Patterns for Improving User Contribution

Mohammad Daud Haiderzai
Valentino Vranić
haiderzai@gmail.com
vranic@stuba.sk
Institute of Informatics, Information Systems and Software Engineering
Faculty of Informatics and Information Technologies
Slovak University of Technology in Bratislava
Slovakia

ABSTRACT

User contributions is an important part in software development, the pursuit of identifying effective means of user contributions in software development is a crucial area of interest for software companies and their development teams. Determining the most opportune moments, locations, and methods for such contributions presents a significant challenge for organizations, which necessitates the exploration of generic and proven solutions. Applying patterns, established and observed solutions may provide the answers to such challenges and similar ones. This paper offers insights into the practical implementation of seven newly discovered organizational patterns in a real-life environment. The patterns are comprehensively described, accompanied by illustrative examples, and subsequently applied in actual practices.

CCS CONCEPTS

• **Software and its engineering → Patterns**.

KEYWORDS

organizational patterns, software development, software engineering

ACM Reference Format:
Mohammad Daud Haiderzai and Valentino Vranić. 2023. Patterns for Improving User Contribution. In *28th European Conference on Pattern Languages of Programs (EuroPLoP 2023), July 5–9, 2023, Irsee, Germany*. ACM, New York, NY, USA, 9 pages. https://doi.org/10.1145/3628034.3628064

1 INTRODUCTION

User contribution is widely acknowledged as an essential component of software development by software companies, researchers, and practitioners. It is crucial to comprehend the areas in which users can contribute and how to determine their contributions in software development, which necessitates attention from both software developers and researchers.

Users can contribute at various stages of software development, ranging from conception to final implementation. However, it is crucial to identify the user contributions that can be efficient and effective in specific development processes. Providing solutions to problems through proven strategies in a particular context is referred to as patterns [1]. Patterns are an effective means of improving user contributions and resolving related issues. The pattern community has established patterns as the best solutions to such types of problems in real-life practice [7].

Although problems can occur in different contexts, organizing and applying the best strategies as solutions can be facilitated through patterns. These multiple solutions can be transferred from real-life experience and recurring solution strategies. Patterns are documented and written based on real-life practices, experiences, and proven strategies described by many authors and published in conferences or books. Observing and documenting prominent, recurring, generic, and proven approaches [1, 6, 8, 11, 12] are considered essential in documenting and writing these patterns.

The contributions of users in business requirements and software development efficiency have been studied, and patterns have been proposed that can help organizations identify users' main contributions and improve the software development process. Providing opportunities to efficiently utilize users in the development process is vital.

Due to the scarcity of user contributions in previous research, a closer assessment of their importance in both practical and theoretical contexts is required. Identifying unstructured difficulties in software development resulting from user interaction is critical for understanding and increasing user participation. In this paper, the best practices of improving user contribution in software development are explained and documented in the form of patterns.

The rest of the paper is structured as follows. Section 2 explains the importance of user contribution and their impact on software development team. Sections 3 to 9 describe all the discovered and documented patterns. Section 10 draws the final conclusions.

2 IMPORTANCE OF USER CONTRIBUTION

The role of user contributions in software development has been recognized as crucial from the very inception of software product development. Software companies have long involved users in all phases of software engineering and considered their participation a key success factor for product development. User contributions in software development are commonly acknowledged by software

EuroPLoP 2023, July 5–9, 2023, Irsee, Germany

ACM ISBN 979-8-4007-0040-8/23/07…$15.00
https://doi.org/10.1145/3628034.3628064

companies and developers during the business requirements gathering process, and are viewed as an important factor that can benefit all parties involved.

Users play a significant role in product development, providing contributions as project leaders, active designers, generalists, communicators, passive designers, and observers [9], all of which help to achieve product quality. Knowledge sharing and the provision of resources in the development process are often linked to user involvement, which in turn leads to software quality [5]. Different users may contribute at varying levels based on their abilities. While there are different types of users in computer science, the user we propose is an individual who is engaged in the software development cycle and serves as a resource for information sharing, data access, and as the final software user.

User contributions may differ in software development and can vary in terms of their level of impact. Users may be involved in information sharing, product design, system acceptance, testing, and software quality assurance, among other aspects of the development process. User involvement can enhance efficiency in software development, and their representation is an essential part of ensuring acceptable satisfaction throughout the software development process [3].

The company wanted to identify the potential users to be the most required part of the software development process who can be beneficial to the software development team and use their skills in the development phase. We were responsible for identifying the users contributions in the entire project cycle and utilization's of the required business requirements. Knowing the facts, it was required to understand the patterns and identify their forces as solutions to the domain problems. Our role was to involve the specified and skilled users and determine their roles, during this stage,e we felt a user who could be an expert amongst all others and documented this as *Selective User Specifications*. The software development team was responsible for collecting data and understanding the requirements so they documented their experience as *Requirements Collections*. There were lots of data and business requirements, but were not clear to the software development team, they felt to have clear and readable data so needed *Understandable Requirements*. In business requirements collections users were not able to provide sufficient information to the development team, it was necessary to *Build User Capacity* and understand where to involve them, in which part of the software development, they need *Process Identification*. We were also responsible for software quality control, improvement and user satisfactions so we need *User Feedback*. However, there were extra and different types of business requirements which were not the part of the business requirements, than we prefer to implement the pattern to *Remove Unnecessary Requirements*. Writing a pattern is an iterative and creative process that takes time to document the proven experience. There are many methods for pattern mining and writing proposed by several authors. Pattern mining and writing has been described in several methods as best solutions to problems in context. Enaggine the users with agile team through patterns are described in various reserach by authors such as organizational patterns, Scrum patterns and other instructive guidelines [7, 15, 16]. for the user contributions and these observed and documented patterns are structured using Coplien's form [7]. These patterns are described through a flow diagram highlighted in italic way for effective user contribution in and possibilities of engaging them with software development team. In Figure 1, the organization starts with an initial goal and setup the user through a specific flow of contributions as pattern sequence.

The initial goal of the project startup is to involve the user in the initial plan and go through their specific business requirements and specifications *Selective User Specifications*. The next step is to collect requirements from the user and store them as *Storage of Information Access*, then analyze the requirements which can be effective and useful in the development process *Understandable Requirements*. To determine which process of the software need the user to contribute *Process Identification*. Sometime user is not skilled and inexpert in business requirement specification and process flow then it's required to *Building User capacity* for effective. The process can be improve if user is concerned with all process and activities through their feedback *User Activity Feedback* and to avoid the consumption of extra resource, reduce cost and time its important to go through the end of the development process of *Remove Unnecessary Requirements*. Considering these patterns from the initial goal of project and involving the user whenever required to contributes for any part of the software development leads a smart goal which is the success and acceptance of the project for both company and customer.

In this paper, we present seven newly discovered patterns that were documented based on our practical experience as software developers and software development team leaders. We have successfully applied these patterns in various real projects, including those in the public sector and market-oriented software development.

3 SELECTIVE USER SPECIFICATIONS

In software development process, developers are limited to the business requirements and their accessibility to understand user needs, and other software development required activities. The pattern is illustrated in Figure 2.

Context: User requirements specification (URS) is a written and a planning document that specifies the software performance and what the user needs and expect? It is proposed and written from the end user points of view and they may be technical/ nontechnical or complicated, but are required activities. The URS are usually from the part of business requirements. Business requirements are all process and documents of an organization that are required to identify their objectives and solutions to their business. It is necessary to understand the user views about the system and important to translate the business owner's views into a structure of architects views and determine the fundamental structure and functions of the organizations [10]. It is the responsibility of the software development team to identify the user specification, analysis the requirements and make accountability of the user to provide them sufficient resources.

Problem: User specifications have several different meaning or interpretations whenever requirements engineers are identifying the user needs and collecting the business requirements. software engineers collecting the business requirements, but they are often traced as incomplete, incorrect, irrelevant and requirements are written by the developer without clear understanding of what the

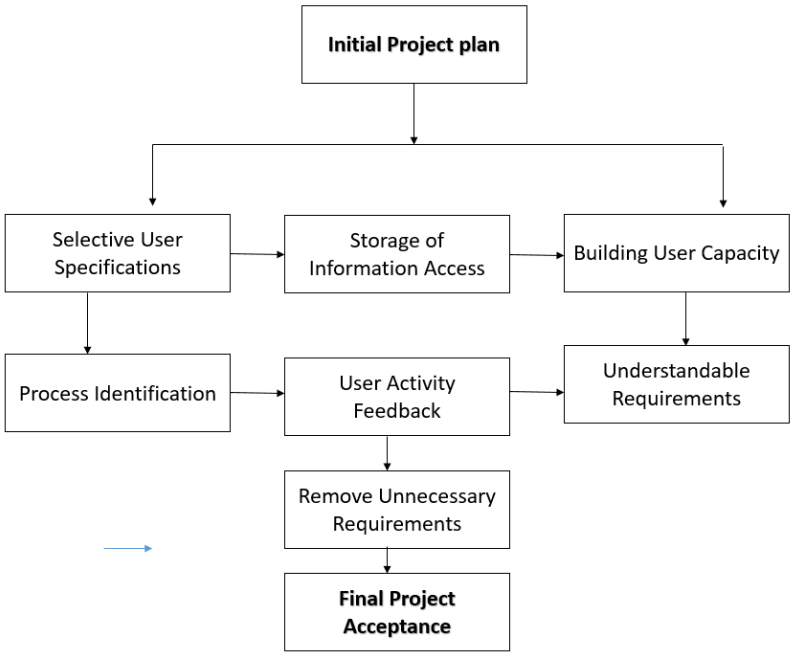

Figure 1: Organization of user contribution patterns.

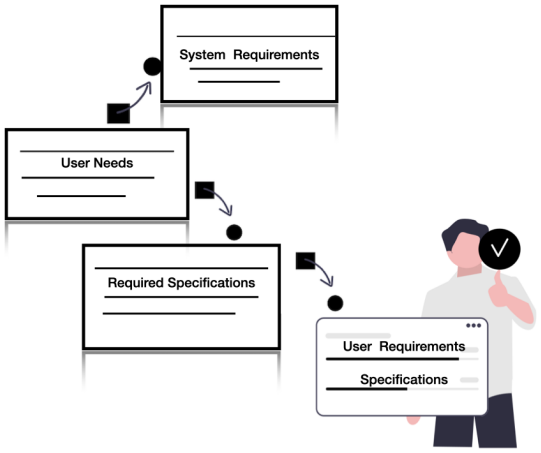

Figure 2: Selective user specifications.

user needs. These requirements are technical which is not inline with user language that creates misunderstanding and faults. These requirements do not capture the functionality from the user, and company. When the user specifications are often written poor and ambiguous, unverified and invalidated will creates risk, if the user requirements are not fully understood and process are not clear in the first stage of the process. When software development process and requirements are not clear then it is very difficult to identify the key persons to get the business requirements and involve them in the process.

Solution: Establish a better way to identify the target users specifications and business requirements as the most important parts of software development process. Focus more on those users who has skills, potential, and understanding the objective of the software. as an example the selective user specification are shown a Figure 2. The users requirements are shown as user persona for business requirements collecting by the agile teams. Time should not be wasted on all other users for feedback or business requirements. Possible way of identifying the selective users can be communications with users and selecting those who are interested to participate in the development process. User selection can be done by observing their skills, interest, responsibilities and provide them workshop and the essence of software benefits. Requirements engineers must be trained well to get the right resources and understand the necessary

information. User requirements specifications should be written clear, understandable, complete and clear. User should be involved from the beginning in all process. Try to clarify the ambiguity, developers should not be allowed to act and assume he understands the user's requirements. Organize a committee to review, verify and validate all the requirements.

Satisfying the users by adopting their requirments we considered the*Selective User Specifications* pattern to software development team for business requirements which would help them to understand the user needs, and all the requirements of the project.

Discussion: This patterns has a close relationship with other observed patterns,*Storage of Information Access* and *Understandable Requirements* when requirements engineers are collecting the user requirement. Software development team need to engage users with developers and consider *Selective User Specifications* patterns to understand the user expectations and all other functions. The pattern *Engage Customers* from Coplien suggest to involve customer in a domain view to avoid blind-sided by a single customer [7].

4 STORAGE OF INFORMATION ACCESS:

Business requirements are the most important part of software development process, but mostly software development team are not considering this fact as initial requirements in the business process. The data storage and access to user data is not fully granted to software development team or not understandable without involving the users in requirement gathering. Developing a software requires sufficient information, access to user data and users engagement with software development team. It is a challenge to identify data and the users needs to build their confidence to provide you access to all information. Setting up a*Data Storage Accessibility* is a good way to overcome the unavailability of data accessibility and user needs. Data should be accessible to the software development team to support the development process and this can be possible to have *Data Accessibility Storage* available to the agile team. The pattern of Storage of Information Access is illustrated in Figure 3.

Context: Software companies have different projects and several teams, it's required to have accessibility to all data required for the software development process. If there are multiple teams working on different projects it's crucial for them to have accessibility to all the required business data that can be very easy when needed.
Problem: Working on business requirements for any project without information storage and understanding the user needs it is hard to understand the development process criteria. Mostly concerned problems are below. Development Processes are not clear and undocumented, there are Conflicting requirements in projects, there is a Lack of user requirements access. User, software development communication problems, and No success criteria set up.

Solution: Establish a *Storage of Information Access* so that every software development team can access the required data and understand the user needs. This pattern *Storage of Information Accessibility* will be a place for the software development team to business requirements and to identify user specifications that can be accessed by developers based on their needs.

Discussion: This pattern and *Selective User Specifications* are very close to each other where you first identify and select the user in software development and then try for the business requirements accessibility. We observed this pattern when there was no sufficient information for software development and had a tough deadline without understanding the requirements. We have arranged a meeting and use this pattern *Storage of Information Access* where software development team first creates information storage earlier to the requirements gathering stage.

5 BUILDING USER CAPACITY

Software development processes may be different or in different phases. It is required to identify and arrange them through the team, but understanding and receiving the actual business requirements to ensure high quality of data is concerned with organization and company standards. Illustrated in Figure 4.

Context: Building User Capacity is the process when software development team is planning to ensure the users inputs, insights and specifications meet the system requirements. It defines the procedures how useful the information can help the team and understand the role and responsibility of users by providing the business requirements of how a user can do and how he can provide these requirements and specifications to the team, allowing them for accountability and consistency through out the process.

Problem: Usually the team is collecting the business requirements from users through repeated manner, often understanding the user skills, requirement's specifications and needs. Software developers are trying to automate process, but they forget to analyze the user skills and capacity based on the accurate business requirements delivery. Ignoring the users capacity and understanding the actual requirements creates risk, if this risk is not resolve in the early stage of software development than the system won't meets the quality, system requirements and acceptance of software.

Solution: Therefore train users and build their capacity in process flow, information sharing and requirements collection before engaging them in business requirements. Analyze users skills and observe them than involve them in the process so the requirements can be efficient in software development for software development team.

Discussion: Some developers prefer to involve users only in some part of the process and get their feedback on business requirements. Even though there are many other people who are willing to be engaged with the software development team in all processes for acceptance and missing features. There are different ways to involve users in different parts of the development process and get the user activity feedback on specific phases of the system and some engage them on different parts of the system development. This pattern is also connected with *Storage of Information Access* when the information is being stored from different stockholders in the project and then illustrating the process by building a prototype *Building Prototype*[7] of each work and share with users for quality work. An example of this pattern is when working in the software development team and engaging users in the business requirements and users are not skilled to explain the flow and describe their needs, it is better to *Build User Capacity* before getting his feedback and collecting the requirements.

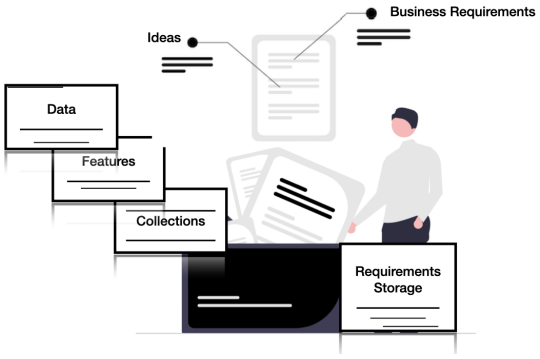

Figure 3: Information repository and storage.

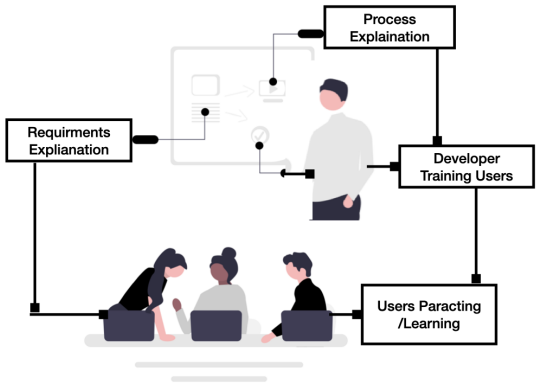

Figure 4: Building user capacity.

6 UNDERSTANDABLE REQUIREMENTS

Requirements understanding is a complex phenomena in software development which often challenge to developers and all stockholders. Data collection and requirements analysis are one of the most important part of software development and getting the understandable requirements in accordance with user need and company regulation for effective operations is difficult. The The concept Illustrated in Figure 4 provides a visual representation of user capacity in the context of the project requirement.

Context:

Requirements can be collected any time, the valid and accurate information can be difficult to gain in first insight. There might be hug information to analyze and validate the accurate information from users that can not be easy understandable. Identifying the requirements for a certain process to fulfill the requirements and easy understandable for software development team to be access at ease. These information can be accessible to everyone in team whenever needed.

Problem:

There are different activities and processes undertaking for software development. These processes include conflicting requirements, lack of user needs understanding and analyzing the actual requirements specifications which creates a risk to development team during implementation. Business requirements collections is an important part of software development, but it become a challenge when not fully understood.

These are the common problems when the development team encounter during misleading requirements: There are undocumented and undefined processes, requirements changes every time, user can not provide actual and accurate information, flow is not well known to users and developers, different and conflicting requirement specifications, unskilled users engaged in process, requirements collected, but not analyzed comprehensively and comparatively that are misunderstanding between users and developers.

Solution: Therefore setup a development team to collect the requirements where every member has to carefully analyze the requirements, identify the conflicting requirements by giving priority to activities that are mostly in high demand of the users needs. An example for this problem can be to document the process and simplify them when accessing the information, set timeline for every requirements to avoid iterative changes and engages the skilled users to provide you the accurate data and flow of the process. Analyze requirements and Separate them as valid and invalid documents. Establish communication between developers and users to *Build Prototype* [7] of each activity. Involve users in every process, meeting, and test every part of the process by them. **Discussion:** The patterns *Storage of Information Access* is used together that can be a first sequence to access to information and than *Build*

Figure 5: Pattern of Understandable Requirements.

User Capacity to design the system based on the actual and right information by having *Understandable Requirements*. We have discussed this pattern during a project with software development team while working on digitization of public sector in Afghanistan and Nepal. An example to this pattern from our experience during the project implementation *Understandable Requirements* result when the developer doesn't know the need to test requirements or the level of specificity required information for effective software development. We analyze the requirements as valid and invalid business requirement's and separate the understandable information from the unclear, misunderstanding and complex information and share with software development team to move with the required process identifications. Establish a bridge between the users and the developers for requirements information so that a flow can be smoothly and not be lost in a hole, the user requirements document has to be in a form that is understandable by both the users and the developers. It is the first step for efficient software development [2].

7 PROCESS IDENTIFICATION

In software development users are often unskilled, and unable to provide sufficient information's to the development team and their involvement need to be scheduled and planned as required in every process. Users involvement in software development process need to be specified so that user activities can be better utilized. Illustrated in Figure 6.

Context: A user can be involved in different phases of software development based on the requirements. This involvement starts from the concept to the final implementations, users and software requirements and their activities to perform in the whole software development process. The software development process phase and involving the user participation on each phase of the process with software development team must be specified. The right phase for user to be involved at each activity in the process is often mislead by the developers. **Problem:** Users involvement and identifying their contributions in the whole software development process is vague and tedious process that happened through several phases and has lack of sufficient information. This happened often lack of understanding by developers for the right place and specific process where the user contributions can help the software development team. Users are often involved in business requirements, but there are other activities to utilize the user participation through several other process. Most common problems in process identifications include the following.

- Informal process documentations: the software development project's improper and poorly defined procedures, which can cause confusion and inefficiencies.
- Unclear phase of the software development process:Mismanagement and delays arise from a lack of clarity on the different stages and phases of the software development process.
- Users are involved once in project startup: User participation is restricted to the project's early stages and might not continue throughout the development lifetime, which could result in miscommunications and low expectations.
- Process is not identified to engage the user with developers: This can happen when a deficiency of well-established techniques to promote efficient user-developer communication and cooperation.
- User specifications not utilized: This mean that failing to properly incorporate into the project user requirements and specifications
- Unskilled developers participation: This could be the lack of the required knowledge and experience of agile team member, which could affect the project's success and quality.
- complex bureaucracy process: Identify any complex and onerous bureaucratic procedures that may impede the effectiveness of a project or the ability to make decisions.

Solution: Start with the initial process startup, discuss software process improvement with quality software developers in team and individual focus groups through several iterations. Analyze the process phase and assign skilled member from the software development team to efficiently utilize the resources and work with users in each activity. Visualize the process flow diagram and draw the business process mapping(BPM) [14] diagram for developers and users to understand the whole process startup and end through steps. Improve the communication with software development team members, users and make it easier to understand process documentation. Identify the process stage and areas for process improvement and Measure the effectiveness of a specific process for developers to go through next process as solutions. Establish roles and responsibilities for software development team members and improve each member performance and productivity through training and observations.

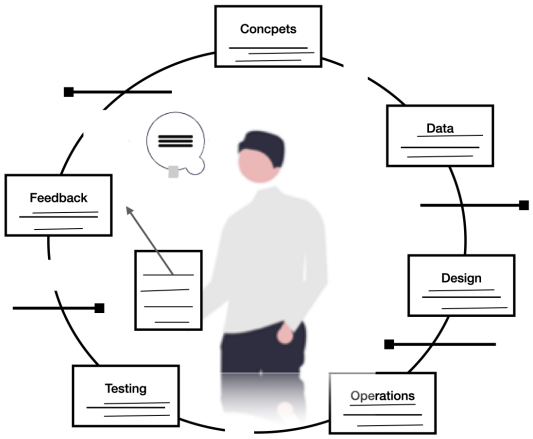

Figure 6: Process identification pattern.

Use Process mapping tools [1] to visualize or model processes and workflows to understand the current and existing business flow of the system. The Process mapping tools can support the software development team and users to identify the workflow and information visibility. And can help both users and developers to understand the relationships between stockholders including people, processes, and systems flow. When process phase is identified than engage the user in each process with software development team to effectively utilize the users preferences. Software process improvement is an important factor for a systematic and continuous improvement of producing organization's ability to produce and deliver quality software within time and budget constraints[4].

7.0.1 Discussion: This pattern and *Understandable Requirements* are tightly connected where first the requirements are analyzed and validating for accuracy than process is identifying from the concept through end of implementations. This pattern is always supportive in software companies and it's commonly considered to be as most important part of the software development. More examples of this pattern would be software developers sitting the initial plan for software implementations startup and business requirements collections. Roles and responsibilities were assigned to each developer based on their skills and interest in communicating with users and involving them for resource sharing in each process identified. The software development team was collaborating with all team members using the *Process Identification* effectively, these members were punctual, collaborative, and accountable for efficient working.

8 USER ACTIVITY FEEDBACK

Software requirements from the user perspective are their input, likes, dislikes, insights, services, and all their request and process involvement activities. It is critical to understand their specifications and the relevant business requirement collection. It is important for the software development team to analyze their feedback to create prioritized tasks that can fit into their software development plan. During the project implementations, we discuss this pattern with other development team members to setup goals for process identifications at different stages and engage the user with developers so the process can be documented formally and effectively. Illustrated in Figure 7.

Context: Mostly software developers engage users for business requirements and set the plan to receive their feedback, but to make sure their feedback is useful and based on their insights and expectations, analyze all issues from their activities and provide iterative feedback in the right directions.

Problem: Receiving user activity feedback is important for the software development team in work environments. The interactions between users and developers can improve productivity, and software quality and keep the software development team with users updated about their product activities. Understanding the requirements and ideas is difficult for all users because the software development team can not get the actual service feedback from all user's activities. There may be users with positive and negative feedback and incorporating them takes more time and repeat the process more iterative. Users' ideas are not constant, they can be change any time, it is important to get the main idea and get their main objectives that affects productivity positively.

Solution: Prioritize user activities and classify the user types to get their meaningful insights. Analyse the users expectations for missing features and identify to collect user feedback for process improvement and provide a constructive output. Collect all those users information that are understandable and feasible for business requirements and software development. Conduct interviews and survey with users, analyze the user requirements and make an iterative approach to understand what are users expecting and what developers gain understanding of them. Another solutions for this problem is to design space for users and developers to interact and share their insights. Build a testing case, arrange software development team with users to test every process by user and move forward for next step. For example arrange meeting between users and software developers to change dialogues during their business requirements collection and whatever activities are undertaking in the development process. These interactions between them can be happened based on the user skills and knowledge.

[1] GitMind, Edraw, LucidChart, Microsoft Viseo, Visual Paradigm, and Google

Figure 7: User feedback.

Discussion: Getting the user feedback in every process is key for the process improvement in software development and understanding the needs of users. Feedback from users activities allows developers to provide suggestions and improvement of their process identifications and helps in software quality assurances. User feedback in software developments plays a crucial role for modern software organizations and increase the product quality [13], it helps developers to understand user needs, filter and aggregate user feedback to get the actual requirements and understand the usage and context of data. The efficiency can be determined by how the developers quickly understand the user's activities and their feedback. This can be achieved through user involvement for example engaging developers with users in each phase and discussing workflow and user needs in an iterative manner. The *Process Identification* pattern is followed by a supporting pattern named *User Activity Feedback*, where the user is put in an identified process rather than getting their insights from each activity. Another example from a practical work is when a software development team engaged the user in software development for business requirements where users and developers exchange views and talk about their needs and system features.

9 REMOVE UNNECESSARY REQUIREMENTS

During the project implementations users and developers are interacting for business requirements and the developers received huge information's and expectations from users. All expectations and extra requirements incorporating into project will increase the project delivery time and complexity.Illustrated in Figure 8. **Context:** The software development team is working to collect requirements, receive user insights, input and feedback to make their work easier, but what user really need. The project goal specific and the unnecessary requirements needs to be removed to deliver project on time and reduce the project complexity. **Problem:** There are huge business requirements collecting for software development and users has a huge list of things to be incorporated that the system should perform. This is often against the project scope and waste of time because it can increase the delivery time, and cost and slowing the performance of adding more functionality. Incorporating unnecessary requirements received from users often blooms and slows down the system, more consideration needs to be undertaken for the actual specifications to avoid the lowering of system performance and complexity of understanding the system usage. **Solution:** Collect the necessary business requirements and identify the users needs, and distinguish between what the users expected and what is necessary needed. Collect the business requirements and consider the project goal and the mandatory data. Try to focus on the project goal to avoid the recreating of existing work, and make things clear and understandable.

- What would you do with these requirements and information: It describe to understand the purpose and actions that need to be considered with the collected software requirements and information.
- What members/software development teams use this collected information and for what purpose they want to use it: Finding the precise groups or people who will make use of the information acquired and the intended uses for which they will put it to use are the goals of this.
- Who are taking decision for information sharing, usage and analysis: The purpose of this item is to make clear who has the power to make decisions about the sharing, using, and analyzing of project information.
- what is the scope of the project: It outlines what is included and what is outside of the project's scope and discusses the project's boundaries.
- what type of requirements need to be Incorporated in the project taken from users: Here, you're asking about the particular specifications that users have given that need to be incorporated into the project.

If you want to make things easier and keep the project simple avoid users hug listed functionalities out of project scope, than you can implement the software/project quicker and on time. The best solution for this type of problem would be to remove many trivial business requirements, and user expectations from the project because it will take more time and increase cost. Avoid extra list of information, and removing unnecessary requirements can helps to deliver on time and complete project on budget. **Discussion:** Business requirements are crucial for software development. In some cases incorporating unnecessary features and functionalities increase the delivery time and budget. This unnecessary requirements are often a hug list of users needs, extra functionalities and out of project scope information. Long lists of requirements are common in projects; these might contain a variety of items, such as all of the user's demands, extra features not initially envisaged, and data unrelated to the project's objectives. An effective method for addressing this problem is to implement *Remove Unnecessary Requirements* The 'Remove Unnecessary Requirements' pattern seeks to get rid of these extraneous components. with *Process Identification* and *User Activity Feedback* patterns are more connected

Figure 8: Remove Unnecessary Information.

and the goal is to implement the project on time and on the fixed budget. This problem is observed mostly with software developers/software development teams in several software companies and it is common with every user whenever engaged with software development team. The users always expect several features and functionalities to be in the system, but it is the responsibility of the software development team to pickup the actual, what is necessary needed in the system. The real work experience example would be users involved during business requirements collection and they propose a huge list of expectations from the system functionalities.

10 CONCLUSIONS

In this study, we have presented seven newly documented patterns for enhancing user contributions in software development, based on our own experience and observation. By documenting these patterns, we aim to help organizations recognize the significance of user engagement in defining business requirements and enhancing the efficiency of software development teams. The identified patterns have been presented in a sequential manner and analyzed in detail in the discussion section, highlighting their interrelationships and efficacy as effective solutions to commonly encountered problems. The integration of these patterns can potentially contribute to the success of software development projects and further improve the overall user experience. We aim to identify additional patterns in user contributions that can provide insights into user engagement, specific user behaviors, and domain-related patterns. These patterns will be valuable for organizations seeking to better understand user involvement in the software development process. We also intend to create a pattern language to effectively visualize and articulate these patterns.

ACKNOWLEDGMENTS

We would like to thank Rossana M. C. Andrade for being our shepherd and for her constructive remarks. Our sincere thanks also go to our writers' workshop group members: Simon Schulte, Andreas Fießer, Eduardo Guerra, and Diego Moreira Da Rosa.

The work reported here received funding from the Operational Program Integrated Infrastructure for the project: Support of Research Activities of Excellence Laboratories STU in Bratislava, project No. 313021BXZ1, co-funded by the European Regional Development Fund (ERDF) and from the Erasmus+ ICM 2020 program under the grant agreement No. 2020-1-SK01-KA107-078196.

REFERENCES

[1] Christopher Alexander. 1977. *A pattern language: towns, buildings, construction.* Oxford university press.
[2] C.E. Bancroft. 1989. Understandable requirements: the first step for efficient software development. In *Proceedings. IEEE Energy and Information Technologies in the Southeast'*. 68–72 vol.1. https://doi.org/10.1109/SECON.1989.132323
[3] Muneera Bano, Didar Zowghi, and Francesca da Rimini. 2017. User satisfaction and system success: an empirical exploration of user involvement in software development. *Empirical Software Engineering* 22, 5 (2017), 2339–2372.
[4] Sarah Beecham, Tracy Hall, and Austen Rainer. 2003. Software process improvement problems in twelve software companies: An empirical analysis. *Empirical software engineering* 8, 1 (2003), 7–42.
[5] Barbara Begier. 2010. Users' involvement may help respect social and ethical values and improve software quality. *Information Systems Frontiers* 12, 4 (2010), 389–397.
[6] Frank Buschmann, Kevlin Henney, and Douglas C Schmidt. 2007. *Pattern-oriented software architecture, on patterns and pattern languages.* John wiley & sons.
[7] James O Coplien and Neil B Harrison. 2004. *Organizational patterns of agile software development.* Prentice-Hall, Inc.
[8] Erich Gamma, Richard Helm, Ralph Johnson, Ralph E Johnson, John Vlissides, et al. 1995. *Design patterns: elements of reusable object-oriented software.* Pearson Deutschland GmbH.
[9] Wei Guo, Qing Zheng, Weijin An, and Wei Peng. 2017. User roles and contributions during the new product development process in collaborative innovation communities. *Applied ergonomics* 63 (2017), 106–114.
[10] David C Hay. 2003. *Requirements analysis: from business views to architecture.* Prentice Hall Professional.
[11] Patrik Honíšek and Valentino Vranić. 2020. Mining drama patterns in dramatic situations. In *27th Conference on Pattern Languages of Programs, PLoP.*
[12] Doble J Meszaros and Jim Doble. 1997. G. A pattern language for pattern writing. In *Proceedings of International Conference on Pattern languages of program design (1997)*, Vol. 131. 164.
[13] Dennis Pagano and Walid Maalej. 2013. User feedback in the appstore: An empirical study. In *2013 21st IEEE international requirements engineering conference (RE).* IEEE, 125–134.
[14] Hajo A Reijers. 2006. Implementing BPM systems: the role of process orientation. *Business Process Management Journal* 12, 4 (2006), 389–409.
[15] Kenneth S Rubin. 2012. *Essential Scrum: A practical guide to the most popular Agile process.* Addison-Wesley.
[16] Tim Wellhausen and Andreas Fießer. 2011. How to write a pattern? A rough guide for first-time pattern authors. In *Proceedings of the 16th European Conference on Pattern Languages of Programs.* 1–9.