating distractions and interruptions, setting clear goals and objectives, and considering the personality traits of software developers can contribute to reducing TP and improving project success. The adoption of these findings has the potential to enhance productivity, software quality, and developer well-being. Moreover, by effectively managing TP, organizations can mitigate risks associated with software failures, security vulnerabilities, and financial losses. By aligning their practices with the synthesized outcomes of this review, practitioners can navigate TP challenges more effectively and achieve improved outcomes in software development endeavors.

5. Recommendation

Based on the specific failure factors identified in this paper that cause TP in software development projects, the following recommendations are made:

1. Define and consistently follow well-defined project management processes, updating them as needed.
2. Ensure projects are properly estimated and provide teams with the necessary resources, knowledge, and skills.
3. Minimize task switching by prioritizing tasks, establishing clear goals and objectives, and reducing distractions.
4. Maintain comprehensive documentation and effectively manage requirements throughout the project.

5. Emphasize the selection of software developers based on personality traits, including stress tolerance, communication skills, and teamwork abilities.

Implementing these recommendations can help reduce the impact of TP and improve the effectiveness of software development projects.

6. Conclusion

TP is one of the factors that can contribute to software failure. It can lead to developers making mistakes or cutting corners to meet deadlines. This study focuses on the factors that trigger TP, which results in software failure. TP has a negative impact on software developers, which has a negative effect on employee performance. In the field of SE, there is a plethora of work done on software failure but a shortage of work under TP. As a result, there is a greater need to focus on the factors that cause TP and find a way to reduce the negative impacts of TP on software developer's performance.

Studying TP and its impact on software development can help organizations identify potential issues and implement strategies to reduce software failure risk. It is crucial to consider organizational strategies and industry-wide initiatives. Organizations can play a pivotal role in alleviating TP by adopting effective project management practices, such as implementing agile methodologies, establishing clear communication channels, and fostering a supportive work environment. Furthermore, industry-wide initiatives can significantly contribute to managing TP in software development. Collaboration among industry stakeholders can lead to the establishment of best practices, the standardization of project management methodologies, and the promotion of industry-wide training and certification programs. By sharing experiences and insights, industry professionals can collectively work towards reducing TP and improving the overall success of software development projects. Beyond the developer level, these organizational strategies and industry-wide initiatives can create a supportive ecosystem that empowers software development teams to effectively manage TP. To minimize the impact of TP and ensure the success of software projects, understanding these factors, taking action to lessen their effects, and placing a priority on effective project management, stable requirements, and clear communication. By doing so, software development teams can improve the chances of delivering successful software products and reduce software failure risk.

Acknowledgment

We express gratitude to our university teachers for the valuable insight and expertise that greatly aided our study. We also extend our appreciation to the authors of the papers we collected as well as the participants in their research, without whom this study would not have been possible. Additionally, we acknowledge the contribution of the anonymous editors and reviewers who assisted in strengthening this study.

Compliance with ethical standards

Conflict of interest

The author(s) declared no potential conflicts of interest with respect to the research, authorship, and/or publication of this article.

References

Ahmadi A, Delkhosh F, Deshpande G, Patterson RA, and Ruhe G (2023). Learning software project management from analyzing Q&A's in the stack exchange. IEEE Access, 11: 5429-5441. https://doi.org/10.1109/ACCESS.2023.3235953

Akbar MA, Sang J, Khan AA, Mahmood S, Qadri SF, Hu H, and Xiang H (2019). Success factors influencing requirements change management process in global software development. Journal of Computer Languages, 51: 112-130. https://doi.org/10.1016/j.cola.2018.12.005

Al-Ahmad W, Al-Fagih K, Khanfar K, Alsamara K, Abuleil S, and Abu-Salem H (2009). A taxonomy of an IT project failure: Root causes. International Management Review, 5(1): 93-104.

Attarzadeh I and Ow SH (2008). Project management practices: The criteria for success or failure. Communications of the IBIMA, 1(28): 234-241. https://doi.org/10.2139/ssrn.1628612

Basten D (2017). The role of time pressure in software projects: A literature review and research agenda. eProceedings of the 12th International Research Workshop on Information Technology Project Management (IRWITPM). Available online at: https://aisel.aisnet.org/irwitpm2017/1

Byrne KA, Silasi-Mansat CD, and Worthy DA (2015). Who chokes under pressure? The Big Five personality traits and decision-making under pressure. Personality and Individual Differences, 74: 22-28. https://doi.org/10.1016/j.paid.2014.10.009 **PMid:28373740 PMCid:PMC5376094**

Chong DS, Van Eerde W, Chai KH, and Rutte CG (2010). A double-edged sword: The effects of challenge and hindrance time pressure on new product development teams. IEEE Transactions on Engineering Management, 58(1): 71-86. https://doi.org/10.1109/TEM.2010.2048914

Dullemond K, Van Gameren B, and Van Solingen R (2011). Overhearing conversations in global software engineering-requirements and an implementation. In the 7th International Conference on Collaborative Computing: Networking, Applications and Worksharing, IEEE, Orlando, USA: 1-8. https://doi.org/10.4108/icst.collaboratecom.2011.247099

Ferreira S, Collofello J, Shunk D, and Mackulak G (2009). Understanding the effects of requirements volatility in software engineering by using analytical modeling and software process simulation. Journal of Systems and Software, 82(10): 1568-1577. https://doi.org/10.1016/j.jss.2009.03.014

Ford D and Parnin C (2015). Exploring causes of frustration for software developers. In the 2015 IEEE/ACM 8th International Workshop on Cooperative and Human Aspects of Software Engineering, IEEE. Florence, Italy: 115-116. https://doi.org/10.1109/CHASE.2015.19 **PMid:25048606**

Gila AR, Jaafa J, Omar M, and Tunio M Z (2014). Impact of personality and gender diversity on software development teams' performance. In the 2014 International Conference on

Computer, Communications, and Control Technology (I4CT), IEEE, Langkawi, Malaysia: 261-265. https://doi.org/10.1109/I4CT.2014.6914186

Gilal AR, Jaafar J, Abro A, Umrani WA, Basri S, and Omar M (2017a). Making programmer effective for software development teams: An extended study. Journal of Information Science and Engineering, 33(6): 1447-1463.

Gilal AR, Jaafar J, Capretz LF, Omar M, Basri S, and Aziz IA (2018). Finding an effective classification technique to develop a software team composition model. Journal of Software: Evolution and Process, 30(1): e1920. https://doi.org/10.1002/smr.1920

Gilal AR, Jaafar J, Omar M, Basri S, and Aziz IDA (2019a). A set of rules for constructing gender-based personality types' composition for software programmer. In Proceedings of the International Conference on Data Engineering 2015, Springer, Singapore, Singapore: 363-374. https://doi.org/10.1007/978-981-13-1799-6_38

Gilal AR, Jaafar J, Omar M, Basri S, and Waqas A (2016). A rule-based model for software development team composition: Team leader role with personality types and gender classification. Information and Software Technology, 74: 105-113. https://doi.org/10.1016/j.infsof.2016.02.007

Gilal AR, Jaafar J, Omar M, Basri S, Aziz IA, Khand QU, and Hasan MH (2017b). Suitable personality traits for learning programming subjects: a rough-fuzzy model. International Journal of Advanced Computer Science and Applications, 8(8): 153-162. https://doi.org/10.14569/IJACSA.2017.080820

Gilal AR, Omar M, Jaafar J, Sharif KI, Mahesar AW, and Basri S (2017c). Software development team composition: Personality types of programmer and complex networks. In the 6th International Conference on Computing and Informatics, Kuala Lumpur, Malaysia: 153-159.

Gilal AR, Omar M, Qureshi F, and Gilal R (2019b). A decision tree model for software development teams. International Journal of Innovative Technology and Exploring Engineering 8(5s): 241-245.

Gilal AR, Tunio MZ, Waqas A, Almomani MA, Khan S, and Gilal R (2022). Task assignment and personality: Crowdsourcing software development. Research Anthology on Agile Software, Software Development, and Testing, IGI Global: 1795-1809. https://doi.org/10.4018/978-1-6684-3702-5.ch086

Gilal R, Omar M, Gilal AR, Rejab MM, Waqas A, and Sharif KIM (2019c). Can time pressure and personality make any sense together in software engineering? International Journal of Innovative Technology and Exploring Engineering, 9: 2278-3075. https://doi.org/10.35940/ijitee.A4287.119119

Girardi D, Lanubile F, Novielli N, and Serebrenik A (2021). Emotions and perceived productivity of software developers at the workplace. IEEE Transactions on Software Engineering, 48(9): 3326-3341. https://doi.org/10.1109/TSE.2021.3087906

Girardi D, Novielli N, Fucci D, and Lanubile F (2020). Recognizing developers' emotions while programming. In the ICSE '20: Proceedings of the ACM/IEEE 42nd International Conference on Software Engineering, Association for Computing Machinery, Seoul, South Korea: 666-677. https://doi.org/10.1145/3377811.3380374

Graziotin D, Fagerholm F, Wang X, and Abrahamsson P (2017). On the unhappiness of software developers. In Proceedings of the 21st International Conference on Evaluation and Assessment in Software Engineering, Association for Computing Machinery, Karlskrona, Sweden: 324-333. https://doi.org/10.1145/3084226.3084242

Guillaume-Joseph G and Wasek JS (2015). Improving software project outcomes through predictive analytics: Part 1. IEEE Engineering Management Review, 43(3): 26-38. https://doi.org/10.1109/EMR.2015.2469451

Gupta SK, Gunasekaran A, Antony J, Gupta S, Bag S, and Roubaud D (2019). Systematic literature review of project failures: Current trends and scope for future research. Computers and Industrial Engineering, 127: 274-285. https://doi.org/10.1016/j.cie.2018.12.002

Hassan M, Hussain M, and Irfan M (2019). A policy recommendations framework to resolve global software development issues. In the 2019 International Conference on Innovative Computing (ICIC), IEEE, Lahore, Pakistan: 1-10. https://doi.org/10.1109/ICIC48496.2019.8966719

Ibraigheeth M and Fadzli SA (2019). Core factors for software projects success. JOIV: International Journal on Informatics Visualization, 3(1): 69-74. https://doi.org/10.30630/joiv.3.1.217

Jayatilleke S and Lai R (2018). A systematic review of requirements change management. Information and Software Technology, 93: 163-185. https://doi.org/10.1016/j.infsof.2017.09.004

Johnson J (2018). CHAOS report: Decision latency theory: It is all about the interval. The Standish Group, Boston, USA.

Khan TI, Khan AZ, and Khan S (2019). Effect of time pressure on organizational citizenship behavior: Moderating role of agreeableness. Sir Syed Journal of Education and Social Research (SJESR), 2(1): 140-156.

Kitchenham B and Charters S (2007). Guidelines for performing systematic literature reviews in software engineering. Joint Report, Keele University and Durham University, Newcastle, UK.

Krasner H (2021). The cost of poor software quality in the US: A 2020 report. Consortium for Information and Software Quality: 1-46. Available online at: https://www.it-cisq.org/pdf/CPSQ-2020-report.pdf

Kuutila M, Mäntylä M, Farooq U, and Claes M (2020). Time pressure in software engineering: A systematic review. Information and Software Technology, 121: 106257. https://doi.org/10.1016/j.infsof.2020.106257

Kuutila M, Mäntylä MV, Claes M, and Elovainio M (2017). Reviewing literature on time pressure in software engineering and related professions: Computer assisted interdisciplinary literature review. In the 2017 IEEE/ACM 2nd International Workshop on Emotion Awareness in Software Engineering, IEEE, Buenos Aires, Argentina: 54-59. https://doi.org/10.1109/SEmotion.2017.11

Langer N, Slaughter SA, and Mukhopadhyay T (2014). Project managers' practical intelligence and project performance in software offshore outsourcing: A field study. Information Systems Research, 25(2): 364-384. https://doi.org/10.1287/isre.2014.0523

Lavallée M and Robillard PN (2015). Why good developers write bad code: An observational case study of the impacts of organizational factors on software quality. In the 2015 IEEE/ACM 37th IEEE International Conference on Software Engineering, IEEE, Florence, Italy, 1: 677-687. https://doi.org/10.1109/ICSE.2015.83

Mäntylä MV, Petersen K, Lehtinen TO, and Lassenius C (2014). Time pressure: A controlled experiment of test case development and requirements review. In Proceedings of the 36th International Conference on Software Engineering, Association for Computing Machinery, Hyderabad, India: 83-94. https://doi.org/10.1145/2568225.2568245

Martini A, Bosch J, and Chaudron M (2014). Architecture technical debt: Understanding causes and a qualitative model. In the 2014 40th EUROMICRO Conference on Software Engineering and Advanced Applications, IEEE, Verona, Italy: 85-92. https://doi.org/10.1109/SEAA.2014.65

Maruping LM, Venkatesh V, Thatcher SM, and Patel PC (2015). Folding under pressure or rising to the occasion? Perceived time pressure and the moderating role of team temporal leadership. Academy of Management Journal, 58(5): 1313-1333. https://doi.org/10.5465/amj.2012.0468

Ohly S and Fritz C (2010). Work characteristics, challenge appraisal, creativity, and proactive behavior: A multi-level study. Journal of Organizational Behavior, 31(4): 543-565. https://doi.org/10.1002/job.633

Raunak MS and Binkley D (2017). Agile and other trends in software engineering. In the 2017 IEEE 28th Annual Software Technology Conference (STC), IEEE, Gaithersburg, USA: 1-7. https://doi.org/10.1109/STC.2017.8234457

Sardjono W and Retnowardhani A (2019). Analysis of failure factors in information systems project for software implementation at the organization. In the 2019 International Conference on Information Management and Technology (ICIMTech), IEEE, Jakarta/Bali, Indonesia, 1: 141-145. https://doi.org/10.1109/ICIMTech.2019.8843725

Sawyer S and Southwick R (2002). Temporal issues in information and communication technology-enabled organizational change: Evidence from an enterprise systems implementation. The Information Society, 18(4): 263-280. https://doi.org/10.1080/01972240290075110

Shafiq M, Zhang Q, Akbar MA, Khan AA, Hussain S, Amin FE, Khan A, and Soofi AA (2018). Effect of project management in requirements engineering and requirements change management processes for global software development. IEEE Access, 6: 25747-25763. https://doi.org/10.1109/ACCESS.2018.2834473

Shah H, Harrold MJ, and Sinha S (2014). Global software testing under deadline pressure: Vendor-side experiences. Information and Software Technology, 56(1): 6-19. https://doi.org/10.1016/j.infsof.2013.04.005

Sharma T and Spinellis D (2018). A survey on software smells. Journal of Systems and Software, 138: 158-173. https://doi.org/10.1016/j.jss.2017.12.034

Shepherd DA, Patzelt H, Williams TA, and Warnecke D (2014). How does project termination impact project team members? Rapid termination, 'creeping death,' and learning from failure. Journal of Management Studies, 51(4): 513-546. https://doi.org/10.1111/joms.12068

Singh MP and Vyas R (2012). Requirements volatility in software development process. International Journal of Soft Computing, 2(4): 259-264.

Smith P (2014). Project cost management–Global issues and challenges. Procedia-Social and Behavioral Sciences, 119: 485-494. https://doi.org/10.1016/j.sbspro.2014.03.054

Speier-Pero C (2019). Using aggregated data under time pressure: a mechanism for coping with information overload. Journal of Decision Systems, 28(2): 82-100. https://doi.org/10.1080/12460125.2019.1623533

Thomas SM (2021). Re: Patterns of mechanical thrombectomy for stroke before and after the 2015 pivotal trials and US national guideline update. Journal of Stroke and Cerebrovascular Diseases, 30(2): 105493. https://doi.org/10.1016/j.jstrokecerebrovasdis.2020.105493 **PMid:33253983**

Verner J, Sampson J, and Cerpa N (2008). What factors lead to software project failure? In the 2008 Second International Conference on Research Challenges in Information Science, IEEE, Marrakech, Morocco: 71-80. https://doi.org/10.1109/RCIS.2008.4632095

Zahid AHA, Haider MW, Farooq MS, Abid A, and Ali A (2018). A critical analysis of software failure causes from project management perspectives. VFAST Transactions on Software Engineering, 6(1): 62–68.

Zarndt F (2011). Project management 101: Plan well, communicate a lot, and don't forget acceptance criteria! OCLC Systems and Services: International Digital Library Perspectives, 27(3): 170-174. https://doi.org/10.1108/10650751111164542

Zhu YM (2017). Failure-modes-based software reading. In: Zhu YM (Ed.), Failure-modes-based software reading: 29-37. Springer International Publishing, New York, USA. https://doi.org/10.1007/978-3-319-65103-3

Zykov SV and Attakorah JA (2020). Survey of human factors in crisis responsive software development. ArXiv Preprint ArXiv:2007.12019. https://doi.org/10.48550/arXiv.2007.12019