Technoarete Transactions on Advances in Computer Applications (TTACA)
Vol-1, Issue-3, September 2022
e-ISSN: 2583-3472

An Analysis on the Techniques for Spotting and Handling Conflicts in Software Design

Dr.Mohd Zuber [1*], Dr.K.Balaji [2]

[1] Madhyanchal professional university, India.
[2] Surana College, Bangalore, India.
*Corresponding Author Email: [1] Mzmkhanugc@gmail.com

Abstract

This paper discusses the techniques for spotting and handling conflicts in software design. It examines the various approaches that can be used to identify potential conflicts and how those conflicts can be addressed. It also explores the importance of conflict management in software design, as well as the tools and techniques that can be used to help resolve conflicts. Finally, the paper provides an overview of some of the best practices that can be employed to ensure that conflicts are handled in an effective manner.

Keywords

Conflict Resolution, Software Design, Spotting Techniques.

INTRODUCTION

Software design is the process of creating a software system that meets a set of specific requirements. As part of this process, conflicts can arise between different stakeholders, including developers, users, and customers. In order to ensure the successful delivery of a software system, it is important to identify and manage conflicts as early as possible. This paper will provide an analysis on the techniques for spotting and handling conflicts in software design.

Spotting Conflicts

The first step in managing conflicts in software design is to identify them. There are several techniques that can be used to spot potential conflicts, including brainstorming, stakeholder analysis, and risk analysis.

Brainstorming is a creative problem-solving technique that encourages stakeholders to brainstorm ideas and solutions to potential conflicts. This allows all stakeholders to be aware of potential conflicts and encourages them to work together to come up with solutions.

Stakeholder analysis is a technique used to identify stakeholders who may have a vested interest in the software project. This analysis can help identify potential conflicts between stakeholders, such as competing interests or incompatible goals.

Risk analysis is a technique used to identify potential risks that may arise during the software design process. This analysis helps to identify potential conflicts that may arise due to the complexity of the project or due to unforeseen circumstances.

Handling Conflicts

Once conflicts have been identified, it is important to address them in a timely manner. There are several techniques that can be used to handle conflicts in software design, including negotiation, compromise, and arbitration.

Negotiation is a technique used to reach agreement between two or more parties. This involves discussing the conflict and attempting to come up with a mutually beneficial solution. Compromise is a technique used to reach agreement between two or more parties by compromising on certain aspects of the conflict. This involves each party agreeing to accept some of the losses in order to reach an agreement (Liu et al. 2020). Arbitration is a technique used to resolve conflicts by having an impartial third party make a decision. This is usually done by an impartial expert who is knowledgeable in the subject matter.

Software design conflicts can be difficult to spot and manage. In order to ensure the successful delivery of a software system, it is important to identify and manage conflicts as early as possible. This paper provided an analysis on the techniques for spotting and handling conflicts in software design, including brainstorming, stakeholder analysis, risk analysis, negotiation, compromise, and arbitration.

Definition of Conflict

Conflict in software design occurs when two or more interests or ideas clash. This clash can arise from different requirements, different approaches to solving a problem, or simply from different personal preferences. It can cause delays in the progress of a project or even lead to its complete failure.

Techniques for Spotting and Handling Conflicts

1. Identifying Stakeholders: The first step in spotting and handling conflicts in software design is to identify the stakeholders. Stakeholders are people who have a vested interest in the success of the project, including users, developers, and stakeholders. By identifying the stakeholders, it is easier to spot potential issues before they become a major problem.

2. Setting Clear Goals and Expectations: Setting clear goals and expectations at the start of a project can help to avoid conflicts. When everyone is on the same page about what is expected of them, it makes it easier to identify any potential conflicts before they become a major problem.
3. Establishing an Open Dialogue: Establishing an open dialogue between all stakeholders is an important part of spotting and handling conflicts. By allowing all stakeholders to voice their ideas and concerns, it is easier to identify potential conflicts before they become a major problem.
4. Encouraging Collaboration: Encouraging collaboration between all stakeholders is another important step in spotting and handling conflicts. By working together to identify solutions to any potential conflicts, it is easier to resolve them before they become a major problem.
5. Establishing Conflict Resolution Processes: Establishing a conflict resolution process is an important step in spotting and handling conflicts (Aghajani et al. 2019). This process should include a set of rules and procedures for resolving conflicts quickly and effectively.

Conflict in software design can be a major issue, but it can be avoided with the right techniques. By identifying stakeholders, setting clear goals and expectations, establishing an open dialogue, encouraging collaboration, and establishing conflict resolution processes, conflicts in software design can be spotted and handled before they become a major problem.

Types of Conflict

Software design conflicts can broadly be divided into two categories: technical conflicts and interpersonal conflicts. Technical conflicts arise when two design options appear to be equally viable, but incompatibilities arise when the two are combined. These conflicts often require the designers to choose one option over the other. Interpersonal conflicts, on the other hand, are disagreements between two or more team members about the best way to approach a design.

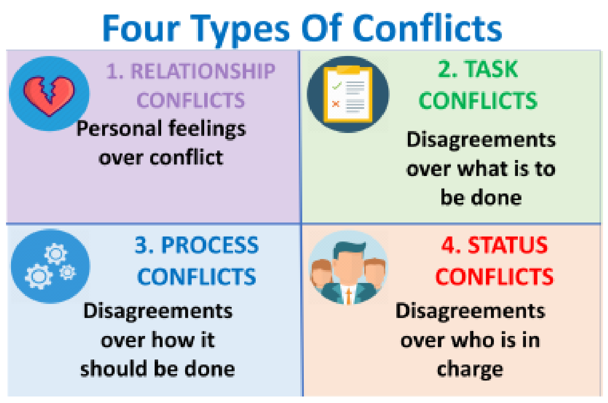
Four Types Of Conflicts
1. RELATIONSHIP CONFLICTS
Personal feelings over conflict
2. TASK CONFLICTS
Disagreements over what is to be done
3. PROCESS CONFLICTS
Disagreements over how it should be done
4. STATUS CONFLICTS
Disagreements over who is in charge

Figure 1 : Types of Conflicts
(Source : Berger et al. 2020)

Techniques for Spotting and Handling Conflicts

The most effective way to spot and handle conflicts in software design is through effective communication and collaboration. When working as a team, it is important to have open and honest conversations, as this helps everyone to understand the different perspectives and opinions at play. Additionally, it is important to have a clear understanding of the overall design goals. With this knowledge, the team can work together to identify potential conflicts before they arise.

When a conflict does arise, it's important for the team to work together to find a resolution. This may involve brainstorming different solutions, running through various scenarios, or even working with a third-party to gain an objective perspective. Ultimately, it's important to reach a consensus that everyone can agree on and move forward with.

In addition to communication and collaboration, it's also important to have effective conflict-management tools in place. This can include setting clear expectations, creating a timeline for project deliverables, and establishing a process for resolving disagreements. All of these tools can help the team to identify, address, and resolve conflicts quickly and efficiently.

Conflicts in software design can be difficult to manage, but with effective communication, collaboration, and conflict-management tools in place, they can be effectively addressed. By having open and honest conversations, understanding the overall design goals, and having a clear process for resolving disagreements, teams can work together to reach a consensus and move forward with the project.

CAUSES OF CONFLICT

Table 1: Causes Impact

Causes	Concern
Communication gap	33%
Contradiction of understanding	41%

(Source: Created by Author)

Poor Communication

Conflicts in software design can arise from a variety of sources. These sources include miscommunication and misunderstanding among stakeholders, competing priorities and goals, mismatched expectations, lack of clarity in project scope and objectives, and lack of resources or technical skills. Additionally, conflicts can arise from the inherent complexity of software design and development, including the need to integrate multiple technologies, the unpredictable nature of software development, and the large number of stakeholders involved in the process.

Techniques for Spotting and Handling Conflicts

1. Communication: Good communication is essential for spotting and resolving conflicts in software design. Effective communication involves listening to other stakeholders, asking questions to clarify expectations, and providing feedback to ensure that everyone is on the same page. Additionally, it is important to ensure that all stakeholders understand the project objectives and are

aware of the resources and capabilities available.

2. Prioritization: Prioritizing tasks and goals is an important technique for spotting and handling conflicts in software design. By understanding the project’s overall objectives, stakeholders can determine which tasks are the most important, and agree on which tasks can be delayed or postponed. Additionally, it is important to prioritize tasks based on their importance to the project’s overall success.
3. Collaboration: Collaboration is key to resolving conflicts in software design. By working together, stakeholders can identify areas of disagreement, brainstorm potential solutions, and work towards a shared understanding of the project’s goals and objectives (Mascardi et al. 2019). Additionally, collaboration can help stakeholders come to an agreement on how to manage the project’s resources and capabilities.
4. Conflict Resolution: When conflicts arise, it is important to take steps to resolve them as quickly as possible. This may involve engaging in open and honest conversations with stakeholders, developing a plan to address the conflict, and ensuring that all stakeholders are comfortable with the solutions proposed. Additionally, it is important to make sure that all stakeholders are aware of the potential consequences of not resolving the conflict.
5. Flexibility: Finally, it is important to remain flexible when dealing with conflicts in software design. Stakeholders should be open to considering different solutions and approaches, and be willing to make changes if necessary. Additionally, stakeholders should be willing to compromise and make adjustments to the project’s goals and objectives if necessary.

Ambiguous Specifications

Conflicts in software design can be caused by ambiguous specifications and can be difficult to spot and handle. One technique for spotting and handling conflicts is to ensure that all stakeholders are involved in the design process and that their requirements and expectations are adequately documented. This helps ensure that stakeholders are on the same page and that any potential conflicts between their requirements can be identified and resolved early on.

Another technique for spotting and handling conflicts is to use testing and verification techniques that are designed to uncover any issues with the design (Lo et al. 2021). This can include using automated testing tools to examine the code for any potential issues, as well as manual testing to ensure that the software meets the requirements. Additionally, code reviews can be used to identify any potential conflicts in the design.

Finally, the use of design patterns can help to minimize conflicts in software design. Design patterns are reusable solutions to common problems that can help ensure that the software meets the requirements and avoids any potential conflicts. Additionally, design patterns can also help to reduce complexity and enable better maintainability.

The ambiguity of software design specifications can lead to conflicts between parties involved in the design and development process. Jabbar et al. (2020) stated that to prevent such conflicts, it is important to identify and address potential issues before they become a problem. There are a number of techniques that can be used to identify and handle conflicts in software design.

One technique is to ensure that all stakeholders in the software design process are involved in the specification process. This allows everyone to have a better understanding of the problem and ensures that everyone is on the same page. This also allows for easier communication between the stakeholders and makes it easier to resolve any potential conflicts.

Another technique is to use a common language for the specification. This helps to reduce the ambiguity of the software design and makes it easier for everyone to understand. It also minimizes the chances of discrepancies between the different stakeholders.

Another technique is to use diagrams and visualizations to help explain the specifications. This can make it easier for everyone to understand the specifications and helps to reduce any potential confusion.

Finally, it is important to have a process for resolving any conflicts that do arise. This can involve the stakeholders discussing the issue and coming to a consensus on how to proceed (Charlton et al. 2021). Having a clear process in place can help to avoid any misunderstandings and ensure that the design and development process runs as smoothly as possible.

Software design is a complex process that requires frequent collaboration between different stakeholders, and conflicts are bound to arise. Conflict resolution is a critical part of software design, as it helps to ensure that the final product meets the requirements set out by the stakeholders. In order to identify and handle conflicts, software designers must be aware of the various techniques available to them.

One of the most common techniques for spotting and handling conflicts in software design is to identify and analyze ambiguous specifications. Ambiguity can arise from a variety of sources, such as misunderstandings between different stakeholders, conflicting opinions, or unclear requirements. By looking at the specifications from multiple perspectives, software designers can often identify ambiguities and resolve them before they become larger issues.

Another technique for spotting and handling conflicts in software design is to identify and address potential sources of conflict early on in the design process (Naseer et al. 2020). This includes understanding stakeholder interests and expectations, identifying areas of potential disagreement, and establishing ground rules for resolving conflicts. By addressing potential sources of conflict early on, software designers can help to prevent disputes from escalating into larger issues that could delay the development process.

Finally, software designers can use collaborative problem-solving techniques to resolve conflicts. This includes using active listening to understand the perspectives of all stakeholders, brainstorming ideas to identify potential solutions, and negotiating an agreement that meets the interests of all parties involved. By using collaborative problem-solving techniques, software designers can ensure that conflicts are resolved in a timely and efficient manner.

Software design requires frequent collaboration between different stakeholders, and conflicts are bound to arise. As per Vilutiene et al. (2019) by understanding the various techniques available for spotting and handling conflicts, software designers can help ensure that the development process is efficient and successful. These techniques include identifying and analyzing ambiguous specifications, addressing potential sources of conflict early on, and using collaborative problem-solving techniques to resolve disputes.

Overlapping Responsibilities

One of the most common techniques for spotting and handling conflicts in software design is to identify overlapping responsibilities. This is especially important in software design, as there can be a lot of stakeholders with different interests and objectives that can lead to conflicting requirements. It is important to identify where responsibilities overlap in order to ensure that all stakeholders are included in the software design process and that their objectives are met.

In order to identify overlapping responsibilities, it is important to take a step back and analyze the big picture. This includes looking at the various stakeholders involved in the software design process and their roles within the project. It is also important to look at the different requirements and objectives that each stakeholder has for the software design and how they interact with one another. Once the big picture has been analyzed, it is much easier to identify areas where responsibilities overlap.

Once overlapping responsibilities have been identified, it is important to handle them in a way that is fair and beneficial to all stakeholders. This could involve assigning specific tasks to different stakeholders, making sure that all stakeholders are aware of the requirements of each other, or even coming up with a compromise that all stakeholders can agree on (Leite et al. 2021). It is important to keep communication open and to ensure that all stakeholders are able to voice their opinions and concerns.

Overall, it is important to identify overlapping responsibilities in software design in order to ensure that all stakeholders are included in the process and that their objectives are met. By analyzing the big picture and communicating openly, it is possible to handle overlapping responsibilities in a way that is beneficial to all stakeholders.

STRATEGIES FOR CONFLICT RESOLUTION

Table 2: Conflicts Resolving Scope

Way of Resolving	Standard Benefits
Improved expecting	17%
Source Knowledge	32%

(Source: Created by Author)

Establish Clear Expectations

Having clear expectations is essential for spotting and handling conflicts in software design. This involves communicating the purpose of the project, the timeline, and the individual roles and responsibilities of each team member. This helps to ensure that everyone is on the same page and working towards the same goal. It also allows for open dialogue between team members and management, which helps to identify potential issues early on and avoid conflicts from arising. Establishing a shared understanding of the project goal and timeline helps to keep everyone accountable and on track. Additionally, expectations should be flexible enough to accommodate changes in the scope or timeline of the project. By having clearly defined expectations, teams can proactively plan for potential conflicts and come up with strategies to address them.

The first step in spotting and handling conflicts in software design is to establish clear expectations. This means communicating to all stakeholders what the goals of the project are, what the timeline is, and what the budget is. Setting expectations early on in the process can help to prevent misunderstandings down the line. Having a unified vision of the project at the start can also help to prevent design conflicts from occurring. Additionally, establishing who is responsible for what can help to ensure everyone is on the same page and that there is a clear understanding of who is responsible for what tasks. This can help to prevent conflicts from arising in the first place.

In order to effectively spot and handle conflicts in software design, it is important to establish clear expectations between stakeholders. This begins with setting out clear goals and objectives for the design and development process(Dargan et al. 2020). Stakeholders should be made aware of their roles and responsibilities, the timeline for the project, the budget, and any other relevant details. It is also important for stakeholders to understand the scope of the project and the expected outcome. By establishing clear expectations, stakeholders can be held accountable for their contributions and any potential conflicts can be addressed before they arise. Additionally, communication between stakeholders should be encouraged in order to keep everyone on the same page and ensure that all parties are aware of any changes that may occur.

Identify the Source of Conflict

Conflict in software design can originate from a variety of sources. These can include competing objectives between stakeholders, cultural differences between teams, lack of clarity around project goals, and disagreements over technology choices (Eck et al. 2019). Other sources of conflict can include disagreements over how features should be implemented, how resources should be allocated, or how timelines should be managed. Additionally, conflicts arise when there is a lack of communication or collaboration between teams, when teams have difficulty understanding each other's perspectives, or when teams lack the tools and processes needed to effectively collaborate.

The source of conflict in software design can vary depending on the particular project. Common sources of conflict include communication issues between designers and developers, resource allocation problems, and differences in design goals. Communication issues often arise due to misunderstandings between team members, or a lack of communication between designers and developers. Sharma et al. (2020) stated that resource allocation problems may arise due to limited budgets or a lack of skilled workers. Lastly, differences in design goals can lead to disagreements between team members regarding the overall direction of the project.

Conflicts in software design can arise from a variety of sources. These include disagreements between software engineers, disagreements between software engineers and stakeholders, and conflicts between different stakeholders. Additionally, conflicts may arise due to differences in software design philosophies, coding styles, feature sets, and other factors. Consequently, it is important to be able to identify and address the source of conflict in order to prevent it from escalating and damaging the project.

The source of conflict in software design can stem from a variety of sources. These include the team's individual goals and values, the stakeholders' goals, the varying technical skills of the team members, and the design of the software itself (Jha et al. 2019). Disputes can also arise from different interpretations of the same problem or different approaches to solving a problem. Moreover, disputes may also arise from a lack of communication and understanding between team members or between the team and stakeholders. Ultimately, the source of conflict can vary depending on the context, but the underlying causes are typically rooted in the different goals and values of the individuals andstakeholders involved.

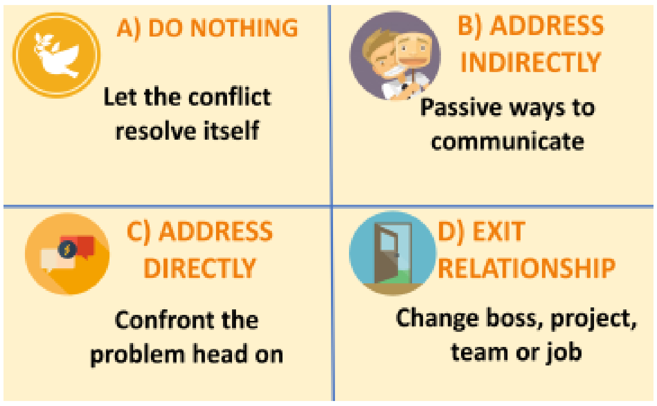
A) DO NOTHING
Let the conflict resolve itself
B) ADDRESS INDIRECTLY
Passive ways to communicate
C) ADDRESS DIRECTLY
Confront the problem head on
D) EXIT RELATIONSHIP
Change boss, project, team or job

Figure 2 : Conflicts Resolving Scope
(Source: Aledhari et al. 2020)

Take Time to Listen

One of the most important techniques for spotting and handling conflicts in software design is to take time to listen. This involves actively listening to the team members involved in a project, as well as any stakeholders, to identify potential conflict points. This means actively engaging with people, asking questions, and taking notes. It is important to try to understand the different perspectives at play, in order to identify potential areas of conflict and begin to work towards resolving them. This can be done through discussions, brainstorming sessions, and other collaborative techniques. By listening carefully, the team can avoid potential conflicts before they become too big of an issue, and also identify areas of agreement in order to move forward with the project.

One of the most important techniques for spotting and handling conflicts in software design is to take the time to listen. It is important for the software designers to actively listen to all stakeholders and to take into consideration their needs and opinions. By doing this, software designers can identify potential conflicts before they arise and come up with a plan on how to handle them. Taking the time to really listen to all stakeholders can also help to resolve conflicts before they even begin, as it can foster open communication and create a shared understanding between all parties (Vassllo et al. 2020). Furthermore, taking the time to listen can also help to build trust between stakeholders, which can be beneficial for the software design process and the eventual outcome.

Brainstorm Solutions

As per clarinval et al. (2020) one technique for spotting and handling conflicts in software design is to identify the stakeholders involved in the project. Stakeholders can include the software developers, product owners, and users. By understanding the different needs and interests of each stakeholder, it is possible to identify potential conflicts before they become a problem. Additionally, communication between the stakeholders should be established early on in the project, to ensure that any conflicts can be addressed promptly.

Another technique for spotting and handling conflicts in software design is to use automated tools to track changes and to detect any potential conflicts. By using automated tools, it is possible to monitor changes to the code and alert the team when a potential conflict arises. This can help to reduce the amount of time spent manually searching through code.

Finally, a third technique for spotting and handling conflicts in software design is to use version control systems. This allows developers to easily track changes to the code and to backtrack if any conflicts arise (Banga et al. 2021). Version control systems also make it easier to collaborate on projects, as they provide an easy way to share code and track changes.

Create a Win-Win Solution

Creating a win-win solution is one of the most effective techniques for spotting and handling conflicts in software design. This approach involves all parties coming together to identify a solution that benefits everyone involved. As per Tobias and Spanier (2020) the goal is to find a way to satisfy both sides, such that both parties gain something from the agreement. This can be done by brainstorming possible solutions, looking for common ground, and negotiating to reach a mutually beneficial outcome. With this technique, it is important to ensure that the interests of both parties are considered and that all parties are given a chance to express their opinions. This approach can help to create a positive environment that encourages collaboration and innovation, allowing for more efficient and effective software design.

The best way to handle conflicts in software design is to create a win-win solution. This involves both parties discussing the issue and coming to an agreement that satisfies everyone's needs. In creating a win-win solution, both parties should focus on the interests of each other and the overall goal of the project, instead of focusing too much on individual wants and needs (Gul et al. 2019). This can be achieved through active listening, open communication and compromise. It is important to recognize that both parties may have legitimate points of view and that there may be a solution that works for both. Through compromise and open dialogue, it is possible to come to an agreement that everyone can be happy with.

CONCLUSION

In conclusion, there are a variety of techniques for spotting and handling conflicts in software design. Implementing a collaborative and iterative approach to system design is essential to successful conflict resolution. Additionally, it is important to ensure that all stakeholders in the design process are kept informed of any changes to designs and that the team is open to feedback. With these strategies in place, software design teams can identify and resolve conflicts quickly and efficiently.

In conclusion, the techniques for spotting and handling conflicts in software design are essential for successful development. Identifying the source of the conflict, assessing its scope and severity, and understanding the underlying needs of stakeholders is a critical first step. Once the source of the conflict is understood, the team can work together to brainstorm and develop solutions. When conflicts arise, it is important to remain calm, communicate clearly, and focus on the best interests of the project. By following these techniques, teams can effectively address conflicts and ensure the success of their software designs.

In conclusion, conflict resolution in software design is an important part of the development process. There are a variety of techniques that can be used to identify conflicts early on and to provide solutions to them. These techniques include peer reviews, user feedback, prototyping, and automated conflict detection. By leveraging these techniques, software developers can ensure that their designs are optimized for the best possible user experience.

REFERENCES

[1] Aghajani, E., Nagy, C., Vega-Márquez, O.L., Linares-Vásquez, M., Moreno, L., Bavota, G. and Lanza, M., 2019, May. Software documentation issues unveiled. In *2019 IEEE/ACM 41st International Conference on Software Engineering (ICSE)* (pp. 1199-1210). IEEE.

[2] Aledhari, M., Razzak, R., Parizi, R.M. and Saeed, F., 2020. Federated learning: A survey on enabling technologies, protocols, and applications. *IEEE Access*, *8*, pp.140699-140725.

[3] Banga, H.K., Kalra, P., Kumar, R., Singh, S. and Pruncu, C.I., 2021. Optimization of the cycle time of robotics resistance spot welding for automotive applications. *Journal of Advanced Manufacturing and Processing*, *3*(3), p.e10084.

[4] Berger, T., Steghöfer, J.P., Ziadi, T., Robin, J. and Martinez, J., 2020. The state of adoption and the challenges of systematic variability management in industry. *Empirical Software Engineering*, *25*(3), pp.1755-1797.

[5] Charlton, N.P., Swain, J.M., Brozek, J.L., Ludwikowska, M., Singletary, E., Zideman, D., Epstein, J., Darzi, A., Bak, A., Karam, S. and Les, Z., 2021. Control of severe, life-threatening external bleeding in the out-of-hospital setting: a systematic review. *Prehospital Emergency Care*, *25*(2), pp.235-267.

[6] Clarinval, A., Simonofski, A., Vanderose, B. and Dumas, B., 2020. Public displays and citizen participation: a systematic literature review and research agenda. *Transforming Government: People, Process and Policy*.

[7] Dargan, S., Kumar, M., Ayyagari, M.R. and Kumar, G., 2020. A survey of deep learning and its applications: a new paradigm to machine learning. *Archives of Computational Methods in Engineering*, *27*(4), pp.1071-1092.

[8] Eck, M., Palomba, F., Castelluccio, M. and Bacchelli, A., 2019, August. Understanding flaky tests: The developer's perspective. In *Proceedings of the 2019 27th ACM Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering* (pp. 830-840).

[9] Gul, M., Hameed, M.H., Nazeer, M.R., Ghafoor, R. and Khan, F.R., 2019. Most effective method for the management of physiologic gingival hyperpigmentation: A systematic review and meta-analysis. *Journal of Indian Society of Periodontology*, *23*(3), p.203.

[10] Jabbar, A., Akhtar, P. and Dani, S., 2020. Real-time big data processing for instantaneous marketing decisions: A problematization approach. *Industrial Marketing Management*, *90*, pp.558-569.

[11] Jha, S., Kumar, R., Abdel-Basset, M., Priyadarshini, I., Sharma, R. and Long, H.V., 2019. Deep learning approach for software maintainability metrics prediction. *Ieee Access*, *7*, pp.61840-61855.

[12] Leite, L., Pinto, G., Kon, F. and Meirelles, P., 2021. The organization of software teams in the quest for continuous delivery: A grounded theory approach. *Information and Software Technology*, *139*, p.106672.

[13] Liu, Z., Chen, C., Wang, J., Huang, Y., Hu, J. and Wang, Q., 2020, September. Owl eyes: Spotting ui display issues via visual understanding. In *2020 35th IEEE/ACM International Conference on Automated Software Engineering (ASE)* (pp. 398-409). IEEE.

[14] Lo, S.K., Lu, Q., Wang, C., Paik, H.Y. and Zhu, L., 2021. A systematic literature review on federated machine learning:

From a software engineering perspective. *ACM Computing* Surveys *(CSUR)*, *54*(5), pp.1-39.

[15] Mascardi, V., Weyns, D., Ricci, A., Earle, C.B., Casals, A., Challenger, M., Chopra, A., Ciortea, A., Dennis, L.A., Díaz, Á.F. and El Fallah-Seghrouchni, A., 2019. Engineering multi-agent systems: State of affairs and the road ahead. *ACM SIGSOFT Software Engineering Notes*, *44*(1), pp.18-28.

[16] Naseer, M., Zhang, W. and Zhu, W., 2020. Early prediction of a team performance in the initial assessment phases of a software project for sustainable software engineering education. *Sustainability*, *12*(11), p.4663.

[17] Sharma, T., Singh, P. and Spinellis, D., 2020. An empirical investigation on the relationship between design and architecture smells. *Empirical Software Engineering*, *25*(5), pp.4020-4068.

[18] Tobias, G. and Spanier, A.B., 2020. Developing a mobile app (iGAM) to promote gingival health by professional monitoring of dental selfies: user-centered design approach. *JMIR mHealth and uHealth*, *8*(8), p.e19433.

[19] Vassallo, C., Panichella, S., Palomba, F., Proksch, S., Gall, H.C. and Zaidman, A., 2020. How developers engage with static analysis tools in different contexts. *Empirical Software Engineering*, *25*(2), pp.1419-1457.

[20] Vilutiene, T., Kalibatiene, D., Hosseini, M.R., Pellicer, E. and Zavadskas, E.K., 2019. Building information modeling (BIM) for structural engineering: A bibliometric analysis of the literature. *Advances in Civil Engineering*, *2019*.