Optimizing Feature Prioritization in Software Development Through Structured Usage Analytics: A Data-Driven Approach to Enhancing Development Efficiency

Manoj Kesavulu

M.Eng. in Information and Network Security

Supervised by Dr Marija Bezbradica, Prof Cathal Gurrin and Prof Markus Helfert (Maynooth University)

DCU
Ollscoil Chathair
Bhaile Átha Cliath
Dublin City University

A thesis presented for the degree of Doctor of Philosophy

School of Computing

Dublin City University

Date: June 2025

Declaration

I hereby certify that this material, which I now submit for assessment on the programme of study leading to the award of Doctor of Philosophy is entirely my own work, that I have exercised reasonable care to ensure that the work is original, and does not to the best of my knowledge breach any law of copyright, and has not been taken from the work of others save to the extent that such work has been cited and acknowledged within the text of my work.

Signed (Manoj Kesavulu): ________________ ID No.: 16212435 Date: 30-June-2025

Acknowledgements

I want to thank my supervisors Dr Marija Bezbradica, Prof Markus Helfert and Prof Cathal Gurrin for their generous support, guidance, encouragement and wisdom far beyond the call of duty and I am incredibly grateful and proud to be associated with them. This thesis would not have been possible without them. I would also like to thank the staff and colleagues at the School of Computing, DCU and Lero. I am also grateful to Dr Martin Crane for all the support and guidance. My sincere thanks are also due to Anna Carrigan, James Flynn, Bhumit Patel, John Farren and all team members at the IBM Research Labs, Dublin for allowing me to visit their offices, and explore the tools, systems and applications. My time during the research internship helped me immensely to gain valuable industry knowledge, identify real-world problems, and conduct experiments and interviews.

This thesis is dedicated to my late father Kesavulu, who was always a source of inspiration and encouragement for me. He taught me the value of hard work, perseverance and curiosity. He supported me throughout my academic journey and believed in my potential. He is not only my father but also my mentor, my friend and my hero. I miss him every day and I hope to honour his memory with this work. My mother Gayathramma and my brother Mahesh for all their support and motivation. My wife, Anamika for whom I am eternally thankful for her patience and understanding during the numerous days working late sharing my joy and frustrations, her sacrifices to reach my goals, and her unwavering love, humour, motivation and compassion. She is the best partner I could ever ask for and I am grateful for her presence in my life.

This work was supported with the financial support of the Science Foundation Ireland grant 13/RC/2094 and co-funded under the European Regional Development Fund through the Southern & Eastern Regional Operational Programme to Lero - the Irish Software Research Centre (www.lero.ie)

Publications

1. **Kesavulu, Manoj**, Helfert, Markus, and Bezbradica, Marija, “Towards Refactoring in Cloud-Centric Internet of Things for Smart Cities,” *Pre-ICIS Workshop, Dublin*, 2016.

2. **Kesavulu, Manoj**, Bezbradica, Marija, and Helfert, Markus, “Generic Refactoring methodology for cloud migration-Position paper”, *International Conference on Cloud Computing and Services Science*, vol. 2, pp. 692–695, 2017, SciTePress.

3. **Kesavulu, Manoj**, Helfert, Markus, and Bezbradica, Marija, “A usage-based data extraction framework for cloud-based application—A human-computer interaction approach,” *International Conference on Computer-Human Interaction Research and Applications (CHIRA 2017)*, pp. 85–92, 2017, SciTePress.

4. **Kesavulu, Manoj**, Dang-Nguyen, Duc-Tien, Helfert, Markus, and Bezbradica, Marija, “An Overview of User-level Usage Monitoring in Cloud Environment,” *UK Academy for Information Systems Annual Conference*, 2018. Available: [Link]

5. Dang-Nguyen, Duc-Tien, **Kesavulu, Manoj**, and Helfert, Markus, “Usage Analytics: Research Directions to Discover Insights from Cloud-based Applications,” *International Conference on Smart Grids and Green IT Systems*, 2018. Available: [Link]

6. **Kesavulu, Manoj**, Dang-Nguyen, Duc-Tien, Bezbradica, Marija, and Helfert, Markus, “A usage analytics model for analysing user behaviour in IBM academic cloud,” *International IBM Cloud Academy Conference*, 2018. Available: [Link]

7. Lux, Mathias, Halvorsen, P., Dang-Nguyen, Duc-Tien, Stensland, Håkon Kvale, **Kesavulu, Manoj**, Potthast, Martin, and Riegler, M., “Summarizing E-sports matches and tournaments: The example of Counter-Strike: Global Offensive,” *Proceedings of the 11th ACM Workshop on Immersive Mixed and Virtual Environment Systems*, 2019. Available: [Link]

8. **Kesavulu, Manoj**, Dang-Nguyen, Duc-Tien, Bezbradica, Marija, and Helfert, Markus, "Usage Analytics: A Process to Extract and Analyse Usage Data to Understand User Behaviour in Cloud," *International Conference on Computer-Human Interaction Research and Applications*, 2017. Available: [Link]

9. Palbar, Tenzin, **Kesavulu, Manoj**, Gurrin, Cathal, and Verbruggen, Renaat, "Prediction of Blood Glucose Using Contextual LifeLog Data," *Conference on Multimedia Modeling*, 2022. Available: [Link]

10. Hordvik, Maria Tysse, Teilstad Østby, Julie Sophie, **Kesavulu, Manoj**, Nguyen, Thao-Nhu, Le, Tu-Khiem, and Dang-Nguyen, Duc-Tien, "LifeLens: Transforming Lifelog Search with Innovative UX/UI Design," *Proceedings of the 6th Annual ACM Lifelog Search Challenge*, 2023. Available: [Link]

11. Steffensen, Iselin Ågotnes, Nystad, Sara Tumey Celik, Liahjell, Margrethe, Dang-Nguyen, Duc-Tien, **Kesavulu, Manoj**, Ninh, Van-Tu, Gurrin, Cathal, Vuong, Gia-Huy, Ho, Van-Son, and Tran, Minh-Triet, "T@Retrospect: A Journey Through Time: A User-Centered Prototype Enabling Seamless Information Retrieval Across Expertise Levels," *Proceedings of the 7th Annual ACM Workshop on the Lifelog Search Challenge*, 2024. Available: [Link]

12. Pagani Vavik, Eirik, Wilhelmsen, Michela, Østensen, Jenny Dal, Dang-Nguyen, Duc-Tien, Ninh, Van-Tu, **Kesavulu, Manoj**, Gurrin, Cathal, Vuong, Gia-Huy, Ho, Van-Son, and Tran, Minh-Triet, "VitaChronicle: Applying UX/UI Principles and Guidelines to Enhance Lifelog Retrieval System Design," *Proceedings of the 7th Annual ACM Workshop on the Lifelog Search Challenge*, 2024. Available: [Link]

13. Hordvik, Maria Tysse, Østby, Julie Sophie Teilstad, Dang-Nguyen, Duc-Tien, **Kesavulu, Manoj**, Nguyen, Thao-Nhu, Le, Tu-Khiem, Gurrin, Cathal, and Tran, Minh-Triet, "LifeLens 2.0: Improving Efficiency and Usability in Lifelog Retrieval Systems through UX/UI Design," *Proceedings of the 7th Annual ACM Workshop on the Lifelog Search Challenge*, 2024. Available: [Link]

14. **Kesavulu, Manoj**, Helfert, Markus, and Patel, Bhumit, "System and Methods for Identifying End-user Usage Behaviour to Improve Software Design Decisions," *Patent No. IPCOM000261887D*, Irish Patent Office, 2020. Available: [Link]

Contents

List of Figures

List of Tables

Optimizing Feature Prioritization in Software Development Through Structured Usage Analytics: A Data-Driven Approach to Enhancing Development Efficiency

Manoj Kesavulu

Abstract

In the rapidly evolving field of software development, the ability to efficiently prioritize and enhance software features based on user feedback is crucial for maintaining competitiveness and development efficiency. Deciding which features to develop or update is often complex and inefficient. Developers and product managers typically use traditional feedback methods, which are slow, subjective, and hard to prioritize. The main issue is the overwhelming amount and subjective nature of feedback from surveys, bug reports, and customer interactions. This often leads to analysis paralysis, where teams struggle to determine the most critical issues to address. This thesis proposes a structured approach to usage analytics aimed at addressing these challenges by helping developers identify and prioritize features that have the most significant impact on users. This method enables developers to understand and assess the impact of updates and new features on user engagement. By creating a more efficient and responsive feedback loop, usage analytics can help teams prioritize development tasks based on actual user interactions rather than subjective feedback, reduce delays in addressing critical issues, and improve overall development efficiency. Case studies demonstrate the practical benefits of usage analytics in software development. The first study on the IBM Academic Cloud project identified challenges in feature prioritization, such as unclear feature definitions and time-consuming data preparation. The second study, on IBM Watson Workspace, revealed issues in data identification, analytics metrics, and feature-action mapping, stressing the need for systematic data collection and iterative testing. The third study applied usage analytics to Odoo Notes, showing the impact of changes on user behavior and helping prioritize future features. These studies demonstrate usage analytics offers actionable insights for data-driven feature prioritization, enhancing decision-making and improving software to better meet user needs.

Chapter 1

Introduction

In the dynamic field of software development, the ability to quickly adapt and enhance software applications is crucial. Modern software development practices, such as Agile and DevOps, emphasize iterative development, continuous integration, and rapid deployment to meet user needs and expectations. Agile methodologies, such as Scrum and Kanban, enable teams to deliver software in short cycles, providing continuous improvements and flexibility in responding to user feedback (Beck et al., 2001; Rigby et al., 2016a). DevOps extends this by integrating development and operations, facilitating automated testing and deployment, thus ensuring reliability and speed in releasing new features (Kim et al., 2016). However, a critical challenge remains: how to effectively prioritize which features to develop or update next. Traditionally, this decision-making process relies heavily on user feedback, which can be voluminous and often subjective. For example, surveys and bug reports can generate a large amount of data that is difficult to analyze systematically, leading to potential biases and misinterpretations (Chen and Jin, 2016a). Additionally, the feedback loop can be slow, causing delays in addressing user needs and implementing necessary updates (Jiang and Naudé, 2011). There is often a lack of standardized processes for integrating this feedback into development cycles, resulting in inconsistent prioritization of features (Bosch et al., 2014). Recent studies continue to highlight the challenges and evolving solutions in Agile practices. A systematic review of Scrum adaptations reveals that while Scrum remains the most popular Agile methodology, its application varies widely, reflecting the need for continuous adaptation to fit diverse project contexts (Science and Organization, 2023). Additionally, the integration of Agile with AI and automation is emerging as a trend to enhance efficiency and accuracy in development processes (SoluteLabs, 2023). The adoption of hybrid approaches, such as Water-Scrum-Fall, indicates a trend towards blending traditional and Agile methods to leverage the strengths of both (CGI, 2023).

To address these challenges, this thesis proposes a systematic approach to usage analytics to aid developers in identifying and prioritizing features that have the most significant impact on users by analyzing user interactions. Usage analytics involves collecting and analyzing data on how users interact with software features. This method can provide objective, quantitative insights into user behavior, helping developers to understand which features are most and least used, and how changes affect user engagement (FullScale, 2023). By leveraging detailed usage data, developers can make more informed decisions about feature prioritization, enhancing the overall efficiency and effectiveness of the development process (ProductHQ, 2023). The software industry employs various methodologies and tools to streamline development processes and improve product quality. Agile methodologies, such as Scrum and Kanban, emphasize iterative progress, customer collaboration, and flexibility in responding to changing requirements. These practices enable teams to deliver incremental updates, allowing for continuous improvement based on user feedback (Beck et al., 2001; Rigby et al., 2016a). The Agile Manifesto outlines principles that prioritize individuals and interactions, working software, customer collaboration, and responding to change over following a fixed plan. Recent trends in Agile development have expanded the scope of these methodologies to apply their principles across a broader range of activities within organizations. This holistic approach enhances organizational agility, enabling businesses to respond swiftly and effectively to changes (CGI, 2023). Additionally, new Agile methodologies emphasize value delivery, optimizing the entire value stream from concept to delivery, ensuring that all activities contribute to customer value (SoluteLabs, 2023). DevOps integrates development and operations, promoting automation, continuous integration, and continuous delivery (CI/CD). This approach enhances collaboration between teams and ensures rapid, reliable software releases (Kim et al., 2016). DevOps practices help streamline the development pipeline by incorporating infrastructure as code, automated testing, and deployment processes that reduce the risk of human error and improve efficiency (Fitzgerald and Stol, 2017a). CI/CD pipelines automate testing and deployment, enabling quick and safe software releases. This practice helps identify and resolve integration issues early, reducing the risk of deployment failures. Continuous integration involves developers frequently integrating code into a shared repository, while continuous deployment ensures that code changes are automatically deployed to production environments after passing predefined tests (Duvall et al., 2007).

Despite these advancements, several inefficiencies persist in the software development process. To address these inefficiencies, the gap lies in systematically bridging user interaction data and actionable insights for feature prioritization. Existing methodologies, while emphasizing iterative development, often lack mechanisms to integrate real-time behavioral data into decision-making processes effectively. The research presented in this thesis focuses on closing this gap by leveraging user interaction analytics, a growing area in software engineering, which combines usage data, contextual insights, and advanced metrics to prioritize features dynamically. This approach aligns with recent trends advocating data-driven decisions in Agile and DevOps methodologies, as highlighted by FullScale (2023) and Bosch et al. (2014). User feedback, often gathered through surveys, bug reports, and customer support, can be overwhelming in volume and subjective in nature. This makes it challenging to systematically prioritize feedback (Chen and Jin, 2016a). The sheer amount of feedback can lead to analysis paralysis, where teams struggle to determine which issues to address first. Studies have shown that traditional feedback mechanisms can be biased and do not always capture the complete user experience (Bosch et al., 2014). The time required to collect, process, and act on user feedback can introduce delays in the development cycle, slowing down the implementation of necessary updates and improvements (Jiang and Naudé, 2011). Delays in addressing critical issues can result in user dissatisfaction and increased churn rates. For example, Jiang and Naudé highlight how delayed feedback loops can cause significant setbacks in the agile development process, undermining the very principles of agility and responsiveness (Jiang and Naudé, 2011). There is often a lack of standardized processes for integrating user feedback into development cycles, leading to potential misalignment with user needs. Without a systematic approach, feedback can be inconsistently addressed, resulting in some issues being overlooked or improperly prioritized. This inconsistency can lead to a misalignment between development efforts and user expectations, ultimately affecting the quality and relevance of the software product (Bosch et al., 2014). Recent studies emphasize the importance of integrating user feedback and usage analytics to improve software development processes. For instance, analyzing usage patterns and user interactions can provide valuable insights into user behavior, helping to identify critical areas for improvement (FullScale, 2023). Additionally, leveraging AI-powered tools for feedback analysis can enhance the accuracy and efficiency of data processing, enabling developers to prioritize features more effectively (Appsero, 2023).

This research aims to address these inefficiencies by introducing a usage analytics method designed to analyze user interactions systematically. The core objective is to help developers identify and prioritize features that significantly impact user experience. The proposed method includes analyzing detailed user interaction data to uncover patterns and trends, providing insights into how different features are utilized. By understanding usage patterns, developers can identify which features are most and least used, enabling more informed decisions about where to focus development efforts (ProductHQ, 2023).

Evaluating how updates and new features influence user behavior, distinguishing between beneficial and detrimental impacts, allows teams to understand the real-world effects of their changes and adjust their strategies accordingly. For instance, Buse and Zimmermann found that detailed usage analytics could reveal critical insights into feature adoption and user satisfaction, thereby guiding development priorities more effectively (Buse and Zimmermann, 2012c).

Using data-driven insights to prioritize development tasks ensures that the most critical issues are addressed promptly, streamlining the feedback loop and making it more efficient and responsive to user needs. This approach is supported by studies indicating that usage analytics can significantly enhance decision-making processes in software development by providing objective, actionable insights (FullScale, 2023; Çökeli, 2024). Moreover, advanced analytics can help identify patterns and trends that are not immediately apparent from raw data, enabling more accurate predictions of user needs and behaviors. By systematically analyzing usage data, developers can reduce the reliance on subjective feedback and make data-driven decisions that align more closely with actual user behavior (Appsero, 2023).

1.1 Software Platforms Used in the Research

In this research, three primary software platforms were utilized to facilitate the collection, analysis, and interpretation of user interaction data: IBM Academic Cloud, IBM Watson Workspace, and Odoo Notes. These platforms were chosen for their robust capabilities in handling large volumes of data, providing detailed analytics, and supporting the development and deployment of software features. The following subsections provide an overview of these software platforms and their relevance to this research. In addition to discussing the features of these platforms, it is crucial to highlight the user actions they facilitate, as these actions form the primary source of data for the proposed usage analytics method. User actions, such as deploying applications, analyzing data, and collaborating on tasks, generate detailed interaction logs that provide insights into how users engage with software features. These logs, when analyzed systematically, enable the identification of patterns, trends, and critical points of interaction, which are essential for prioritizing features and improving user experience. By incorporating a discussion on user actions, this section emphasizes their foundational role in capturing the behavioral data necessary for deriving actionable insights through the usage analytics method. This linkage between features and actions underscores how the platforms' capabilities directly influence the depth and quality of data collected, which ultimately shapes the effectiveness of the proposed method.

1.1.1 IBM Academic Cloud

The IBM Academic Cloud is a comprehensive cloud computing platform designed specifically for academic and research purposes. It offers a scalable, secure, and flexible environment that supports a wide range of computational and data-intensive tasks. The platform provides access to a variety of IBM services and tools, such as IBM Watson for AI and machine learning, IBM Cloud Databases for data storage, and IBM Cloud Functions for serverless computing. These capabilities enable researchers to efficiently process large datasets, perform complex analyses, and collaborate on research projects.

The IBM Academic Cloud is built to accommodate the demanding needs of academic research, offering a high degree of customization and integration. Researchers can leverage powerful computing resources to run simulations, model complex scenarios, and analyze vast amounts of data. The platform's robust infrastructure ensures reliability and performance, even under heavy workloads. Additionally, IBM's commitment to data security and privacy makes the Academic Cloud a trusted environment for handling sensitive research data. The platform's integration with IBM Watson services allows researchers to incorporate advanced AI and machine learning techniques into their analyses, enhancing the depth and breadth of their research capabilities.

Features of IBM Academic Cloud:

- **Scalability:** The IBM Academic Cloud offers scalable computing resources, allowing researchers to handle large datasets and perform complex computations efficiently.
- **Flexibility:** It provides a flexible environment that supports various programming languages and tools, making it adaptable to different research needs.
- **Data Security:** The platform ensures high levels of data security, which is crucial for handling sensitive research data.

- **Integration:** Seamlessly integrates with other IBM tools and services, such as IBM Watson, enabling comprehensive data analysis and machine learning capabilities.

- **Collaboration:** Facilitates collaboration among researchers by providing shared resources and collaborative tools.

Relevance and Suitability:

The IBM Academic Cloud was selected for this research due to its ability to support the extensive data processing and storage needs inherent in user interaction analysis. Its scalability ensured that the research could accommodate increasing amounts of data without performance degradation. The platform's flexibility allowed the use of various data analysis tools and programming environments, making it easier to implement and test different analytical models. Data security features ensured that user data was protected, complying with ethical standards for research. Integration with IBM Watson services enabled advanced analytics, such as natural language processing and machine learning, which were crucial for deriving insights from user interaction data.

1.1.2 IBM Watson Workspace

IBM Watson Workspace is a collaborative communication platform designed to enhance team productivity and collaboration. It leverages the power of IBM Watson AI services to provide intelligent and context-aware features that facilitate efficient communication and project management. The platform includes tools for real-time messaging, video conferencing, file sharing, and task management. Additionally, IBM Watson Workspace integrates AI capabilities such as sentiment analysis, natural language processing, and information retrieval, which help in understanding and improving user interactions. This makes it an ideal choice for research focused on analyzing collaborative behaviors and communication patterns.

IBM Watson Workspace is particularly valuable for teams that require seamless communication and coordination. The platform's real-time collaboration tools enable users to interact and share information instantly, which is essential for maintaining productivity in fast-paced environments. The integration of AI services enhances these interactions by providing insights into the tone and sentiment of communications, helping teams to better understand and respond to each other. Furthermore, Watson Workspace's task management features allow users to organize and track their work efficiently, ensuring that projects stay on track and deadlines are met. The platform's advanced search capabilities also make it easy to locate specific information within conversations and documents, saving time and improving overall efficiency.

Features of IBM Watson Workspace:

- **Real-Time Collaboration:** Provides tools for real-time communication and collaboration, including messaging, video conferencing, and file sharing.
- **AI Integration:** Integrates with IBM Watson AI services such as Watson Conversation for chatbots, Watson Discovery for information retrieval, and Watson Tone Analyzer for sentiment analysis.
- **Search and Discovery:** Advanced search capabilities to find information quickly across conversations and documents.
- **Task Management:** Built-in tools for managing tasks and projects, enhancing productivity and organization.
- **APIs:** Offers APIs for custom integrations, allowing developers to extend the functionality of the workspace.

Relevance and Suitability:

IBM Watson Workspace was integral to this research as it provided a rich dataset for analyzing user interactions in a collaborative environment. The platform's real-time collaboration tools generated valuable data on how users communicate and collaborate, which was essential for understanding feature usage and engagement. AI integration allowed for deeper analysis of user conversations, enabling the research to identify sentiment and extract meaningful insights from textual data. The search and discovery features facilitated efficient data retrieval, supporting the analysis of large volumes of communication data. Task management tools provided additional data on user productivity and workflow, contributing to a comprehensive understanding of user behavior.

1.1.3 Odoo Notes

Odoo Notes is a module within the Odoo suite of business applications that provides comprehensive note-taking and task management functionalities. It is designed to help users organize their thoughts, manage tasks, and collaborate effectively. Odoo Notes is highly customizable, allowing users to tailor the application to their specific needs and integrate it seamlessly with other Odoo modules. This platform is particularly useful for capturing detailed user interaction data related to note-taking and task management activities, providing insights into user productivity and organizational behaviors. The integration capabilities and user-friendly interface make Odoo Notes a valuable tool for research focused on improving task management and collaborative work.

Odoo Notes supports a wide range of functionalities that enhance its utility for both individual and team use. Users can create and organize notes with ease, leveraging features such as rich text formatting, tags, and categories to keep information well-structured and accessible. The task management tools allow users to set deadlines, assign tasks, and monitor progress, ensuring that projects are completed on time. Collaboration is a key strength of Odoo Notes, with features that enable users to share notes and work together on tasks in real-time. The platform's analytics capabilities provide valuable insights into how notes and tasks are being used, helping users to identify areas for improvement and optimize their productivity.

Features of Odoo Notes:

- **Note-Taking:** Allows users to create, edit, and organize notes efficiently.
- **Task Management:** Provides tools for managing tasks, setting deadlines, and tracking progress.
- **Customization:** Highly customizable to meet specific user needs and integrate with other Odoo applications.
- **Collaboration:** Supports collaborative note-taking and sharing, enhancing teamwork and information sharing.
- **Analytics:** Offers basic analytics features to track note usage and task completion.

Relevance and Suitability:

Odoo Notes was chosen for its robust note-taking and task management capabilities, which were crucial for analyzing user productivity and task organization behaviors. The platform's extensive customization options allowed the research to tailor the application to specific study requirements, ensuring that relevant data was captured effectively. Integration with other Odoo applications provided a comprehensive view of user activities across different contexts, enhancing the depth of the analysis. Collaborative features enabled the study of how users interact and share information, providing insights into teamwork dynamics. The analytics capabilities of

Odoo Notes supported the tracking of user engagement with notes and tasks, helping to identify key features that improve productivity.

The platforms used in this research, IBM Academic Cloud, IBM Watson Workspace, and Odoo Notes, support a variety of user actions that serve as the primary source of data for the proposed usage analytics method. These user actions reflect the way individuals interact with platform features, providing a detailed record of behavior that is critical for understanding software usage patterns.

In the IBM Academic Cloud, user actions such as deploying applications, managing resources, and analyzing data generate valuable interaction logs. These actions provide a granular view of how users engage with scalable computing resources and leverage analytical tools to meet their objectives. The timing, frequency, and duration of these actions serve as key metrics for analyzing the effectiveness of resource utilization and identifying potential bottlenecks in workflows.

IBM Watson Workspace captures a wide range of user actions centered on real-time communication and collaboration. Sending messages, participating in video calls, and sharing files not only facilitate teamwork but also produce interaction data that reveals patterns in collaboration dynamics. Additionally, the platform's AI-enhanced features, such as sentiment analysis and task automation, provide contextual information that further enriches the data collected from user actions.

Similarly, Odoo Notes focuses on user actions related to task management and note-taking. Creating, organizing, and sharing notes are fundamental actions that generate actionable data about user productivity and collaboration. These actions, along with task tracking and usage monitoring, help identify trends in how teams prioritize and complete their work.

By focusing on user actions across these platforms, this research captures the essential behavioral data necessary for deriving actionable insights through usage analytics. User actions form the backbone of the proposed method, enabling the identification of critical touchpoints, usage trends, and areas for improvement. These actions are not isolated activities but are deeply intertwined with the features they support, creating a rich dataset that informs feature prioritization and software enhancement strategies.

The combination of IBM Academic Cloud, IBM Watson Workspace, and Odoo Notes provided a robust foundation for this research, enabling the collection and analysis of detailed user interaction data. Each platform contributed unique features that supported different aspects of the research, from data storage and processing to real-time collaboration and task management. By leveraging the capabilities of these platforms, the research was able to identify and prioritize features that significantly impact user experience, ultimately enhancing the overall efficiency and effectiveness of the software development process.

1.2 Motivations and Research Problems

In the realm of software development, the continuous enhancement and adaptation of software applications are critical to meeting evolving user needs and maintaining competitiveness in the market. Traditional methods of gathering user feedback, such as surveys and direct user interviews, often prove to be inefficient and limited in scope. Traditional methods, while useful, fail to capture implicit feedback that is naturally embedded in user behavior and interaction logs. Studies by Vozniuk et al. (2016) and Bosch and Olsson (2016) emphasize the importance of moving beyond explicit feedback to harness the potential of implicit feedback for real-time, adaptive decision-making. This shift is vital for addressing scalability and relevance challenges, especially in complex systems where traditional mechanisms can lead to information overload or misaligned priorities. This research aims to address these limitations by leveraging advanced usage analytics to analyze user interactions and prioritize software features effectively.

1.2.1 Motivations

Usage analytics plays a pivotal role in understanding user behavior, optimizing software features, and prioritizing development efforts. By analyzing patterns of user interactions, developers and stakeholders can make informed decisions to enhance software functionality and user satisfaction. Despite its significance, the process of developing a systematic Usage Analytics Method (UAM) encounters numerous challenges, particularly in environments with diverse platforms, user bases, and data sources. These challenges hinder the ability to extract actionable insights and necessitate a structured framework that addresses these barriers comprehensively.

Inefficiencies in Traditional Feedback Mechanisms: Traditional feedback mechanisms, such as surveys and user interviews, are time-consuming and often result in low response rates. They rely heavily on users' willingness to participate and provide honest feedback, which may not always be forthcoming. Additionally, these methods capture only explicit feedback, missing out on the wealth of implicit feedback generated through actual user interactions with the software.

This limitation is highlighted in the work of Vozniuk et al. (2016), who emphasize the need for more efficient feedback mechanisms in software development. Additional studies by Lethbridge et al. (2005) and Pagano and Maalej (2013) further underscore the challenges and limitations associated with traditional feedback methods in software engineering.

Fragmented Data Collection Practices: In distributed software systems, data is often collected across multiple teams, tools, and platforms. Zimmermann and Nagappan (2010) highlight how fragmented data sources lead to inconsistencies, duplications, and missing information, making it difficult to standardize logging practices. For instance, cloud-based applications frequently employ heterogeneous logging mechanisms, where some components generate high-granularity logs while others provide only minimal metadata. This discrepancy complicates efforts to integrate data into a unified analysis framework. Furthermore, Wang et al. (2017) emphasize the difficulty in reconciling data collected across modular systems, where the lack of standardized protocols amplifies integration challenges. These barriers delay decision-making and increase the risk of misinterpretation.

Complex Feature Mapping: Mapping user interactions to specific software features is another significant challenge. Pagano and Maalej (2013) emphasize that in complex systems with overlapping functionalities, distinguishing user-driven actions from automated processes can lead to ambiguities. For example, in enterprise collaboration platforms, users may trigger workflows that span multiple modules, making it difficult to attribute behaviors to individual features. Additionally, feature mapping challenges extend to environments where third-party integrations obscure the origin of specific user actions. This lack of clarity undermines the reliability of usage metrics and hampers the identification of high-priority features.

Metric Validation Challenges: Validating metrics like frequency, time spent, and consistency is critical for ensuring their applicability across diverse contexts. Ghezzi et al. (2014) argue that without robust validation frameworks, metrics may yield misleading conclusions, especially in environments where user behavior varies significantly. For instance, metrics such as "time spent" can indicate engagement in one context but reflect usability issues in another, necessitating rigorous testing to confirm their reliability. Moreover, Nguyen and Wu (2018) highlight the complexity of ensuring metric consistency across time-sensitive applications, where user patterns are influenced by external factors.

Need for Real-Time Feedback: The dynamic nature of software development requires real-time feedback to make timely adjustments and improvements. Traditional methods fail to provide real-time insights, resulting in delayed responses to user needs and issues. Real-time usage analytics can bridge this gap by continuously monitoring user interactions and providing instant feedback to developers. According to Bosch and Olsson (2016), real-time feedback mechanisms are crucial for maintaining the agility and responsiveness of software development processes. The significance of real-time feedback is also supported by the works of Rodden et al. (2010b) and Litoiu et al. (2010), who highlight the importance of real-time data in adaptive and responsive software systems.

Scalability Issues: As software applications grow in complexity and user base, the volume of feedback also increases exponentially. Traditional methods struggle to scale effectively, leading to information overload and difficulty in identifying the most critical issues. Usage analytics, on the other hand, can handle large volumes of data and extract meaningful insights, making it easier to prioritize features and address user concerns efficiently. Chen and Jin (2016b) discuss the importance of scalable feedback mechanisms in modern software development. Additionally, Kittur et al. (2008) and Kim et al. (2011) provide further evidence on the challenges of scalability and the role of analytics in managing large datasets effectively.

Understanding Implicit User Feedback: Implicit feedback, such as user behavior patterns and interaction logs, provides valuable insights that are often overlooked by traditional feedback methods. Analyzing this implicit feedback can reveal user preferences and pain points that are not captured through explicit feedback channels. The importance of capturing and analyzing implicit feedback is underscored by the work of Kapoor et al. (2021) and Sun et al. (2019), who demonstrate how implicit feedback can lead to more accurate and actionable insights in software development. Studies by Jin and Lee (2013) and Claypool et al. (2001b) further emphasize the value of implicit feedback in understanding user behavior and improving software systems.

Enhancing User-Centric Development: User-centric development focuses on creating software that meets the needs and preferences of users. By leveraging usage analytics, developers can ensure that the most impactful features are prioritized, enhancing the overall user experience. Studies by Lee et al. (2020) and Martin et al. (2018) show that user-centric development approaches, informed by usage analytics, lead to higher user satisfaction and retention rates. The benefits of user-centric development are also highlighted by Schwaber and Sutherland (2017) and Nielsen (2012), who discuss the principles of user-centered design and its impact on software quality and usability.

To address these challenges, the research adopted a systematic literature review to derive a comprehensive super-set of data types and metrics from the literature. This process critically evaluated metrics frequently used in software development, assessing their utility for feature prioritization. The resulting super-set formed the foundation for further refinement. In collaboration with IBM developers, brainstorming workshops, experiments and structured interviews were conducted to tailor these metrics to meet the organization's specific needs. This iterative process ensured that the selected metrics were both practical and relevant, aligning with IBM's development goals while being generalizable for application in other software platforms.

1.2.2 Research Problem

The central research problem addressed in this study is the identification and prioritization of software features based on user interactions using advanced usage analytics. This involves several key challenges:

Data Collection and Integration: Collecting comprehensive and accurate data on user interactions with software applications is the first challenge. This data must be integrated from various sources, including web and mobile applications, to provide a holistic view of user behavior. The work of Buse and Zimmermann (2012a) highlights the complexities involved in collecting and integrating user interaction data from diverse sources. Further insights are provided by Harker and Eason (1983) and Jensen and Scacchi (2005), who discuss the challenges of data integration in software engineering environments.

Data Analysis and Interpretation: Analyzing the collected data to extract meaningful insights requires advanced analytical techniques, including machine learning and statistical analysis. The interpretation of these insights to inform feature prioritization decisions is another significant challenge. According to Fitzgerald and Stol (2017c), effective data analysis and interpretation are critical for making informed decisions in software development. This view is supported by Aggarwal and Yu (2009) and Witten et al. (2016), who highlight the role of advanced analytics in deriving actionable insights from complex datasets.

Feature Prioritization: Prioritizing features based on user interactions involves assessing the impact of each feature on user satisfaction and overall software performance. This requires a systematic approach to evaluate the importance and urgency of different features. The framework proposed by Olsson and Bosch (2014b) provides a valuable reference for developing such a systematic approach. Additional perspectives are offered by Karlsson et al. (2007a) and Racheva et al. (2010), who discuss various methods for feature prioritization in agile and iterative development processes.

Real-World Application and Validation: Implementing the proposed usage analytics method in real-world software development environments and validating its effectiveness is essential for demonstrating its practical value. This involves collaborating with software development teams and collecting empirical evidence of the method's impact on feature prioritization and user satisfaction. The importance of real-world validation is emphasized by Rigby et al. (2016b), who argue for the need to validate new methods and tools in actual practice to ensure their relevance and effectiveness. Supporting studies by Kitchenham et al. (2004) and Sjoberg et al. (2005) further highlight the importance of empirical validation in software engineering research.

The motivation for this research stems from the limitations of traditional feedback mechanisms and the need for real-time, scalable solutions to prioritize software features effectively. By leveraging advanced usage analytics, this research aims to address the challenges of data collection, analysis, and feature prioritization, ultimately enhancing the efficiency and responsiveness of software development processes.

1.3 Hypothesis and Research Questions

To systematically address the outlined research problem and to achieve the research objective described in the previous section, the following hypothesis is formulated and validated in this research: **User interaction data can be systematically analyzed to improve the efficiency of the software development process by enabling feature prioritization**. In order to address the identified research problem and to test the designed hypothesis, the methodology used here will aim to understand how features of an application are used by the users by analyzing the application usage data to help improve the software development process. This involves focusing on key activities to identify and analyze usage data, gaining insights into how end-users engage with the features of a software application and understanding the impact of developer-implemented changes on user interactions. In order to achieve this, a usage analytics method will be developed, providing the key activities such as usage data source identification, usage data extraction, and usage data analysis that can be applied in practice. The demonstration and evaluation of the method will be conducted through constrained experiments and case studies.

RQ1: (*Problem scoping and objective definition*) What are the customized key metrics and data points, tailored to industrial requirements, from user interactions that most significantly influence feature prioritization decisions?

Identifying the most relevant metrics and data points is crucial for effective usage analytics. This question seeks to determine which aspects of user interactions (e.g., frequency of use, amount of time spent on each feature etc.) are most indicative of feature importance and should be prioritized in the analysis. The Usage Analytics Method (UAM) proposed in this research provides a structured approach for analyzing user interactions and prioritizing features. While this thesis demonstrates a specific subset of metrics frequency, time spent, and consistency; these were selected from a broader super-set derived through rigorous literature analysis and tailored to industrial needs. Developers using UAM are encouraged to select and customize metrics based on their specific organizational requirements, leveraging the method's flexibility.

RQ2: (*Design and development*) What are the key challenges faced by the developers to identify and utilize usage data and key metrics related to the feature prioritization effectively within the software platform?

This question focuses on the methodologies and tools necessary for capturing and analyzing usage data. It seeks to explore best practices for data collection, data cleaning, and data analysis to ensure that the metrics gathered are accurate and actionable. Additionally, it looks at how to integrate this data into the software development lifecycle.

RQ3: (*Application and evaluation*) How can activities be systematically structured to identify and utilize usage data for feature prioritization?

This question aims to develop a structured approach or framework for incorporating usage data into the feature prioritization process. It involves defining a set of activities, from initial data collection and analysis to decision-making and implementation, ensuring that the process is repeatable and scalable.

1.4 Research Process and Methodology

This research systematically addresses the posed research questions by employing an iterative empirical design. The platforms selected for this study IBM Academic Cloud, IBM Watson

Workspace, and Odoo Notes were chosen for their diversity in application contexts, allowing for a robust evaluation of the proposed Usage Analytics Method (UAM). These platforms represent varied domains, including cloud-based academic environments, collaborative workspace tools, and task management systems. This selection was carefully informed by the need to balance practicality and relevance to industrial applications, as well as to ensure the research findings have broad applicability.

The selection process involved consultations with industry practitioners to ensure alignment with real-world needs. These consultations included discussions with experienced developers, project managers, and software analysts to validate the appropriateness of the chosen platforms. An additional reason for the choice of these platforms was the opportunity they presented to leverage existing access and familiarity, reflecting a pragmatic and opportunistic approach. Each platform provided unique opportunities for analyzing user interactions and offered complementary contexts for validating the UAM. For instance, IBM Academic Cloud facilitated the analysis of large-scale data-intensive operations, essential for understanding user behaviors in resource-heavy applications. Meanwhile, IBM Watson Workspace supported the exploration of real-time collaboration dynamics, enabling the research to study immediate feedback loops and team interactions. Odoo Notes offered a contrasting perspective by focusing on task management and user engagement in a smaller, more localized software environment, demonstrating the flexibility and adaptability of the UAM across varied use cases.

By combining these platforms, the study effectively illustrates the adaptability and generalizability of the UAM, providing a bridge between theoretical insights and practical applications. This approach ensures that the research outcomes are not only relevant to the platforms studied but also broadly applicable to other software environments with similar user interaction characteristics.

The remainder of this thesis is structured as follows, Chapter 2 contains the discussion on the systematic literature review conducted and the related work, details the systematic process of deriving a super-set of data types and metrics, providing the foundation for the empirical studies. followed by Chapter 3, which describes the research methodology used, the hypothesis formed and the detailed discussion on the research questions. Outlines the iterative design of the UAM, incorporating feedback from each empirical study. Discusses the role of participatory design methods and collaborative workshops in refining the metrics. Chapter 4 describes the role, importance and problems with features and usage data in the software maintenance domain. Chapter 5 presents the description of the case studies conducted to identify and formulate the problem; design, develop, apply and test the artifact. Chapter 6 details the Usage Analytics method, developed based on the case studies, including its key activities and the novel analytics algorithm. Chapter 7 presents the results obtained from the case studies and experiments conducted and the evaluation of the developed artifact. Chapters 5-7 consolidate the empirical designs, including platform selection, participant details, and data analysis methods. These chapters demonstrate how each study contributed to the refinement of the UAM. Provide detailed narratives on the challenges encountered and the solutions implemented during each iteration of the research. Chapter 8 describes the conclusion drawn and the contributions of this research.

1.5 Limitations of the Research

The research was carried out according to the requirements and expectations of the funding organization, IBM. Consequently, decisions taken and the research directions are both guided by, and restricted by the funding organization. This extends to various aspects of the research process and outcomes.

One of the significant limitations is the selection of key metrics. The metrics of frequency, time spent, and consistency were identified based on the specific needs of the developers at IBM. Although these metrics were used to demonstrate the utility of the Usage Analytics method, they do not represent the state-of-the-art metrics to analyze user interactions in the software development domain. The focus on these three metrics, while providing valuable insights into user behavior, might have excluded other potentially useful metrics that could offer a more comprehensive understanding of user interactions. For example, metrics such as user satisfaction scores, task completion rates, and error rates could provide additional dimensions to the analysis, offering a richer perspective on user engagement and application performance. The developers of the application may still choose metrics suitable for their application and the specific requirements of their analysis. Therefore, the aim of the research focuses on the Usage Analytics method and the related techniques that could be used to prioritize features in the software development domain, rather than presenting an exhaustive set of user interaction metrics.

Another limitation pertains to the researchers' ability to publish substantial parts of this research in open forums. Due to proprietary constraints and the sensitive nature of the data involved, some findings and methodologies were not fully disclosed in public domains. This restriction limits the broader academic scrutiny and validation that open publications would facilitate. The inability to share detailed data and methods openly hinders the potential for other researchers to replicate the study or build upon its findings, thereby restricting the overall contribution to the field. This limitation also impacts the perceived transparency and thoroughness of the research process, as external validation is a critical aspect of robust scientific inquiry.

Furthermore, the research process and decision-making could appear to lack depth and breadth in certain areas due to these constraints. For instance, the study's focus was predominantly aligned with IBM's immediate needs and strategic interests, potentially overlooking broader industry trends or alternative methodologies that might be relevant in different contexts. This alignment with the funding organization's priorities means that some innovative or emerging metrics and techniques in usage analytics might not have been considered or explored in this research. As a result, the findings and outcomes, while highly relevant to IBM, may not be entirely generalizable to other organizations or contexts within the software development industry.

However, considerable effort has been put into addressing some of these limitations. To demonstrate the broader applicability of the Usage Analytics method beyond IBM, a case study involving the Odoo Notes application was included. This application does not belong to IBM nor does it have any overlap or commonality with IBM applications. By applying the Usage Analytics method developed through work with IBM applications to the Odoo Notes application, it was possible to demonstrate that the method can be applied to other software applications as well. This external validation is crucial in showing the method's versatility and relevance across different contexts.

Additionally, the same key metrics of frequency, time spent, and consistency were intentionally used in the evaluation of the Odoo Notes application. This decision was made to ensure that the Usage Analytics method remains relevant and applicable regardless of the specific application. The consistent use of these key metrics across different applications reinforces the idea that the method's core principles and techniques are robust and not dependent on the type of software or its unique characteristics.

Moreover, the initial part of the research was conducted within the specific organizational context of IBM, which has its unique workflows, tools, and user base. This context-specific focus may limit the applicability of the findings to other organizations with different structures, technologies, or user demographics. The particular software development practices and user interaction patterns at IBM might not reflect those in smaller companies, startups, or different sectors of the software industry, leading to potential challenges in adapting the Usage Analytics method to other environments.

The iterative development and refinement of the Usage Analytics method, while demonstrating a robust process of continuous improvement, were constrained by the specific case studies chosen for the research. The method was evaluated primarily through its application to IBM Watson Workspace and the Odoo Notes application, and while this provided valuable feedback, the method's effectiveness in other types of software applications or in different stages of the software lifecycle was not extensively tested. This limitation suggests that further research is necessary to validate the method's generalizability and effectiveness across a wider range of software development contexts and application types.

In summary, the limitations of this research stem from the constraints imposed by the funding organization, the specific selection of metrics, the restricted ability to publish findings openly, and the context-specific focus of the study. However, significant efforts have been made to address these limitations by applying the Usage Analytics method to a non-IBM application and using consistent metrics to validate the method's applicability. These steps enhance the generalizability, transparency, and comprehensiveness of the research, suggesting that while there are inherent limitations, the findings and the Usage Analytics method remain relevant and robust across different software development contexts.

Chapter 2

Literature Review and Related Work

In order to obtain a sound scientific basis for further steps in the course of the current research and to consider the contribution of the topic, an extensive, systematic literature review was conducted following the process described by Webster and Watson (Webster and Watson, 2002). The primary focus of this research is to design a novel data-driven method to understand which features of the application are important to the end-user by analyzing the end-user usage of the applications. The user-level usage data in the context of this thesis is the usage data generated due to user interaction with the application; by analyzing these data we can determine which features are critical and important for the end-user. Although the concept of understanding how the applications are used by the users by analyzing the application logs is not new, it is still difficult to understand how to prioritize and select the features for the next version of the application from the users' perspective. These applications are often monitored by Application Performance Monitoring (APM) tools to understand how much of the underlying resources are used by the application, errors, bugs, and usability issues and to identify outdated services and monitoring tools to understand the usage of the features of the applications by the software developers. The data could be analyzed by the software developers to fix the errors and bugs, improve the application, roll out updates and optimize the resources. This constitutes the development life-cycle of a typical software development environment as shown in Figure 2.1.

The first step in the software development life-cycle is *Requirement Elicitation*, which describes gathering requirements from the business model and design of the application based on which features are needed by the end-users. The second step, *Software Construction*, includes the development, testing and verification of the features of the application. The third step, *Software Deployment* describes the design of the Service Level Agreement (SLA), platform pol-

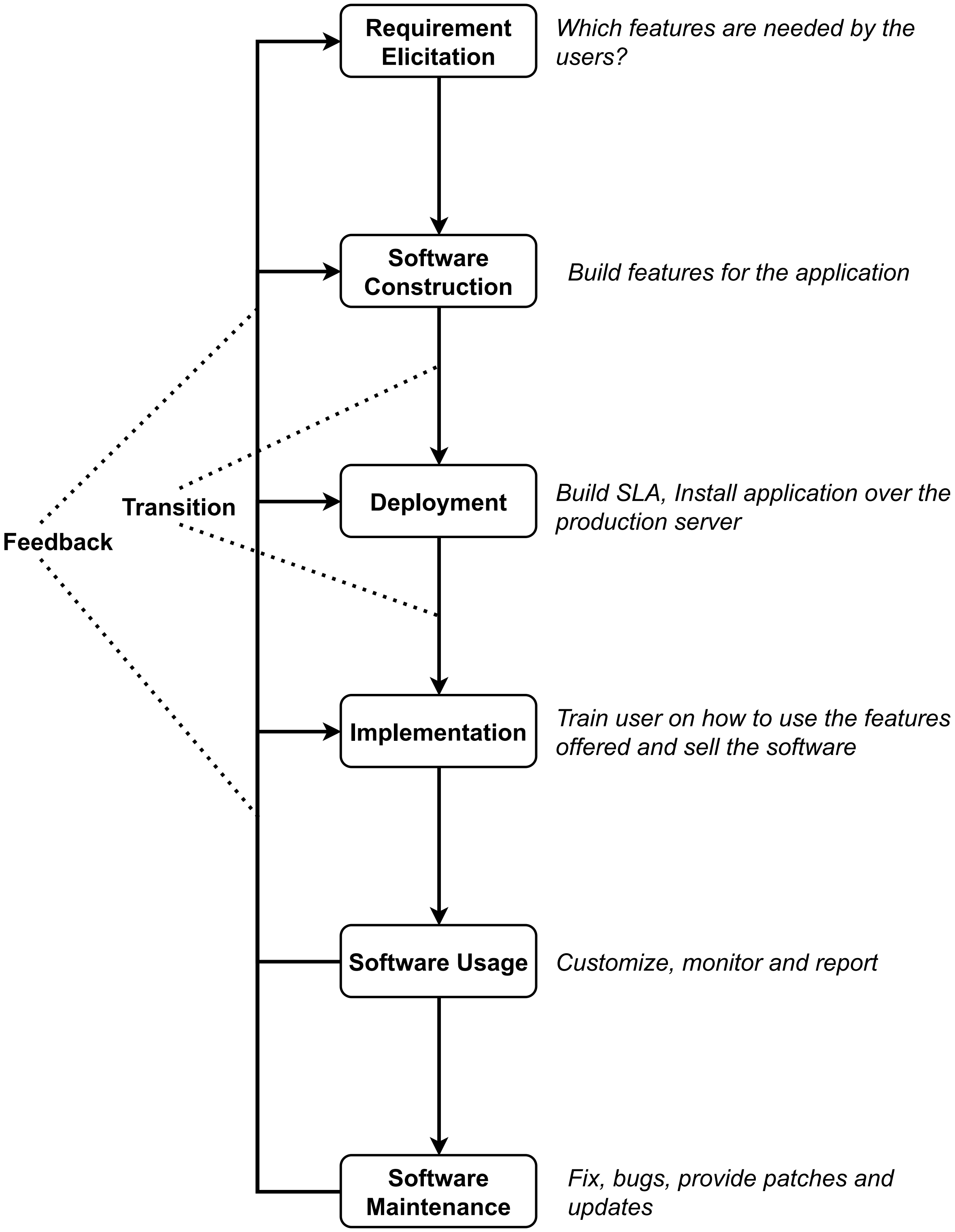

Requirement Elicitation
Which features are needed by the users?
Software Construction
Build features for the application
Deployment
Build SLA, Install application over the production server
Implementation
Train user on how to use the features offered and sell the software
Software Usage
Customize, monitor and report
Software Maintenance
Fix, bugs, provide patches and updates
Transition
Feedback

Figure 2.1: Software Development Life-cycle

icy compliance verification and installation of the application over the platform. The fourth step, *Software Implementation*, includes training the end user of the application and selling the application to the end user. The fifth step, *Software Usage*, includes software customization, platform and application monitoring, reporting and escalation of the incidents which includes understanding which features are preferred by the users and any problems experienced by them. The last step, *Software maintenance*, includes providing application patches and upgrades.

The software developers use APM tools (for example, CloudWatch in Amazon Web Services) to monitor the status of the deployed applications, and the number of resources used by the applications based on the agreement between the application provider and the users called Service Level Agreement (SLA). The application developers can also use various third-party monitoring tools such as New Relic, Binadox and so on. User-level usage monitoring plays a critical role in understanding how end-users interact with software applications, providing valuable insights for developers and architects. These insights help optimize future updates, personalize features based on user preferences, and evaluate application performance in real-world environments. This thesis leverages the principles of user-level usage monitoring to develop a comprehensive usage analytics method, addressing key challenges in understanding and improving user interactions. Existing monitoring solutions for cloud environments predominantly focus on application and infrastructure levels, addressing aspects such as resource utilization and service-level agreements (SLAs). However, user-level usage monitoring, which captures and analyzes end-user interactions with applications, remains under-explored (Kesavulu et al., 2018a). The lack of comprehensive solutions in this domain creates challenges in understanding user satisfaction, behavior patterns, and feature utilization. This research builds on these identified gaps to propose a novel user-level usage monitoring framework, aiming to bridge the disconnect between end-user behavior and application optimization. But these tools mainly focus on monitoring application-oriented features and cloud service-oriented features such as measuring the number of users logged in to the application, identifying rare logins, cloud resource usage, idle times, license types etc. Understanding usage data of features of an application has various uses such as personalizing the application according to the end-user's preferences (Yang et al., 2017), profiling users for security (Al-Bayati et al., 2016), facilitating improvement in marketing of software products (Bucklin and Sismeiro, 2009a), and analyzing the performance of the application in the deployed environment for maintenance purposes (Petruch et al., 2012). From the literature exploration, we see that the idea of monitoring user behavior is to understand how users interact with the application and this is mainly done through analyzing the clickstreams (Banerjee and Ghosh, 2001; Bucklin and Sismeiro, 2009a; Pachidi et al., 2014; Wang et al., 2016). Cito et al. (2015a) provide a high-level taxonomy of types of operation data:

1. Monitoring data (Operational application metadata, Collected from state-of-the-art APM tools)
2. Performance data – service response times, database query times
3. Load data – incoming request rate, server utilization
4. Costs data – hourly cloud virtual machine costs, data transfer costs per 10,000 page views
5. User behavior data - clickstreams, paths taken by the user(s)
6. Production data - bug and errors encountered, application usage statistics
7. Data produced by SaaS application itself - placed orders, customer information.

The usage data is generated after the applications are deployed and while they are being used in real time by the end users. This usage data is essential for software developers and architects, supporting tasks such as providing software updates, fixing bugs, performing run-time analysis, and more. The majority of the research in the software monitoring domain focus on collecting software operational data, event logs, and resource usage monitoring in order to identify performance issues, errors and other usability problems (Fabijan et al., 2015). Since applications in a SaaS environment are deployed over the internet, web usage mining plays a crucial role in identifying and extracting usage data. Additionally, the field of web usage mining has experienced significant advancements in recent years (Ghezzi et al., 2014; Gasparetti, 2017). While these works are mainly concentrated on understanding the usage of features of the application and infrastructure level, understanding of user-level features is seldom considered in both academia and industry. Identifying and analysing usage of user-level features facilitates developers, designers and software architects to understand which features of the software application are critical to the end user. To achieve this, it is essential to consider all the ways through which an end-user can access the SaaS application, which include the interfaces discussed earlier such as a web browser, smartphone app and a command-line interface along with the server and database at the back-end of the cloud system that facilitates the execution of the application. Consideration of usage data only through clickstreams is mainly under the assumption that the end-user has access to the SaaS application only through a web browser. This may be true for a traditional web application but for SaaS applications, other interfaces such as mobile apps and command line interfaces are also used to access the application and these interfaces should be considered as sources for the extraction of the usage data from a cloud-based application. Although we discussed various techniques, methods and tools that could be used to monitor and extract usage data in the cloud, it is critical to understand which elements of the complex multi-layered application architecture have to be considered as features to analyze by the software development team. The different stakeholders such as application developers, cloud service providers, infrastructure providers, software architects and end-users have different understandings of a feature in a cloud-based application environment. Consequently, the secondary focus of this research is to develop a novel way to identify and classify features from a different stakeholder perspective.

In today's software systems, system complexity and the frequency of upgrades are increasingly imposing various risks to development organizations. One such risk is that often once the software update is rolled out, the changes implemented may be of little value to the users. This requires companies to continuously discover what customers need through direct customer feedback and observation of usage behavior. To achieve that, companies are recognizing the need to transition their traditional research and development (R&D) activities which are mainly backed up by opinion-based decision-making towards data-driven systems that support continuous customer feedback loop (Fagerholm et al., 2014). The use of data to inform decision-making in software development is well-established, with numerous studies highlighting its benefits through various approaches and case studies (Schellong et al., 2017; Berndtsson et al., 2018). The rise of smartphones, cloud computing, and web applications, combined with advanced techniques such as web and data mining, has significantly increased the availability of usage data. This data has been utilized in areas like personalization, system optimization, site enhancement, business intelligence, and usage characterization. For instance, Han et al. (2012) employs locally running software on users' devices to analyze the resulting data and identify performance issues. Similarly, (Pachidi et al., 2014) propose a framework for analyzing operational software data, utilizing three distinct data mining techniques — classification analysis, user profiling, and clickstream analysis to support data-driven decision-making. While many existing monitoring solutions focus on individual interfaces, such as web browsers, or operational metadata, there is a notable gap in addressing usage data across multiple interfaces comprehensively (Kesavulu et al., 2017b). The usage data extraction framework proposed in this research bridges this gap by integrating data from web browsers, mobile applications, and command-line interfaces. This holistic approach ensures a complete understanding of user interactions and critical features, aligning with the goals of this thesis to provide actionable insights for application optimization.

Another notable approach to understanding how users use the software application is the experiment-driven method where the deployment of software is seen not as the delivery of a final product but as a way to start, test, and revise a small functionality of the software. This requires the ability to build data collection components and the capability to use the collected data effectively (Olsson et al., 2012; Fagerholm et al., 2014). Incorporating experimentation into the development process of a software product not only allows for quick delivery of updates but also helps the software developers make decisions based on user data rather than on just opinions from the experts (Rissanen and Munch, 2015). With many organizations moving toward agile development and adopting experiment-driven approaches to rapid delivery of software updates, several models have been developed that aim to support and implement the experimentation process. Although there are differences in the models, many of them resemble the build-measure-learn feedback loop which starts by forming one or more hypotheses that need to be tested. The build step focuses on creating a minimum viable product (MVP) that has been instrumented for data collection. The Measure step focuses on using the MVP in a test, thereby collecting data. Once the test has been conducted, the collected data is analyzed in order to validate or invalidate the formed hypotheses based on which a decision can be made to move the idea to the next stage (i.e., implement a full product or feature), correct the course to test a new fundamental hypothesis or stop (Ries, 2011). These experiment-driven models mainly focus on the hypotheses formed, which inhibits the range of outcomes which could be exhibited by the users once the feature or functionality is implemented. It is possible for some uncertainty to exist once the decided update is deployed into production.

2.1 Typical Problems in Software Development

In this section, we discuss the typical problems faced by software developers and architects in the process of developing and deploying software applications. The problems discussed in this section are mainly related to the software development process and the software architecture.

Since the software development process is a complex process, it is difficult to identify the exact cause of a problem. Problems are often caused by a combination of factors such as the software architecture, the software development process, the software development team, the software development tools and the software development environment. Additional factors that influence the problems in the software development process are the environment in which the software is deployed and served, and recently the cloud computing environment.

2.1.1 Resource Provisioning

A typical problem in a cloud-based environment is the network resource management, for example, the acceptable Virtual Machine (VM) configuration to minimize the resource consumed by certain services deployed on these VMs. A common problem experienced in data centers and utility clouds is the lack of knowledge about the mappings of the services being run by, or offered to, external users to the sets of virtual machines (VMs) that implement them (Wang et al., 2011). It can be done by exploiting analytics methods, for example by predictive analysis of the usage data from the systems logs from each VM, to predict the suitable configuration for future VM deployments.

For in-time decision-making, Wang et al. (2011) proposed a system integrating monitoring with analytics, termed Monalytics, which can capture, aggregate, and incrementally analyze data on-demand and in real-time, thus increasing accuracy and reducing human intervention in the analysis process. It was done by applying a clustering algorithm and a top-k flow analysis on the data gathered from the CPU usage data on each VM, identifying the VMs that are responsible for the majority of the traffic flow in the group (Kumar et al., 2004). This provides information on critical VM combinations to include in the same group to achieve maximum cost benefit during VM migration.

Usage data on VM can also be exploited to predict VM states that can be used as the inputs of the existing networking capacity management techniques. For example, the authors Sun (2016) proposed a method named Smart Predictive Capacity Management (SPCM) that is designed to assist cloud networking deployment in estimating the acceptable network capacity for a specific configuration of interdependent VMs by predicting individual VM states. It is done by applying Markov chain techniques to address the data analytics for potential states in a heterogeneous cloud computing environment. This work could help enterprises to optimize the VM configurations to attain significant performance improvement.

2.1.2 Problem Diagnosis

With the increasing scale and complexity of cloud-based applications, it has become more and more difficult for system operators to understand the behaviors of systems for tasks such as system problem diagnosis. For example, system operators need to understand system execution behaviors to identify symptoms and root causes of the anomalous nature of the system. System behaviors include a series of actions executed by the system and the corresponding changes in the system states. Although operators usually investigate a system starting from a specific state of interest, e.g., a hang state or failure state, it is critical to identify the series of states the system traversed to reach the current unstable state.

The authors Fu et al. (2013) proposes a new approach for the contextual analysis of system logs to understand a system's behaviors better. In particular, they used execution patterns extracted from the system logs that ultimately reflect the run-time behavior of the application, and propose an algorithm to mine execution patterns from the system logs. Based on the execution patterns, their approach further learns the essential contextual factors by modeling the relationships among execution patterns that are responsible for the execution of specific branches of the system.

Inspired by a study by the authors Fu et al. (2013), analyzing the application logs could help to understand better the correlation between user behaviors and the corresponding system. In particular, using the Formal Concept Analysis (Ganter and Wille, 1997) to mine execution patterns from the usage data from all sources: application, hosting VM(s), and underlying cloud logs. The execution patterns in this context can be considered reflections of the user's interactions with the application. The mining and learning results can help system operators understand both the behaviors of the customers as well as the execution logic of their services and applications.

2.1.3 Understanding User Satisfaction

A typical way to access cloud-based applications by the end-user is through a web browser. Data analytics techniques, e.g., web-mining, in this way, can be employed to obtain interaction insights. In (Bucklin and Sismeiro, 2009a), the authors provided an overview of *Clickstream* data, defined as the electronic record of a user's activity, representing the traces an end-user takes while accessing the cloud application. Analyzing such kind of information can discover user satisfaction with the provided services based on the interaction of the users (obtained via the number of clicks, time spent, and other usage data).

User satisfaction is a critical aspect of software development, often assessed post-deployment. However, existing literature lacks detailed methods for analyzing user satisfaction at a granular level. Studies by Smith and Brown (2019); Johnson and Smith (2019) highlight the importance of user feedback but do not provide robust frameworks for real-time satisfaction analysis. Additionally, empirical studies such as those conducted by Martin et al. (2018) emphasize the need for continuous user feedback loops, yet practical implementations remain sparse. This gap underscores the necessity for advanced usage analytics to capture nuanced user interactions and satisfaction metrics throughout the software lifecycle.

2.2 Concept of Features in Software Development

In software development, a feature is a component of software functionality, complementing the core software while adding additional value (adapted from (De Chaves et al., 2011)). Typically, features are added incrementally, at various stages in the life-cycle, often by different groups of developers. Although the companies are aware that some features, at some point in time, cost more than what they add in terms of value, they may not have the data to support this and to understand when this happens. Although there are techniques available for collecting user feedback to understand which features are useful to the users, yet rarely answer why those features are important for them and also these feedback are typically not applied as part of a continuous feedback loop. As a result, the selection and prioritization of features are far from optimal, and the product may deviate from what the users need (Creswell, 2002).

Over the decades, methodologies like Feature Modeling (Riebisch, 2003) and the integration of features in agile frameworks (Beck et al., 2001) have emphasized iterative and user-centered design. These approaches align with the goals of identifying core user needs and adapting features to fulfill them dynamically. Studies by Olsson and Bosch (2014a) highlight the significance of continuous feedback loops in refining feature relevance and performance. Features are commonly classified into functional and non-functional types, each playing distinct roles in a software system's architecture. Functional features address specific user requirements, such as a file upload system, while non-functional features improve underlying aspects like security or scalability (Boehm, 1988).

To bridge the gap between action-based results and feature-centric insights, Chapter 4 provides an in-depth discussion on the definition and role of features in software analytics. It emphasizes the distinction between functional and non-functional features, highlighting their impact on software performance and user satisfaction. The chapter outlines the methodologies used to define and identify features, such as reviewing application catalogs, analyzing version logs, and conducting stakeholder interviews. These approaches ensure that features are clearly understood and categorized to support meaningful usage analysis. By connecting features with user interaction data, the chapter demonstrates how these insights are instrumental in refining the Usage Analytics Method to prioritize features effectively and align them with user needs

2.2.1 Action-based Usage Analysis

User actions like clicks, hovers, or text entries are integral to analyzing software usage. These actions reflect how users engage with specific features, offering measurable insights into their frequency, duration, engagement patterns, and overall behavioral trends. In the Usage Analytics Method (UAM), focusing on action-based metrics enables developers to assess the effectiveness and relevance of software features, ensuring alignment with user needs, expectations, and system goals (Olsson and Bosch, 2014a; Beck et al., 2001). However, action-based results are not synonymous with feature-based analyses. User actions often represent only fragments of a feature's lifecycle or composite functionality. For example, "time spent" on a feature captures interaction duration but may not reveal deeper contextual insights like task success, error rates, or user satisfaction levels (Tang and Zhang, 2011). Similarly, click frequencies can indicate feature popularity but may fail to provide insights into the efficiency or quality of the interactions (Riebisch, 2003).

Mapping Actions to Features

To extract more meaningful insights, it is critical to map user actions to the broader goals and lifecycle of software features. This requires deconstructing features into measurable components and aligning these with action-based metrics. For instance:

1. A file upload feature might be analyzed based on metrics like upload success rates, upload speeds, and file sizes.

2. A chat feature could be evaluated based on metrics such as message delivery times, message read receipts, and user engagement rates.

3. A search feature can be analyzed based on metrics such as query completion time, accuracy

of results, and frequency of use.

By bridging user actions with these granular analyses, developers can better understand both the direct and indirect impacts of features on user experience and system performance.

Adoption of action-based analysis in UAM

Action-based metrics were adopted in the Usage Analytics Method (UAM) for several reasons, rooted in their practicality and capacity to provide insightful data for software evaluation and refinement. User actions, such as clicks, navigation paths, and interaction durations, are directly observable through modern analytics tools embedded within software applications. These tools enable developers to collect data efficiently and with minimal disruption to the user experience, facilitating real-time and post-hoc evaluations of system performance and user engagement (Karlsson et al., 2007a). The use of these metrics extends beyond data collection, as they serve as proxies for understanding complex user behaviors. By analyzing patterns in user actions, developers can infer intentions, identify usability bottlenecks, and prioritize features that align with user needs. This foundational understanding of user behavior is critical for iterative improvements, ensuring that software systems remain relevant and user-centered (Pachidi et al., 2014). Additionally, action-based metrics are highly scalable, making them suitable for applications ranging from small-scale experiments to enterprise-level deployments. They allow for the consistent collection and analysis of data across diverse user bases and platforms, adapting to dynamic and evolving software ecosystems. This scalability ensures that the insights derived from these metrics remain robust and applicable across various contexts, enhancing their utility in guiding software development decisions (Kang et al., 1990).

Despite their utility, action-based metrics should be seen as the starting point rather than the endpoint of usage analysis. Complementary methods, such as qualitative user feedback, A/B testing, and machine learning models, can provide additional depth. For example, feedback loops derived from user surveys can contextualize quantitative metrics, revealing underlying motivations or pain points. Machine learning techniques can identify patterns in user actions that may be indicative of latent needs or preferences Tang and Zhang (2011). To address these gaps, the UAM integrates a layered approach that combines action-based metrics with higher-level feature evaluations and contextual analysis. This hybrid approach ensures that software developers are equipped with a comprehensive toolkit for assessing user engagement and feature utility, ultimately driving better-informed decisions in the design and refinement processes.

2.3 Role of Usage Data in Software Development Life-cycle

Agile software development is well-known for its focus on close customer collaboration and customer feedback (Abrahamsson et al., 2002; Mougouei and Powers, 2017). In emphasizing flexibility, efficiency and speed, agile practices have led to a paradigm shift in how software is developed. The concept of continuous deployment, i.e. the ability to deliver software functionality frequently to customers and subsequently, the ability to continuously learn from real-time customer usage of software, has become attractive to companies due to the potential of having even shorter feedback loops (Creswell, 2002; Chen et al., 2011; Fabijan et al., 2016b). To realize faster release cycles of software applications, companies embraced cloud technologies for their built-in database, security, workflow, user interface, and other tools that help in building powerful business applications, mobile applications, and websites. Since the entire application is hosted on the cloud, customers need not worry about IT Infrastructure, upgrades, updates, up-time and backups. The adoption of cloud computing has shown a different approach to managing requirements, adding frequent and rigorous experimentation to the development process. The combination of continuous deployment and cloud paradigms enables faster delivery of applications to gain knowledge regarding the success of the changes implemented to the application in the post-deployment stage. Applications that run in the cloud are delivered as a service so companies do not have to buy and maintain hardware and software to run them or IT teams to manage and maintain complicated deployments. Although the responsibility of software development activities such as Requirement Gathering, Analysis, Design, Construction (Development) and Testing are shared between the software developer and the cloud provider, the deployment process is mainly under the responsibility of the cloud provider (Olsson and Bosch, 2014c). However, while agile practices have succeeded in involving the customer in the development cycle, there is an urgent need to continuously learn from customer usage of software also after delivering and deployment of the software product (Iivari et al., 2000; Krusche and Alperowitz, 2014). Typically, feedback on a software system is collected during pre-deployment phases, i.e. before and during development. Most often, this is done by applying techniques that allow customers to engage in problem definition, requirements engineering and system evaluation and validation. The collection of customer feedback has always been important for R&D teams in order to understand better what customers want, R&D teams are becoming increasingly multi-disciplinary to include all functions, that the full potential of customer data can be utilized (Montes et al., 2013). Applications evolve over time and hence, application characteristics need to be adjusted, adapted and updated according to emerging customer requirements and needs. This implies that mechanisms for post-deployment customer collaboration are as important as those used during the pre-development and development phases of a system. One technique that has emerged due to the online nature of most software-intense systems today is the opportunity to continuously collect post-deployment data, i.e. data generated by the product after commercial deployment. This data can be operational data reflecting product performance, diagnostic data recording product behavior, or it can be data indicating feature usage. For online technologies such as Web 2.0 software, software-as-a-service systems, and cloud computing services, the collection of post-deployment data is a well-established technique used for continuous collection of information about product usage (Bosch, 2012; Helfert et al., 2012; Montes et al., 2013; Bauer et al., 2017).

2.4 Analysis Techniques for Software Feature Usage

In software development, a feature is a component of software functionality, complementing the core software while adding additional value (adapted from (De Chaves et al., 2011)). Typically, features are added incrementally, at various stages in the life-cycle, often by different groups of developers. Although the companies are aware that some features, at some point in time, cost more than what they add in terms of value, they may not have the data to support this and to understand when this happens. Although there are techniques available for collecting user feedback to understand which features are useful to the users, yet rarely answer why those features are important for them and also these feedback are typically not applied as part of a continuous feedback loop. As a result, the selection and prioritization of features are far from optimal, and the product may deviate from what the users need (Creswell, 2002).

2.4.1 Usage Data and User Feedback Collection Techniques

Five electronic database resources were used to primarily extract data for synchronizations in this research. These include IEEE Xplore, ACM Digital Library, ScienceDirect, Springer, and Google Scholar. Title, abstract and index terms were used to conduct a search for published journals papers, conference proceedings, workshops, symposiums, book chapters and IEEE bulletins. The literature review was conducted by searching for articles with the keyword combination “quantitative feedback” OR “usage data” AND “software” AND “deployment” OR “development” in the title, abstract or index terms resulting in around 300 articles. Included papers are only written in the English language, and published in conferences and journals in the relevant computer science and engineering domains resulting in 140 papers. Thereafter, papers are excluded by carefully examining the title and abstract limiting the number of articles to around 60. Furthermore, in the next iteration, additional relevant papers are selected based on backward citations in the selected papers. Finally, a total of 24 papers were deemed highly relevant to this research.

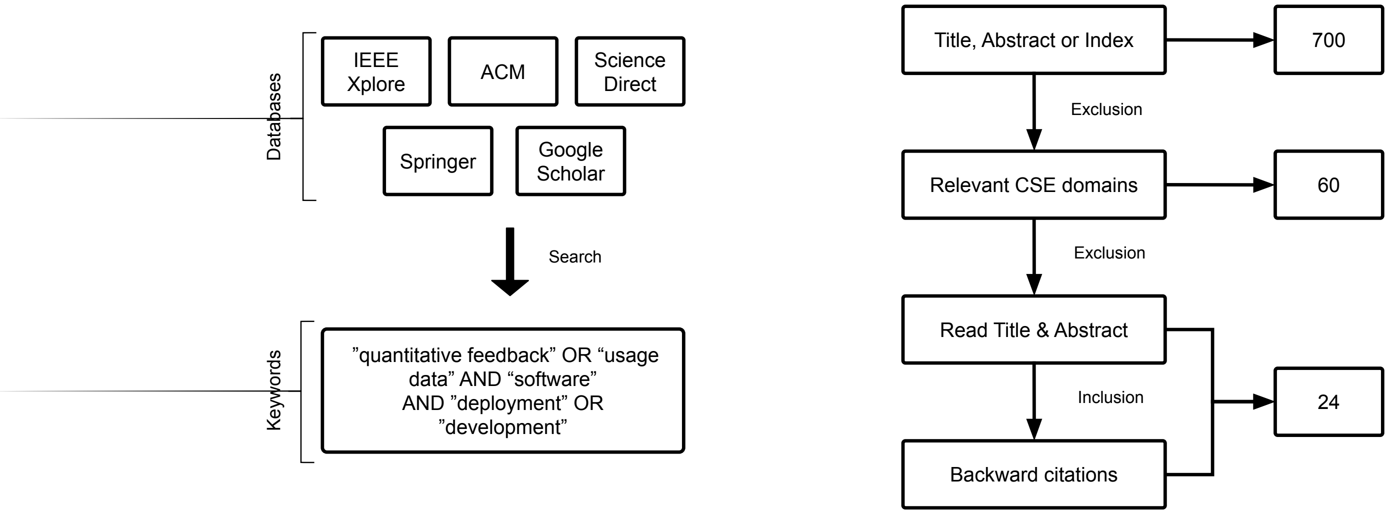


Figure 2.2: Summary of Systematic literature review process followed for reviewing usage data and user feedback collection techniques

Post-deployment feedback is critical in the software development lifecycle to ensure that applications evolve based on real user needs and behavior. This section consolidates the user data collection techniques employed by various researchers, emphasizing their significance in gathering comprehensive and actionable user feedback post-deployment. Table 2.1 lists different user data collection techniques employed in software development process, highlighting their application in pre-development, development, and post-deployment stages

1. **Automated Feedback Mechanisms** are embedded within software applications to collect user data in real-time. These tools can prompt users for feedback during their interaction with the software and collect data on usage patterns without requiring active user

Table 2.1: Table summarizing the user data collection techniques used in different stages of the software development process

	Software development stage												
	Pre-development					**Development**				**Post Deployment**			
Reference	F2F Comm.	Prototypes	Comm. tools	User as Developer	User Activities	Experiments	Developer as User	Standalone Appn.	Integrated Feedback	Product Usage Data	Social Media	Bug/Crash Reports	Online Forums
Maalej et al. 2009	X	X		X	X	X		X					
Muthitacharoen and Saeed 2009						X					X	X	
Wilcox et al. 2010		X			X								
Labib et al. 2010		X											
Schneider et al. 2010	X							X					
Chen et al. 2011	X	X			X								
Arias et al. 2012	X			X	X								
Olsson et al. 2012				X						X			
Poppendieck and Cusumano 2012							X						
Hess et al. 2013	X		X	X	X			X			X		X
Jakobi and Stevens 2013	X									X			
Lee et al. 2013	X	X			X					X			
Ogonowski et al. 2013	X	X	X	X	X	X		X		X	X	X	X
Holmström Olsson and Bosch 2013				X		X				X			
Pagano and Bruegge 2013			X					X	X			X	X
Fagerholm et al. 2014		X				X		X					
Krusche and Alperowitz 2014	X	X	X					X	X	X		X	
Krusche and Bruegge 2014		X											
Meijer and Kapoor 2014						X							
Claps et al. 2015						X							
Fabijan et al. 2015	X	X			X	X	X			X	X	X	X
Lindgren and Münch 2016		X				X							
Bauer et al. 2017									X	X		X	X
Mougouei and Powers 2017	X		X		X								
Hamiot and Verlaine 2024											X		
Rahimi and Jahanian 2023			X							X		X	X
Amininiaki and Saidi 2024		X				X				X		X	X
Ferreira and Costa 2024	X	X			X	X							
Iqbal and Ahmed 2024		X				X							X
Masoudi and Ghassemi 2024						X							
Sabbagh and Khalil 2024										X		X	X
Rangel and Martinez 2024		X			X					X			
Tarmuji and Abdullah 2024	X		X							X		X	X

input. For instance, Maalej et al. (2009) and Chen et al. (2011) highlight the efficiency of automated feedback systems in capturing immediate insights into user interactions, facilitating timely adjustments and continuous improvement. Hamiot and Verlaine (2024) further discuss the integration of automated feedback with social media analysis, enhancing the scope of real-time feedback collection. The study explore the utilization of social media for gathering post-deployment feedback. The study demonstrates how social listening tools can be used to monitor mentions of software products across various social media platforms. By analyzing these mentions, developers can identify emerging issues and trends that might not be reported through formal channels like bug reports. This proactive approach to feedback collection helps in maintaining a high level of user satisfaction and quickly addressing potential public relations issues. However, while efficient, these systems can sometimes be intrusive, leading to user fatigue. Additionally, they may not capture the depth of qualitative insights that manual feedback methods can provide, which can limit the understanding of the context behind user actions.

2. **In-app feedback forms** are another popular method for collecting user feedback directly within the software interface. These forms can be triggered contextually based on specific user actions or at predetermined intervals, allowing for immediate and relevant feedback collection. Wilcox et al. (2010) and Arias et al. (2012) discuss the benefits of in-app feedback forms in collecting qualitative user insights that are contextually relevant and immediate. Rahimi and Jahanian (2023) highlight the use of in-app feedback combined with social media insights to provide a more comprehensive view of user experiences. The study highlights how user-generated content on platforms like Twitter, Reddit, and dedicated forums can provide invaluable insights into user satisfaction and pain points. By employing natural language processing (NLP) techniques, developers can systematically analyze this feedback to identify recurring issues and gauge the sentiment surrounding their software. This method enhances the post-deployment phase by offering a continuous, real-time stream of user opinions and experiences, which can be directly fed into the development cycle to prioritize bug fixes and feature enhancements. Despite these advantages, in-app feedback forms can interrupt the user experience, potentially leading to incomplete or hurried responses. They also rely on user initiative to provide feedback, which can result in lower response rates. Social media monitoring tools analyze various forms of user-generated data, such as engagement metrics, sentiment analysis,

and behavioral trends, to provide actionable information for decision-making. While the primary context of social media monitoring tools is marketing and content optimization, their underlying principles of data collection, analysis, and application align closely with the objectives of usage analytics methods in software systems.

3. **Log analysis and event monitoring** involve capturing detailed records of user interactions with the software. Event logs track every action a user takes within the application, and analyzing these logs can reveal usage patterns, errors, and other significant events that indicate user behavior and software performance. Hess et al. (2013); Holmström Olsson and Bosch (2013) emphasize the value of event logs in detecting subtle usage patterns and anomalies, providing a comprehensive understanding of how users interact with software. Amininiaki and Saidi (2024) discuss the use of advanced analytics techniques to interpret log data, enhancing its utility in the development process. The study highlights how these tools can be used throughout the software development lifecycle to maintain a continuous feedback loop. By integrating feedback mechanisms directly into the software, developers can collect usage data in real-time, allowing for more agile and responsive development practices. This approach ensures that user feedback is always at the forefront of development decisions, leading to a more user-centric product. However, log analysis can generate large volumes of data, making it challenging to extract meaningful insights without advanced analytical tools. Moreover, it may not capture the context or reasons behind user actions, limiting its effectiveness in understanding user intent.

4. **Surveys and questionnaires** are widely used to collect structured feedback from users post-deployment. These tools can be distributed via email, embedded within the application, or presented on the company website, allowing for a broad reach. Muthitacharoen and Saeed (2009); Fabijan et al. (2015) highlight the importance of surveys in gathering detailed user opinions and satisfaction levels, complementing automated data collection methods. Ferreira and Costa (2024) highlight the integration of surveys with other user feedback mechanisms to improve response rates and the quality of collected data. By involving users early through prototypes and observing their activities, developers can gather detailed feedback on feature usability and performance. Structured experiments, such as A/B testing, further refine this feedback, ensuring that only the most effective and user-friendly features make it into the final product. This method contributes to a more

efficient development process by reducing the number of iterations needed to achieve a satisfactory product. However, surveys and questionnaires often suffer from low response rates and can be subject to response biases. They also typically provide feedback only at specific intervals, which may miss real-time issues, making them less effective for immediate feedback needs.

5. **A/B testing** is a method where two versions of a software application are compared to determine which version performs better in terms of user engagement and satisfaction. Users are randomly assigned to one of the two versions, and their interactions are monitored and analyzed. Lee et al. (2013); Olsson et al. (2012) demonstrate the effectiveness of A/B testing in isolating the effects of specific changes, enabling data-driven decisions about software improvements. Iqbal and Ahmed (2024) discuss how A/B testing, combined with online forum feedback, can enhance the reliability of the findings and provide more nuanced insights into user preferences. The paper highlights how controlled experiments can be used to test new features or changes in a live environment with a subset of users. Feedback collected from these experiments, combined with discussions and reports from online forums, provides a robust dataset for analyzing user behavior and preferences. This dual approach ensures that changes are data-driven and user-approved before being rolled out to the entire user base. However, A/B testing can be complex to design and interpret, especially if multiple variables are involved. It also requires a sufficiently large user base to produce statistically significant results, which may not always be feasible.

6. **Clickstream data analysis** involves tracking the sequence of clicks made by a user while navigating through a software application. Analyzing this data helps identify user navigation patterns and potential friction points. Holmström Olsson and Bosch (2013); Bauer et al. (2017) discuss how clickstream analysis provides insights into user journeys, helping developers optimize the user interface and overall user experience. Masoudi and Ghassemi (2024) explore the use of enhanced clickstream analysis techniques to better understand the context of user actions. By implementing controlled experiments, developers can test hypotheses about user behavior and feature effectiveness. The results of these experiments provide empirical evidence that guides development decisions, ensuring that the final product aligns with user needs and preferences. Despite its advantages, clickstream data can indicate what users do but not why they do it, lacking the context behind user actions.

It also generates vast amounts of data that require significant processing and analysis to derive actionable insights.

7. **Bug tracking and incident reports** are systems that collect reports of software defects and incidents submitted by users. These reports help identify and prioritize issues that need to be addressed to improve software reliability and performance. Pagano and Maalej (2013); Labib et al. (2010) emphasize the role of bug tracking in maintaining software quality and addressing user-reported problems promptly. Sabbagh and Khalil (2024) discuss the integration of bug tracking with usage data and online forums to create a more comprehensive feedback loop. By aggregating data from these sources, developers can create a comprehensive view of how users interact with their software and where they encounter problems. This holistic approach allows for more precise identification of issues and better prioritization of development tasks, ultimately leading to a more stable and user-friendly product. However, bug reports often require users to take the initiative to submit them, which can result in under-reporting. Additionally, they tend to focus on negative experiences, potentially missing positive user feedback, which is also valuable for understanding what works well in the software.

8. **Real-time dashboards and analytics platforms** provide immediate visibility into user behavior and software performance metrics. These tools aggregate data from various sources and present it in a visual, easily interpretable format. Krusche and Alperowitz (2014); Mougouei and Powers (2017) highlight the benefits of real-time dashboards in facilitating quick responses to emerging issues and trends in user behavior. Tarmuji and Abdullah (2024) expand on the use of real-time dashboards integrated with communication tools to improve collaboration and feedback interpretation. They argue that direct interactions between developers and users, facilitated by modern communication tools like video conferencing and collaborative platforms, ensure a clearer understanding of user needs and more precise feedback. These techniques help bridge the gap between developers and end-users, fostering a more user-centric development process where feedback is not only collected but also interpreted correctly and acted upon swiftly. While offering valuable real-time insights, these platforms can be resource-intensive to set up and maintain. They also require careful interpretation to avoid misinterpreting data trends, which can lead to incorrect conclusions about user behavior.

9. **User interviews and focus groups** involve direct interaction with users to gather in-depth qualitative insights into their experiences and preferences. These methods provide rich contextual information that complements quantitative data. Ogonowski et al. (2013); Bauer et al. (2017) discuss how user interviews and focus groups uncover nuanced user needs and satisfaction drivers that may not be evident from automated data collection alone. Rangel and Martinez (2024) highlight the combination of prototypes with user interviews to validate design concepts early in the development process. The paper discusses the implementation of interactive prototypes that allow potential users to engage with a simplified version of the software. Feedback from these interactions is collected through user activities and structured experiments, providing early validation of design concepts and feature functionality. This approach helps in refining the user interface and user experience (UI/UX) before substantial resources are committed to full-scale development, thereby reducing the risk of costly redesigns. However, these methods can be time-consuming and may not be scalable for large user bases. They also rely on the subjective interpretation of user feedback, which can introduce biases, making it challenging to generalize the findings.

These studies contribute significantly to the software development process by emphasizing the importance of integrating user feedback at various stages. They highlight how modern techniques such as social media analysis, interactive prototypes, controlled experiments, and integrated feedback mechanisms can provide more detailed and actionable insights into user behavior and preferences. By leveraging these techniques, developers can make more informed decisions, reduce the time and cost associated with iterative development, and ultimately deliver higher-quality software that meets user needs more effectively.

The user data collection techniques employed throughout the software development process, while invaluable, come with their own set of challenges that developers must navigate. Automated feedback mechanisms, although efficient in capturing real-time user interactions, can sometimes be intrusive, leading to user fatigue and potentially limiting the depth of qualitative insights obtained. In-app feedback forms, while contextually relevant, can interrupt the user experience, resulting in incomplete or hurried responses, and often suffer from low response rates due to reliance on user initiative. Log analysis and event monitoring generate vast amounts of data, making it challenging to extract meaningful insights without advanced analytical tools, and they often lack the context behind user actions. Surveys and questionnaires, despite their broad reach, frequently suffer from low response rates and response biases, and typically provide feedback only at specific intervals, missing real-time issues. A/B testing, although effective in isolating the effects of specific changes, can be complex to design and interpret, requiring a sufficiently large user base to produce statistically significant results. Clickstream data analysis provides valuable insights into user navigation patterns but lacks the context behind user actions and generates data that require significant processing to derive actionable insights. Bug tracking and incident reports depend on user initiative for submission, leading to potential under-reporting, and focus primarily on negative experiences, potentially overlooking positive feedback. Real-time dashboards and analytics platforms, while offering immediate insights, are resource-intensive to set up and maintain, and require careful interpretation to avoid misinterpreting data trends. Lastly, user interviews and focus groups provide rich qualitative insights but are time-consuming, may not be scalable for large user bases, and rely on subjective interpretation, introducing biases that can challenge the generalization of findings. Collectively, these challenges underscore the need for a balanced approach that integrates various feedback mechanisms, advanced analytical tools, and thoughtful interpretation to effectively leverage user feedback in the software development process.

2.4.2 Traditional Methods of Feature Prioritization

A new systematic literature review was conducted to explore and to establish state-of-the-art methods and techniques for prioritizing features of the software applications. The systematic literature review conducted for this research involved a rigorous and iterative process to ensure the inclusion of high-quality and relevant studies. The final selection of 32 papers provides a solid foundation for understanding the current state of feature prioritization and usage analytics, highlighting both the strengths and limitations of existing methods and underscoring the need for the proposed research. The detailed insights gained from this review inform the development of a novel usage analytics method aimed at improving feature prioritization in software development.

Traditional methods of feature prioritization in software development often rely on techniques such as stakeholder analysis, cost-value analysis, and voting mechanisms. These methods are well-documented in the literature but have several limitations that restrict their effectiveness in dynamic and complex development environments.

1. **Stakeholder Analysis** involves identifying and prioritizing features based on input from key stakeholders. While this method ensures that the most influential voices are heard, it

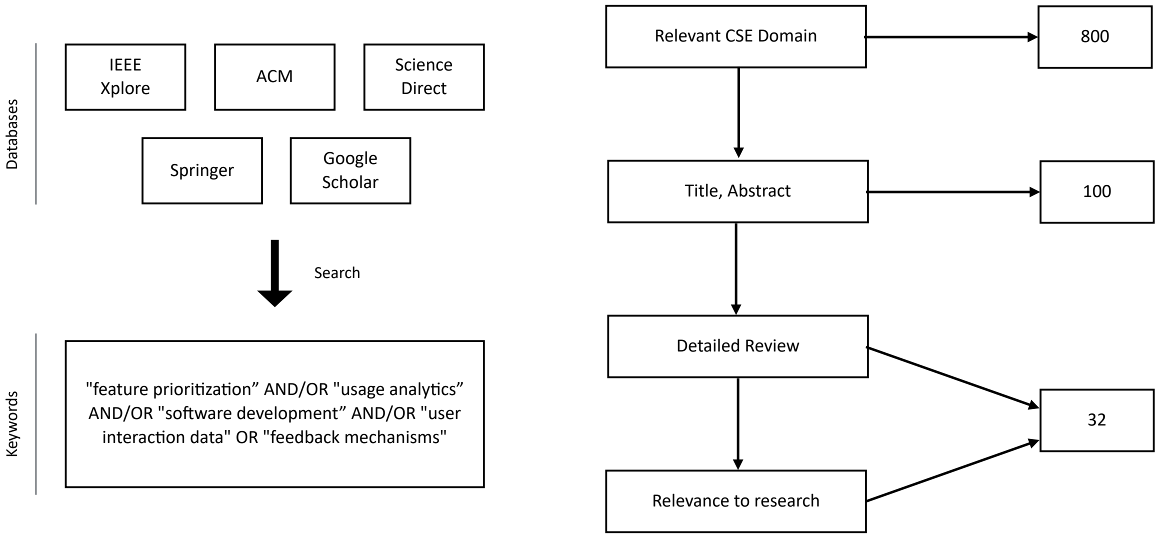


Figure 2.3: Summary of Systematic literature review process followed for reviewing feature prioritization process in software development

can lead to biases and does not always reflect the actual usage patterns and needs of end-users. Studies by Karlsson et al. (2007b) highlight the limitations of stakeholder-driven approaches, particularly in large-scale projects. More recent work by Gorschek et al. (2017) has further explored the limitations and potential biases inherent in stakeholder analysis, emphasizing the need for more objective methods to complement stakeholder input.

2. **Cost-value analysis** assesses features based on their implementation cost and expected value to the user. This method provides a structured approach to decision-making but often lacks real-time feedback and can be limited by the accuracy of cost and value estimates. As noted by Ruhe and Greer (2002), the subjective nature of these estimates can result in sub-optimal prioritization. More recent studies by Zhang et al. (2018); Xia et al. (2019) have explored advanced techniques to improve the accuracy of cost-value analyses, including the use of machine learning to predict implementation costs and user value more accurately

3. **Voting mechanisms**, such as the Planning Poker technique used in Agile methodologies, allow team members to vote on feature priorities. While this democratic approach encourages team involvement, it can be influenced by group dynamics and may not always lead to the selection of the most critical features. Research by Moløkken-Østvold and Jørgensen (2003) discusses the potential for consensus bias in these methods. A study by Kakar (2020) has further examined how group dynamics and the social influence of team members can affect the outcomes of voting mechanisms, suggesting ways to mitigate these biases.

Advanced Usage Analytics in Feature Prioritization

Advanced usage analytics involves the systematic collection and analysis of data generated by users as they interact with software applications. This approach leverages various data sources, including log files, user feedback, and telemetry data, to provide a detailed understanding of how features are used and their impact on user experience.

1. **User Interaction Metrics:** Key metrics such as frequency of use, session duration, and error rates can provide valuable insights into which features are most important to users and where improvements are needed. Studies by Buse and Zimmermann (2012b) emphasize the importance of these metrics in identifying critical features. More recent

research by Shams et al. (2020) has expanded on this by identifying additional metrics, such as user engagement levels and feature abandonment rates, that can provide deeper insights into user behavior

2. **Behavioral Analysis:** Analyzing user behavior patterns, such as navigation paths and feature sequences, can reveal underlying user needs and preferences that may not be evident from explicit feedback. Research by Claypool et al. (2001c) demonstrates how behavioral analysis can enhance feature prioritization. Recent studies by Müller et al. (2019); Gupta et al. (2021) have utilized advanced techniques like sequence mining and clustering algorithms to uncover complex behavioral patterns that inform feature prioritization.

3. **Real-Time Feedback:** Usage analytics can provide real-time feedback to developers, allowing for more agile and responsive decision-making processes. As discussed by Rodden et al. (2010a), real-time analytics enable continuous improvement and rapid adaptation to changing user needs. Recent advancements in real-time analytics platforms, as detailed by Liu et al. (2020), have made it easier for developers to integrate real-time data into their workflows, enhancing their ability to respond quickly to user feedback.

Table 2.2 provides a comparative analysis of key methodologies used in the field of feature prioritization and usage analytics, highlighting the studies and their key findings. Each study has been carefully selected to illustrate the evolution of feature prioritization methods and the integration of usage analytics into these processes. This analysis provides a comprehensive overview of the strengths and limitations of each methodology, illustrating the evolution and integration of these techniques in software development.

Methodology	**Studies**	**Key Findings**
Stakeholder analysis	Karlsson et al. (2007b), Gorschek et al. (2017)	Identified biases in stakeholder-driven prioritization, suggesting the need for more objective approaches to complement stakeholder input. Explored the limitations and potential biases in stakeholder analysis, suggesting the need for more robust methods to capture diverse perspectives.

Cost-value analysis	Ruhe and Greer (2002), Zhang et al. (2018), Xia et al. (2019)	Discussed the limitations in the accuracy of cost and value estimates, highlighting the subjectivity involved in these assessments. Improved the accuracy of cost-value analysis using machine learning techniques including neural networks, support vector machines, and ensemble methods like random forests to better predict implementation costs and user value. Further refined methods for predicting implementation costs and user value, integrating more sophisticated analytical models.
Voting mechanisms	Moløkken-Østvold and Jørgensen (2003), Kakar (2020)	Highlighted the potential for consensus bias in group decision-making, emphasizing the influence of group dynamics on the outcomes. Examined the influence of group dynamics on voting mechanisms, offering insights into mitigating consensus bias in feature prioritization.
Usage analytics	Buse and Zimmermann (2012b), Shams et al. (2020)	Emphasized the importance of user interaction metrics, such as frequency of use and error rates, in identifying critical features. Identified additional metrics for user engagement and feature abandonment, enhancing the understanding of user interactions with software.
Behavioral analysis	Claypool et al. (2001c), Müller et al. (2019), Gupta et al. (2021)	Demonstrated the value of analyzing user behavior patterns to enhance feature prioritization, particularly through navigation paths and feature sequences. Utilized sequence mining to uncover complex behavioral patterns that inform feature prioritization, providing deeper insights into user needs. Applied clustering algorithms to analyze user behavior, identifying distinct user segments and their preferences to guide feature development.

Real-time feedback	Rodden et al. (2010a), Liu et al. (2020)	Discussed the benefits of real-time analytics for agile decision-making, allowing for continuous improvement and rapid adaptation to user needs. Detailed advancements in real-time analytics platforms, highlighting the impact of immediate feedback on software development efficiency.
Hybrid approaches	Olsson and Bosch (2014c), Alahyari et al. (2017)	Provided frameworks for integrating stakeholder input with usage data, showing the effectiveness of combining traditional methods with data-driven insights. Demonstrated the effectiveness of hybrid approaches in feature prioritization, combining qualitative and quantitative data for better decisions.
Data-driven decision-making	Fitzgerald and Stol (2017b), Waller and Fawcett (2020)	Outlined best practices for incorporating analytics into software development processes, stressing the importance of data quality and governance. Highlighted the importance of data governance and quality in ensuring successful data-driven decision-making frameworks in software development.
Empirical validation	Kitchenham and Charters (2004)	Highlighted the importance of validating research methods through empirical studies to ensure the reliability and generalizability of findings.
Challenges and best practices	Sjøberg et al. (2005), Deka and Vemuru (2021)	Provided strategies for overcoming challenges in implementing usage analytics, such as data integration and privacy concerns. Provided practical recommendations for modern software development environments, addressing common challenges in implementing usage analytics.

Table 2.2: Current feature prioritization techniques and practices employed in the software development domain

The comparison in Table 2.2 highlights several important trends and limitations in existing tools and methodologies. Most tools excel in capturing operational metadata, such as performance metrics and resource utilization, but often fall short when it comes to user-level behavior analysis. This gap is particularly significant for tools that rely exclusively on web-based monitoring, overlooking alternative interfaces like mobile applications and command-line tools.

Another notable observation is the limited emphasis on data completeness and reliability, which are critical for generating actionable insights. While certain methodologies address specific aspects of user interaction, such as clickstream analysis or session tracking, they fail to provide a holistic view of user behavior across multiple interfaces. This limitation reduces the utility of these tools in scenarios where comprehensive user behavior insights are necessary.

The analysis in Table 2.2 highlights the diversity of approaches used for feature prioritization, ranging from stakeholder-driven methods to advanced data-driven techniques. While each method offers unique strengths, the table reveals several critical gaps and limitations that necessitate the development of more comprehensive solutions.

Stakeholder-driven prioritization, for instance, is widely used due to its accessibility and involvement of key decision-makers. However, as Karlsson et al. (2007b) and Gorschek et al. (2017) identified, this approach is prone to biases, particularly when influential stakeholders disproportionately shape priorities. Such imbalances often result in misaligned decisions that may not accurately reflect end-user needs, underscoring the necessity of complementing stakeholder input with objective data.

Data-driven methods, such as cost-value analysis and usage analytics, attempt to address these limitations by introducing metrics and algorithms for prioritization. Cost-value analysis, as discussed by Ruhe and Greer (2002), Zhang et al. (2018), and Xia et al. (2019), provides a structured framework to assess features based on their implementation cost and user value. However, the subjective nature of cost and value estimates remains a challenge, as noted by Ruhe and Greer (2002). Recent advancements, such as machine learning-based enhancements proposed by Zhang et al. (2018) and Xia et al. (2019), have improved the accuracy of these estimates, but further refinements are needed to ensure their scalability and reliability.

Usage analytics and behavioral analysis stand out for their ability to derive insights directly from user interactions. Buse and Zimmermann (2012b) and Shams et al. (2020) demonstrated the potential of metrics like engagement levels and feature abandonment rates to inform decision-making. However, these methods require robust data collection and integration mechanisms to ensure the completeness and reliability of the extracted data. Behavioral analysis methods, such as those employing sequence mining and clustering (Müller et al., 2019; Gupta et al., 2021), provide deeper insights into user patterns but are computationally intensive, which may limit their practical applicability.

Hybrid approaches that combine stakeholder input with data-driven insights, as proposed by Olsson and Bosch (2014c) and Alahyari et al. (2017), offer a promising middle ground. These approaches leverage the strengths of qualitative and quantitative methods to balance diverse perspectives, addressing the limitations of standalone techniques. Despite their potential, the implementation of hybrid methods requires careful consideration of data integration and governance to maximize their effectiveness.

Finally, the increasing emphasis on real-time feedback and empirical validation highlights the evolving needs of modern software development. As noted by Rodden et al. (2010a) and Liu et al. (2020), real-time feedback mechanisms enable agile decision-making and continuous improvement. Empirical validation, as advocated by Kitchenham and Charters (2004) and Sjøberg et al. (2005), ensures that proposed methods are reliable and generalizable, addressing practical challenges such as data privacy and integration (Deka and Vemuru, 2021).

These findings from Table 2.2 underline the necessity for a comprehensive usage analytics framework that integrates multi-interface data sources, prioritizes reliability, and bridges operational and behavioral insights. The proposed framework in this research addresses these gaps, offering a holistic approach to feature prioritization and decision-making.

2.5 Comprehensive Overview of Data Types and Analytics Metrics

Understanding user behavior and prioritizing software features through advanced usage analytics require a robust framework built on systematically identified data types and analytics metrics. This section expands upon the foundational concepts of data collection and analysis, providing an exhaustive catalog of data types and metrics relevant to software platforms. Furthermore, it describes the methodological rigor employed to identify, select, and apply these elements to the research platforms.

2.5.1 Key Data Types

Data types form the foundational layer for extracting actionable insights from user interactions. They represent raw inputs gathered during user engagement with software platforms. Table 2.3

outlines a detailed taxonomy of data types, emphasizing their relevance across various stages of analysis.

Table 2.3: Summary of Data Types and Their Applications

Data Type	**Definition**	**Use Cases**
Event Logs	Records of user actions, including timestamps and context.	Tracking user behavior, feature usage, and workflow patterns.
Session Data	Data capturing the start and end of a user's interaction with the application.	Identifying user engagement levels and patterns.
Error Logs	Records of errors encountered by users during their interactions.	Diagnosing usability issues and improving system reliability.
Navigation Data	Sequence of pages or screens visited by a user.	Understanding user workflows and pain points in navigation.
Interaction Logs	Details about specific interactions, such as button clicks, form submissions, or drag-and-drop actions.	Analyzing feature adoption and usability.
Demographic Data	Information about user attributes, such as age, location, or experience level.	Segmenting users for personalized insights.
Feedback Data	Explicit feedback provided through surveys, ratings, or comments.	Correlating user opinions with interaction data.
Performance Data	Metrics like load times, latency, and responsiveness of features.	Evaluating system performance and user experience.
Device and Platform Data	Details about the device, operating system, and browser used.	Tailoring the user experience across different platforms.
Usage Context	Environmental data such as time of day or location of use.	Understanding contextual factors influencing user behavior.

The identification and utilization of key data types form the cornerstone for extracting actionable insights from user interactions in software systems. These data types not only act as raw inputs but also enable the translation of complex behavioral patterns into meaningful software development decisions. Particularly, they play a crucial role in prioritizing and enhancing features based on user behavior.

Informed by an extensive systematic literature review encompassing 32 significant studies discussed earlier, this research derived a comprehensive taxonomy of data types frequently employed in software analytics. The review synthesized insights from diverse sources to identify data points and metrics that are foundational to understanding user behavior and their interaction with software systems. Each study was carefully analyzed to identify recurring themes and evidence supporting the inclusion of these data types.

The importance of event logs has been consistently highlighted across studies such as those by Lou et al. (2013); Wang et al. (2017) and Zhang et al. (2011). These logs, capturing granular user actions complete with timestamps, context, and system responses, serve as critical tools for debugging and comprehending interaction patterns. Similarly, session data, encompassing the start and end points of user interactions, has been shown to enhance understanding of engagement and retention trends, with studies like Chen et al. (2013) and Reyes (2015) providing compelling evidence for its inclusion. This data type, enriched with metadata such as session duration, offers valuable insights into user behavior.

Navigation data, which records sequenced logs of pages or screens visited by users, has been another essential category supported by studies such as Xu et al. (2008); Gasparetti (2017). These studies demonstrate how navigation data uncovers usability bottlenecks and optimizes navigation workflows. Demographic data, encompassing attributes such as age, role, or location, has also been pivotal in tailoring software to user needs. This is substantiated by research like that of Muthitacharoen and Saeed (2009) and Huang and Rust (2018), which highlight its role in enabling personalized software adaptations. Additionally, performance metrics such as load times and system latency emerge as vital indicators, as emphasized by Puthal et al. (2015); Montes et al. (2013) in evaluating system reliability and user satisfaction.

The critical narrative accompanying each data type underscores their categorization into foundational and advanced metrics. Foundational metrics, such as event logs and session data, are universally applicable for understanding basic user behaviors. Advanced metrics, like feedback data, discussed in studies such as Pagano and Maalej (2013), and performance data, supported by the findings of Fu et al. (2013), facilitate nuanced insights into predictive analytics and system optimization. This categorization ensures that the taxonomy serves both generalizable and context-specific needs in software analytics.

The derivation of this superset of data types is firmly rooted in their demonstrated utility across varied contexts in prior research. The iterative process of refining this taxonomy incorporated systematic filtering and collaborative sessions with IBM developers, ensuring that theoretical constructs aligned with practical applications. This collaborative effort validated the industrial relevance of the data types and metrics while ensuring their adaptability to diverse software environments.

Furthermore, the superset was rigorously validated across multiple platforms, including IBM Watson Workspace and Odoo Notes. Validation efforts demonstrated the adaptability of the derived metrics in both industrial and non-industrial contexts, with the studies by Bezemer et al. (2010) and Pachidi et al. (2014) providing examples of cross-platform applicability. The findings highlight the necessity of extensible frameworks capable of accommodating diverse software environments, especially in cases where access to source code may be limited. This adaptability ensures that the research remains relevant across a spectrum of real-world applications, bridging the gap between theoretical insights and practical software development needs.

The taxonomy of data types presented in Table 2.3 serves as a foundational framework for the subsequent development of usage analytics methods. By systematically categorizing and validating these data types, the research ensures that the resulting analytics framework is robust, adaptable, and aligned with the evolving needs of software development. The taxonomy not only captures the breadth of user interactions but also provides a structured approach to deriving actionable insights from complex user behaviors. This foundational taxonomy forms the basis for the subsequent identification and application of analytics metrics, enabling a comprehensive understanding of user behavior and feature prioritization in software systems.

2.5.2 Analytics Metrics

Analytics metrics are derived from raw data types and offer actionable insights into user behavior. Table 2.4 presents the metrics grouped into categories based on their purpose and utility in feature prioritization and behavioral analysis.

Analytics metrics are derived from raw data types and offer actionable insights into user behavior, serving as pivotal tools for feature prioritization and behavioral analysis. The derivation of these metrics is supported by a systematic review of 32 key studies, as outlined earlier in this chapter, which provided a robust foundation for categorizing and refining metrics based on their purpose and utility.

Usage metrics, encompassing metrics such as frequency of use, time spent, and session length, are widely recognized in the literature for their role in quantifying user engagement. Lou et al. (2013) and Reyes (2015) emphasized the criticality of measuring frequency and duration to identify highly engaged features and assess user interaction levels. For instance, frequency of use tracks the number of interactions with a feature over a defined period, while time spent highlights the intensity of user engagement. These metrics are foundational in identifying patterns of user

Table 2.4: Summary of Analytics Metrics and Their Applications

Category	**Metric**	**Definition and Use Cases**
Usage Metrics	Frequency of Use	Number of interactions with a feature per session or time period. Useful for identifying highly engaged features.
	Time Spent	Duration of interaction with a specific feature. Indicates user engagement levels.
	Session Length	Total time spent in a user session. Helps identify patterns of user activity.
Behavioral Metrics	Feature Abandonment Rate	Percentage of users starting but not completing a task. Highlights usability issues.
	Transition Patterns	Sequences of user actions. Useful for understanding workflows and transitions.
	Task Completion Rate	Proportion of successfully completed tasks. Measures feature effectiveness.
Performance Metrics	Latency and Response Time	Time taken for a feature or application to respond. Helps optimize user experience and identify bottlenecks.
	Error Rates	Number of errors encountered during interactions. Identifies problematic areas of the application.
Engagement Metrics	Retention Rate	Percentage of users returning to use a feature after the first interaction. Assesses feature stickiness.
	Interaction Depth	Number of levels accessed within a feature. Determines the complexity of feature usage.
Satisfaction Metrics	Net Promoter Score (NPS)	Measures user satisfaction through a single question: likelihood of recommending a product.
	Sentiment Analysis	Emotional tone of user comments derived through natural language processing. Useful for understanding user perception.

activity, as demonstrated in studies by Xu et al. (2008) and Gasparetti (2017).

Behavioral metrics, such as feature abandonment rate, transition patterns, and task completion rate, delve into the nuances of user workflows. Feature abandonment rate, explored in studies by Pachidi et al. (2014) and Pagano and Bruegge (2013), identifies usability issues by analyzing the percentage of users who initiate but do not complete a task. Transition patterns, discussed by Ghezzi et al. (2014), reveal sequences of user actions, enabling a deeper understanding of user workflows and transitions between features. Task completion rate, a focus of

Chen et al. (2013), provides insights into the effectiveness and usability of features by measuring the proportion of successfully completed tasks.

Performance metrics, such as latency and response time, alongside error rates, are integral to optimizing user experience. Studies by Puthal et al. (2015) and Montes et al. (2013) underscore the importance of monitoring system responsiveness to identify bottlenecks and enhance application performance. Latency and response time measure the time taken for a feature or application to respond, while error rates track the frequency of issues encountered during interactions, as highlighted by Fu et al. (2013).

Engagement metrics, including retention rate and interaction depth, evaluate the sustained appeal and complexity of feature usage. Retention rate, discussed by Huang and Rust (2018), assesses feature stickiness by analyzing the percentage of users returning to use a feature after their first interaction. Interaction depth, examined in studies such as Gasparetti (2017), measures the number of levels accessed within a feature, offering insights into user exploration and engagement with complex functionalities.

Satisfaction metrics, such as Net Promoter Score (NPS) and sentiment analysis, provide direct indicators of user sentiment and satisfaction. Sentiment analysis, explored by Reyes (2015) and Pagano and Bruegge (2013), leverages natural language processing to decode the emotional tone of user comments, offering valuable perspectives on user perception. The NPS metric measures user satisfaction through a straightforward question about the likelihood of recommending a product, serving as a quick yet effective tool for gauging overall satisfaction.

The Usage Analytics method offers a systematic approach to analyzing user behavioral changes. By integrating data types and metrics into a structured framework, it bridges the gap between raw interaction data and actionable insights. This method's flexibility enables its adoption across diverse platforms, ensuring that developers can tailor it to their specific needs. Additionally, the approach emphasizes the iterative refinement of metrics, ensuring their relevance over time. Developers are not constrained to the metrics demonstrated in this research but are empowered to customize their approach, leveraging the super-set for guidance. This comprehensive overview of data types and metrics demonstrates the rigor and adaptability of the Usage Analytics method. The systematic process of identification, refinement, and application ensures that the method remains relevant across various software development contexts, providing developers with the tools needed to analyze user behavior effectively.

Developers from both IBM Academic Cloud and IBM Watson Workspace participated in a comprehensive year-long collaborative process that involved iterative applications of the usage analytics method. The participants were selected for their diverse expertise, which provided a robust foundation for refining and validating the customization of metrics and data types from the super-set. This process emphasized inclusivity and critical evaluation at each stage, with feedback loops designed to ensure the practicality and relevance of the findings. Participants ranged from senior developers and architects to operational managers, bringing together a wide array of skills including data analytics, software architecture, and full-stack development.

Each iteration involved developers systematically evaluating the proposed metrics and data types, offering detailed feedback regarding their relevance, usability, and industrial applicability. This iterative approach facilitated continuous improvement, ensuring that the resulting framework addressed real-world challenges in feature prioritization and behavioral analysis within software platforms.

Table 2.5: Industry Participant Demographics for Iterative Development of UAM, participants were stakeholders of both IBM Academic Cloud and IBM Watson Workspace.

Participant	**Role**	**Experience Level**	**Expertise**
Developer 1 (Case Study 1)	Senior Developer	10+ years	Python, Java, Data Analytics, Software Architecture
Developer 2 (Case Study 1)	Senior Developer	10+ years	Python, JavaScript, Software Architecture, Machine Learning
Developer 3 (Case Study 1)	Mid-Level Developer	5-10 years	Python, JavaScript, Full-Stack Development
Participant 1 (Case Study 2)	Operational Manager	15+ years	Project Management, Agile Methodologies
Participant 2 (Case Study 2)	Architect	12+ years	Software Architecture, Cloud Computing
Participant 3 (Case Study 2)	Senior Developer	10+ years	Python, Java, Data Analytics
Participant 4 (Case Study 2)	Developer	5-7 years	JavaScript, Full-Stack Development
Participant 5 (Case Study 2)	Developer	5-7 years	Java, Python, DevOps

The table above summarizes the demographics of the participants involved in the iterative refinement process for the usage analytics method, specifically for identifying the data types and key metrics suitable for the real-world applications they were responsible for development and maintenance. The selection of participants from diverse roles and backgrounds ensured a balanced and comprehensive approach to validating the proposed metrics and data types. By leveraging the expertise of senior developers and operational managers alongside mid-level contributors, the process incorporated a wide range of perspectives. This diversity was instrumental in refining the framework to ensure both its theoretical robustness and practical relevance.

This research significantly contributes to bridging the gap between theoretical insights and practical applications in software development. By systematically connecting user behavior analysis to feature prioritization decisions, it offers a rigorous and adaptable framework tailored to industrial needs. The iterative refinement and validation processes ensured that the metrics and methods were not only theoretically robust but also practically implementable across diverse organizational contexts. This approach highlights the novelty and industrial relevance of the research, showcasing its potential to inform and improve decision-making in software development practices.

2.6 Gap Analysis of Existing Literature

The review of the literature on feature prioritization methods in software development reveals several significant limitations in the current techniques. These limitations establish the need for the research conducted in this thesis and highlight the potential contributions of a new approach focused on usage analytics.

Stakeholder Analysis: Bias and subjectivity in stakeholder analysis often lead to skewed prioritization decisions. Influential stakeholders can disproportionately impact decisions, potentially overlooking the actual needs and behaviors of end-users Karlsson et al. (2007b); Gorschek et al. (2017). Moreover, stakeholder analysis typically fails to capture the diverse perspectives and needs of the entire user base, resulting in a narrow view that does not reflect actual usage patterns.

Cost-Value Analysis: The cost and value estimates used in cost-value analysis are often subjective and can be inaccurate, leading to suboptimal prioritization decisions. The complexity and unpredictability of software development projects exacerbate these inaccuracies Ruhe and Greer (2002); Zhang et al. (2018); Xia et al. (2019). Additionally, cost-value analysis does not typically incorporate real-time data, making it difficult to adapt to changing user needs and preferences, which can lead to outdated or irrelevant prioritization.

Voting Mechanisms: Voting mechanisms are prone to consensus bias, where the opinions of more vocal or influential team members can sway the results, leading to decisions that may not accurately reflect the best interests of the project Moløkken-Østvold and Jørgensen (2003);

Kakar (2020). The social dynamics within a team can significantly affect voting outcomes, making it challenging to achieve balanced and objective prioritization.

Data Overload: The vast amount of data generated from user interactions can be overwhelming, making it difficult to extract meaningful insights without sophisticated data processing and analysis techniques Buse and Zimmermann (2012b); Shams et al. (2020).

Complexity of Behavioral Analysis: Analyzing user behavior patterns requires advanced analytical skills and tools, which can be a barrier for many development teams Claypool et al. (2001c); Müller et al. (2019); Gupta et al. (2021).

Integration Challenges: Integrating real-time feedback mechanisms into existing development workflows can be complex and resource-intensive, often requiring significant changes to processes and systems Rodden et al. (2010a); Liu et al. (2020).

Given the limitations of traditional and advanced usage analytics methods, there is a clear need for a novel approach that integrates the strengths of both while addressing their weaknesses. The research conducted in this thesis aims to fill this gap by developing a usage analytics method that enhances feature prioritization in software development through systematic data analysis and real-time feedback mechanisms. By integrating stakeholder input with data-driven insights from usage analytics, this research aims to create a balanced and comprehensive feature prioritization framework that leverages both qualitative and quantitative data Olsson and Bosch (2014c); Alahyari et al. (2017). The research provides practical recommendations and best practices for overcoming common challenges in implementing usage analytics, such as data integration and privacy concerns, ensuring that the proposed methods can be effectively adopted in real-world development environments Kitchenham and Charters (2004); Sjøberg et al. (2005); Deka and Vemuru (2021). In summary, the limitations of current feature prioritization techniques underscore the need for a new approach that integrates usage analytics with traditional methods. This research aims to develop a robust framework that addresses these limitations and enhances the efficiency and effectiveness of the software development process through systematic analysis of user interaction data.

Despite significant advancements in usage analytics, persistent gaps highlight the limitations of existing methodologies. Challenges such as integrating heterogeneous data sources (Fagerholm et al., 2014; Appsero, 2023), accurately mapping user actions to features (Fabijan et al., 2016a; Joshi, 2024), and deriving insights that developers can act upon (Bosch, 2000; Lee et al., 2020) illustrate the disconnect between theoretical models and practical application. These issues underscore the necessity for comprehensive frameworks that link user behavior data with actionable feature prioritization strategies.

Another notable gap is the limited discussion on the scalability of metrics across varying software platforms. As organizations increasingly adopt cloud-based and SaaS solutions, the ability to generalize and adapt metrics becomes critical. Addressing these gaps requires a dual approach that combines robust theoretical grounding with empirical validation, enabling a seamless transition from academic insights to industrial application. Metrics identified in the literature, including frequency, time spent, and consistency, are intricately tied to the research objectives of this thesis. These metrics provide a structured framework for evaluating user behavior, facilitating the assessment of how features impact engagement and usability. By systematically reviewing these metrics, this research establishes a clear pathway to address RQs 2 and 3, which focus on feature prioritization and the challenges developers face in utilizing user data. Moreover, these metrics align with broader trends in software development, such as the shift toward data-driven decision-making and the emphasis on user-centered design. By grounding the research questions in well-established metrics, this thesis ensures that its findings are both theoretically sound and practically relevant. By identifying and analyzing key metrics from the literature, this chapter establishes a robust theoretical foundation for the research. The gaps and challenges discussed emphasize the need for the Usage Analytics method outlined in later chapters. Metrics derived from the super-set not only address the research questions but also advance software development practices by bridging the gap between theory and application.

This chapter sets the stage for transitioning from theoretical insights to practical implementation, detailed in Chapters 5, 6, and 7. By highlighting the interplay between foundational metrics and real-world challenges, it ensures a seamless connection between the literature review and the empirical findings. Ultimately, this integration reinforces the relevance and applicability of the research, contributing to the broader field of usage analytics and software development.

Chapter 3

Research Methodology

In the literature, research is defined as a logical process of steps applied to collect and analyze data in order to improve the knowledge and understanding of a topic or issue respectively to solve a problem perceived (Cito et al., 2015a). Many different approaches to conducting research can be found in literature (Hess et al., 2013). The computer science and software engineering domain have followed action research and design science implicitly for decades (Hevner, 2007). Information systems research to date has produced knowledge by two complementary but distinct paradigms: behavioral sciences and design sciences (Qu et al., 2013). While the behavioral science paradigm usually starts with a defined hypothesis, it aims at exploring the truth, Action Research is a research methodology used by researchers in the Information Systems domain. This methodology is context-bound while attempting to address a specific client's problems, which produces narrow learning in its context and inhibits the reproducibility of the research. Design Science Research Methodology (DSRM) follows a different approach and positions itself as a problem-solving paradigm (Olsson and Bosch, 2015) with the objective of producing an artifact which must be designed and then evaluated thoroughly (Fabijan et al., 2016b; Fitzgerald and Stol, 2017b; Ogonowski et al., 2013). Design science is technology-oriented and its outcomes (the artifacts) have to be assessed against criteria of value and utility (Krusche and Alperowitz, 2014).

The objective of this research is to examine and understand how features of an application are used by the users. This is done by analyzing the application usage data to help improve the application development process. The main component of this research is the design of an artifact in the form of a practical Usage Analysis method applied to improve the continuous deployment process in the software domain. Unlike DSRM, Action Research does not necessarily include a need for the development and evaluation of innovative artifacts Iivari and Venable (2009). Furthermore, the need for a structured approach in the design, development and evaluation of the artifact to solve the outlined problem through research and the aim to develop a generic solution that can be applied to any software application that falls under the software development context eliminates Action Research as a viable research methodology. In line with the identified problem statement, the defined research objective and the developed research questions, Design Science Research Methodology (DSRM) (Peffers et al., 2007; Hevner and Chatterjee, 2010) can be considered as an appropriate research methodology in this particular case. Some examples of research work related to the design and development of software methods and models similar to this thesis using DSRM are Féris et al. (2017); Piccoli et al. (2020); Ballandies et al. (2022).

3.1 Using Design Science Research Methodology

Several variants of design science methodologies exist in the scientific literature, describing distinct approaches to conducting research. A design science process with its five steps, namely awareness of a problem, suggestion, development, evaluation and conclusion, is presented to establish a computable design process model (Poppendieck and Cusumano, 2012). A three-cycle view of design science research comprises the relevance, design, and rigor cycles to manage research projects (Ghezzi et al., 2014). An approach to design research is divided into steps covering three general phases: problem identification, solution design, and evaluation (March and Smith, 1995). A two-dimensional framework is driven by the distinction between research outputs (e.g. representational constructs, models, methods, and instantiations) and research activities (e.g. build, evaluate, theorize, and justify) (Krusche and Alperowitz, 2014). Peffers et al. (2007) provide a set of phases for implementing design science research methodology following a sequential yet iterative process with six steps including (1) problem identification and motivation, (2) definition of the objectives for a solution, (3) design and development, (4) demonstration, (5) evaluation and (6) communication.

This research follows the design-science-oriented approach proposed by Peffers et al. (2007) due to its clear definition of steps to address the problem, motivation, objectives, and the designed evaluation of an artifact, all essential steps to achieve the objectives of this research. Figure 3.1 provides an overview of the DSRM adapted for this research. The following subsections describe the steps of the DSRM as shown in Figure 3.1 in more detail.

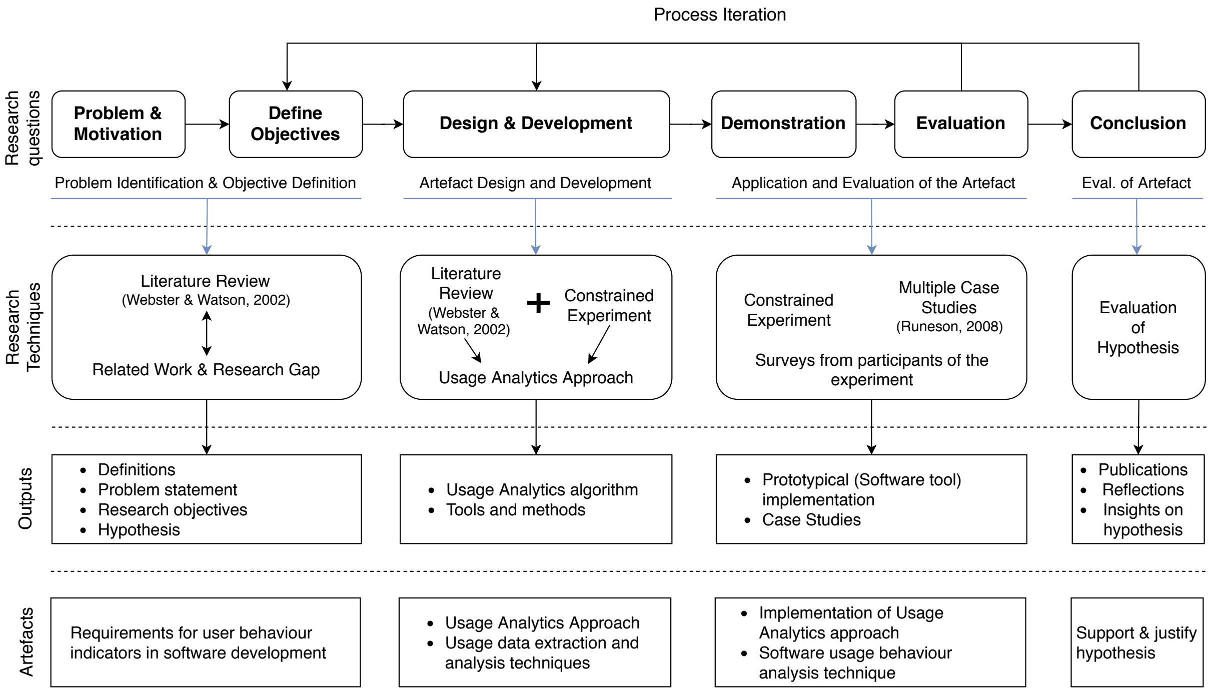

Process Iteration
Research questions
Problem & Motivation
Define Objectives
Design & Development
Demonstration
Evaluation
Conclusion
Problem Identification & Objective Definition
Artefact Design and Development
Application and Evaluation of the Artefact
Eval. of Artefact
Research Techniques
Literature Review (Webster & Watson, 2002)
Related Work & Research Gap
Literature Review (Webster & Watson, 2002)
Constrained Experiment
Usage Analytics Approach
Constrained Experiment
Multiple Case Studies (Runeson, 2008)
Surveys from participants of the experiment
Evaluation of Hypothesis
Outputs
Definitions
Problem statement
Research objectives
Hypothesis
Usage Analytics algorithm
Tools and methods
Prototypical (Software tool) implementation
Case Studies
Publications
Reflections
Insights on hypothesis
Artefacts
Requirements for user behaviour indicators in software development
Usage Analytics Approach
Usage data extraction and analysis techniques
Implementation of Usage Analytics approach
Software usage behaviour analysis technique
Support & justify hypothesis

Figure 3.1: Design Science Research Methodology

3.1.1 Problem Identification

Problem identification is the first step of the research approach as shown in Figure 3.1. At this stage, the research problem is identified and the importance and motivation of the research are justified Peffers et al. (2007). A detailed description of Problem Identification and Motivation is presented in Chapter 2 Section 2.6 of the thesis. The second step of the DSRM is concerned with the definition of expected outcomes and research objectives; these are defined and presented in Chapter 1 Section 1.3 of the thesis. The research gap is identified by carefully analyzing the literature as described in Chapter 2. The resulting artifact from this step is a set of data types and their sources in a software environment which can be used to understand application feature usage discussed in Chapter 6.

3.1.2 Design, Development and Demonstration

The next step in the DSRM is the design and development phase. Based on the defined problem and identified data types, key activities and techniques such as usage data collection, extraction and analysis of the feature usage data are designed and developed in this step. The resulting artifact will be a feature analysis method that includes the key activities as described in Chapter 6 Section 6.1. A prototype demonstrating the developed feature usage analysis method will be built which can extract and analyze the usage data.

The metrics incorporated in this research were derived through a rigorous multi-stage process, beginning with an extensive systematic literature review discussed earlier in Chapter 2. The derived super-set of metrics was refined collaboratively with IBM developers through structured workshops and interviews. These sessions involved a diverse group of participants, including data analysts, senior developers, and software architects, who evaluated each metric for its scalability, relevance to ongoing development workflows, and ease of integration into existing systems. Key considerations during these sessions included how well each metric could be applied across different software platforms, the extent to which it provided actionable insights, and its adaptability to varying project needs.

The iterative refinement process ultimately led to the inclusion of frequency, time spent, and consistency as central metrics. These metrics were chosen for their ability to address core research questions while remaining flexible enough for future adaptation. For instance, frequency helps identify high-use features, time spent provides insights into user engagement, and consistency evaluates behavioral patterns across application updates. This systematic approach ensures that the metrics are both theoretically grounded and practically applicable, providing a robust framework for usage analytics.

3.1.3 Evaluation and Communication

The evaluation consists in observing and measuring how well the artifact supports the solution to the problem (Peffers et al., 2007). This involves comparing and verifying the objectives of the developed solution to the actual results obtained from the use of the proposed feature usage analysis method and prototype demonstration. The key activities of the analysis method will be evaluated through a constrained experiment as described in Chapter 7 and use cases as described in Chapter 5 of this thesis. At the end of the evaluation, it is necessary to decide whether to iterate back to the design phase to try to improve the effectiveness of the artifact or to move to communication.

The evaluation of the Usage Analytics Method involved a comprehensive three-tiered approach:

1. **Qualitative Feedback:** Structured interviews and focus group discussions with developers and project stakeholders provided detailed insights into the usability and perceived value of the artifact. Developers shared their experiences regarding the ease of integrating

the metrics into their workflows and highlighted areas for improvement.

2. **Quantitative Validation:** Metrics such as consistency and time spent were compared against baseline data collected from earlier software versions. Statistical analyses were performed to determine the artifact's impact on user behavior insights, revealing clear trends in feature adoption and usage patterns.

3. **Iterative Refinement:** The results from evaluations conducted in IBM Watson Workspace and Odoo Notes informed iterative improvements to the artifact. Each refinement cycle focused on addressing identified gaps, enhancing the artifact's scalability, and ensuring its relevance to both industrial and academic contexts.

This systematic evaluation framework highlights the robust applicability of the proposed method, ensuring its utility across a wide range of software development scenarios. The use of both qualitative and quantitative measures ensures a balanced assessment, providing comprehensive insights into the artifact's effectiveness.

To enhance the rigor and applicability of the Design Science Research Methodology (DSRM), this research emphasizes iterative refinement and multi-platform validation. By employing structured brainstorming sessions and structured interviews with IBM developers, along with input from academic experts, the methodology ensures alignment between theoretical frameworks and industrial requirements. This collaborative approach not only addresses the nuances of feature prioritization in diverse software environments but also provides a robust pathway to ensure that the designed artifact remains adaptable and scalable across applications. Additionally, the inclusion of iterative testing cycles allows for continuous feedback, ensuring that both functional and non-functional requirements are consistently met.

This research also integrates feedback loops at each phase of DSRM, emphasizing the importance of stakeholder engagement. The feedback collected during each phase informs subsequent iterations, ensuring a dynamic balance between theoretical insights and practical constraints. By engaging multiple stakeholders, including software developers, project managers, and system architects, the methodology highlights the real-world relevance of its solutions.

3.2 Research Design

This section presents the empirical designs for developing and evaluating the Usage Analytics Method (UAM) across multiple case studies. The design encompasses iterative development, participant feedback, and rigorous evaluation to address the research questions. The iterative refinement of UAM ensures its adaptability and relevance in real-world scenarios. The research design includes case studies used to explore and analyze the research problem, design and develop an artifact as a solution to the problem and application and improvement of the artifact and its evaluation. The high-level diagram of the research design is shown in Figure 3.2. Each stage of the research design describes the key activities of the artifact and is followed by an experiment to evaluate the artifact.

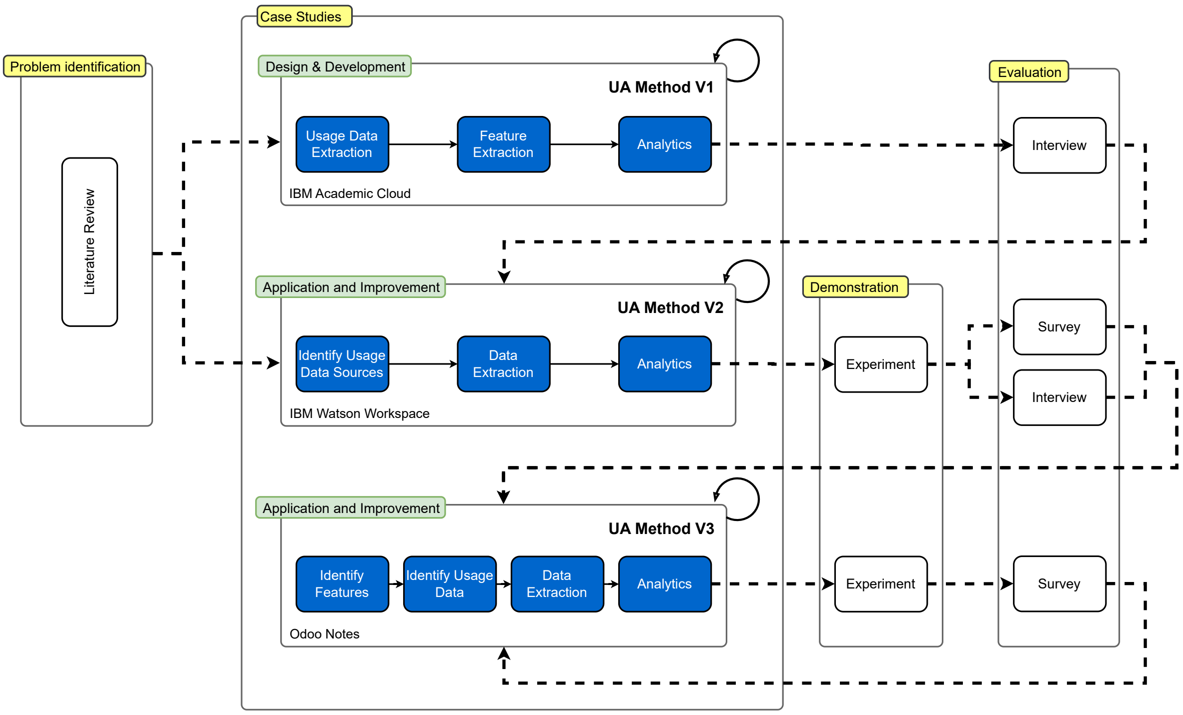


Figure 3.2: Research Design Overview illustrating the iterative development and evaluation of the Usage Analytics Method (UAM) across three case studies: IBM Academic Cloud, IBM Watson Workspace, and Odoo Notes. Each case study follows distinct stages, including design and development, application and improvement, and evaluation.

3.2.1 Platform Selection

The study utilized three distinct platforms: IBM Academic Cloud, IBM Watson Workspace, and Odoo Notes to evaluate the adaptability and generalizability of the Usage Analytics Method (UAM). These platforms were opportunistically chosen based on practical availability and their unique contributions to the study. The IBM platforms provided unparalleled value to this research. By offering direct access to source code, the platforms enabled the development and integration of analytics tools with minimal restrictions. Additionally, access to IBM's development teams provided industry insights that significantly enhanced the customization and refinement of the UAM. The availability of real-world data from these platforms ensured that the study's outcomes were grounded in practical, actionable insights rather than theoretical abstractions. Furthermore, the proprietary industrial environment of IBM allowed the research to simulate high-stakes enterprise scenarios, making the findings directly applicable to real-world software development challenges. Conversely, Odoo Notes was deliberately selected to illustrate that UAM is not confined to IBM systems but is generalizable to any software application. This inclusion highlighted the methodology's flexibility by demonstrating its applicability in an open-source, community-driven environment. The deliberate contrast between industrial and open-source platforms strengthened the methodology's credibility and broadened its applicability across diverse contexts.

Similarities with Leading Platforms The selected platforms share key functionalities with leading industry platforms, ensuring that the research outcomes are directly applicable to real-world scenarios. For instance, IBM Watson Workspace and Odoo Notes offer collaborative tools and user engagement features similar to those of Microsoft Teams or Google Workspace. This alignment ensures that the UAM methodology can be seamlessly integrated into existing software environments, providing actionable insights for feature prioritization and user engagement tracking. The platforms' scalability and flexibility mirror cloud-based infrastructures like AWS or Azure, making the UAM methodology suitable for testing generalizability and adaptability across diverse software environments.

Diverse User Base The user demographics of IBM Watson Workspace and Odoo Notes cater to professional and enterprise users, providing a comparable user base to that of industry-leading platforms. This alignment ensures that the UAM methodology is tailored to meet the needs of professional developers and project managers, enhancing its relevance and applicability in real-world scenarios.

Scalability and Flexibility The scalability of IBM Academic Cloud mirrors the cloud-based infrastructure provided by platforms like AWS or Azure, making it suitable for testing generalizability. The platform's diverse user base and extensive feature set provide a robust testing ground for the UAM methodology, ensuring that the research outcomes are applicable across a wide range of software environments.

The research started with the identification of the research problem exploring the literature and validating the problem in the industry working closely with developers of IBM Academic Cloud. The research problem is identified as the lack of understanding of how features of an application are used by the users. Specifically, it is a time-intensive process to determine the impact of application changes on user behavior. The traditional techniques employed by the developers created a bottleneck in the development process. To address the first research question (RQ1), which focuses on identifying customized metrics and data-points, a rigorous multi-stage process was employed. This process began with an extensive systematic literature review, detailed in Chapter 2 to establish a comprehensive “super-set” of data-points and metrics commonly used in user behavior analysis and feature prioritization. The derived super-set was then refined collaboratively with industrial partners to identify the most relevant and feasible metrics for implementation within their organizational context. Following the derivation of the super-set, the metrics were refined in collaboration with a team of six IBM developers, including senior software engineers, system architects, and data analysts. Through structured brainstorming sessions and workshops, the developers evaluated the super-set for its applicability to IBM’s specific development pipelines. This iterative refinement process emphasized scalability, resource efficiency, and relevance to industrial contexts. The sessions provided valuable insights into how metrics like consistency of usage and error rates could be tailored to meet organizational needs. This collaborative approach ensured that the super-set not only reflected academic rigor but also aligned with practical requirements.

A set of experiments were designed to implement the UA method, gather the usage data, perform analysis on the usage data and gather feedback on the software usage experience of the users. Furthermore, the results of the analytics are compared with the results of the user feedback to evaluate the proposed UA method. Following the DSRM approach, the experiments were iteratively applied to each version of the UA method to identify the improvements and evaluate the method. The experiments were conducted in the IBM Academic Cloud environment and the IBM Watson Workspace application. The experiments were conducted in the Odoo Notes application to validate the final version of the UA method.

Work with IBM Academic Cloud resulted in the initial version of the Usage Analytics Method (V1) described in Chapter 5 Section 5.2.3. The initial version of the method (V1) was applied to the IBM Watson Workspace application, based on the results of the experiment, the method was improved and is described in Chapter 5 Section 5.3.3. Five developers of the IBM Watson Workspace application were recruited to participate in the experiment. The results and feedback obtained were used to improve the UA method further. The improved version of the method (V2) was applied to the Odoo Notes application resulting in the final version of the Usage Analytics

Method (V3) described in Chapter 5 Section 5.4.3. The final version of the Usage Analytics method is presented in Chapter 6 of this thesis. Results of experiments, insights gathered and the evaluation of the method are presented in Chapter 7 of this thesis.

3.2.2 Participants in Evaluation of UAM

The participants involved in this study were carefully selected to ensure their expertise and relevance to the research objectives. Each case study involved participants with distinct roles and technical expertise, enabling a comprehensive evaluation of the Usage Analytics Method (UAM) as shown in Table 3.1.

Table 3.1: Demographics and Roles of the participants in the Evaluation of the Usage Analytics Method (UAM)

Case Study	Participant Role	Expertise Level	Contribution
IBM Academic Cloud	Developer	Senior (10+ years)	Provided insights on data analytics and cloud computing challenges.
IBM Academic Cloud	Data Scientist	Senior (10+ years)	Focused on refining metrics for feature prioritization.
IBM Academic Cloud	Software Architect	Mid-Level (5-10 years)	Assisted in integrating UAM into existing frameworks.
IBM Watson Workspace	Operational Manager	Senior (15+ years)	Focused on collaborative workflows and feature prioritization.
IBM Watson Workspace	Architect	Senior (12+ years)	Provided insights into software architecture and cloud computing.
IBM Watson Workspace	Senior Developer	Senior (10+ years)	Contributed to Python, Java, and data analytics for the project.
IBM Watson Workspace	Developer	Mid-Level (5-7 years)	Focused on JavaScript and full-stack development.
IBM Watson Workspace	Developer	Mid-Level (5-7 years)	Worked on Java, Python, and DevOps integration.
Odoo Notes	End-User	Novice to Expert	Highlighted open-source platform adaptability and unbiased assessments.

Case Study 1: IBM Academic Cloud

The participants for Case Study 1 included three developers with extensive experience in data analytics, software architecture, and cloud computing. Their professional expertise allowed for valuable insights into challenges related to feature prioritization. The developers' demographics, including their tenure and areas of expertise, highlighted their suitability for this study.

Case Study 2: IBM Watson Workspace

For Case Study 2, five participants were recruited, representing diverse roles such as operational managers, architects, and developers. Their experience levels ranged from mid-level to senior, with technical expertise in areas like project management, software architecture, and full-stack development. A semi-structured interview approach was used, focusing on challenges in applying advanced analytics metrics, feature-action mapping, and other critical aspects.

Case Study 3: Odoo Notes

Case Study 3 extended the participant base to include nine users with varying levels of expertise. This diverse participant pool was essential for validating the adaptability of UAM to non-IBM platforms. The inclusion of participants unfamiliar with Odoo Notes eliminated potential biases related to platform familiarity, providing an objective assessment of the method.

Additionally, insights from Chapter 4 on software feature usage data provided essential background for addressing challenges in mapping user interactions to application features. This iterative approach ensured the artifact remained adaptable to evolving project requirements while maintaining a high level of performance and reliability. For further evaluation of these solutions and their effectiveness, see the results and discussion presented in Chapter 7.

The empirical design seamlessly addresses the research questions through a structured, iterative methodology. Specifically, RQ2 ("What are the key challenges faced by developers in identifying and utilizing usage data and key metrics related to feature prioritization effectively within the software platform?") is tackled by uncovering and addressing the complexities developers encounter when integrating analytics into feature prioritization processes. These include difficulties in mapping user interactions to features and challenges in data integration. The iterative refinement of the UAM directly responds to these obstacles, ensuring a pragmatic and systematic approach. Similarly, RQ3 ("How can activities be systematically structured to identify and utilize usage data for feature prioritization?") is addressed by embedding structured activities into the UAM framework. These activities include generating actionable metrics, developing feature-action mappings, and integrating participant feedback, thereby demonstrating how detailed analytics can refine feature prioritization with precision and depth. By combining these elements, the empirical design serves as a robust mechanism to investigate and answer both research questions comprehensively. The empirical design seamlessly addresses the research questions through a structured, iterative methodology. The application of UAM in real-world contexts elucidates the challenges inherent in leveraging usage data and metrics for feature prioritization, directly responding to RQ2. Simultaneously, the structured activities embedded within the method demonstrate its utility in refining feature prioritization through detailed analytics, addressing RQ3 with precision and depth. By combining these elements, the empirical design serves as a robust mechanism to investigate and answer both research questions comprehensively.

3.2.3 Iterative Refinement Process

The iterative refinement process forms the cornerstone of the methodological framework, focusing on the continuous improvement of the UAM through systematic applications and participant collaboration. The methodology integrates practical implementations with reflective evaluations, ensuring a dynamic interplay between theory and practice. Participants engaged in real-world applications of the UAM, generating empirical data, which was then analyzed to identify areas for enhancement. Feedback was gathered exclusively through structured interviews and surveys. This targeted feedback informed adjustments to the method, including refinements to metrics, the data collection framework, and analytical techniques. By concentrating on these structured tools, the feedback process ensured consistency and depth in participant insights.

For instance, in the IBM Watson Workspace case study, feedback highlighted the necessity of incorporating feature-action mapping to accurately track user interactions. This insight prompted the inclusion of explicit steps in the UAM framework for identifying and correlating user actions with specific features. These refinements ensured the method's adaptability to diverse scenarios while maintaining its focus on actionable insights.

The iterative process was highly structured and involved distinct phases. First, participants applied the UAM to practical scenarios, generating a wealth of usage data. This data was analyzed not only for its immediate insights but also for its broader implications regarding the method's applicability and scalability. Feedback collection followed, encompassing structured interviews that explored the method's clarity, relevance, and effectiveness. Finally, these insights were meticulously analyzed, leading to iterative improvements that strengthened the UAM framework's robustness and usability.

3.2.4 Evaluation Methods

The research employed an integrated approach combining qualitative and quantitative methods to ensure a robust dataset and comprehensive evaluation of UAM. This approach was essential for addressing the study's research objectives and ensuring the validity of the findings.

Structured interviews formed a cornerstone of the data-gathering process. Conducted with developers, analysts, and end-users, these interviews provided deep insights into the challenges and opportunities related to feature prioritization. For instance, in IBM Watson Workspace, interviews highlighted the importance of capturing team dynamics, including patterns of collaboration and decision-making. Similarly, IBM Watson Workspace interviews underscored the need for metrics like the behavioral score to assess user engagement across different versions.

Case study observations complemented these methods by capturing real-world application scenarios. For IBM Watson Workspace, observations focused on collaborative workflows and their impact on feature prioritization. In the Odoo Notes case, observations provided critical validation of UAM's adaptability to an open source application settings. This multi-method approach ensured that the data collected was both comprehensive and contextually relevant to the study's aims.

Interaction data logs were systematically collected across platforms to quantify user engagement through metrics such as frequency, time spent, and consistency. These logs were especially crucial for platforms like IBM Academic Cloud, where detailed logs enabled granular analysis of professional use cases. In contrast, Odoo Notes relied on indirect instrumentation and community feedback to overcome data access challenges, reflecting the platform's open-source nature.

Cross-platform comparisons were conducted to assess the robustness and adaptability of the metrics and methods across varying platforms. For example, in the Odoo Notes case study, the inclusion of the behavioral score provided a more nuanced analysis of user engagement and behavioral changes. This composite metric was validated against the patterns observed in IBM Watson Workspace, where consistency and frequency metrics highlighted team dynamics. Developer feedback played a critical role in refining the metrics, ensuring their alignment with practical needs and real-world application scenarios, such as integrating frequency and time-spent metrics into IBM Academic Cloud's existing analytical systems. These iterative refinements underscored the adaptability of the UAM methodology and strengthened its generalizability.

The analysis incorporated four pivotal metrics, each designed to capture different dimensions of user behavior and engagement. *Frequency,* a metric focusing on the number of interactions, served as a key indicator of engagement levels across features and platforms. For instance, in IBM Academic Cloud, frequency highlighted high-priority features based on user interaction density. *Time spent* provided a complementary perspective by measuring the duration of user engagement with specific features, shedding light on their practical utility and user appeal. This metric was particularly effective in identifying features that required prolonged interaction, as observed in IBM Watson Workspace's collaborative tools. *Consistency* analyzed patterns of user behavior across multiple sessions and user groups, revealing trends and stability in feature utilization. In the case of IBM Watson Workspace, consistency metrics uncovered valuable insights into team collaboration habits and feature adoption rates. *Behavioral Score,* introduced in the Odoo Notes case study, emerged as a composite metric integrating frequency, time spent, and consistency. This metric was instrumental in providing a holistic understanding of user behavior, particularly when analyzing changes across different versions of the software. By combining individual metrics, the behavioral score facilitated a nuanced evaluation of user engagement, supporting a cross-platform comparison that underscored the robustness and adaptability of the UAM.

Participant feedback emerged as a pivotal component of the iterative refinement process. In the IBM Academic Cloud case study, developers underscored the difficulties of mapping user interactions to specific features, leading to the inclusion of comprehensive mapping guidelines within the UAM framework. However, it was evident that the primary analytics metrics of interest to the IBM developers and other stakeholders were frequency, time spent, and consistency. These three metrics were prioritized due to their direct relevance and applicability to feature prioritization and user engagement analysis. Other metrics, such as session duration and clickstream data, remained part of the theoretical super-set derived from the literature. The super-set, obtained through an extensive exploration of academic studies, served as a comprehensive foundation, but its elements were selectively applied based on participant feedback and practical considerations.

The empirical studies conducted as part of this research were pivotal in validating the artifact's utility across diverse contexts. These studies included:

1. **Case Study 1:** Initial implementation in IBM Academic Cloud to identify baseline challenges and metrics. This phase highlighted the importance of clear feature definitions and robust data collection methods, as discussed in Chapter 4. The findings revealed gaps in user interaction mapping and inspired targeted refinements to improve metric clarity and scalability.

2. **Case Study 2:** Advanced testing in IBM Watson Workspace to refine metrics and pro-

cesses. This phase incorporated both developer feedback and usage analytics to adjust the artifact to real-world workflows. Specific challenges addressed included data synchronization across systems and implementing dynamic feedback loops. These insights align closely with the discussion of challenges in Chapter 5, demonstrating iterative progress.

3. **Case Study 3:** Final validation in Odoo Notes, demonstrating the method's adaptability to non-IBM platforms. This study provided critical insights into the artifact's generalizability and scalability across different software environments. The results, detailed further in Chapter 7, underscore the artifact's robustness in varying operational conditions.

The design of these studies ensured comprehensive evaluation through qualitative interviews and quantitative metrics analysis. The cross-platform application demonstrates the artifact's ability to integrate into diverse industrial and academic contexts. By systematically addressing challenges and leveraging iterative improvements, these empirical evaluations confirm the artifact's effectiveness in meeting the research objectives while offering a scalable framework for future use.

These studies provide a comprehensive evaluation of the artifact, illustrating its robustness and generalizability. Building on the empirical findings, the subsequent chapter delves into the nuanced understanding of software feature usage data, laying the groundwork for aligning metrics with actionable insights. Chapter 4 specifically explores how data-driven approaches to feature prioritization can bridge the gap between user behavior analysis and practical software development, ensuring a seamless progression from theoretical methodologies to their real-world applications.

Chapter 4

Features and Usage data in Software Development

In order to obtain quick, actionable and meaningful insights based on the software usage, the analytics system should use real-time data (Menzies and Zimmermann, 2013). This chapter offers an exhaustive and expansive analysis of the data types and metrics foundational to the Usage Analytics Method. These data types, systematically categorized into a theoretical super-set derived from extensive literature, form the backbone of the classification discussed in this chapter. This super-set was meticulously constructed through a rigorous review of academic studies and further refined through iterative applications and structured participant feedback. By synthesizing theoretical insights with practical requirements, this chapter underscores the UAM's adaptability and indispensable role in addressing diverse software development challenges. This process not only highlights the UAM's scalability but also directly answers Research Question 1, which seeks to identify the most relevant metrics and data points for feature prioritization decisions. By integrating frequency, time spent, and consistency as prioritized metrics based on participant feedback, the UAM bridges theoretical understanding with industrially viable applications. The deliberate integration of theoretical depth and industrial relevance ensures that the method is robust and scalable across a wide range of contexts, making it a critical tool for contemporary analytics.

RQ1: (*Problem scoping and objective definition*) What are the customized key metrics and data points, tailored to industrial requirements, from user interactions that most significantly influence feature prioritization decisions?

The software maintenance phase is an important stage in the software development process. Software developers use the maintenance phase to monitor the deployed software. One of the many benefits of monitoring software deployment is to understand how end-users use the different features particularly to know which features are useful to them. Section 4.1 provides an in-depth analysis of existing definitions of the term *feature* to construct a generic definition with an intention to differentiate different features of a software application and the corresponding users of these features. For example, some features of the application are designed specifically for only developers to use such as “debug mode”, “maintenance mode” or “service mode”; some features are only designed to be accessible by the administrators of the software such as “account management”, “network management” and so on. On the other hand, the same term feature is used by the analysts of the software to refer to the data metrics used for the analysis of software usage statistics. Hence, to provide clarity and narrow the focus of this research, a general definition for the term feature in the software domain is defined which provides the ability to identify the set of features and the associated role of actors who can access these features. While the following section of the thesis may not be the primary focus of the research, it addresses a critical issue: the ambiguity in defining what constitutes a feature in a software application. This ambiguity, influenced by the varying roles individuals play throughout the software development lifecycle, is a foundational challenge that significantly impacts the effectiveness of usage analytics. By clarifying this aspect, this part of the work not only complements the research but also enhances its applicability and relevance in real-world scenarios. Section 4.1.1 uses the derived definition of the term *feature* to highlight the end-user-level features of a software application to align with the focus of this research. Section 4.2 describes the role of usage data in software development and maintenance, typical types and sources of usage data and so on.

4.1 The Concept of Features in Software Development

The term “feature” is abstract and has different understandings depending on the domain, user role and its application in the domain. Even in a specific domain such as traditional software development, the term feature may refer to non-functional attributes of the application to the software architect/designer, but to the end-user, feature refers to application-specific functionality. Furthermore, cloud-based software development introduces additional elements that could be considered as features. Each layer includes different types of features and people responsible for the development and maintenance of those features. The IaaS layer of the cloud includes features such as data storage, data transfer, CPU allocation, network interfaces, network configuration, data redundancy etc. maintained by the cloud resource providers. The PaaS layer of the cloud includes features such as programming languages, database management systems, libraries, and development tools which are developed and maintained by cloud service providers. The SaaS layer of the cloud includes features such as scalability, interoperability, redundancy, configurability etc developed and maintained by software developers. Although the term *feature* is used to refer to all these layer-specific characteristics, existing definitions of the term feature cannot be applied to define them.

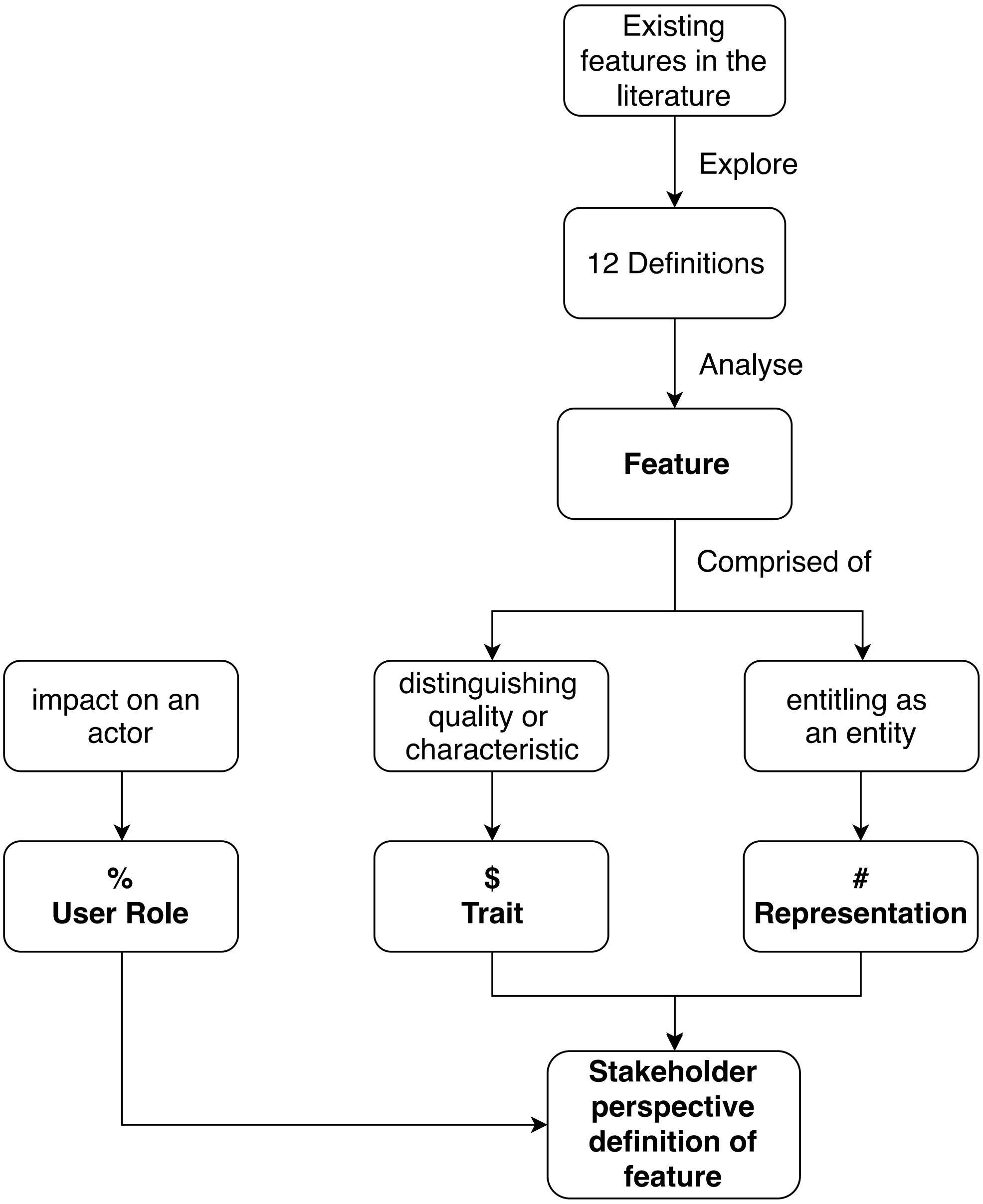

Existing features in the literature
Explore
12 Definitions
Analyse
Feature
Comprised of
impact on an actor
distinguishing quality or characteristic
entitling as an entity
%
User Role
$
Trait
#
Representation
Stakeholder perspective definition of feature

Figure 4.1: Analysis procedure of feature definition

An exhaustive literature survey is conducted to find papers that provide definitions of a feature as shown in Figure 4.1. The search terms used are “feature” and “definition”. The databases considered for the search are Google Scholar and Scopus. The resulting papers were filtered using inclusion and exclusion criteria, where the inclusion criteria included the Computer Science and Information Systems domain and other domains were excluded. Then the papers are read thoroughly, and only those papers are considered for the analysis that provided a definition of a feature. Definitions for the term feature were provided as early as the year 1990 and found novel definitions in different software engineering and information systems domains until the year 2010. As a result, the analysis includes feature definitions between 1990 and 2010. Classen et al. (2008) made a detailed analysis of different definitions of a feature in the contexts of requirements engineering, software product lines and feature-oriented software development. This analysis describes a feature using three descriptions but does not consider the role of stakeholders in the environment. Kang et al. (1990) provided the first definition for the term feature in the context of software product lines as “a prominent or distinctive user-visible aspect, quality or characteristic of a software system or systems”. The definition they proposed can be regarded as insufficiently precise, as it fails to effectively identify features in a complex and diverse cloud computing environment, where multiple stakeholders and application domains coexist. While other definitions exist in the literature (Batory et al., 2004; Bosch, 2000; Riebisch, 2003), they tend to be either domain-specific, overlook the role of stakeholders, or lack the clarity needed for broader applicability.

We analyzed definitions of a feature in various domains as shown in Table 4.1. Based on detailed observation of the definitions of a feature, it is evident that a feature is comprised of basic elements such as a trait, representation and possibly an actor role. The result of the analysis reveals a feature should have a distinguishing quality or characteristic known as a trait, an entity known as a representation and an impact on a specific Actor role or roles.

Analyzing the definition of features in the software development domain reveals that the term “features” may relate to a different aspect of the software for different actor roles. Considering the focus of this research, the term *feature* always refers to the functionality of the software and the *actor* role always refers to an end-user of the software. The following section identifies the user-level features of IBM Watson Workspace and Odoo Notes applications using these feature definition assumptions.

Following the above analysis, we establish a comprehensive definition for a feature broadly in the Software domain as follows:

*"**A feature represents a** functionality / requirement / aspect **that possess a** distinct / distinguishable / incremental / functional / non-functional **quality attribute visible-to / satisfy a** user / customer / stakeholder"*

To generalize the definition of the feature based on the above analysis:

"***A feature represents a*** *trait* (identified by the symbol '$') ***that possess a*** *representation* (identified by the symbol '#') ***of a quality attribute visible-to / satisfy an*** *actor role* (identified by the symbol '%')"

Table 4.1: Analysis of Existing Feature Definitions in the Literature

Paper	**Definition**	**$Trait**	**#Representation**	**%Actor Role**
(Kang et al., 1990)	a prominent$ or distinctive$ user-visible% aspect#, quality#, or characteristic# of a system	Prominent, Distinctive	Aspect, Quality, Characteristic	User
(Kang et al., 1998)	a distinctively$ identifiable$ functional$ abstraction# that must be implemented$, tested$, delivered$ and maintained$	Distinctive, Identifiable, Functional	Abstraction	-
(Bosch, 2000)	a logical$ unit of behaviour# specified by a set of functional$ and non-functional$ requirements#	Logical, Functional, Non-functional	Behaviour, Requirements	-
(Czarnecki et al., 2002)	a distinguishable$ characteristic of a concept# (e.g., system, component, and so on) that is relevant to some stakeholder% of the concept	Distinguishable	Concept	Stakeholder
(Riebisch, 2003)	an aspect# valuable$ to the customer%	Valuable	Aspect	Customer
(Zave, 2003)	an optional$ or incremental$ unit of functionality#	Optional, Incremental	Functionality	-

Paper	**Definition**	**$Trait**	**#Representation**	**%Actor Role**
(Batory et al., 2004)	a product characteristic# that is used in distinguishing$ programs within a family of related programs	Distinguishing	characteristic	-
(Batory, 2006)	an elaboration$ or augmentation$ of an entity(s) that introduces a new service# , capability# or relationship#	Elaboration, Augmentation	Service, Capability, Relationship	-
(Batory et al., 2006)	an increment$ in product functionality#	Increment	Functionality	-
(Apel et al., 2008)	a structure# that extends$ and modifies$ the structure of a given program in order to satisfy a stakeholder's% requirement#, to implement$ and encapsulate$ a design decision#, and to offer$ a configuration-option#	Extends, Modifies, Implement, Encapsulate, Offer	Structure, Requirement, Design Decision, Configuration-option	Stakeholder
(Classen et al., 2008)	a triplet, f = (R, W, S) where R represents the requirements# the feature satisfies$, W the assumptions# the feature takes about its environment and S its specification#	Satisfies	Requirements, Assumptions, Specifications	-

Paper	Definition	$Trait	#Representation	%Actor Role
(Revelle et al., 2010)	a code# that implements$ a functionality# sometimes also referred to as a concept# or a concern#	Implements	Code, Functionality, Concept, Concern	-

4.1.1 User-level Features

This research focuses on the user-level features of the software application. The actor role in the definition of the feature is the end-user of the application. Typically, the description of new features (and updated changes to the existing features) of an application is provided by the developers of the application in the form of a human-readable document called a version log or change log, specifically documented for both prospective and current customers (users) of the application. This document is usually available for the users on the features section of the application's or application developers' official website. The version log document is updated when a new version of the application is rolled out (deployed) to the customers, describing the new features or specific changes made to existing features of the application. Sections 4.5.1 and 4.5.2 describe the user-level features of the IBM Watson Workspace and Odoo Notes applications respectively. These features are designed to meet the user needs and improve their overall experience with the application. Features can range from basic functionalities, such as user login and file upload, to more complex operations, like data analytics and real-time collaboration tools.

In the context of software development, features are often documented as part of the requirements collection process. They are typically prioritized based on various factors, including stakeholder input, cost-value analysis, and usage data. Understanding which features are most critical to users and how they interact with these features is crucial for making informed development decisions. Recent studies have highlighted the importance of integrating user feedback and interaction data into the feature prioritization process. For instance, Chen and Liu (2017) demonstrated that user-centric approaches to feature prioritization could significantly enhance the relevance and usability of software applications. Similarly, Johnson and Smith (2019) emphasized the role of continuous user feedback in refining and prioritizing features throughout the software development lifecycle. Moreover, Bosch and Olsson (2014) discussed how the integration of user feedback into agile development processes could improve the responsiveness and adaptability of software development teams, ultimately leading to more user-centric products.

The ability to adapt features based on user interaction data ensures that the software remains relevant and useful to its user base. This approach is supported by Gurp et al. (2009), who argued that the features should evolve based on real-world usage to remain competitive and valuable. By continually refining features based on how they are used, developers can ensure that software remains aligned with user needs and expectations.

4.2 Usage Data

Usage data refers to the information collected from users when they interact with a software application. This data can be collected through various means, such as log files, user feedback, and telemetry data. The types of usage data that can be analyzed include:

1. **Interaction Data:** Information about how users interact with the software, such as clicks, navigation paths, and session duration.
2. **Performance Data:** Metrics related to the performance of the software, including response times, error rates, and system uptime.
3. **Engagement Data:** Measures of user engagement, such as the frequency of use, number of active users, and user retention rates.
4. **Feedback Data:** User-generated feedback, including comments, ratings, and support tickets.

Each type of usage data provides unique insight into different aspects of user interactions and software performance. For example, interaction data can reveal how users navigate through the application and which features they use most frequently. Performance data can help identify bottlenecks and areas for improvement, while engagement data can indicate the overall popularity and effectiveness of the software.

The superset of data types presented in this section was derived through a structured academic research process, ensuring rigor and traceability. The methodology began with an extensive systematic literature review, as described in Chapter 2, following the guidelines of Webster and Watson (2002). The review utilized databases such as IEEE Xplore, ACM Digital Library, ScienceDirect, Springer, and Google Scholar. Search terms included several combinations including "usage data", "software monitoring metrics", "user behavior analytics" and "software performance metrics" resulting in an initial pool of 240 papers. This pool was refined based on inclusion and exclusion criteria that prioritized relevance to feature prioritization, user behavior analysis, and actionable insights for software systems, specifically their relevance to the aim of this research.

From this pool, a subset of 32 key studies was selected for detailed analysis, as outlined in Chapter 2. These studies were chosen for their methodological rigor and alignment with research questions focused on user behavior analysis, feature prioritization, and the development of actionable insights for software systems. The categorization of data types was informed by a systematic extraction of themes, focusing on their utility across varied software environments. Furthermore, iterative validation was conducted through expert feedback from industry professionals and structured discussions with developers engaged in the empirical studies described in this thesis. This feedback process was rigorously aligned with the methodology outlined in Chapter 2, ensuring consistency and traceability. Each data type was critically evaluated for its theoretical robustness, practical applicability, and potential to enhance decision-making in software analytics.

This methodological rigor ensured that the data types presented in the superset reflect both theoretical advancements and practical considerations, bridging the gap between academic research and industry needs. By following this structured approach, the superset presented in Table 4.2 aims to align with the overarching goals of this research to provide a robust foundation for feature prioritization and usage analytics in software systems. The following table summarizes the superset of data types derived from the literature, providing a foundation for the subsequent detailed discussions. Each data type is analyzed for its theoretical contributions and practical applications, bridging the gap between research insights and industrial needs.

User Identity data includes unique identifiers such as user IDs, IP addresses, and device details. Zimmermann and Nagappan (2010) highlighted how tracking specific user patterns allows for tailoring software features to individual needs, enhancing user satisfaction and engagement. Furthermore, Shams and Hashemi (2018) demonstrated the security advantages of analyzing identity data, such as detecting unauthorized access and anomalies in real-time. However, while critical for personalization and segmentation, relying solely on identity data may neglect broader interaction trends, necessitating its integration with complementary data types for a holistic analysis.

Application Host Details provide key insights into the underlying infrastructure, including server, database, and container information, which directly impact system performance. Menzies and Williams (2011) and Wang et al. (2017) emphasized how host-level data aids in diagnosing performance issues, identifying bottlenecks, and optimizing resource allocation. For instance, tracking database query times helps pinpoint latency issues, ensuring smoother user experiences. However, this type of data, while invaluable for backend diagnostics, provides limited insights into user behavior without contextual enrichment from other data sources.

Interaction Logs capture granular details of user actions, such as clicks, navigation paths,

Table 4.2: Superset of Data Types for Usage Analytics

ID	**Data Type**	**Description**	**Examples of Related Data**
S1	User Identity	Captures unique identifiers such as user IDs, IP addresses, and device details to distinguish between users.	User ID, IP Address, Device ID, Session Tokens
S2	Application Host Details	Tracks server, database, and container information, providing insights into system-level operations and hosting environments.	Server Name, Database Instance, Container ID, Host IP Address
S3	Interaction Logs	Records actions performed by users, such as clicks, button presses, navigation paths, and API calls.	Clicks, Button Presses, API Calls, Navigation Paths
S4	Temporal Data	Captures timestamps and session durations associated with user interactions, offering a timeline of user activity.	Timestamps, Session Start/End Times, Duration Metrics
S5	Session Metrics	Provides aggregated data on session start and end times, user retention, and overall engagement trends.	Session Duration, Retention Rates, Active Session Count
S6	Error Logs	Tracks occurrences, types, and frequencies of errors encountered during user interactions.	Error Codes, Error Frequency, Exception Messages
S7	Performance Metrics	Includes system response times, uptime, latency, and related performance indicators.	Response Times, Latency, Uptime Percentage, Throughput
S8	Clickstream Data	Tracks sequences of user interactions, mapping navigation paths within the application.	Click Paths, Navigation Sequences, Time on Page
S9	Engagement Metrics	Measures user retention, frequency of use, and time spent on features.	Frequency of Use, Time Spent, User Retention Rates
S10	Operational Details	Monitors background tasks, resource utilization, and system events occurring during user interactions.	Background Task Logs, CPU Utilization, Record Load Counts
S11	Behavioral Patterns	Includes repeated actions, focus metrics, and consistency in user behavior over time.	Repeated Actions, Focus Durations, Behavioral Consistency

and API calls. These logs are instrumental in understanding how users engage with software features. Pagano and Bruegge (2013) illustrated their role in identifying high-usage features, which informs development priorities. Additionally, Buse and Zimmermann (2012c) highlighted the utility of real-time interaction tracking to uncover friction points in user workflows. Nevertheless, the high volume of data generated from interaction logs demands robust analytical frameworks for effective interpretation.

Temporal Data focuses on timestamps and session durations, offering a chronological perspective of user interactions. Lo and Nagappan (2015) and Kim and Park (2020) utilized temporal data to uncover seasonal trends and usage peaks, enabling strategic resource allocation and feature deployment. However, temporal data often lacks the depth needed to analyze user behavior independently and serves best as a supplementary data type when combined with more specific interaction metrics.

Session Metrics including session durations and retention rates, provide macro-level insights into user engagement. Dyckhoff et al. (2012) and Graf et al. (2011) used session data to assess user motivation and overall satisfaction. While these metrics offer valuable overviews, they require augmentation with interaction-level data to evaluate feature-specific impacts and understand the nuances of user engagement more precisely.

Error Logs document the occurrences and types of errors encountered by users, serving as diagnostic tools for improving reliability. De Chaves et al. (2011) and Ghezzi et al. (2014) emphasized the critical role of error data in identifying usability challenges and mitigating system vulnerabilities. While primarily reactive, error logs become powerful when combined with performance metrics to facilitate proactive system improvements and maintenance strategies.

Performance Metrics such as response times, latency, and uptime, are essential for backend optimization and ensuring system efficiency. Nguyen and Wu (2018) and Crowston and Kammerer (2003) demonstrated their utility in pinpointing bottlenecks that impact user experience, such as slow-loading pages or server downtime. However, these metrics need to be paired with user-facing engagement data to bridge the gap between system performance and user satisfaction effectively.

Clickstream Data provides a detailed map of user navigation, capturing the sequence of interactions within an application. Atterer et al. (2006) and Banerjee and Ghosh (2001) showcased its role in workflow optimization by identifying navigation inefficiencies and improving user journeys. While offering granular insights, implementing and processing clickstream data requires significant computational resources and advanced analytics.

Engagement Metrics such as retention rates, frequency of use, and time spent on features, are pivotal for understanding user loyalty and feature success. Shams et al. (2020) and Lo and Zimmermann (2013) demonstrated their role in predicting user churn and identifying high-value features for prioritization. These metrics, while insightful, must be contextualized to differentiate between meaningful engagement and superficial activity effectively.

Operational Details encompass background tasks, resource utilization, and system events, providing a comprehensive view of application performance. Lo and Zimmermann (2013) and Arora and Malik (2017) identified these metrics as critical for proactive maintenance and capacity planning. When combined with user interaction data, operational metrics enhance understanding of system behavior and its impact on user experience.

Behavioral Patterns including repeated actions and consistency metrics, reveal long-term trends in user behavior. Fabijan et al. (2016a) and Buse and Zimmermann (2012c) highlighted their importance for tracking the lifecycle of features and monitoring user retention over time. While offering rich longitudinal insights, these patterns require substantial data collection and advanced analysis techniques to generate actionable outcomes.

System and application logs contain a wealth of information to help manage systems. Most systems print out logs during their executions to record system runtime actions and state that can directly reflect system runtime behaviors. System developers and architects usually use these logs to track a system to detect and diagnose system anomalies.

The second type of this data is the user-level usage data generated as a result of user interaction with a cloud-based application. Some examples of usage data are application logs, for example, the assessment data (wiki, forum, message), the activity data (clicks, time spent), server logs, and so on. They can be extracted by the applications themselves, third-party monitoring tools or via Web cookies (from a web browser). Such data in the cloud is spread across various interfaces such as Web browsers, mobile applications and command line interfaces on the front end and servers, virtual machines, containers and databases on the back end. The last type of usage data is the VM logs, typically generated from the VMs running the applications or services. This type of log contains the usage of the CPU, memories, as well as running tasks, time of starting and stopping and others.

Table 4.3 presents the classification of usage data in the software environment. This classification was specifically designed to address the fragmented nature of usage data in cloud-based

Table 4.3: Usage Data Classification: Types of the usage data available in the software environment, their characteristics the data representing the context of the data in the analytics process, description of the data characteristic and the specific type of data.

ID	Characteristic	Description	Data Type
C1	User Identity	Who is using the application	a) User ID b) IP Address c) User Name d) Device ID
C2	Application host	Where the application hosted	a) Web server b) Database c) Instance ID d) container ID e) IP Address
C3	Actions performed	What the end user does	a) Page b) Method c) Function d) Button clicked e) View f) Focus g) API call h) swipe
C4	Action moment	When the user performs the action	a) Date and time b) Session ID
C5	Action Duration	How long it takes to complete the action	a) Duration b) Query duration
C6	Operational details	Complementary information to the actions performed	a) Errors b) Background tasks c) Number of records loaded

applications, considering both front-end and back-end data sources. The framework categorizes usage data into the following seven dimensions:

- **Who is using the application:** Captures user-related details, including user ID and IP address.
- **Where the application is being hosted:** Focuses on server and database locations to track data and operations.
- **What the end user does:** Includes actions such as accessing specific application pages, invoking methods, or utilizing features (e.g., buttons and functions).
- **When the user performs the operation:** Records the timing of user interactions through timestamps and session identifiers.
- **How long it takes to complete an operation:** Measures the duration of user actions and system responses, such as query execution times.
- **Other operational details:** Accounts for errors, background tasks, and the number of records loaded during user interactions.
- **User behavior:** Introduces metrics such as clickstreams, focus patterns, and viewing activities to analyze user engagement.

This updated classification integrates additional insights to account for advancements in multi-interface environments, reflecting the growing complexity of user interactions with modern software systems. By systematically categorizing usage data, this framework provides a structured approach to analyzing user behavior, enabling a deeper understanding of feature utilization and system performance.

The selection of this specific data classification is well supported by the literature and serves as a comprehensive basis for analyzing user interactions. The classification is adapted from Kesavulu et al. (2017b), which provides a well-structured and detailed categorization of usage data that aligns with the requirements of usage analytics. The primary aim is to capture a broad range of interaction details that can provide actionable insights into user behavior.

4.2.1 Justification of the Selected Usage Data

The classification of usage data into specific categories is critical for effectively capturing and analyzing user interactions within software applications. Each category serves a unique purpose and provides valuable insights into different aspects of user behavior and system performance. This section provides an in-depth justification for the selected categories of usage data, supported by extensive references from recent academic literature.

1. *User Identity (C1)* Capturing user identity data such as User ID, IP Address, and Device ID is fundamental for distinguishing between different users and understanding individual user behaviors. According to Zimmermann and Nagappan (2010), identifying individual user patterns helps tailor features to meet user-specific needs and improve user satisfaction. User identity data enables developers to track usage patterns and identify unique behaviors across different user segments. This capability is essential for personalizing user experiences and optimizing features for specific user groups. User identity data also enhances security measures by monitoring access patterns and detecting anomalies. For instance, Shams and Hashemi (2018) demonstrated that analyzing user identity data can help identify unauthorized access and potential security threats, ensuring the integrity and safety of the application.

2. *Application Host (C2)* Understanding where the application is hosted, such as on a web server, database, or virtual machine, is crucial for diagnosing performance issues and optimizing resource allocation. The application host data provides insights into the infrastructure supporting the application, which is essential for maintaining optimal performance. Menzies and Williams (2011) emphasized that knowing the hosting environment helps developers optimize system performance and ensure scalability. Additionally, Wang et al. (2017) highlighted that application host data is critical for performance tuning and capacity planning. By analyzing host data, developers can identify bottlenecks and implement strategies to enhance system efficiency. Understanding the hosting environment also ensures data integrity and security, as different hosting configurations may have unique vulnerabilities and performance characteristics.

3. *Actions Performed (C3)* Recording the actions performed by users, such as button clicks, API calls, and page views, provides direct insights into how users interact with the application. This data type is fundamental for understanding feature usage and is widely

recognized in the literature for its importance in user behavior analysis. Pagano and Bruegge (2013) emphasized that Actions Performed data allows developers to track user interactions in real-time, providing a detailed view of how features are utilized. This information is invaluable for identifying popular features and those that may require improvement. Furthermore, analyzing user actions helps understand user workflows and identify potential friction points within the application. By optimizing these workflows, developers can enhance the overall user experience and improve feature adoption rates. Buse and Zimmermann (2012b) also highlighted that detailed tracking of user actions can help in identifying usage patterns that inform feature enhancement decisions.

4. *Action Moment (C4)* Tracking the exact time when actions are performed allows developers to analyze usage patterns and trends over time. Temporal data is essential for identifying peak usage times and understanding how usage varies across different periods. Lo and Nagappan (2015) stated that action moment data provides a temporal context to user interactions, enabling developers to analyze trends and patterns in user behavior. Temporal data helps developers understand when users are most active and how their interactions evolve over time. By leveraging this data, developers can schedule maintenance and updates during periods of low activity to minimize disruption. Additionally, Kim and Park (2020) highlighted that temporal data can help identify seasonal trends and inform marketing strategies by aligning feature releases with peak usage periods.

5. *Action Duration (C5)* Measuring how long it takes to complete actions provides insights into user engagement and the efficiency of different features. Duration metrics are particularly useful for identifying features that may require optimization to enhance user experience. Crowston and Kammerer (2003) discussed that action duration data captures the time users spend on specific tasks, providing insights into feature engagement and usability. By analyzing action duration data, developers can identify features that are time-consuming or cumbersome for users and prioritize them for optimization. Duration metrics also help assess the effectiveness of new features or updates by comparing engagement times before and after implementation. This data is crucial for ensuring that features are intuitive and efficient, contributing to a positive user experience. Nguyen and Wu (2018) emphasized that understanding the time users spend on specific features can guide developers in improving the user interface and functionality of the software.

6. *Operational Details (C6)* Collecting data on errors, background tasks, and the number of records loaded helps diagnose operational issues and ensure the application runs smoothly. These operational metrics are critical for maintaining high performance and reliability in software applications. Lo and Zimmermann (2013) highlighted that operational details data provides a comprehensive view of the application's performance and reliability. Monitoring errors and background tasks allows developers to quickly identify and resolve issues that may impact user experience. This data is essential for maintaining system stability and ensuring that the application performs optimally under various conditions. Operational metrics can also inform capacity planning and resource allocation by highlighting areas that require additional support or optimization. Arora and Malik (2017) discussed that operational details are vital for proactive maintenance and ensuring a seamless user experience.

The classification of usage data into these categories is supported by several studies in the field of software engineering and usage analytics. For example, Nguyen and Wu (2018) emphasizes the importance of detailed action logs in understanding user behavior and improving feature prioritization. Additionally, Arora and Malik (2017) discusses how consistency in user interactions can be used as a metric to enhance software usability, further justifying the inclusion of detailed action and duration metrics. The comprehensive nature of this classification ensures that a wide range of user interactions is captured, providing a holistic view of how users engage with the software. This approach aligns with the recommendations of Aldhaheri and Abdullah (2020), who advocate for a multi-metric approach to feature prioritization, combining various types of usage data to capture a complete picture of user behavior. By adopting this classification, the research leverages a structured and validated framework that has been shown to be effective in previous studies. This ensures that the analysis is both thorough and aligned with established best practices in the field, making it a robust basis for the usage analytics method developed in this thesis.

4.3 The Relationship between Software Features and User Actions

The relationship between software features and user actions is crucial for understanding how users derive value from a software application. Features define what users can do, while user actions reflect how they utilize these capabilities. Analyzing this relationship helps identify which features are most valuable to users and where there may be gaps in functionality or usability. For instance, if a feature intended to streamline a particular task is underutilized, it may indicate usability issues or a lack of awareness among users. Conversely, features that see high levels of engagement can be prioritized for further development and enhancement.

Recent studies have underscored the value of this relationship in driving feature prioritization and software improvement. Smith and Brown (2019) discussed how mapping user actions to features could reveal critical insights into user needs and preferences, enabling more targeted development efforts. Similarly, Lee and Taylor (2021) highlighted that understanding this relationship is essential for creating intuitive and user-friendly software experiences. Furthermore, Zimmermann and Nagappan (2010) pointed out that aligning software features with user actions could improve the overall efficiency and effectiveness of the software development process by ensuring that development efforts are focused on the most impacted areas. For example, in a study by Tang and Zhang (2011), the authors analyzed user interactions with a content management system (CMS) to understand which features were most frequently used. They found that certain features, such as the WYSIWYG editor and media management tools, were heavily utilized, while others, like advanced search functionalities, were rarely used. This analysis led to a decision to prioritize improvements to the editor and media tools, while de-emphasizing less critical features. Another example can be seen in the work of Murphy and Weiss (2013), who investigated user interactions with a project management tool. By analyzing usage data, the researchers identified that users frequently accessed task management and time tracking features, but rarely used the built-in chat functionality. This insight prompted the development team to enhance the task and time tracking features and consider integrating third-party chat tools to better meet user needs.

The relationship between features and user actions also extends to identifying and addressing usability issues. For instance, Guo and Barnes (2012) analyzed user actions in an e-learning platform and discovered that users often abandoned the platform during complex quiz interactions. This finding led to a redesign of the quiz feature to simplify navigation and reduce user frustration, resulting in higher completion rates and improved user satisfaction.

In software development, defining features is essential to ensure clarity and actionable insights. However, challenges in mapping user interactions to features were highlighted in the IBM Academic Cloud case study (Chapter 5). These challenges stemmed from the diverse interpretations of features among stakeholders and the complexity of mapping abstract definitions to tangible user actions. By focusing on action-based results rather than abstract feature categorizations, this research provided a more practical approach to analyzing user behavior. This focus allowed for the development of structured data pipelines tailored to align user interactions with feature usage, addressing ambiguities and enhancing the analytical utility of metrics.

For example, in IBM Academic Cloud, structured logging mechanisms captured detailed user interactions, such as session durations and specific feature access points. This action-based focus proved instrumental in identifying critical features for refinement and optimization, demonstrating the value of linking user behavior directly to software functionalities.

4.3.1 Challenges With Usage Data and Analyzing User Actions

As mentioned above, usage data can be extracted at any stage and they can be in any form and format bringing many challenges to the analysis of the usage data. The main questions for usage data extractor are what usage data should be extracted and how to map the raw usage data with the right applications or services. Considering the multi-tenant architecture of the cloud, different applications share the same physical and virtual resources. This raises challenges of how to separate and extract the logs that represent each application from the instance (VM) co-hosting the applications.

Another important challenge is handling different contextual information. A system usually has a lot of branches, and thus the system's behaviors may be quite different under different input data or environmental conditions. Knowing the execution behavior under different inputs or configurations can greatly help system operators to understand system behaviors. However, there may be a large number of different combinations of inputs or parameters under different system behaviors. Such complexity poses difficulties in analyzing contextual information related to the state of interest.

Analyzing user actions involves examining the specific tasks and activities that users perform while using the software. This analysis is critical for understanding how features are utilized and identifying opportunities for enhancement. By focusing on user actions, developers can gain a deeper understanding of user needs and preferences, which can inform feature prioritization and development efforts.

The relationship between software features and user actions is inherently interconnected. Features provide the capabilities that users interact with, and user actions generate the data needed to assess the effectiveness of these features. For example, a feature that allows users to upload files will generate data on the number of uploads, the types of files uploaded, and any errors encountered during the process. This data can then be analyzed to improve the feature and enhance the user experience.

Recent literature supports the importance of analyzing user actions to inform feature development. Nguyen and Wu (2018) highlighted that understanding user behavior through detailed action analysis could lead to more user-centric design and improved feature prioritization. Kim and Park (2020) further emphasized that analyzing user actions provides actionable insights that can drive continuous improvement in software development. Additionally, Crowston and Kammerer (2003) discussed how user action analysis could help identify usability issues and areas for feature enhancement, contributing to a more intuitive user experience.

Analyzing user actions is central to this research, as the results of this analysis are later translated to understand how features are used by the users. By dissecting the user actions, developers can map these actions to specific features, thereby gaining a comprehensive view of feature utilization and user engagement. This mapping process is essential for identifying which features are most valuable to users and where there may be gaps in functionality or usability.

The process of analyzing user actions involves several steps. First, data must be collected from various sources, such as log files, telemetry data, and user feedback. This data is then processed and analyzed to identify patterns and trends in user behavior. Advanced analytical techniques, such as machine learning and data mining, can be employed to uncover deeper insights into how users interact with the software Zhou and Chen (2017). Finally, the insights gained from this analysis are used to inform feature prioritization and development decisions.

4.4 Key Metrics for Analyzing Usage Data

To address the research question *[RQ1:] (Problem scoping and objective definition) What are the customized key metrics and data points, tailored to industrial requirements, from user interactions that most significantly influence feature prioritization decisions?*, it is essential to develop a comprehensive understanding of the metrics and data points that provide actionable insights into user interactions. These metrics serve as a foundation for analyzing user behavior and making informed decisions about feature prioritization.

The derivation of the super-set of analytics metrics and data types was informed by an exhaustive exploration of academic literature aimed at capturing both foundational and advanced metrics that facilitate user behavior analysis. Foundational metrics, including frequency and time spent, were identified for their universal applicability and consistent relevance in existing research. Advanced metrics, such as consistency and clickstream data, were noted for their ability to deliver granular, context-rich insights essential for deeper behavioral analysis. The following table encapsulates the primary metrics and data types within the super-set, emphasizing their descriptions and supporting evidence:

Table 4.4: Comprehensive Super-Set of Metrics Derived from Literature

Metric/Data Type	**Description**	**Evidence from Literature**
Frequency	Tracks the number of times a feature is used within a given timeframe.	Frequently cited as critical in understanding user engagement Menzies and Zimmermann (2013); Pagano and Maalej (2013)
Time Spent	Measures the duration users interact with a feature or task.	Emphasized in studies of user engagement and retention Shams and Whittaker (2020); Lo and Zimmermann (2013)
Consistency	Examines the stability of user behavior over time and across updates.	Recognized for its utility in assessing longitudinal behavioral trends Buse and Zimmermann (2012a); Fabijan et al. (2016a)
Clickstream Data	Captures the sequence of clicks and navigation paths taken by users.	Foundational for mapping user workflows and interaction paths Atterer et al. (2006); Banerjee and Ghosh (2001)
Session Data	Provides trends related to session start and end times, indicating overall engagement.	Valuable for contextualizing user interaction patterns Dyckhoff et al. (2012); Graf et al. (2011)
Error Rates	Monitors the frequency of errors encountered during user interactions.	Highlighted as a diagnostic tool for usability challenges De Chaves et al. (2011); Ghezzi et al. (2014)

The Table 4.4 underscores the theoretical richness of the super-set, providing a comprehensive foundation for iterative refinement and selective application, while also setting the stage for addressing emergent challenges in user analytics.

Frequency measures how often a feature is used within a specific timeframe. Widely acknowledged in user engagement studies, frequency is critical for understanding core feature usage. Menzies and Zimmermann (2013) highlighted its application in feature prioritization for software maintenance, demonstrating its role in identifying high-impact features. Similarly, Pagano and Maalej (2013); Maalej et al. (2016) emphasized frequency as a foundation for identifying essential features in open-source software communities. While frequency is invaluable for immediate insights, its standalone application lacks depth. It cannot distinguish between consistent and sporadic users or provide context for interactions, which necessitates combining it with time-based metrics for richer insights.

Time Spent captures the duration of user interaction with a feature or task. Evidence and Use in Literature: Shams and Whittaker (2020) and Lo and Zimmermann (2013) underlined the importance of time spent in evaluating user engagement and task difficulty. Shams and Whittaker (2020) demonstrated its efficacy in identifying features that are either overly complex or engaging, while Lo and Zimmermann (2013) applied time spent metrics to assess software usability. Although effective, interpreting time spent data can be challenging. Extended durations might indicate either high engagement or usability issues. Its reliance on contextual understanding makes it a complementary metric rather than a standalone indicator.

Consistency evaluates the stability of user behavior across time and updates. Buse and Zimmermann (2012c)Zimmermann and Nagappan (2019) explored the utility of consistency in tracking the impact of feature updates on user engagement. Their research showed that stable usage patterns often correlate with user satisfaction and feature reliability. Additionally, Fabijan et al. (2016a) used consistency to monitor the adoption of new features in iterative software development. Consistency offers significant insights but demands robust datasets and longitudinal analysis. The resource-intensive nature of this metric limits its immediate applicability in smaller-scale studies.

Clickstream Data captures sequences of user interactions, offering a detailed map of navigation paths. Atterer et al. (2006) demonstrated the application of clickstream analysis in understanding user workflows and identifying navigation bottlenecks in web applications. This metric has also been instrumental in e-commerce, where Banerjee and Ghosh (2001) and leveraged it to optimize user journeys. While granular and insightful, clickstream data requires substantial infrastructure and analytical capabilities, which may not be feasible for all teams. Its implementation can also pose privacy challenges.

Session Data focuses on when and how users interact with an application, providing high-level engagement trends. Dyckhoff et al. (2012) demonstrated the utility of session data in learning analytics, correlating session lengths with learner outcomes. Graf et al. (2011) used session trends to assess user motivation and retention in e-learning environments. Although session data provides macro-level insights, it lacks specificity. It must be complemented with interaction-level data to provide actionable insights.

Error Rates quantify user-facing issues encountered during interactions. De Chaves et al. (2011) employed error rates to diagnose usability problems in cloud environments. Ghezzi et al. (2014) highlighted their role in improving user satisfaction by identifying systemic issues in feature implementation. Error rates are effective for reactive usability analysis but offer limited foresight into proactive feature optimization. They function best as part of a diagnostic toolkit rather than as standalone metrics.

The iterative refinement process was essential to tailoring the UAM's metrics for industrial applicability. IBM developers and other stakeholders participated in structured interviews and surveys, offering consistent and actionable feedback on the super-set. This feedback emphasized the critical importance of metrics such as frequency, time spent, and consistency, which directly address challenges in feature prioritization and user engagement analysis. Participants highlighted frequency as indispensable for identifying core features, time spent as a means to gauge engagement levels, and consistency for tracking behavioral stability across updates.

This refinement process involved multiple cycles of implementation and feedback, during which other metrics in the super-set, such as clickstream data and error rates, were acknowledged for their theoretical value but deprioritized for immediate industrial application. Practical constraints, including integration complexities and resource limitations, informed these decisions. These metrics remain part of the super-set for potential future implementations, reflecting the method's flexibility and adaptability. By focusing on the most actionable metrics, the UAM ensured a balance between immediate practicality and long-term scalability.

The prioritization of frequency, time spent, and consistency stems from their alignment with the practical needs of industrial stakeholders and their demonstrable impact on software development decisions. The decision-making process was deeply rooted in structured participant feedback obtained through brainstorming sessions, interviews and surveys conducted during iterative refinement cycles.

In the IBM Academic Cloud case study, participants highlighted the indispensable role of frequency in identifying core features that aligned with user needs. Developers explained that tracking the frequency of feature usage allowed them to prioritize updates and optimizations effectively. One participant shared, "We use frequency like a heartbeat monitor, it helps us see which features are alive and thriving with users and where we need to focus next."

In contrast, feedback from IBM Watson Workspace emphasized the significance of time spent as a metric for understanding user engagement and identifying potential usability challenges. A senior developer explained, "Time spent offers a window into how deeply users interact with our features. Short engagement times often indicate areas where improvements are needed."

Consistency emerged as a critical metric during discussions with both IBM Academic Cloud and Odoo Notes participants. Stakeholders underscored its value in tracking longitudinal user behavior, especially following updates or feature releases. An operational manager remarked, "Consistency helps us ensure that updates are enhancing user experiences rather than disrupting them."

Metrics such as clickstream data and error rates, though acknowledged for their theoretical significance, faced challenges in immediate industrial application. Feedback revealed integration difficulties and resource constraints as primary barriers. One architect noted, "While clickstream data is valuable, its implementation requires significant infrastructure, which isn't feasible in our current environment."

This systematic feedback-driven approach ensured that the selected metrics were both theoretically sound and practically implementable. The iterative discussions with stakeholders provided clarity on the trade-offs between various metrics, leading to informed and context-sensitive decisions. The prioritization of frequency, time spent, and consistency stems from their alignment with the practical needs of industrial stakeholders and their demonstrable impact on software development decisions. Frequency emerged as a critical metric for identifying high-value features warranting prioritization, while time spent offered detailed insights into user engagement and usability. Consistency, as a longitudinal measure, enabled the tracking of user behavior across different application versions, highlighting the impacts of feature updates and changes. Metrics such as clickstream data and error rates, though theoretically significant, posed challenges in resource-intensive integration and immediate industrial utility. Grounding these decisions in structured feedback ensured that the UAM remained both theoretically robust and practically implementable. This collaborative process ensured that the selected metrics were not only functional but also highly relevant to real-world software environments.

1. **Frequency of Use (F):** Frequency of use is a fundamental metric that measures the number of times a specific feature or action is accessed by users within a given time frame (t). It reflects the relative importance and relevance of a feature to the user base. For example, high-frequency features are often considered critical to user workflows (Buse and

Zimmermann, 2012b). This metric can be computed as:

$$F = \sum_{u \in U} \text{count}(A_u, t)$$

where U represents the set of users, A_u denotes the actions performed by a user u, and t is the time period. Features with high frequency are typically candidates for optimization or enhancement to maintain their usability.

2. **Time Spent (T):** Time spent on a feature provides insights into user engagement and the complexity of the feature. This metric is calculated as the cumulative duration users spend interacting with a feature, determined from the timestamps of log entries. High time spent can indicate a feature's importance or, conversely, its inefficiency if the time spent does not align with its purpose (Shams and Whittaker, 2020). The metric is expressed as:

   $$T = \sum_{u \in U} \text{time}(A_u)$$

   This data helps prioritize features that require usability improvements or those that hold significant value for users.

3. **Consistency (C):** Consistency measures the stability of user behavior in accessing a feature after updates or modifications. It evaluates whether changes have altered the frequency of use or the time spent on a feature. Consistency can be analyzed through two sub-metrics:

   - **Consistency of Frequency (C_F):** Compares the frequency of feature usage across different versions of the application.
   - **Consistency of Time Spent (C_T):** Examines variations in the time spent on a feature before and after changes.

   These sub-metrics are critical for understanding the impact of updates on user behavior and can guide feature refinement (Buse and Zimmermann, 2012b).

4. **Error Rate (E):** The error rate captures the frequency of user-reported or system-logged errors associated with a feature. High error rates can indicate usability or functionality

issues that require immediate attention. Error rate is calculated as:

$$E = \frac{\text{number of errors}}{\text{number of interactions}}$$

This metric provides actionable insights into areas where user satisfaction might be negatively impacted (Zhang and Wang, 2018).

5. **Abandonment Rate (AR):** Abandonment rate measures the proportion of users who begin interacting with a feature but do not complete the intended action. This metric is particularly useful for identifying features with high dropout rates, suggesting a need for usability improvements (Shams and Whittaker, 2020). It is expressed as:

$$AR = \frac{\text{number of incomplete actions}}{\text{total actions initiated}}$$

6. **User Engagement (UE):** Engagement is a higher-order metric that combines frequency, time spent, and interaction depth. It measures how actively users interact with a feature over time, providing insights into user satisfaction and feature relevance (Claypool et al., 2001a).

7. **Task Success Rate (TSR):** Task success rate evaluates the effectiveness of a feature in enabling users to complete their intended actions. It is calculated as:

$$TSR = \frac{\text{number of successful interactions}}{\text{total interactions}}$$

A low TSR may indicate a need for redesign or additional support mechanisms for the feature (Muller and Stein, 2019).

The inclusion of these metrics provides a comprehensive framework for analyzing user behavior and its impact on feature prioritization. A subset of these metrics (Frequency, Time Spent and Consistency) are practically included for demonstration purposes in this research. By leveraging these data points, this research contributes to the development of a robust usage analytics method capable of systematically identifying and prioritizing features based on user interactions.

Identifying these key metrics involves a systematic approach, starting with the collection of comprehensive usage data. This data is then analyzed using various analytical techniques, such as statistical analysis and data mining, to uncover patterns and trends. By focusing on the most significant metrics, developers can prioritize features that have the greatest impact on user satisfaction and engagement. The utilization of these metrics in feature prioritization involves integrating them into the decision-making process. This can be achieved through the development of dashboards and reporting tools that provide real-time insights into user interactions. Additionally, incorporating these metrics into regular review meetings and development cycles ensures that feature prioritization decisions are data-driven and aligned with user needs.

Recent studies support the importance of these metrics in feature prioritization. For example, Buse and Zimmermann (2012b) emphasized the value of interaction data, such as frequency of use and session duration, in identifying critical features. Shams et al. (2020) highlighted the significance of user engagement metrics, such as user retention rates, in determining feature importance. Furthermore, Nguyen and Wu (2018) demonstrated how navigation paths and user feedback could provide actionable insights for improving feature prioritization. Additionally, Aldhaheri and Abdullah (2020) discussed the effectiveness of combining various usage metrics, including frequency and time spent, to prioritize features that enhance user engagement and satisfaction. The study emphasized the importance of a multi-metric approach in capturing a comprehensive view of user behavior and preferences. Furthermore, Pereira and Silveira (2018) explored the use of machine learning techniques to analyze usage data for feature prioritization. The study found that integrating advanced analytics with traditional metrics, such as frequency and time spent, significantly improved the accuracy and relevance of feature prioritization decisions. Finally, Soltani and Moussavi (2021) highlighted the role of user feedback in refining and validating the metrics used for feature prioritization. The study demonstrated that incorporating qualitative feedback alongside quantitative metrics provided a more nuanced understanding of user needs and preferences, leading to better-informed development decisions.

The choice to focus exclusively on the Frequency, Time Spent, and Consistency metrics in this research is primarily grounded in the practical constraints and opportunities associated with the data source. The research collaboration with IBM Research Labs provided access to a valuable dataset derived from real-world platforms. Such access is highly significant, as it offers authentic user interaction data that aligns with the goals of this research, ensuring the findings are both relevant and generalizable to real-world applications. However, the availability and scope of data collection were subject to specific organizational policies and data-sharing agreements.

IBM Research Labs permitted the use of only Frequency, Time Spent, and Consistency metrics for analyzing the data due to internal data governance policies, which emphasize user privacy, data security, and compliance with organizational guidelines. These metrics were chosen as they effectively capture user interaction patterns while adhering to the restrictions on data usage. While other metrics, such as Error Rate, Abandonment Rate, and Task Success Rate, could provide additional insights into user behavior, their exclusion does not diminish the significance of this research. Instead, it underscores the practical realities of working with real-world datasets governed by strict access policies. The deliberate focus on Frequency, Time Spent, and Consistency ensures the findings remain robust and actionable within the constraints of the available data. This methodological choice not only highlights the practical importance of real-world data but also underscores the value of balancing theoretical comprehensiveness with practical applicability.

Given the limitations on metrics that could be applied to IBM's dataset, the same three metrics were intentionally extended to the analysis of Odoo Notes to maintain consistency in methodology. This uniform approach ensures comparability of results across platforms, enabling a coherent analysis framework that aligns with the research objectives. While additional metrics such as Error Rate, Abandonment Rate, and Task Success Rate could provide richer insights into user behavior, their exclusion does not diminish the significance of this research. Instead, it reflects the practical realities of working with real-world datasets governed by strict access policies. Moreover, focusing on these three metrics ensures the findings remain robust and actionable within the constraints of the available data. This deliberate methodological choice not only underscores the practical importance of real-world data but also highlights the value of balancing theoretical comprehensiveness with practical applicability. By leveraging Frequency, Time Spent, and Consistency, this research demonstrates the adaptability of usage analytics methods in real-world environments while respecting organizational constraints.

4.5 Features and Usage Data of selected Software Applications

In this section, the user-level features and the corresponding usage data of the selected software applications are discussed. Considering the focus of this research, for the remainder of the thesis, features of a software application refer to only the user-level features of the application. The features and usage data of the IBM Watson Workspace application are discussed in detail in

Section 4.5.1. The features and usage data of the Odoo Notes application are discussed in detail in Section 4.5.2.

4.5.1 Features and Usage Data of IBM Watson Workspace

The IBM Watson Workspace application is a cloud-based team collaborative communication application developed by IBM which provides 15 user-level features for users of the application to interact with the list of features is as shown below. Each feature contains multiple actions for users to perform. A feature-action map of the IBM Watson Workspace application is shown in Appendix A, showing the list of features and the comprising action list. The feature-action map serves as one of the inputs for the usage analytics method.

A list of features of the IBM Watson Workspace application is shown below, the features are numbered from *A* to *O*. This list of features is necessary for the usage analytics method to identify which features are used by the users. This list forms an input to the usage analytics method and the results of the analysis are presented as a prioritized list of some or all of these features based on the analysis.

A. One-on-one direct messaging

B. Searchable history with modifiers

C. File sharing

D. Persistent Group Chat - Private Spaces

E. One-on-one video & audio calls

F. Screensharing + whiteboard capabilities

G. Group video & audio calls

H. Public spaces

I. Organizational administrative controls

J. File storage

K. Apps and integrations

L. Built-in Watson cognitive technology

M. SAML-based single sign-on (SSO)

N. Member provisioning and de-provisioning

O. Managed guest user access

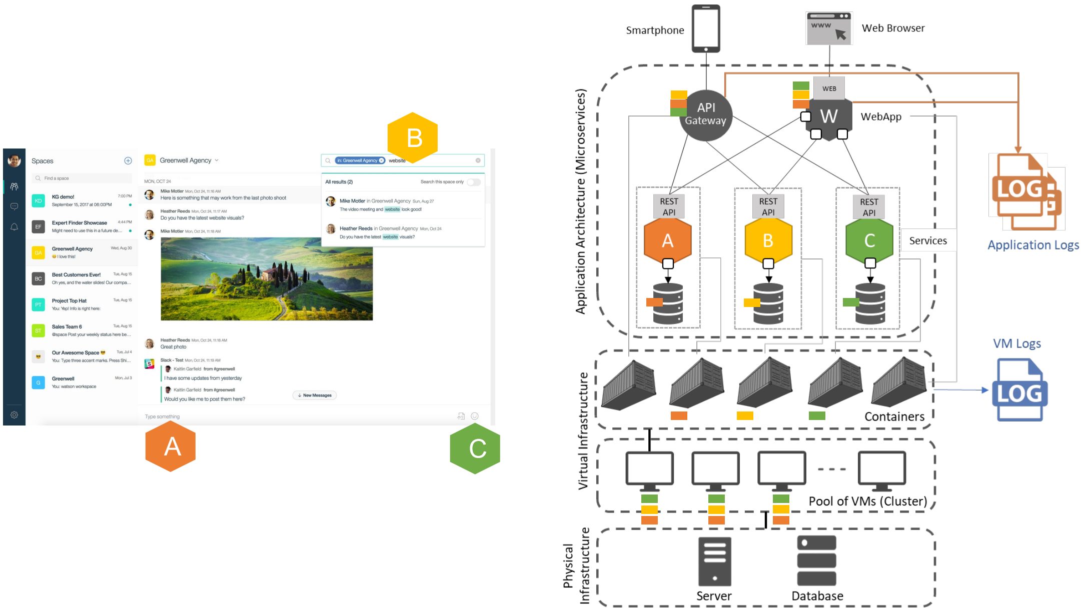


Figure 4.2: User Interface and High-level Microservices Architecture of IBM Watson Workspace application. Highlighted microservices between the architecture and the UI.

The user interface and the high-level architecture of the IBM Watson Workspace application along with the feature-related usage data location are shown in Figure 4.2. The figure also shows three color-coded features as an example: *A. One-on-one direct messaging, B. Searchable history with modifiers and C. File sharing* along with their placement in the UI. The same colour-coded blocks in the architecture show the different components responsible for these features. These components are considered as the source of usage data needed for the usage analytics algorithm, specifically, the API gateway and the WebApp components are used to collect the application logs and containers running the feature services are used to collect the VM logs.

4.5.2 Features and Usage Data of Odoo Notes

Odoo is a suite of open source Customer Relationship Management (CRM) and Enterprise Resource Planning (ERP) applications. Odoo Notes is a collaborative Kanban-style task organizing web application. Odoo Notes provides 19 features (mentioned in bold) grouped in 8 categories as shown in the list below:

1. Build to-do list
    (a) **Create Stages (Columns)** - Break down your to-do list into stages which will be converted to columns in your dashboard
    (b) **Create Notes** - Add notes to your stages. Each note corresponds to a mini-project that you will move from one stage to another as your project moves forward
    (c) **Delete Notes** - Delete a note
    (d) **Edit Notes** - Change the text in the note
    (e) **Kanban View** - Drag and drop notes easily from one stage to another in the Kanban view
2. Organize your notes
    (a) **Text layout** - Insert text styles like headers, bold, italic, lists and fonts with a simple WYSIWYG editor
    (b) **File attachments** - Attach text files, image files document files to your notes
    (c) Tags - Add tags to your notes for a clear organization
        i. **Create new tag** - Create a new tag to the tags list
        ii. **Add tag** - Add a tag to a note
    (d) **Filters and Groups** - Search notes easily with smart filters
    (e) **Colors** - Group your notes by color as a way to categorize your tasks. There are 9 colors to choose from and a color-less option
3. **Import** - Upload any text file or document to your notes
4. **Export** - Export notes as HTML, plain text or DocuWiki text documents
5. Collaborate

   (a) **Invite people** - Add co-workers to your notes so they can follow the discussions and receive notifications

   (b) **Authorship color** - Every author typing some text in a note has a different background color to show who wrote what. You can link a name to a color

   (c) **Timeline slider** - See the history of changes made to a note through a timeline, from the first to the last sentence

6. **Share** - Easily share your notes with your colleagues by sending them as links or embedding URL

7. **Access Settings** - Choose what others can do with your notes by granting viewing or editing access

8. **Chat** - Enable chat for real-time discussion with the people following your notes

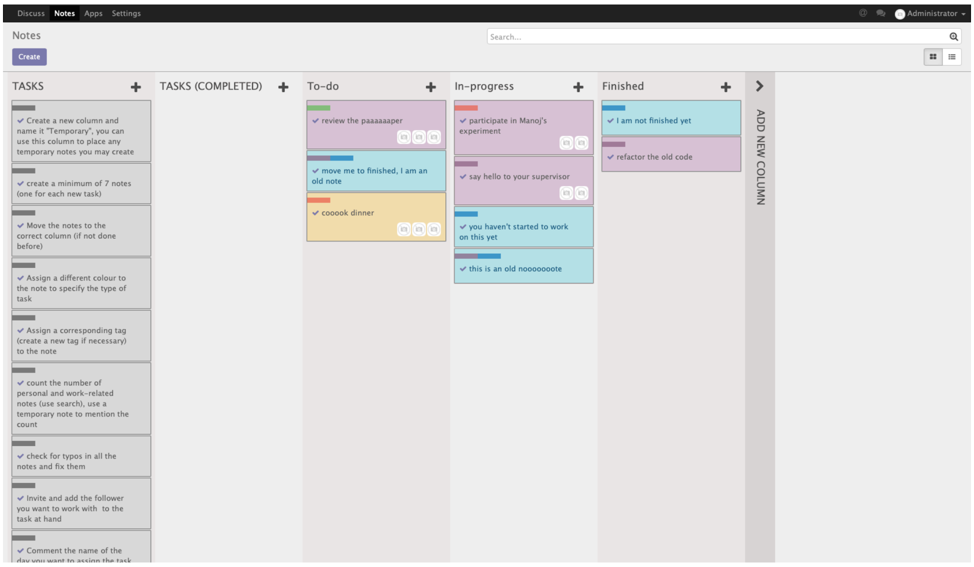
   (a) Show Connected Users - See who is connected to your notes right now

The user interface of the Odoo application version 10 and version 11 are as shown in Figure 4.3 and Figure 4.4 respectively. Odoo uses the Python standard log library logging which is used as the usage data source. However, it uses a special configuration syntax to configure the logging level. Furthermore, the logger is customized to capture the actions performed by the users. The details of the logger customization and the custom setup of Odoo in discussed in detail in Chapter 5 Section 5.4

While the metrics derived from usage data proved valuable, several limitations were encountered during their practical application. For example, integrating metrics like time spent across platforms with distinct user interfaces (e.g., web-based vs. desktop applications) posed significant challenges. Chapter 7 elaborates on how these challenges were addressed through iterative refinements, such as adjusting data pipelines to account for varying interaction patterns and normalizing results for comparative analysis.

Moreover, the Odoo Notes case study highlighted gaps in capturing nuanced user behaviors, such as the subtle differences in how users interact with features across versions. For instance, low-consistency metrics for drag-and-drop functionality provided actionable insights but also revealed the need for more granular tracking mechanisms to fully understand user behaviors.

By iteratively refining these metrics, this research demonstrated their flexibility and adaptability. The cross-platform challenges and solutions discussed in Chapters 5 and 7 further



TASKS

✓ Create a new column and name it "Temporary", you can use this column to place any temporary notes you may create

✓ create a minimum of 7 notes (one for each new task)

✓ Move the notes to the correct column (if not done before)

✓ Assign a different colour to the note to specify the type of task

✓ Assign a corresponding tag (create a new tag if necessary) to the note

✓ count the number of personal and work-related notes (use search), use a temporary note to mention the count

✓ check for typos in all the notes and fix them

✓ Invite and add the follower you want to work with to the task at hand

✓ Comment the name of the day you want to assign the task

To-do

✓ review the paaaaaaper

✓ move me to finished, I am an old note

✓ cooook dinner

In-progress

✓ participate in Manoj's experiment

✓ say hello to your supervisor

✓ you haven't started to work on this yet

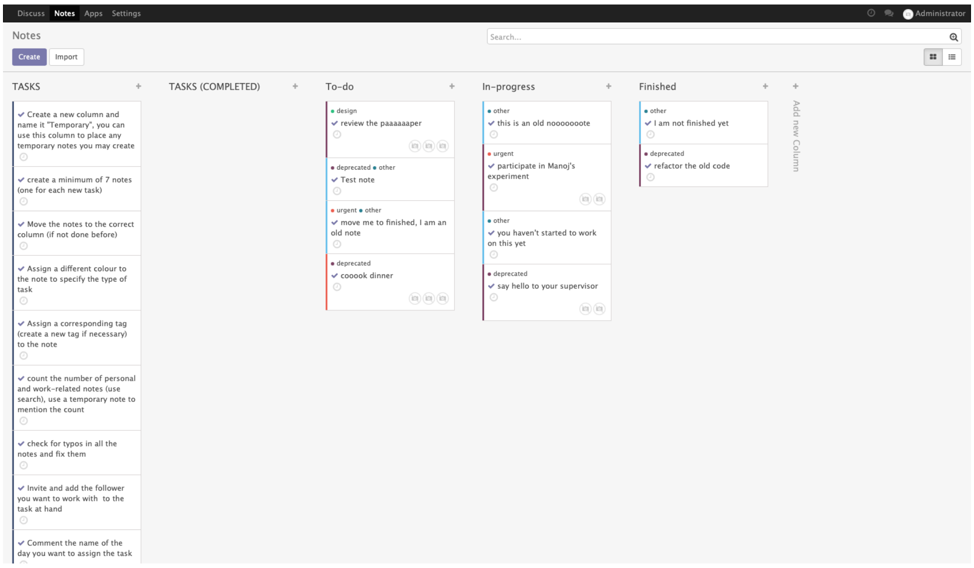
✓ this is an old noooooooote

Finished

✓ I am not finished yet

✓ refactor the old code

Figure 4.3: User Interface of the Odoo Notes application version 10



TASKS

✓ Create a new column and name it "Temporary", you can use this column to place any temporary notes you may create

✓ create a minimum of 7 notes (one for each new task)

✓ Move the notes to the correct column (if not done before)

✓ Assign a different colour to the note to specify the type of task

✓ Assign a corresponding tag (create a new tag if necessary) to the note

✓ count the number of personal and work-related notes (use search), use a temporary note to mention the count

✓ check for typos in all the notes and fix them

✓ Invite and add the follower you want to work with to the task at hand

✓ Comment the name of the day you want to assign the task

To-do

● design
✓ review the paaaaaaper

● deprecated ● other
✓ Test note

● urgent ● other
✓ move me to finished, I am an old note

● deprecated
✓ cooook dinner

In-progress

● other
✓ this is an old noooooooote

● urgent
✓ participate in Manoj's experiment

● other
✓ you haven't started to work on this yet

● deprecated
✓ say hello to your supervisor

Finished

● other
✓ I am not finished yet

● deprecated
✓ refactor the old code

Figure 4.4: User Interface of the Odoo Notes application version 11

emphasize the importance of continuously evolving analytical frameworks to meet diverse operational needs. These refinements ensure that the metrics remain robust and actionable, providing a foundation for future research and development in usage analytics.

This chapter has discussed the importance of understanding software features and analyzing user actions to inform feature prioritization and development decisions. By examining various types of usage data and their relationship to software features, developers can gain valuable insights into user needs and preferences, ultimately enhancing the software development process. Addressing RQ1, the chapter identified key metrics and data points from user interactions that significantly influence feature prioritization decisions and outlined how these can be systematically identified and utilized. The next chapter describes the steps taken to develop the Usage analytics method.

Chapter 5

Case Studies for Development of the Usage Analytics Method

The introduction of this chapter offers a foundational understanding of the challenges developers encounter for prioritizing features in the software development domain. This chapter addresses the following research question 2:

RQ2: (*Design and development*) What are the key challenges faced by the developers to identify and utilize usage data and key metrics related to the feature prioritization effectively within the software platform?

The discussion integrates insights from Case Studies 1 and 2, detailing barriers at each stage of the UA process, such as planning, data collection, design, and validation. By analyzing these challenges comprehensively, the chapter establishes a roadmap for solutions presented in subsequent sections. Furthermore, it highlights the nuanced intricacies developers face in integrating analytics into practical workflows, balancing real-world constraints with theoretical advancements. This introductory context underscores the importance of a structured approach to resolving issues in data-driven feature prioritization. This chapter presents a comprehensive discussion of the challenges encountered during the development and application of the UAM. By consolidating insights from Case Studies 1 and 2, we explore the barriers faced by developers and researchers, structured around the stages of the UA process. This chapter sets the stage for a deeper understanding of the intricacies involved in implementing usage analytics, while serving as a foundation for evaluating the solutions presented in subsequent chapters.

5.1 Overview of the Case Studies

To effectively address the challenges identified in the usage analytics process, it is important to examine real-world applications and their contexts. The case studies discussed in this chapter are instrumental in uncovering practical issues developers encounter and how these issues manifest across different platforms. By analyzing the IBM Academic Cloud, Watson Workspace, and Odoo Notes, this section provides a structured foundation for understanding the nuanced barriers in applying usage analytics methods. These cases demonstrate how platform-specific constraints influence the effectiveness of feature-action mapping, data preparation, and metric utilization, highlighting the complexity of implementing robust analytics solutions in diverse environments.

1. **RQ2: "What are the key challenges faced by developers in identifying and utilizing usage data effectively?"**

   (a) Through the IBM Academic Cloud study, significant challenges such as unclear feature definitions, resource-intensive data preparation processes, and incomplete mappings of user interactions to features were identified. These challenges illustrated the initial barriers developers encounter when attempting to utilize large-scale usage data effectively. The study highlighted how these obstacles hinder the identification of meaningful behavioral patterns, necessitating innovative solutions to streamline data workflows.

   (b) In IBM Watson Workspace, a more complex set of issues emerged related to integrating disparate data sources and applying metrics like consistency and frequency in a meaningful way. These findings emphasized the nuanced requirements for utilizing usage data in dynamic, collaborative development environments. The iterative validation processes implemented during this study provided critical insights into mitigating these challenges through structured adaptation and feedback loops.

2. **RQ3: "How can activities be systematically structured to identify and utilize usage data for feature prioritization?"** The structured development of feature-action maps across the three case studies—IBM Academic Cloud, IBM Watson Workspace, and Odoo Notes—provided a replicable framework for systematically mapping user interactions to features, addressing RQ3 comprehensively:

   (a) **IBM Academic Cloud:** The feature-action map in this case study provided clarity in associating user actions with specific functionalities. By addressing gaps in data mapping and identifying critical user interactions, the study illustrated how systematic mapping could guide informed development decisions. This approach highlighted the importance of establishing robust connections between user behavior and application features to streamline feature prioritization efforts.

(b) **IBM Watson Workspace:** This case study refined the application of feature-action maps by integrating advanced analytics metrics like consistency and frequency. Through iterative alignment of user behavior metrics with actionable insights, the study demonstrated the practical value of using structured mapping for prioritizing features in collaborative environments. This iterative process emphasized how dynamic, real-world scenarios could benefit from a systematic approach to feature prioritization.

(c) **Odoo Notes:** Controlled experiments in this case study showcased the adaptability of feature-action maps in non-industrial contexts. By evaluating user interactions across different application versions, the study validated the flexibility and scalability of the methodology. The findings reinforced how structured mapping could be effectively applied to diverse environments, demonstrating its utility in tailoring feature prioritization to varied user needs.

Collectively, these practical approaches addressed challenges in identifying and prioritizing critical features. They provided concrete examples of how both quantitative metrics and qualitative feedback can be leveraged effectively, underscoring the importance of systematic structuring in feature prioritization activities across varied software environments.

5.1.1 Integration of Empirical Designs

A unified explanation of the empirical designs employed across the case studies highlights their iterative role in refining the UAM. Each study offered unique contributions that were instrumental in addressing specific challenges, enabling iterative adaptation, ensuring broader applicability, and emphasizing the critical importance of contextual relevance. By systematically analyzing the designs, this section uncovers how each case study laid the groundwork for understanding nuanced developer and user behaviors, fostering a cohesive evolution of the method. Through a step-by-step evaluation, the iterative improvements from each case study demonstrate how contextual insights and empirical evidence merged to strengthen the UAM framework, paving the way for its flexible application across diverse environments and use cases.

1. **IBM Academic Cloud:** This study focused on uncovering foundational challenges in collecting and preparing usage data. Developers were engaged through structured interviews to identify gaps in feature-action mapping and inefficiencies in data selection processes. The study illuminated the need for robust data pipelines and better integration of analytics. Detailed analyses from this study laid the groundwork for more advanced refinements explored in subsequent case studies, highlighting its foundational role in shaping the UAM.

2. **IBM Watson Workspace:** Building on insights from the Academic Cloud study, this study refined the UAM by integrating advanced analytics metrics, such as consistency, frequency, and time-spent analysis. These metrics were applied iteratively in a real-world collaborative environment, providing critical insights into the usability, scalability, and adaptability of the method. The study also addressed challenges in validating the relevance of metrics to feature prioritization, emphasizing the role of iterative feedback loops in enhancing their application.

3. **Odoo Notes:** This study extended the UAM's applicability beyond IBM systems to validate its flexibility and adaptability in non-industrial contexts. Controlled user experiments, task-based evaluations, and comparative analyses were employed to assess the method's effectiveness in feature prioritization across different versions of the application. Findings reinforced the generalizability of the UAM while highlighting limitations related to participant diversity, task scope, and data access, demonstrating the method's potential for broader application.

To ensure full transparency, an expansive view of the research process is provided, emphasizing the deliberate and strategic rationale for platform selection, the comprehensive documentation of participant demographics, and the contextual relevance and broader applicability of findings. This approach highlights the careful and methodical considerations that shaped the empirical designs, showcasing their contribution to the trustworthiness and rigor of the research outcomes.

5.1.2 Selection of Platforms for Case Studies

Platform selection involved the thoughtful inclusion of IBM Academic Cloud, IBM Watson Workspace, and Odoo Notes. These platforms represent a diverse range of software development contexts, encompassing academic systems, enterprise-level collaboration tools, and open-source environments. The choice of these platforms was not incidental but opportunistic. The IBM platforms, in particular, provided unparalleled access to real-world data, including actual user interactions, source code, and the direct collaboration of the development teams who built these systems. This level of access, which allowed for a comprehensive and authentic analysis of usage data, is exceedingly rare in academic research and is very valuable for industry-academic partnerships. Such collaboration provided the research with a unique opportunity to refine the UAM in a setting that bridged theoretical exploration with practical application. Furthermore, the inclusion of participants such as developers, architects, and operational managers brought diverse perspectives and expertise to the research. Their involvement enriched the findings by offering practical insights into the challenges of utilizing usage data effectively, enhancing the validity and depth of the research. These participants played a critical role in uncovering nuanced challenges and validating the UAM across different use cases. Finally, the synthesis of data across these distinct contexts underscored the adaptability and robustness of the UAM. By transparently acknowledging limitations related to participant diversity, task scope, and the specificity of the platforms, the research demonstrated how the UAM could be generalized across a broader range of software environments. This comprehensive approach not only validated the method's potential but also provided a roadmap for its future application in diverse and complex scenario.

5.2 Case Study 1: IBM Academic Cloud

IBM Academic Cloud (IBMAC) is a cloud-based application that has been optimized for the educational and research needs of the academic community (Rindos et al., 2014). The IBMAC provides a virtual lab environment for the students with necessary resources provisioned on-demand and can be preconfigured by the lecturers or administrators. IBMAC developers expressed an interest in understanding how users of the application use and interact with the different features of the application and the lecturers and the administrators expressed their interest in understanding how students are affected by the changes implemented by the admins and lecturers to the application. An important additional requirement was to not include additional workload for the students when attempting to understand how they are affected by the changes implemented to the application. As a result, the initial version of the research[1] was designed to investigate how to analyze user behavior with the importance of the impact (positive/negative) of the behavior of the students as well as lecturers, as discussed in (Chandra and Malaya, 2012). The need for frequent updates, and constantly changing requirements in the cloud environment encouraged us to explore and adopt the Agile model. This study resulted in a Usage Analytics approach based on the Agile model and shows how the approach can be implemented in IBM Academic Cloud while explaining the challenges involved and how these challenges could be addressed using Usage Analytics.

In IBM Academic Cloud, lecturers are required to frequently update and handle changing requirements such as new tools, methods, and techniques to improve the quality of courses and modules offered to students. Applications deployed over the cloud and the services provided by the cloud such as IBM Academic Cloud have requirements that change and update frequently. The IBMAC team followed agile development model to handle the changes. Agile development, in opposition to the traditional model, is characterized by a focus on small teams of skilled individual developers, changing requirements, frequent version deliveries, and daily contact with stakeholders making it a suitable candidate for cloud environments. Understanding how these frequent changes and updates affect user behavior is important to improve the quality of the application and services provided by the cloud. This case study is based on the assumption that changes in user behavior is influenced by the changes and updates in the application and services provided by the cloud. To test this assumption, a process model was developed to analyze user interactions in IBM Academic Cloud (Kesavulu et al., 2018b). The process model is based on the generic Agile models used in the software development domain (Wilcox et al., 2010; Abrahamsson et al., 2002) and the resulting process model is shown in Figure 5.1 which consists of the following five phases in each iteration.

- Planning: includes planning on the following goals of the usage analytics process model: *Understanding user's usage Pattern, Understanding user behavior, Identifying Critical features of the application from the user's perspective.*
- Requirements Analysis: involves project managers and software architects dealing with the elicitation of requirements such as *Usage data classification* - refer Sec. 2, *Usage data sources*

[1] Please note that the research was sponsored by IBM as part of the academic-industrial research collaboration between Lero research institute and IBM Research Labs, Dublin, Ireland

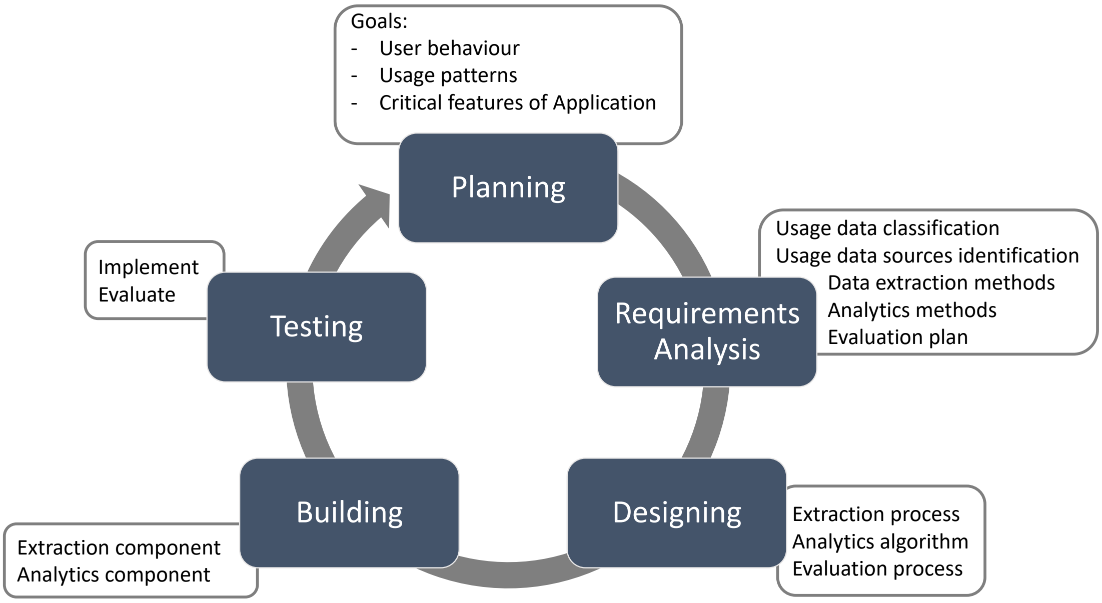


Figure 5.1: Usage Analytics Process Model used by the developers of the IBM Academic Cloud to analyze user interactions with an aim to prioritize features of the application for the next cycle of the development

identification, *Data extraction methods* or *Analytics methods*.

- Designing: includes the software architects dealing with the design of *usage data extraction process, Analytics algorithm and evaluation process.*

- Building: involves developers building the *usage data extraction and analytics component.*

- Testing: involves *implementing* the usage data extraction and analytics components and *evaluation.*

In the following sections, we describe the case study in detail and explain how the process model was applied to IBM Academic Cloud.

5.2.1 Goal - Research Problem Identification

This case study aims to address part of the RQ2: *what challenges are associated with feature prioritization process?*. The goal of this case study is to identify the challenges faced by developers for feature prioritization by critically exploring the current process followed by the developers to prioritize features for the next cycle of the development.

IBM Academic Cloud provides virtual resources (VMs) on-demand basis for universities to help run classes and labs. Users (lecturers, academic administrators and students) access the VMs through an RDP connection with any desktop, or laptop with a basic configuration. Lecturers and administrators can request and change the configurations of the VMs while students can only access the VMs provided to them. A scenario was designed to incorporate the Usage Analytics process model as shown in Figure 7.1. The first phase is to decide the goals, meaning what to achieve from analyzing the user's usage patterns (Kesavulu et al., 2017a). Lecturers and administrators faced challenges in understanding how students interacted with the applications hosted on the VMs provided to them, as well as assessing the positive or negative impact of changes made to the VM configurations on student behavior. Additionally, they sought to identify the most suitable configurations tailored to individual students or specific student groups.

5.2.2 Design - Analysis of the Usage Analytics Process Model

To achieve the goal defined above, a comprehensive analysis of the Usage Analytics Process Model was conducted to identify key stages and potential bottlenecks in the feature prioritization process. The model comprises five main stages: Data Collection, Data Processing, Data Analysis, Feature Prioritization, and Feedback Loop.

The Data Collection stage involves gathering user interaction data from various sources within the IBM Academic Cloud platform. Key metrics such as frequency of use, duration of use, and user satisfaction ratings were identified as significant for prioritization. These metrics align with findings from literature which emphasize the importance of user interaction data in understanding user behavior and preferences (Bucklin and Sismeiro, 2009a; Cito et al., 2015b).

Implementing data collection tools required integration with existing logging systems and setting up new analytics frameworks to capture relevant metrics. This involved deploying tools to track user actions, session durations, and user feedback systematically. The challenges faced during this stage included ensuring data quality and integrating different data sources, which often led to inconsistencies and gaps, making comprehensive data collection difficult (Chen et al., 2011; Olsson and Bosch, 2014c).

The Data Processing stage was critical for cleaning and transforming the collected data into a usable format. This stage was resource-intensive and error-prone, as it required dealing with large volumes of data efficiently. The complexity of this task is well documented in the literature, highlighting the need for robust data preprocessing frameworks (Wang et al., 2011; Cito et al., 2015b).

In the Data Analysis stage, advanced analytical techniques were applied to extract meaningful insights from the processed data, focusing on three key metrics: Frequency, Time Spent, and

Consistency. Frequency analysis was used to identify the most accessed features by calculating the number of interactions per feature within specific time frames. Duration analysis measured the total time spent by users on individual features, providing insights into user engagement and potential inefficiencies. Consistency analysis evaluated the stability of user behavior by comparing Frequency and Time Spent metrics across different versions of the application, enabling an assessment of the impact of updates. These analytical approaches, while straightforward, posed challenges due to the complexity of real-world datasets and the need for precise interpretation (Fabijan et al., 2016b; Tizard et al., 2022).

The Feature Prioritization stage involved translating analytical insights into actionable development priorities. This required close collaboration between analysts and developers to ensure that the insights derived from the data were aligned with development goals. Issues with aligning these insights with development priorities were identified, as supported by the literature (Fagerholm et al., 2014; Lindgren and Münch, 2016).

The Feedback Loop stage aimed at continuously refining the process based on feedback from developers and users. Establishing a robust feedback loop was essential for ensuring the relevance and accuracy of the analytics process over time. This involved gathering real-time feedback and making necessary adjustments to the data collection and analysis techniques (Fitzgerald and Stol, 2017b; Bauer et al., 2017).

Developer interviews provided insights into the challenges they face in prioritizing features and how they use data in their decision-making process. Common issues identified included the difficulty in interpreting complex data and the need for specialized analytical skills. These challenges are supported by findings from (Maalej et al., 2009; Ogonowski et al., 2013).

5.2.3 Result: Initial Design of the Usage Analytics Method and Feature Prioritization Challenges

The implementation of the usage analytics process model in IBM Academic Cloud revealed several challenges related to prioritization of software features and led to the development of the Initial Design of the Usage Analytics (UA) Method (Version 1).

Initial Design of the Usage Analytics Method

The initial design of the usage analytics method was developed based on the insights gathered from the process model analysis and pilot testing. This design is depicted in Figure 7.1 and includes a comprehensive framework for collecting user interaction data from various sources, a robust pipeline for cleaning, transforming, and storing data in a usable format, initial models for analyzing data to identify usage patterns and feature importance, a mechanism for translating analytical insights into feature prioritization decisions, and a system for gathering and incorporating feedback from developers and users to continuously improve the analytics process.

Requirements Analysis of the Usage Analytics process model is to analyze the requirements, for example, application logs contain non-trivial information such as the event data, user ID, access time and so on. System logs reveal information such as workload, processing time, errors and so on. Cross-matching information from both types of logs can help understand students' usage patterns, and impact students' behavior (for example, deviation from typical interactions) as a result of configuration changes made by the lecturer/administrator to the VMs and so on. Here, application and system log files are treated as usage data sources. The third *Designing* and fourth *Building* phases deal with carefully designing and developing suitable usage data extraction processes (e.g., log extraction and analysis) and tools (e.g., logstash, ELK stack), analytics methods (e.g., regular expressions, machine learning, statistical analysis). However,

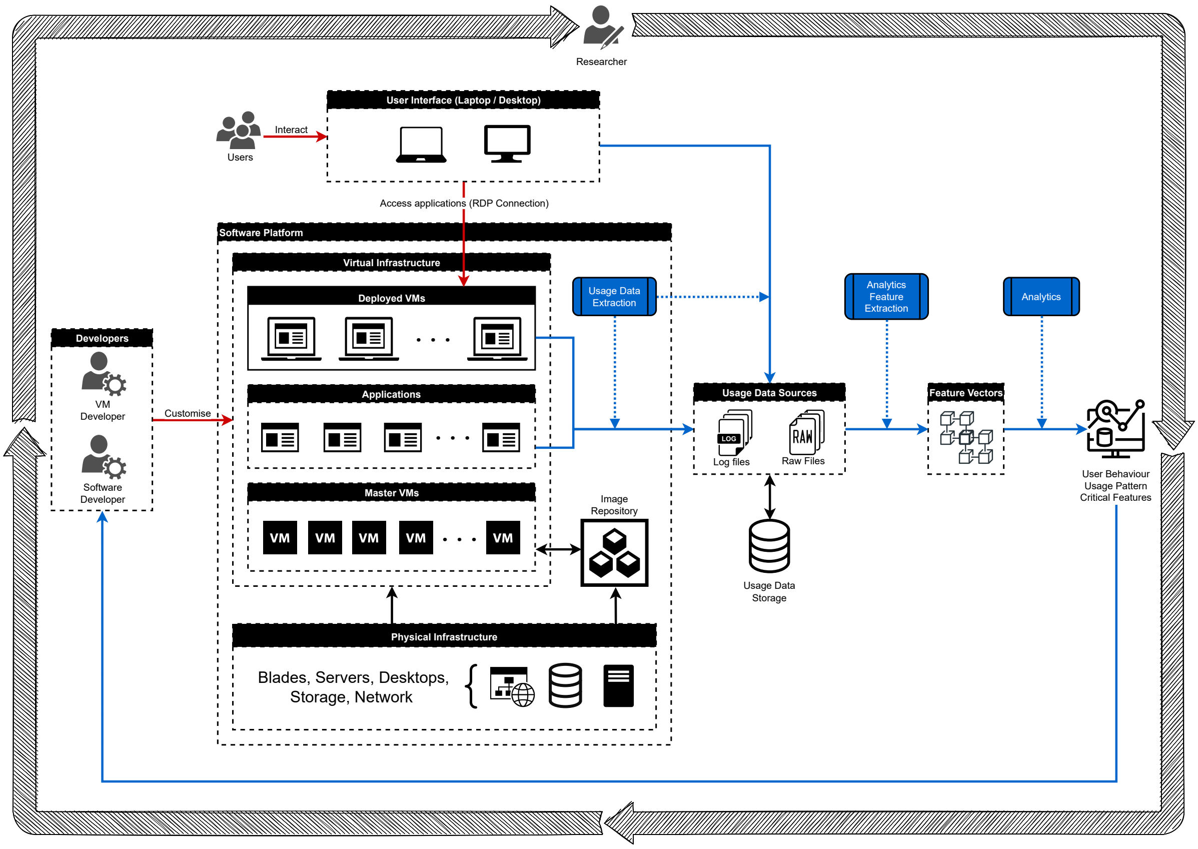


Figure 5.2: Usage Analytics Method Design - Version 1 resulted from the exploration of IBM Academic Cloud Application.

tools vary for different scenarios. The fifth *Testing* stage deals with the evaluation of the model. Questionnaires, surveys and interviews are suitable for the initial iterations to validate the extraction and analytics components designed and developed in the previous phases. This case study resulted in the Usage Analytics experiment design shown in Figure 7.1. The experiment is designed to start with providing the VMs to students and then changing the configuration of the VMs to see how the changes impact user behavior. The experiment is designed to be conducted in two iterations. In the first iteration, the experiment is designed to be conducted in a controlled environment where the VMs are provided to students and the configuration of the VMs is changed by the lecturer/administrator. In the second iteration, the experiment is designed to be conducted in a real-world environment to compare the results of the experiments and examine if a controlled environment affects the results. During both iterations, system and application logs are collected and processed for further analysis. The results of the analysis revealing the impact of the changes made to the VMs on user behavior will be used by the lecturer and the administrator to make informed decisions on the configuration of the VMs in the future. Similar development attempts of process models, software platforms and applications using the agile models can be seen in (Bauer et al., 2017; Claps et al., 2015; Fagerholm et al., 2014).

The Initial Design of the Usage Analytics (UA) Method includes three primary stages: Usage Data Extraction, Analytics Feature Extraction, and Analytics.

1. **Usage Data Extraction:** The first stage, Usage Data Extraction, focuses on gathering raw data from various sources within the IBM Academic Cloud platform. This involves implementing robust data collection frameworks capable of capturing a wide range of user interaction metrics, including frequency of use, duration of use, and user satisfaction ratings. The challenge here is to ensure the data collected is comprehensive and accurate, which requires the integration of different data sources and maintaining high data quality. The literature emphasizes the importance of capturing diverse interaction data to gain a holistic understanding of user behavior and preferences (Bucklin and Sismeiro, 2009a; Cito et al., 2015b). The deployment of data collection tools, such as logging systems and analytics frameworks, was critical in this stage.

   - **IBM Log Analysis with LogDNA:** This tool provided robust logging capabilities, allowing for the collection and analysis of logs from various application components. It facilitated the tracking of user actions by capturing log entries related to application interactions, which were later processed to extract metrics such as Frequency and Time Spent.

- **IBM Cloud Monitoring with Sysdig:** This monitoring tool was utilized to track system performance and resource utilization while also integrating user interaction data. Sysdig provided insights into how users interacted with cloud-hosted applications, helping to identify patterns in feature usage.
- **Custom Analytics Pipelines:** In addition to IBM's proprietary tools, custom analytics pipelines were developed to preprocess and structure the collected data. These pipelines integrated raw logs and user interaction records into a structured format, enabling the calculation of the core metrics—Frequency, Time Spent, and Consistency.

These tools were configured to systematically track user actions, session durations, and user feedback, providing a rich dataset for subsequent analysis. Ensuring the consistency and reliability of this data was a significant challenge, as highlighted by previous studies that stress the importance of robust data collection methods (Chen et al., 2011; Olsson and Bosch, 2014c).

2. **Analytics Feature Extraction:** The second stage, Analytics Feature Extraction, involves processing the raw data collected in the first stage to extract meaningful features that can be analyzed. This stage is crucial for transforming the data into a usable format, which includes cleaning, filtering, and aggregating the data. The complexity and volume of data necessitate efficient preprocessing techniques to handle large datasets without compromising accuracy. The literature underscores the necessity of efficient data preprocessing to prepare the data for analysis (Wang et al., 2011; Cito et al., 2015b). During this stage, the data was cleaned to remove any inconsistencies and transformed to a format suitable for analysis. This included aggregating data points to derive metrics that could provide insights into user behavior, such as average session duration and frequency of feature usage. The challenge here was to ensure that the extracted features accurately represented the underlying user interactions, a task that requires sophisticated preprocessing pipelines as discussed in the literature (Fabijan et al., 2016b; Tizard et al., 2022).

3. **Analytics:** The final stage, Analytics, involves applying advanced analytical techniques

to the processed data to identify patterns and generate insights. This stage is critical for understanding how users interact with the IBM Academic Cloud platform and for making data-driven decisions about feature prioritization. To analyze the key metrics—Frequency, Time Spent, and Consistency—two machine learning models were employed:

- **k-Means Clustering:** Clustering was used to group users or features based on interaction patterns derived from Frequency and Time Spent metrics. For example, k-Means clustering identified user segments with similar usage behaviors, such as power users who frequently interacted with specific features. These clusters provided insights into user engagement levels and guided decisions about feature prioritization and customization.
- **Linear Regression:** Regression analysis was applied to examine the relationships between metrics, such as the correlation between Time Spent and Frequency, or the impact of updates on Consistency. This model helped quantify the effect of changes and identify trends, offering a predictive understanding of user behavior across different application versions.

These two models were chosen for their relevance to the research objectives and their ability to generate actionable insights from the available metrics. By combining clustering to identify patterns in user behavior with regression to analyze and predict relationships between metrics, this analytical framework effectively supported the decision-making process on the IBM Academic Cloud platform. The complexity of these techniques and the need for specialized skills were significant challenges in this stage, as noted in previous research (Fabijan et al., 2016b; Tizard et al., 2022). The insights generated during this stage were used to inform feature prioritization decisions, ensuring that the development efforts were focused on the most impactful features. This required close collaboration between analysts and developers to translate the analytical findings into actionable development priorities. The literature highlights the importance of this collaboration to align data insights with development goals (Fagerholm et al., 2014; Lindgren and Münch, 2016). Additionally, establishing a feedback loop was essential for refining the analytics process based on real-time feedback from developers and users. This continuous improvement cycle is crucial for maintaining the relevance and accuracy of the usage analytics method over time, as emphasized by (Fitzgerald and Stol, 2017b; Bauer et al., 2017).

Feature Prioritization Challenges

IBM's Chief Analytics Office reports handling over 30,000 new ideas and feature requests annually, highlighting the immense volume of data that needs to be processed manually. This situation is analogous to the challenges faced in the IBM Academic Cloud, where developers struggle with the manual processing of vast amounts of user feedback. The absence of a systematic approach results in inconsistencies and inefficiencies, underscoring the necessity for automated solutions to manage and prioritize feature requests effectively (Office, 2020). IBM's research on hybrid anomaly detection and data governance emphasizes the critical challenge of inconsistent data categorization and manual processes. These challenges are prevalent in the IBM Academic Cloud, where the lack of mapping user actions to features hinders effective data-driven analytics. The adoption of advanced analytics and automated data management practices has proven beneficial in other IBM projects, suggesting similar approaches could enhance feature prioritization in the IBM Academic Cloud (Wei et al., 2019; Rouse, 2018).

The developers found it difficult to analyze user interactions because not all platform features were clearly defined. This lack of clarity made it challenging to understand how users were interacting with the application and which features were being utilized. Without a clear definition of platform features, it was nearly impossible to map user interactions accurately and derive meaningful insights from the data. This issue is highlighted in the literature, which emphasizes the importance of well-defined features for effective usage analytics Maalej et al. (2009); Olsson and Bosch (2014c). There was no mapping of user interactions to the features of the application. Developers struggled to understand which actions performed by the user were related to which features of the application. This lack of mapping created significant challenges in analyzing user behavior and identifying the most critical features. Without a clear mapping, it was difficult to prioritize features based on user interaction data. This challenge is supported by previous research, which underscores the need for a detailed mapping of user interactions to application features for effective analytics Fabijan et al. (2015); Fagerholm et al. (2014). There were many different types of data available with the application that could be treated as usage data. Choosing which specific types of data, the source of the data, and the format of the data required a tremendous amount of time and effort from the developers. This created a bottleneck for the development process, as developers had to spend significant time and resources on data selection and preparation. The literature highlights the importance of selecting the right types of data and formats for effective usage analytics and the challenges associated with this process

Chen et al. (2011); Wang et al. (2011).

The process model analysis revealed critical stages where data collection and analysis were time-consuming and prone to errors, directly relating to the key challenges identified:

1. **Data Collection Bottlenecks:** Integrating different data sources and ensuring data quality were major challenges in the Data Collection stage. The lack of clear feature definitions and the absence of a user interaction mapping to features made it difficult to collect relevant and accurate data. The heterogeneity of data sources often led to inconsistencies and gaps, making comprehensive data collection difficult Chen et al. (2011); Olsson and Bosch (2014c).

2. **Data Processing Bottlenecks:** In the Data Processing stage, the complexity of cleaning and transforming large volumes of data efficiently was exacerbated by the challenge of selecting specific types, sources, and formats of data. This stage was resource-intensive and error-prone, as developers had to spend significant time on data preparation. Data preprocessing is a critical step that can be resource-intensive and error-prone, as highlighted by Wang et al. (2011); Cito et al. (2015b).

3. **Data Analysis Bottlenecks:** The Data Analysis stage posed significant challenges due to the complexity of the data and the need for specialized analytical skills. The absence of a clear mapping of user interactions to features made it difficult to apply advanced analytical techniques and interpret the results accurately. Applying advanced analytical techniques and interpreting the results were difficult tasks that required the use of sophisticated tools Fabijan et al. (2016b); Tizard et al. (2022).

4. **Feature Prioritization Bottlenecks:** During the Feature Prioritization stage, aligning analytical insights with development priorities was identified as a major issue. The lack of clear feature definitions and user interaction mappings made it challenging to translate data insights into actionable development priorities. Translating data insights into actionable development priorities required close collaboration between analysts and developers Fagerholm et al. (2014); Lindgren and Münch (2016).

5. **Feedback Loop Bottlenecks:** Establishing a robust Feedback Loop was essential for continuous improvement based on real-time feedback. However, the absence of a clear feature mapping and comprehensive data collection made it difficult to gather and in-

corporate relevant feedback from developers and users. This stage involved gathering and incorporating feedback from developers and users to refine the analytics process Fitzgerald and Stol (2017b); Bauer et al. (2017).

The experiment design in this case study forms the basis for the usage analytics method and future designs of the experiments. The experiment design is further improved to include specific data sources, the process to extract the usage data from the identified data sources and implemented in the following case studies. The next section describes the case study on IBM Watson Workspace.

5.3 Case Study 2: IBM Watson Workspace

IBM Watson Workspace was a team collaboration application built at IBM Labs. It was a cloud-based application that provided a platform for teams to collaborate and share information in real-time. The application was built using IBM Watson services such as Watson Conversation, Watson Discovery, Watson Natural Language Understanding, Watson Language Translator and Watson Tone Analyzer. The application was built using the IBM Bluemix platform. Figure 5.3 shows the process followed with the IBM Watson Workspace use case. The process starts with identifying the features, specifically the user-related features, and the data sources. The data sources are then analyzed to identify the data that can be used to analyze the user behavior. The data is then extracted from the data sources and processed to extract the required information. The extracted information is then analyzed to identify the user behavior. Figure 5.4 shows the high-level architecture of the IBM Watson Workspace application. The application is built using the microservices architecture where each microservice is responsible for a specific feature of the application. Each microservice has an API that is used to communicate with other microservices and for users to access the various features of the application either through an API gateway or a messaging queue (MQTT server). These microservices are served by one or more containers running on a pool of VMs.

Application, infrastructure, and production health checks are monitored using New Relic[2] in IBM Watson Workspace. New Relic is a Software-as-a-Service (SaaS)[3] platform which allows real-time monitoring and instrumentation of bare metal VMs, operating systems, containers

[2] https://newrelic.com

[3] SaaS is a software distribution model in the cloud computing paradigm in which a third-party provider hosts applications and makes them available to customers over the Internet

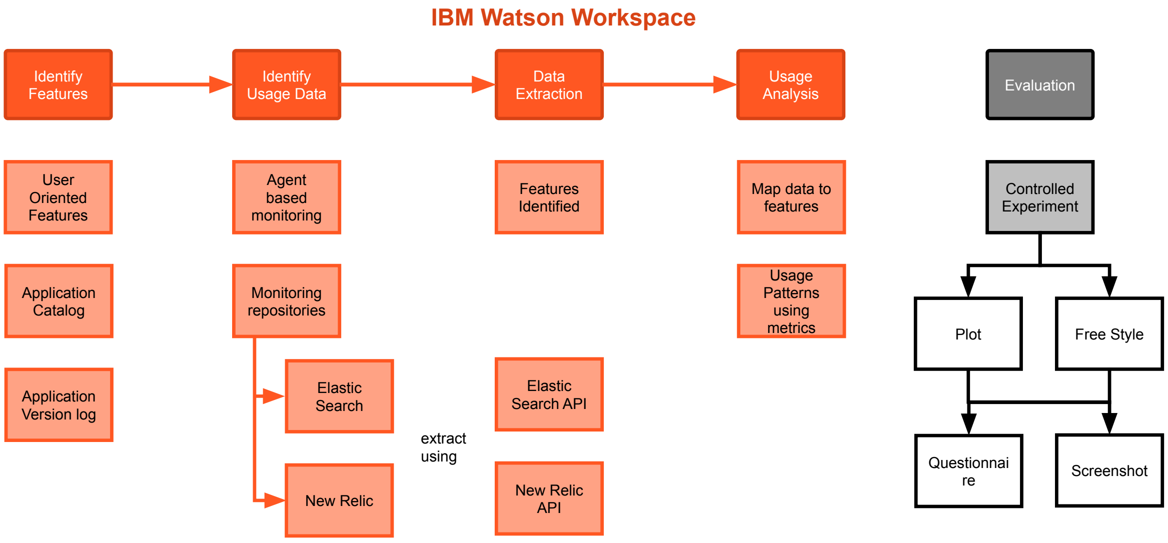


Figure 5.3: Use Case 2: IBM Watson Workspace

and other hardware resources to front-end systems, UI/UX components or API layer of the application. Watson Workspace heavily relies on the capabilities of New Relic to keep its production services up and running. New Relic also runs production health checks for IBM Watson Workspace, these health checks trigger an alert if there are any inconsistencies in defined SLAs and current SLAs for the immediate attention of the team. The practices currently employed by the developers of the application are concentrated on maintaining the application and system health, proper execution with acceptable performance and elimination of bugs and errors. Changes are made to the components of the software for these purposes on a daily basis. Understanding the impact of these changes on the users and their usage behavior to efficiently provide updates with such a high velocity and frequency tends to be a laborious task for the developers.

5.3.1 Goal - Implementation challenges of UA

This case study address the RQ2: *What are the key challenges faced by the developers to identify and utilize usage data and key metrics related to the feature prioritization effectively within the software platform?*, while considering the challenges identified in the previous case study. This question focuses on the methodologies and tools necessary for capturing and analyzing usage data, exploring best practices for data collection, data cleaning, and data analysis to ensure that the metrics gathered are accurate and actionable. This section focuses on the methodologies

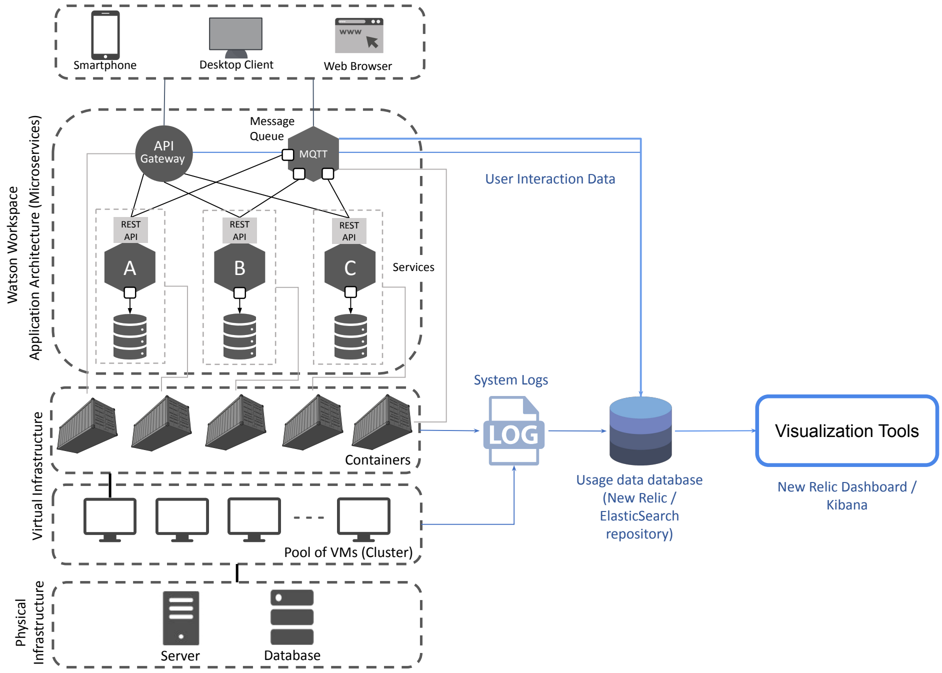


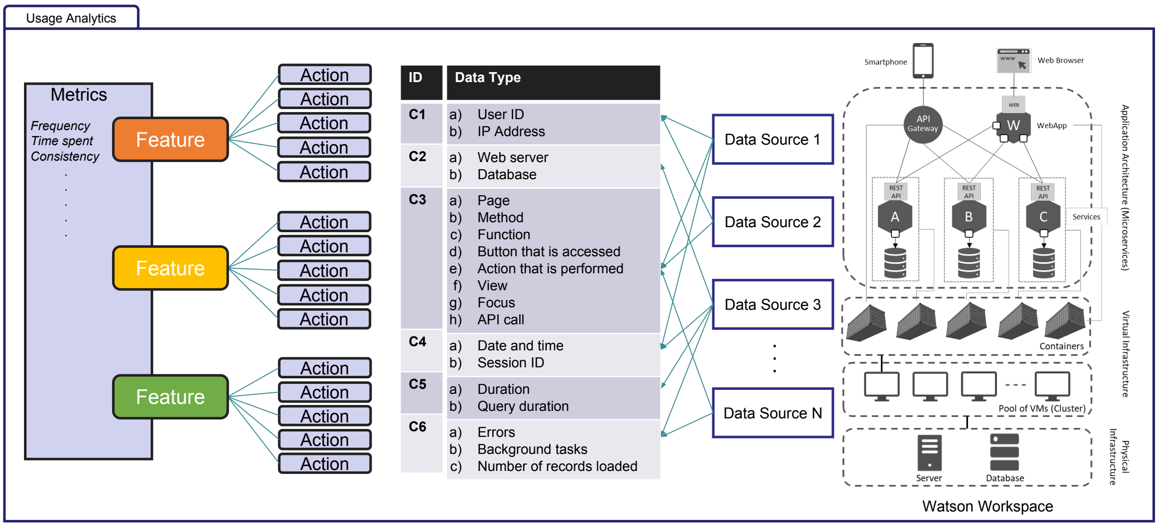
Figure 5.4: High-level architecture of the IBM Watson Workspace application. Highlighting the data sources necessary for the Usage Analytics method.

and tools essential for capturing and analyzing usage data, with an emphasis on best practices for data collection, cleaning, and analysis to ensure accuracy and actionability. Furthermore, it explores strategies for integrating this data into the feature prioritization process, enabling data-driven decision-making.

5.3.2 Design - Development and Application of Usage Analytics Method

Case Study 1 established the foundational three-stage framework of the Usage Analytics (UA) method, focusing on Usage Data Extraction, Analytics Feature Extraction, and Analytics. It provided a proof of concept but revealed limitations in handling complex software systems and diverse data sources. Building on these insights, Case Study 2 expanded the method into a four-stage process—Identify Features of the Application, Identify Usage Data Sources, Data Extraction, and Analytics. This iteration addressed the shortcomings of the initial framework, adding structure and scalability. Together, the two case studies illustrate an iterative refinement process, with Case Study 1 laying the groundwork and Case Study 2 enhancing the method for broader applicability. The Usage Analytics approach aims at analyzing the software usage data to understand the impact of changes made to the application on the usage behavior of the users. The high-level diagram of the Usage Analytics approach is shown in Figure 5.5.

The approach involves four stages:



ID	Data Type
C1	a) User ID b) IP Address
C2	a) Web server b) Database
C3	a) Page b) Method c) Function d) Button that is accessed e) Action that is performed f) View g) Focus h) API call
C4	a) Date and time b) Session ID
C5	a) Duration b) Query duration
C6	a) Errors b) Background tasks c) Number of records loaded

Figure 5.5: High-level diagram of the Usage Analytics approach designed using IBM Watson Workspace application showing the available data sources

- **Identify features of the application**: The first step is to identify the list of features provided by the application and the actions a user can perform that relate to each feature. This involves a comprehensive review of the application catalog, version logs, and developer documentation to compile an exhaustive list of features and functionalities. The application catalog typically provides a detailed description of each feature, including its intended purpose and how it fits into the overall application. By thoroughly analyzing the catalog, developers can gain a clear understanding of the functionalities offered by the application. Version logs are equally important as they document changes made to the application over time. These logs provide insights into the evolution of features, including new additions, modifications, and deprecations. By reviewing the version logs, developers can track the development history and understand how features have been updated and improved. Developer documentation, including technical specifications and implementation guides, offers a deeper insight into how features are built and integrated into the application. This documentation is invaluable for identifying the technical details of each feature, which can help in mapping user actions to features accurately. Once the final updated list of features is identified (as shown in Appendix A), the actions a user can perform are mapped to the corresponding feature they belong to. This feature-action map will serve as input to the analytics stage.

Feature-action map

A feature-action map, while increasingly regarded as a standard tool for monitoring user interactions in modern software applications, is not universally implemented across all systems. Many applications still lack a pre-existing feature-action map, requiring it to be built from scratch. This process involves identifying and mapping specific user actions such as clicks, navigation events, and input submissions to corresponding application features. The absence of a predefined feature-action map can result from factors such as legacy system architectures, lack of standardized monitoring practices, or the complexity of integrating diverse user interfaces. Constructing such a map is a critical step in enabling effective usage analytics, as it provides the foundational structure necessary for tracking and analyzing user interactions to inform feature prioritization and system optimization. A feature-action map is a tool that connects the features of an application to the specific actions performed by users, thereby serving as a guide to understanding how users interact with the application. The theoretical foundation of feature-action maps draws from several key areas within human-computer interaction (HCI), usability engineering, and user-centered design. These disciplines provide the necessary framework and methodologies for understanding and improving user interactions with software applications. The techniques used to create a feature-action map for the IBM Watson Workspace are exploring *action logs and tags* which were already available as the developers were using New Relic tool for application monitoring, this tool included monitoring some features of the application for managing health checks and *heuristic analysis* for identifying actions related to features that were not included in the New Relic tool.

Action Logs and Tags: Action logs and tags involve tagging each user action with a unique identifier that corresponds to a specific feature within the software application. This method requires a well-thought-out design during the development phase to ensure that all actions are accurately tagged and logged. The use of action logs and tags provides a detailed record of user interactions, which can be analyzed to understand how users engage with different features. For instance, tools like Google Analytics and Mixpanel offer functionalities to tag and track user actions. These tools enable developers to define events (such as button clicks, page views, and form submissions) and assign tags to these events. By doing so, each action performed by the user is recorded with an associated tag that identifies the feature being used. This tagged data can then be aggregated and analyzed to gain insights into feature usage patterns (Holmström Olsson and Bosch, 2013). The effectiveness of action logs and tags depends on the granularity and accuracy of the tags. Granular tagging allows for more detailed analysis but requires a comprehensive tagging strategy to ensure that all relevant actions are captured. Accurate tagging, on the other hand, ensures that the recorded actions are correctly associated with the corresponding features. To achieve this, developers must collaborate closely with UX designers and product managers to identify the key actions and features that need to be tracked.

Heuristic Analysis: Heuristic analysis involves using predefined rules or heuristics to infer which user actions are related to specific features. This method can be particularly useful when action logs and tags are not available or when it is not feasible to tag every user action. Heuristic analysis can be based on patterns observed in user behavior, such as sequences of actions that are typically associated with the use of a particular feature. For example, if a user frequently accesses a set of related functions within a short time frame, it can be inferred that these actions are associated with a specific feature. Heuristic analysis can also involve analyzing the context in which actions are performed, such as the type of task the user is attempting to complete or the specific conditions under which actions occur (Newman, 2010). By playing the role of a user and using the developer documentation as reference, the application was used by exploring each feature described in the documentation. In addition to manually recording the actions performed, the application logs generated during this time was recorded and analyzed to identify the list of actions performed. The resulting list of actions are mapped to the features, example of a single feature with its mapped related actions in JSON format as shown below.

```json
{
    "Features": [
        {
            "id": "1",
            "Persistent Group Chat - Private Spaces": [
                "CHAT_MESSAGE_DELIVERED",
                "CHAT_MESSAGE_RECEIVED",
                "CHAT_MESSAGE_SEND",
                "COGNITIVE_MOMENTS_LENS_CLICKED",
                "COGNITIVE_MOMENTS_LOADED",
                    "COGNITIVE_MOMENTS_MOMENT_CLICKED",
                    "COGNITIVE_MOMENTS_UNLOADED",
                    "CREATE_TEAM_COMPLETE",
                    "CREATE_TEAM_START",
                    "DELETE_MESSAGE_CANCEL",
                    "DELETE_MESSAGE_CONFIRM",
                    "DELETE_MESSAGE_START",
                    "DELETE_MESSAGE_SUCCESS",
                    "MENTION_CREATED",
                    "MENTION_LIST_ITEM_OPEN",
                    "MENTION_LIST_PAGINATED",
                    "MENTION_SPACE_CREATED",
                    "MENTION_VIEW_OPEN",
                    "SYNC_SPACES",
                    "SYNC_SPACES_ERROR",
                    "TEAM_ROOM_OPEN",
                    "TEAM_ROOM_VIEW_MESSAGES"
                ]
            },
```

This example illustrates how the feature “Persistent Group Chat - Private Spaces” was identified as a key feature within IBM Watson Workspace. The team listed all relevant actions associated with this feature, such as “CHAT_MESSAGE_SEND” and “COGNITIVE_MOMENTS_LOADED”. Each action was tagged with a unique identifier during the development phase, ensuring accurate tracking. For actions not directly logged, heuristic methods were used to infer their association with the feature based on observed user behavior patterns. All identified actions were mapped to the feature in a structured format (JSON), providing a clear representation of user interactions with the feature. Creating a feature-action map involved several detailed steps, each requiring careful planning and execution.

1. **Setting Up Monitoring Tools** To begin the process, the development team integrated New Relic for application monitoring. New Relic was already used by the

developers to monitor various features of the application, including managing health checks. During the development phase, the team designed the system to ensure that all actions were accurately tagged and logged. This involved collaboration with UX designers and product managers to identify key actions and features. For applications needing to create a feature-action map, tools like Google Analytics[4] and Mixpanel[5] could be employed to define events such as button clicks, page views, and form submissions, and to assign tags to these events.

2. **Data Collection Through Action Logs** With the monitoring tools in place, action logs were utilized to track user interactions. The action logs provided a detailed record of user interactions, which were then aggregated and analyzed to understand how users engaged with different features. The granularity and accuracy of the tags were crucial for this analysis. Granular tagging allowed for more detailed analysis, while accurate tagging ensured that the recorded actions were correctly associated with the corresponding features.

3. **Manual Data Collection and Heuristic Analysis** In addition to automated data collection, manual data collection was also employed and heuristic analysis was used. The IBM Watson Workspace application was manually used in a systematic manner, guided by the developer documentation. Each feature was explored carefully described in the documentation, performing various actions. During this exploration, actions were manually recorded. Additionally, application logs generated during this period were collected and analyzed. Heuristic analysis was used to identify actions related to specific features based on patterns observed in user behavior. For example, if a user frequently accessed related functions in quick succession, it was inferred that these actions were associated with a particular feature.

4. **Mapping Actions to Features** Once the data was collected and analyzed, process of mapping actions to features was performed. This involved aggregating the data from action logs, tags, and manual recordings. The feature-action map was created by mapping the identified actions to the corresponding features. An example of this mapping is provided in JSON format in Appendix A.

The creation of a feature-action map for IBM Watson Workspace involved a combination

[4] https://marketingplatform.google.com/about/analytics/

[5] https://mixpanel.com/

of action logging and heuristic analysis. By leveraging tools like New Relic for monitoring and implementing a comprehensive tagging strategy, the development team was able to accurately track and analyze user interactions. This detailed mapping provided valuable insights into how users engage with different features, informing design and development decisions to enhance user experience.

- **Identify usage data sources**: The process of identifying and selecting the primary usage data source for IBM Watson Workspace is critical to ensure the accuracy and comprehensiveness of user interaction data. This process involves evaluating various data collection tools, considering their capabilities, and determining how well they meet the application's specific requirements. The following detailed explanation outlines the steps taken to select the usage data source.

  1. **Understanding Requirements and Objectives:** The first step in selecting a usage data source is to clearly define the requirements and objectives of data collection. This involves understanding the key questions that the data should answer, such as user behavior patterns, feature usage, and performance issues. According to Preece et al. (2015), the success of any HCI research or evaluation method depends significantly on how well the initial requirements are understood and articulated. For IBM Watson Workspace, the objectives included tracking user interactions, identifying performance bottlenecks, and understanding the context of user actions. The requirements specified the need to capture data such as user IDs, device types, action names, timestamps, session details, and error occurrences.

  2. **Exploring Available Monitoring Solutions:** The next step involved exploring various monitoring solutions available for cloud-based applications. Effective monitoring solutions must provide comprehensive data collection capabilities, real-time analytics, and seamless integration with existing systems. As highlighted in Kesavulu et al. (2018a), monitoring tools for cloud environments often focus on infrastructure or application-level metrics, but there is a growing need for user-level monitoring to capture detailed interaction data. This involves tracking user actions such as navigation patterns, feature usage, and error occurrences across different interfaces. Kesavulu et al. (2018a) categorize monitoring tools into three broad categories:

     - **Infrastructure Monitoring Tools:** Focus on resource utilization, server performance, and uptime metrics, providing insights into the operational health of the system.

- **Application Monitoring Tools:** Measure application-specific metrics such as response times, transaction throughput, and error rates.
- **User-level Monitoring Tools:** Emphasize capturing granular user interaction data, such as feature usage and session behavior, which are essential for understanding user engagement and improving feature prioritization.

For this research, several tools were evaluated based on their ability to support user-level monitoring. Notable examples include:

- *GraphQL*[6]*:* Known for its flexibility and efficiency in querying data, GraphQL allows for precise data fetching tailored to gather specific user interaction metrics. Its ability to query nested data structures makes it particularly useful for capturing complex user behaviors.
- *New Relic*[7]*:* Offers comprehensive application monitoring, real-time data collection, and robust analytics capabilities, enabling the tracking of both application performance and user interactions. Its user-centric features make it suitable for understanding feature usage and session dynamics.

These tools were selected and analyzed for their ability to align with the research objectives, ensuring that the data collected would be both actionable and relevant for usage analytics.

3. **Evaluating Each Tool's Capabilities:** After identifying potential tools, the team conducted a detailed evaluation of each tool's capabilities. This involved setting up trial versions of the tools and testing their data collection, integration, and reporting functionalities. GraphQL was evaluated for its ability to provide flexible and efficient data querying. It allows developers to specify exactly what data they need, reducing the amount of unnecessary data transfer and improving the efficiency of data collection processes. GraphQL's flexibility makes it a powerful tool for customizing data queries to fit specific analytical needs. New Relic, on the other hand, provided a comprehensive suite of monitoring tools that included application performance management, real-time analytics, and detailed logging capabilities. It could seamlessly

[6] https://graphql.org/
[7] https://newrelic.com/platform

integrate with the existing infrastructure of IBM Watson Workspace, making it easier to deploy and use effectively.

4. **Aligning with Data Requirements** The final selection was heavily influenced by how well each tool aligned with the pre-defined data requirements. The chosen tool needed to collect data on:*User Identification (C1):* Capturing unique user identifiers while ensuring privacy. *Device Type (C2):* Recording the type of device used for accessing the application. *Action Details (C3):* Logging specific actions performed by the user. *Timestamp (C4):* Recording the time when each action was performed. *Action Duration (C5):* Measuring the duration of each action. *Error Tracking (C6):* Logging any errors encountered during user interactions. New Relic was able to meet all these requirements effectively. It provided detailed log entries that included user ID, timestamps, device type, session information, action names, and error details. This alignment ensured that the data collected would be comprehensive and useful for analyzing user behavior and application performance.

5. **Making the Final Decision:** Based on the evaluations, New Relic was selected as the primary usage data source for IBM Watson Workspace. The decision was supported by its comprehensive feature set, ease of integration, and ability to provide real-time insights into both user interactions and application performance. This decision-making process is supported by the principles of usability engineering and empirical measurement discussed by Nielsen (1994), which emphasize the importance of choosing tools that can provide reliable and actionable data. Furthermore, the iterative and user-focused approach aligns with the user-centered design principles described by (Norman and Draper, 1986).

In this stage, the data sources of the usage data are identified. Ideally, the data sources that contain information regarding the actions performed by the users with the application (referred to as data type in Figure 5.5) such as who performed the action (C1), which device type was used to interact with the application (C2), what action was performed (C3), when was the action performed (C4), what is the duration of the action performed (C5), any errors occurred during the actions performed by the user (C6) are considered as usage data sources. The log entries of New Relic are treated as the primary data source. New Relic collects various information about the Watson Workspace application such as user

ID, time stamp, device type, session, action name, app version etc. A subset of the data is used as usage data, namely, time stamp, user ID (pseudo ID will be assigned to each user), action name, device type, session, and user agent OS. The updated version 2 of the usage analytics experiment design includes the data sources identified in this stage.

- **Data extraction**: After identifying the primary data source for usage data, the next crucial step is to extract the relevant data. This process involves utilizing New Relic's capabilities to access and retrieve the necessary information. The steps and methods used in this stage are detailed below. The process of data extraction begins with identifying the specific data types needed for analysis. For IBM Watson Workspace, these data types include user identification, device type, action details, timestamps, action durations, and error tracking. These elements are essential for constructing a comprehensive view of user interactions. To retrieve this data from New Relic, a customized query language called New Relic Query Language (NRQL) is employed. NRQL is specifically designed to interact with the New Relic database, allowing users to write queries that can retrieve detailed and specific data. Designing NRQL queries involves creating a set of instructions tailored to fetch the identified data types. An example of such a query is:

```
SELECT * from PageAction, MobileAction WHERE USER_ID =
'ID_OF_THE_USER' SINCE this quarter
```

This query functions by retrieving all records from the PageAction and MobileAction tables in New Relic. These tables contain data about user interactions with the application, both from web pages and mobile devices. The WHERE USER_ID = 'ID_OF_THE_USER' condition filters the data to include only the actions performed by a specific user, identified by USER_ID. This helps in isolating the user's actions for detailed analysis. The SINCE this quarter clause limits the data retrieval to actions performed within the current quarter, ensuring that the data is recent and relevant. The attributes PageAction and MobileAction in New Relic store information about user interactions. The PageAction attribute tracks actions performed on web pages, while the MobileAction attribute captures actions performed on mobile devices. Together, they provide a comprehensive view of user interactions across different platforms. The USER_ID attribute uniquely identifies each user, enabling personalized analysis by filtering actions based on USER_ID. Limiting the data extraction to the current quarter ensures that the analysis focuses on recent user behavior. This time frame can be adjusted as needed to examine different periods.

Implementing the data extraction process involves several steps. First, the team designs a series of NRQL queries to cover all necessary data points. These queries are tested and refined to ensure they retrieve accurate and comprehensive data. Once the queries are finalized, they are executed against the New Relic database. This involves running the queries within New Relic's interface or using its API to automate data extraction. The extracted data is then collected and stored in a suitable format for analysis. This may involve exporting the data to CSV files, databases, or data lakes, depending on the volume and structure of the data. After the data is collected, it is verified for accuracy and completeness. This step is crucial to ensure that the data accurately reflects user interactions and does not contain any gaps or errors. The verification process may involve cross-checking the extracted data with known user interactions or using automated validation scripts to identify inconsistencies.

The extraction of usage data from New Relic using NRQL is a critical step in analyzing user interactions with IBM Watson Workspace. By designing precise queries and leveraging New Relic's powerful data retrieval capabilities, the development team can collect comprehensive and relevant data. This data forms the foundation for detailed user behavior analysis, informing decisions to enhance the application and improve user experience.

- **Analytics**: After the data extraction phase, the next critical step is the analytics stage. This stage involves performing statistical analysis on the extracted logs from New Relic to derive meaningful insights into user behavior and feature usage within IBM Watson Workspace. The following elaboration details the steps and methodologies used in the analytics stage. The analytics stage focuses on interpreting the usage data extracted from New Relic. This involves employing various statistical metrics to analyze how users interact with different features of the application. The key metrics used in this analysis include frequency, timespent, and consistency.

  The frequency metric refers to the number of times a specific action is performed by an individual user or a group of users over a defined period. This metric helps identify the most and least used features within the application. For instance, if a particular feature is used frequently by a large number of users, it indicates the feature's popularity and importance. Conversely, features with low usage frequencies may need to be re-evaluated for their relevance or usability. To calculate frequency, the extracted data is aggregated to count occurrences of each action.

Time Spent refers to the total amount of time users spend interacting with a specific feature within an application. This metric is essential for understanding user engagement, as it provides insights into the depth and duration of user interactions. Features with high Time Spent values may indicate critical functionality, whereas disproportionately high Time Spent could signal usability issues, such as inefficiencies or unnecessary complexity. Conversely, features with low Time Spent may require further investigation to determine whether they lack relevance or suffer from poor discoverability. The Time Spent metric is calculated by summing the duration of all actions associated with a particular feature. For example, if a feature involves multiple user actions, such as clicking a button and navigating through subsequent screens, the total time across these actions is aggregated to determine the feature's overall Time Spent. This method provides a holistic view of how much effort users dedicate to specific functionalities within the application. Using *average time per user* as a metric, while seemingly intuitive, is often not as effective for this type of analysis. Average time values can be heavily influenced by outliers, such as users who either abandon a feature immediately or spend an unusually long time due to distractions or external factors unrelated to the application itself. Additionally, average time fails to account for the volume of interactions. A feature used frequently by many users with moderate Time Spent values could appear less significant than a rarely used feature with high average times. This discrepancy can lead to misinterpretation of a feature's actual importance or relevance. The total Time Spent metric avoids these pitfalls by emphasizing cumulative engagement rather than individual variations. It reflects the overall workload a feature generates for users and helps identify trends at a broader scale. This approach aligns with the objectives of the research by providing actionable insights into feature prioritization, ensuring that decisions are based on meaningful, aggregate patterns rather than potentially misleading averages.

Consistency measures whether users exhibit similar patterns in accessing a feature over time, especially after changes are implemented by developers. Consistency is important for assessing the impact of feature updates and ensuring that changes do not negatively affect user behavior. To analyze consistency, the team compares usage patterns before and after changes are made to a feature. This involves calculating the frequency and timespent metrics for defined periods before and after the update. Statistical tests, such as the paired t-test, can be employed to determine if the differences in usage patterns are statistically significant.

5.3.3 Result - Improvement and Implementation of the Usage Analytics Method - Version 2

The implementation of the usage analytics method in IBM Watson Workspace revealed several challenges related to prioritization of software features and led to the improvement of Usage Analytics (UA) Method (Version 2) as shown in Figure 7.2. Feedback from the development team helped refine the method further. This case study directly addresses RQ2: *What are the key challenges faced by the developers to identify and utilize usage data and key metrics related to the feature prioritization effectively within the software platform?*.

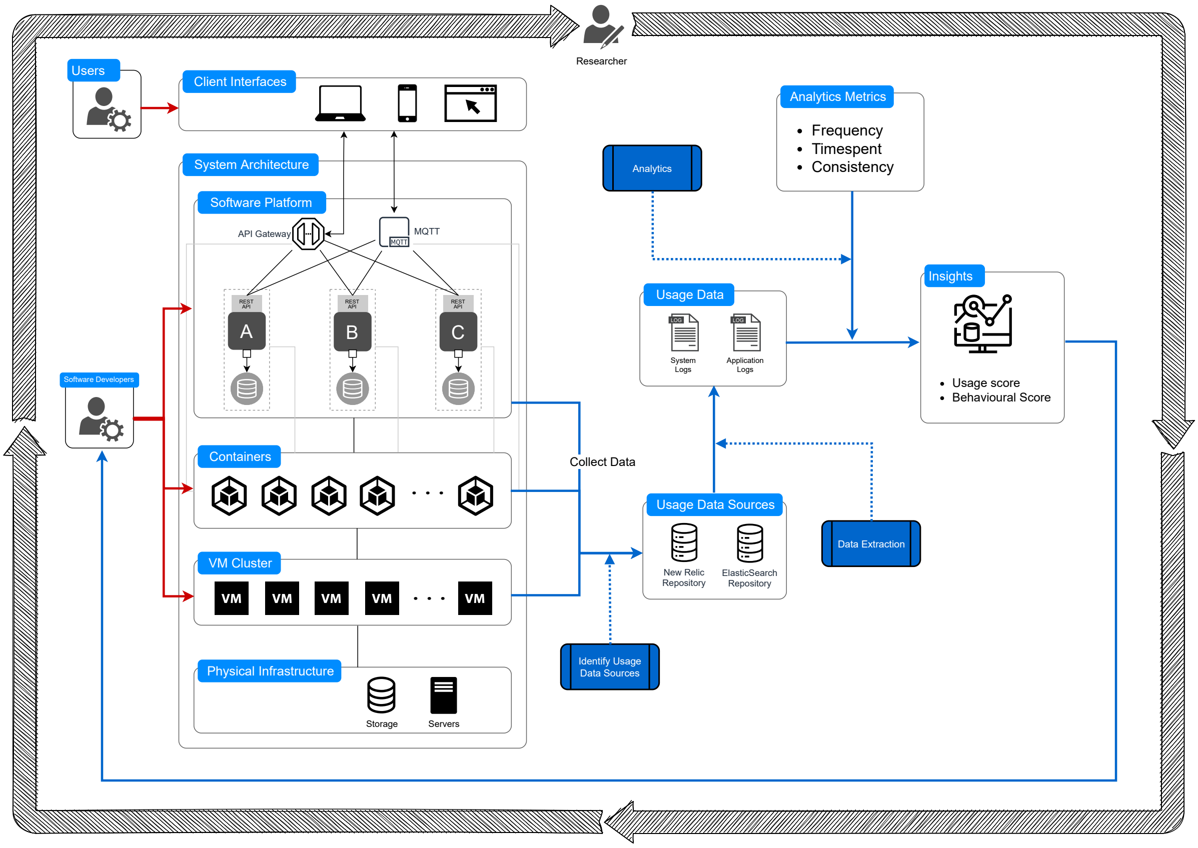


Figure 5.6: Usage Analytics method - Version 2 resulting from the work with IBM Watson Workspace Application

The updated Usage Analytics method was applied to IBM Watson Workspace with the following key observations and results:

1. **Usage Data Identification and Extraction:** Various sources of usage data were identified, including native and third-party monitoring tools. This provided a comprehensive view of user interactions with the application. This included the addition of a new stage in the Usage Analytics Method version 2, "Identify Usage Data Sources". This stage involves systematically exploring the application architecture to pinpoint where relevant data is being collected, whether from in-built monitoring tools or third-party services. This stage ensures that all potential sources of user interaction data are considered, thereby providing a more comprehensive understanding of user behavior. According to Preece et al. (2015), the success of any HCI research or evaluation method depends significantly on how well the initial requirements are understood and articulated. For IBM Watson Workspace, the objectives included tracking user interactions, identifying performance bottlenecks, and understanding the context of user actions. The requirements specified the need to capture data such as user IDs, device types, action names, timestamps, session details, and error occurrences. The selected tools, such as GraphQL and New Relic, were evaluated based on their capabilities to meet these requirements. Ultimately, New Relic was chosen for its comprehensive feature set, ease of integration, and real-time insights into user interactions and application performance.

2. **Application of Analytics Metrics:** Frequency and timespent analysis were conducted for actions performed by users, providing insights into how often and how long users engaged with different features of the application. Consistency of usage was also analyzed by comparing usage patterns before and after updates, identifying significant changes in user behavior.

    (a) How many times has a user performed an action? - **Frequency**: calculated by counting the number of occurrences of action in the usage data and filtering these occurrences with the user ID (for a specific user) or combining multiple user IDs (for a group of users). The calculation is further filtered between a specific range in time, ideally, in the software development domain, the time range is set to the amount of time a specific version of the application was made available to the users of the application. Every time a new version of the application is released, the frequency of the actions performed by the users is calculated. The changes made to the application by the developers are compared with the frequency of the actions performed by the users. The changes in the frequency of the actions performed by the users are correlated with the changes made to the application by the developers.

(b) How much time does a user spend on an action? - **Timespent**: calculated by summing up the duration of all the occurrences of action in the usage data and filtering these occurrences with the user ID (for a specific user) or combining multiple user IDs (for a group of users). The calculation is further filtered between a specific range in time, ideally, in the software development domain, the time range is set to the amount of time a specific version of the application was made available to the users of the application. Every time a new version of the application is released, the timespent of the actions performed by the users is calculated again. The changes made to the application by the developers are compared with the timespent of the actions performed by the users. The changes in the timespent of the actions performed by the users are correlated with the changes made to the application by the developers.

(c) Do users perform the actions in the same way as before? - **Consistency**: calculated by combining the frequency and timespent of the actions performed by the users. Then, check the order of the occurrences of the actions in the usage data and filter these occurrences with the user ID (for a specific user) or combining multiple user IDs (for a group of users). The calculation is further filtered between a specific range in time, ideally, in the software development domain, the time range is set to the amount of time a specific version of the application was made available to the users of the application. Every time a new version of the application is released, the consistency of the actions performed by the users is calculated again. The changes made to the application by the developers are compared with the consistency of the actions performed by the users. The changes in the consistency of the actions performed by the users are correlated with the changes made to the application by the developers.

3. **Development of Feature-Action Map:** The integration of the process to develop a feature-action map was a significant result of the case study. This process is part of the "Identify Features of the Application" stage and involves feature identification, action mapping, data correlation, visualization, and validation, providing a structured approach to understanding user interactions with specific features.

Challenges in Implementation of the Analytics Method

Despite the clear benefits, implementing advanced usage analytics also presents several challenges. One major challenge is the integration with multiple data sources, including inbuilt monitoring tools and third-party services. This requires significant effort to ensure that data is collected consistently and accurately across different platforms. Additionally, the process of mapping user actions to specific features and correlating this data with usage metrics requires careful planning and execution to ensure that the results are meaningful and actionable.

Another challenge is the need for ongoing validation and refinement of the feature-action map. As user behaviors and preferences evolve, the development team must continuously monitor and update the map to ensure that it remains accurate and relevant. This requires a commitment to iterative testing and feedback loops, as well as the ability to quickly adapt to changes in user behavior.

Application of the updated version 2 of the Usage Analytics based on the findings in Case Study 2 to Odoo Notes application is discussed in the next section in detail.

5.4 Case Study 3: Odoo Notes

As discussed in Chapter 4, Odoo Notes is a note-taking and task management application that allows users to create, edit, and organize notes and tasks. It features collaborative capabilities, enabling multiple users to work on the same notes and tasks simultaneously. User interactions in Odoo Notes involve various actions such as creating, editing, and deleting notes and tasks, adding comments, tagging other users, and collaborating in real-time. These interactions generate a wealth of usage data that can be analyzed to understand user behavior and preferences.

5.4.1 Goal - Implementation and validation of the Usage Analytics method

The primary goal of this case study was to implement and validate the usage analytics method developed in Case Study 2, focusing on the key metrics identified earlier (frequency, time spent, and consistency). This case study aimed to demonstrate how the usage analytics method can be applied to a different software platform, Odoo Notes, to address *RQ3: How can activities be systematically structured to identify and utilize usage data for feature prioritization?*

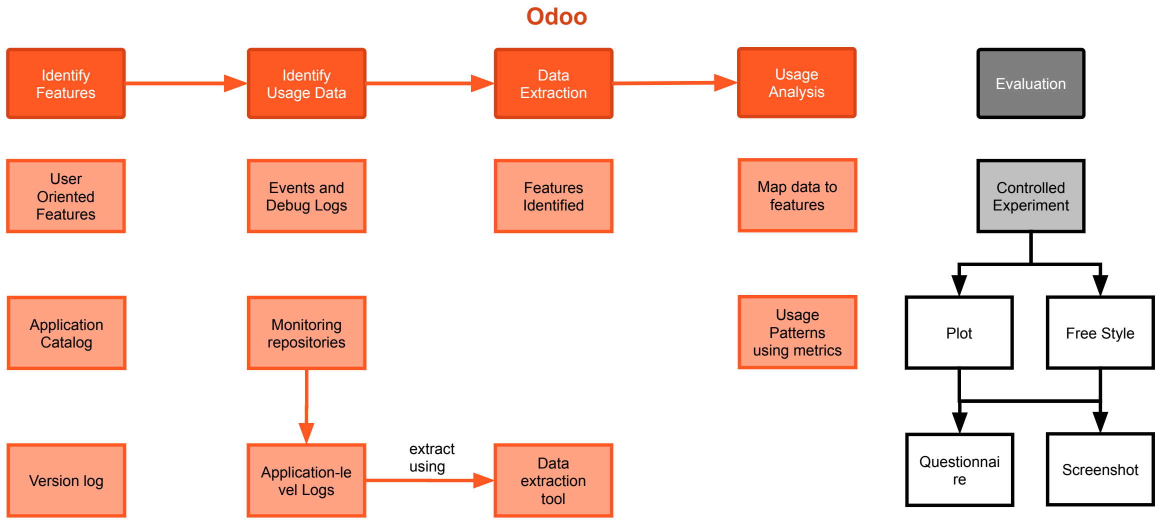


Figure 5.7: Use Case 3: Odoo Notes. Updated activities of the usage analytics method applied to the Odoo Notes application

5.4.2 Design - Development of a custom Usage Data monitoring tool for Odoo Notes application and application of the UA method

The default logging mechanisms in Odoo Notes are primarily focused on basic system logs, capturing events such as note creation, updates, deletions, and user logins. These logs provide a limited view of user interactions and are not sufficient for detailed usage analytics required for feature prioritization. The application lacks built-in monitoring tools capable of capturing comprehensive user interaction data.

The development of a custom monitoring tool for Odoo Notes was essential due to several critical reasons:

1. **Lack of Built-in Monitoring Tools:** Many applications, including Odoo Notes, do not come with built-in tools capable of capturing detailed and comprehensive user interactions. The absence of such tools necessitates the development of a custom solution tailored to the specific needs of the usage analytics method. Studies have shown that off-the-shelf monitoring solutions often lack the flexibility required for detailed analytics, as they may not capture all relevant user actions or provide the necessary granularity.

2. **Limitations of Third-Party Tools:** While third-party tools offer extensive tracking capabilities, they may not fully align with the specific requirements of the usage analytics

method developed in this research. These tools often focus on high-level metrics and general user behavior patterns, which may not be sufficient for the detailed and context-specific analysis needed for feature prioritization. For example, third-party tools might not capture the exact actions within a specific feature set, leading to incomplete or skewed data.

3. **Customization and Integration:** A custom monitoring tool allows for complete control over the data collection and processing pipeline. This customization ensures that all necessary data points are captured accurately and are relevant to the usage analytics method. It allows developers to tailor the tool to the specific features and interactions of Odoo Notes, providing a more accurate representation of user behavior. This level of customization is critical for implementing the usage analytics method accurately and effectively.

4. **Data Privacy and Security:** Custom tools can be designed to comply with specific data privacy and security requirements of the organization or application. Third-party tools might pose risks related to data ownership, compliance with data protection regulations, and security vulnerabilities. By developing an in-house solution, these concerns can be mitigated, ensuring that user data is handled securely and in compliance with relevant laws and standards.

5. **Consistency with Usage Analytics Method:** The usage analytics method developed in this research requires specific data points and metrics that may not be supported by existing tools. A custom solution ensures that the data collected conforms to the method's requirements, enabling precise and meaningful analysis. This alignment is crucial for validating the research hypotheses and achieving the research objectives.

A custom usage data monitoring tool was developed for the Odoo Notes application to facilitate the collection and analysis of detailed user interaction data. This tool was designed to capture various user actions within the application, such as creating notes, editing notes, and managing tasks. The collected data included timestamps, user IDs, action types, and other relevant metadata. The Odoo Notes application uses a third-party monitoring system for debugging purposes. The monitoring system is built using a Python version of the web application library called “werkzeug”[8]. The monitoring system generates log entries in the form

[8] https://werkzeug.palletsprojects.com/

of text files. The werkzeug tool is explored further to identify the log entry format and the information that can be extracted from the log entries. The log entries are in the form of text files. A new instance of the Odoo Notes Application is started and each action is performed to generate a sample log entry which includes the entries for all actions provided by the application. A snapshot of the sample log entry is shown in Log 5.1. The next task was to explore the log entries to identify the actions performed and the corresponding log generated by the monitoring system, a brief description of the log entry is assigned to each entry in the sample log file.

Log 5.1: Odoo Notes application log entries generated by the inbuilt monitoring system

```
2018-01-07 17:40:32,716 4338 INFO DCU_Experiment_1 odoo.models.unlink: User #6 deleted
    ↪  project.task records with IDs: [14]
//User deleted a project task

2018-01-07 17:46:16,937 4338 INFO DCU_Experiment_1 werkzeug: 192.168.0.116 - - [07/Jan
    ↪ /2018 17:46:16] "POST /web/dataset/call_kw/mail.channel/channel_fetch_preview
    ↪ HTTP/1.1" 200 -
//Open message (chat) window by user

2018-01-07 17:47:01,461 4338 INFO DCU_Experiment_1 werkzeug: 192.168.0.116 - - [07/Jan
    ↪ /2018 17:47:01] "POST /web/dataset/call_kw/mail.channel/message_post HTTP/1.1"
    ↪ 200 -
//Message sent by user (to another user)

2018-01-07 17:49:43,248 4338 INFO DCU_Experiment_1 odoo.addons.base.ir.ir_model:
    ↪ Access Denied by ACLs for operation: create, uid: 6, model: project.task.type
//User is denied access to create a new column in the project (only admin can do so)
```

The next step was to explore the docker Odoo source code to make it easier to patch Odoo and execute the source code of the application. In the Odoo dockerfile, the Odoo folder is moved to a different directory after installation.

For example, Odoo10 is installed in the location:

/usr/lib/python2.7/dist-packages/odoo

After the installation, we moved it to the location:

/usr/lib/python2.7/dist-packages/odoo.untouched

We set up docker-compose to start Odoo and the database it needs to store data and share the normal installation directory of Odoo with the host system (via volumes). Docker executes a bash script to start Odoo, we changed the script so it copies Odoo to the host copying it into the normal installation directory that is shared and ensuring that when Odoo halts the logs are copied into the host. By doing so we can work on Odoo source code and easily execute it. Studying Odoo source code we found that when a user executes an action that needs to pass through the server it passes through those two functions, in the api.py file:

```
def call_kw_multi(method, self, args, kwargs)
def call_kw_model(method, self, args, kwargs)
```

These functions act as a generic dispatch for the actions executed in the browser. The calls in this central dispatch are identified via strings: the method name and the object to call the method on. So we add a call to call_kw_multi and call_kw_model so that the action gets logged in an external CSV file.

A sample of the log entries from the Odoo Notes application is shown in Log 5.2. The first and second fields in all the log entries represent the time stamp (**C4**). The keywords *INFO, DEBUG* represent the level of logging, other possible options are *CRITICAL, ERROR and WARN* (**C6**). The fourth field represents the target data store, and a "?" symbol represents requests to load web pages and web scripts. The name of the database in this sample is "Odoo_Database" (**C2**). The next field shows the IP address of the user (or the person who made the request, for example, administrator of the application) (**C1**). The next field of importance is the HTTP request and response between the browser and the application server along with the response code (**C3**). One interesting fact to notice here is that Odoo application logs can be configured to capture the API calls which reveal a vast amount of information about features such as the object ID of the elements (for example, the ID of the tag is 4), the operation performed (for example, the last line of the log entry represents read operation with the variables - the user-defined name of the tag and the color assigned by the user to the note) (**C4**). Although what appears to be lacking in the log entries is the action duration (**C5**) characteristic of the usage data, it can be calculated by considering the timestamps of subsequent entries.

Log 5.2: Odoo Notes application sample log entries

```
2018-03-28 12:24:10,966 8970 INFO ? werkzeug: 136.206.48.84 - - [28/Mar/2018 12:24:10]
    ↪  "GET /web_editor/static/src/js/transcoder.js HTTP/1.1" 200 -
```

```
2018-03-28 12:24:11,256 8970 DEBUG Odoo_Database odoo.api: call ir.ui.view().
    ↪ read_template(u'web_editor.colorpicker')

2018-03-28 12:24:11,260 8970 INFO Odoo_Database werkzeug: 136.206.48.84 - - [28/Mar
    ↪ /2018 12:24:11] "POST /web/dataset/call HTTP/1.1" 200 -

2018-03-28 12:24:11,324 8970 INFO ? werkzeug: 136.206.48.84 - - [28/Mar/2018 12:24:11]
    ↪  "GET /web_editor/static/src/xml/editor.xml?debug=1522239851310 HTTP/1.1" 200 -
```

The first step in analyzing the usage data is to map the feature data to the log entries. This step involves identifying the keywords in the log entries that refer to the actions performed by the user to the keywords in the log entries. A mapping file is created to aid this purpose as shown in Log 5.3. The mapping file will help identify the keywords to search in the logs file for entries that reveal the actions performed by users. Google's colab tool[9] could be used for identifying the moments at which a specific user performed a specific task. Colab tool is built on Jupyter notebook[10], which is an open-source web application used for data cleaning and transformation, numerical simulation, statistical modeling, data visualization, machine learning, and so on. Keywords identified from the mapping file are used to search the log. Google's Colab tool[11] could be utilized for analyzing interaction logs to identify the specific moments at which a user performed a particular task. By leveraging its Python-based environment and integration with powerful data analysis libraries such as *pandas*, *NumPy*, and *matplotlib*, Colab can process timestamped log data to extract actionable insights.

For example, interaction logs containing user IDs, timestamps, and action descriptions can be imported into Colab for analysis. Using Python scripts, the logs can be filtered by user ID to isolate data for a specific individual. Time-based queries can then identify the exact moments when the user performed a specific action, such as accessing a feature, clicking a button, or completing a task.

Additionally, Colab's ability to integrate with external cloud storage systems (e.g., Google Drive or AWS S3) allows for seamless access to large-scale datasets. By combining this functionality with visualization libraries like *matplotlib* or *seaborn*, Colab can generate timelines or activity heatmaps to provide a clear depiction of user interactions over time. This makes it a

[9] https://colab.research.google.com
[10] http://jupyter.org
[11] https://colab.research.google.com

versatile tool for identifying and analyzing user behavior patterns.

Log 5.3: A snapshot of feature mapping

```
12 Create a new note
2019-03-20 15:39:23,261 1 DEBUG odoodb odoo.api: call note.note().create({u'
    ↪ message_follower_ids': False, u'stage_id': False, u'tag_ids': [], u'memo': u'<p
    ↪ >A New note...<br></p>', u'message_ids': False})

13 Delete a note
2019-03-20 15:40:20,772 1 INFO odoodb odoo.models.unlink: User #1 deleted note.note
    ↪ records with IDs: [33]

23 Create a new stage
2019-03-20 15:39:46,139 1 DEBUG odoodb odoo.api: call note.stage().name_create(u'A new
    ↪  stage...')
```

The keywords in the odoo notes application logs are generally found in in the HTTP POST request and API call fields. The action *Create a new note* generates on the log an entry with the keyword "note.note().create." *Delete a note* can be identified using the keyword "odoo.models.unlink" and the log entry also provides the ID of the note deleted. In this sample log, the ID of the note deleted is "33". Similarly, *Create a new stage* can be identified by the keyword "note.stage().name_create". While these keywords are simple to recognize by their names, some keywords are difficult to identify. For example, the log entry for the *Open comment box in note* does not use keywords such as comment or open in the HTTP request. However, the keywords "note.note/message_get_suggested_recipients" reveal that a comment box was opened by the user as the application loaded the message recipient suggestions. Hence, a mapping file is carefully created by anticipating the actions performed and understanding the programming logic of the application. The feature map resulting from this activity serves as a reference for analyzing the extracted usage data, exploring the usage metrics and identifying the usage patterns exhibited by the users, which is part of the future work of this research. Once understood when and how these log entries are generated, the next step is to create machine-readable log entries for the analysis. Odoo logs are difficult to read programmatically, but since Odoo is open-source software, we can patch the code and write the logs we need. Yet, our technique can also be applied to closed-source software, but we would need to convert the existing log text. To find the strings and how the parameters are passed we logged on to standard output and conducted some trial experiments. Here are a few examples:

Create Stage < − > name_create.note.stage

Delete Stage < − > unlink.note.stage

Rename Stage < − > write.note.stage

The resulting CSV files are the results of the changes made to the source code of the application described above. If the users do multiple experiments, the results from the same user are concatenated. It is important to underline that from the point of view of the user none of the changes alters the observable behavior of Odoo.

Log 5.4: Log entries generated by the custom usage data monitoring tool patched to Odoo application

```
timestamp,actionName,odoo-version,userId,ipAddr,id,stageId,partnerId,text
2019-09-21 20:04:00.351600,Open Note,10,1,172.19.0.1,29, , ,
2019-09-21 20:04:25.705395 ,Create new tag , 10 , 1 ,172.19.0.1 , , , ,competition
2019-09-21 20:04:38.593585 ,Change Tag Colour , 10 , 1 ,172.19.0.1 , 1 , , ,
2019-09-21 20:04:45.610882 ,Edit Note , 10 , 1 ,172.19.0.1 ,29 , , ,<p>Look for
    ↪ competing products in the market</p>
2019-09-21 20:04:53.357031 ,Change Note Colour , 10 , 1 ,172.19.0.1 ,29 , , ,
2019-09-21 20:04:57.083195 ,Change Note Colour , 10 , 1 ,172.19.0.1 ,29 , , ,
2019-09-21 20:04:59.886629 ,Open Note , 10 , 1 ,172.19.0.1 ,29 , , ,
2019-09-21 20:05:22.279496 ,Comment in Note , 10 , 1 ,172.19.0.1 , , , ,Look for start
    ↪ -ups in particular.
```

The Log 5.4 is a snapshot of the log entries generated by the custom usage data monitoring tool developed as part of this research to demonstrate the ability to collect usage data with applications that rely on third-party tools for the data extraction stage of the Usage Analytics method.

5.4.3 Results - Improvement of the Usage Analytics Method - Version 3

Application of version 2 of the Usage Analytics method and associated experiment design resulted in the following findings, the Usage Analytics method was further updated (version 3) based on the findings shown in Figure 7.3.

The findings from the work with the Odoo Notes application are as follows:

1. The existing monitoring tools (native or third-party) may not be able to extract the

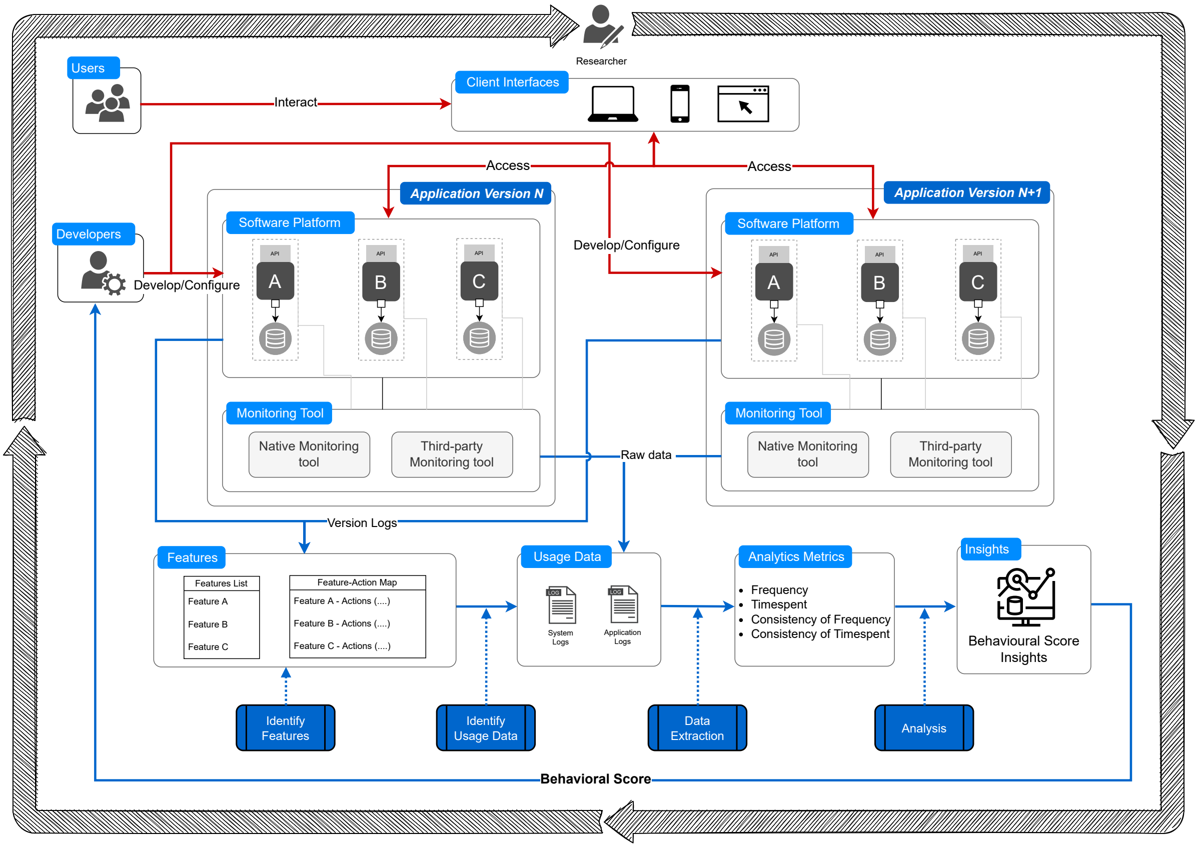


Figure 5.8: Usage Analytics method - Version 3 as a result of work with Odoo Notes

usage data from the software application as they are designed to monitor and analyze the software performance, bug detection and maintenance. These existing tools do not differentiate between actions performed by the user and the actions performed by the system. To analyze and understand usage behavior of the users, the data extracted from the monitoring tools need to be filtered to extract only the actions performed by the users alone and not the actions performed by the system. As a result, the Usage Analytics method needs to identify features of the application a user can interact with and all actions associated with these features. The Usage Analytics method is updated to include *Identify Features* as a step in the method. The features are identified by analyzing the application and identifying the actions that can be performed by the user. The features are then mapped to the actions performed by the user. The mapping is done by analyzing the log entries generated by the application. The log entries are analyzed to identify the actions performed by the user. A mapping file is created to map the actions performed by the user to the features of the application. The mapping file is used to filter the usage data extracted from the monitoring tool to extract only the actions performed by the user. The mapping file is termed the *Feature-Action Map.* The Feature-Action map file can be in CSV or JSON format. In this research, the Feature-Action map file for the Odoo Notes application is in JSON format and can be accessed using the link in Appendix A. The mapping file can be created by the developer of the application by analyzing the source code of the application and identifying the features of the application and the actions associated with these features.

2. Software applications may not always have a monitoring tool that can be used to extract the usage data. In such cases, the usage data can be extracted from the log files. However, the log files may not be in a format that can be used for analysis. In such cases, the logger tool should be modified to generate the required log entries.

3. The Usage Analytics method is further updated to include the application of the method to a minimum of two consecutive versions of the application. The application of the method to two consecutive versions of the application is done to identify the changes in the usage behavior of the users. Let us say, we have an application version N and the next version of the application is $N + 1$. Ideally, for the analysis to yield optimum results the Usage Analytics method should be applied continuously across the lifetime of the application. For the purpose of demonstrating the Usage Analytics method, the application of the method to two consecutive versions of the application is sufficient. The recommendation for the software developers who would use this method to understand the usage behavior of the users is to apply the method to all the versions of the application. If the changes are small and implemented frequently, the developers can choose a specific time period and apply the method to all the versions of the application released during that time period.

4. The *Consistency* metric was updated to include a different way to measure the consistency of the usage behavior of the users. The *Consistency* metric is updated to include the following steps:

    (a) A complementary metric to the Consistency metric, the *order of actions* performed by users across different versions of the application, tends to appear random over shorter time periods. This metric requires users to engage with the application for a significant duration to reliably identify the typical order of actions they follow while interacting with its features. Due to time constraints, the *order of actions* metric was not included in the experiment design. However, it presents an opportunity for future work to enhance the *Consistency* metric by incorporating this additional dimension. Various methods, such as Jaccard similarity and cosine similarity, can be used to quantify the *order of actions* performed by users. Changes in the *order of actions* scores across application versions can then be analyzed using these similarity measures.

(b) Consistency measure is calculated by combining the metrics *frequency of actions* and *time spent on actions* performed by the users between two consecutive versions of the application. The metrics are termed as *Consistency of Frequency* and *Consistency of Timespent* respectively.

The resulting version 3 of the Usage Analytics method is applied again to the Odoo Notes Application. The final result based on the work with the above three case studies is a Usage Analytics method that can be applied to any software product. A detailed description of the method and the experiment design are described in the next chapter.

5.5 Summary of Challenges in Developing Usage Analytics Method

Understanding the challenges encountered during the development of the Usage Analytics Method (UAM) is crucial for identifying areas of improvement and refining the methodology to ensure its applicability across diverse platforms. This section examines the obstacles uncovered in three case studies: IBM Academic Cloud, IBM Watson Workspace, and Odoo Notes. By analyzing these challenges, the research aims to provide a clearer picture of the practical difficulties faced by developers and how these barriers influenced the iterative refinement of UAM. The findings not only shed light on the complexities of data-driven feature prioritization but also highlight the critical need for adaptable and robust analytical frameworks to address these issues effectively.

5.5.1 Challenges in IBM Academic Cloud (Case Study 1)

The IBM Academic Cloud case study uncovered several foundational challenges that underscored the complexities of implementing a robust Usage Analytics Method (UAM). One of the most pressing issues was the absence of a comprehensive feature-action map, which hindered the developers' ability to effectively analyze user behavior. This gap in clarity meant that mapping user interactions to specific features was often inconsistent or incomplete, making it difficult to derive actionable insights from the data. Developers highlighted how this lack of structure often led to confusion and misaligned priorities when interpreting user behavior patterns.

Key insights from the IBM Academic Cloud study highlight the significant bottlenecks and challenges in aligning user data with feature prioritization. Developers struggled with unclear feature definitions that hindered effective mapping of user actions to platform features. This was particularly evident when attempting to correlate user behavior with actionable feature insights, where gaps in definitions led to incomplete data interpretations.

Another significant challenge was the resource-intensive nature of data preparation. The processes of selecting, cleaning, and preparing data for analysis required substantial manual effort, often resulting in delays in development timelines. These bottlenecks not only consumed valuable time but also introduced opportunities for errors, which further compounded the difficulty of drawing accurate conclusions from the data. The reliance on manual intervention emphasized the need for more automated and streamlined approaches to data processing.

Additionally, the design of analytics tools and processes relied heavily on broad assumptions due to the limited understanding of feature-specific interactions. Without a detailed framework to guide the development of these tools, the analytics relied on generalized assumptions that often failed to capture the nuanced behaviors exhibited by users. This lack of precision reduced the effectiveness of the tools in providing targeted and relevant insights, thereby limiting their utility in driving feature prioritization and system improvements.

5.5.2 Challenges in IBM Watson Workspace (Case Study 2)

The IBM Academic Cloud case study uncovered several foundational challenges that underscored the complexities of implementing a robust Usage Analytics Method (UAM). One of the most pressing issues was the absence of a comprehensive feature-action map, which hindered the developers' ability to effectively analyze user behavior. This gap in clarity meant that mapping user interactions to specific features was often inconsistent or incomplete, making it difficult to derive actionable insights from the data. Developers highlighted how this lack of structure often led to confusion and misaligned priorities when interpreting user behavior patterns.

Key insights from the IBM Academic Cloud study highlight the significant bottlenecks and challenges in aligning user data with feature prioritization. Developers struggled with unclear feature definitions that hindered effective mapping of user actions to platform features. This was particularly evident when attempting to correlate user behavior with actionable feature insights, where gaps in definitions led to incomplete data interpretations.

Another significant challenge was the resource-intensive nature of data preparation. The processes of selecting, cleaning, and preparing data for analysis required substantial manual effort, often resulting in delays in development timelines. These bottlenecks not only consumed valuable time but also introduced opportunities for errors, which further compounded the difficulty of drawing accurate conclusions from the data. The reliance on manual intervention emphasized the need for more automated and streamlined approaches to data processing.

Additionally, the design of analytics tools and processes relied heavily on broad assumptions due to the limited understanding of feature-specific interactions. Without a detailed framework to guide the development of these tools, the analytics relied on generalized assumptions that often failed to capture the nuanced behaviors exhibited by users. This lack of precision reduced the effectiveness of the tools in providing targeted and relevant insights, thereby limiting their utility in driving feature prioritization and system improvements.

5.5.3 Challenges in Odoo Notes (Case Study 3)

The challenges encountered in the Odoo Notes case study stemmed from the unique requirements of working within an open-source software platform. Odoo Notes, as part of its community edition, provided full access to the source code, enabling extensive customization and integration with the UAM. However, several significant obstacles shaped the development and refinement of the UAM for this platform.

A key challenge was the need for a custom monitoring tool. The default logging mechanisms in Odoo Notes primarily focused on system-level logs, such as note creation and deletion. These were insufficient for capturing detailed user interaction data necessary for comprehensive analytics. Consequently, a specialized monitoring tool was developed to extract relevant data points and metrics, ensuring alignment with the study's objectives.

Another critical issue was the absence of predefined feature definitions. There was no access to a development team to verify or clarify whether any definitions existed. This challenge highlighted the broader problem of inconsistent feature definitions in software development. To address this, the study proposed a standardized process for defining features, which became a key contribution to knowledge in this domain. This process involved systematically mapping user interactions to well-defined features, ensuring clarity and consistency across iterations.

The lack of a feature-action map further compounded the difficulties. Without an existing map, the study had to create one from scratch, relying on iterative refinement and careful observation of user behaviors. This highlighted the importance of establishing robust connections between features and user actions for deriving actionable insights.

Analyzing multiple versions of the application introduced another layer of complexity. Initially, the UAM scope included only a single version of software. However, with Odoo Notes, comparing interactions across versions 10 and 11 revealed deeper behavioral patterns. This analysis led to the introduction of the "comparative behavioral score," a new metric for evaluating changes in user engagement and feature adoption over time, which added a valuable dimension to UAM.

The scope of participants and tasks limited the generalizability of the findings. While the structured experiments provided focused insights, the participant pool was relatively narrow, and task design did not capture the full spectrum of possible user interactions. This underscored the importance of expanding participant diversity and task breadth to strengthen future analyses.

Despite these challenges, the Odoo Notes case study demonstrated the adaptability and effectiveness of the UAM in open-source environments. The custom monitoring tool, standardized feature definitions, and longitudinal analysis using comparative behavioral scoring highlighted the potential for extending UAM to similar platforms while addressing limitations through iterative refinement.

5.6 Key Challenges for Feature Prioritization

This chapter provides an extensive analysis of the challenges encountered by the software developers during the feature prioritization process and their critical relevance to addressing Research Question 2 (RQ2): "What are the key challenges faced by developers in identifying and utilizing usage data and key metrics related to feature prioritization effectively within the software platform?" By identifying and deeply contextualizing key obstacles such as the absence of feature-action maps, inconsistent feature definitions, and the integration of analytics across multiple versions, this chapter lays a comprehensive foundation for understanding how these challenges shaped the iterative refinement of usage analytics methodologies. These obstacles not only presented immediate issues but also catalyzed innovations that enriched the overall approach, ensuring that the UAM evolved to meet diverse and complex demands.

The discussion in this chapter highlights how the absence of feature-action maps forced the research to develop an adaptable framework capable of bridging gaps in user interaction analysis. This requirement underscored the necessity of a systematic method to identify and map user actions to features, creating a replicable process that could accommodate varying levels of granularity across different platforms. Similarly, the inconsistency in feature definitions exposed the need for standardized processes, which emerged as a pivotal contribution of this research. By establishing a method for defining features systematically, the UAM was able to enhance clarity and consistency, addressing challenges that are pervasive across the software development industry. Additionally, the exploration of analytics integration across multiple versions of platforms revealed valuable insights into longitudinal analysis and comparative metrics, such as the newly introduced “comparative behavioral score,” which became a cornerstone of advanced usage analytics.

In addressing RQ2, the findings emphasized the critical importance of developing robust frameworks that prioritize adaptability and scalability. The feature-action mapping process evolved through iterative testing, ensuring that it could capture nuanced user behaviors effectively while remaining flexible enough to adapt to the evolving needs of diverse software systems. Similarly, the establishment of standardized feature definitions provided a universal language for developers, streamlining communication and reducing ambiguity in the analytics process. These advancements collectively demonstrated how developers can overcome barriers to effectively utilize usage data, transforming obstacles into opportunities for methodological innovation.

This chapter also sets the stage for addressing Research Question 3 (RQ3) in the next chapter by documenting the iterative development and application of UAM. The introduction of metrics like the “comparative behavioral score” illustrates how the research transitioned from identifying challenges to proposing actionable solutions. By focusing on systematic approaches to feature prioritization and evaluation, the study bridges the gap between theoretical frameworks and practical applications. The narrative transitions seamlessly from the complexities of understanding user behavior to providing tools and methodologies that enable the effective application of UAM across diverse software environments. This progression ensures a cohesive linkage between challenges and solutions, paving the way for a more sophisticated and scalable implementation of usage analytics.

Chapter 6

Usage Analytics Method

This chapter provides an in-depth introduction to the Usage Analytics Method (UAM), a comprehensive framework designed to address critical challenges in software development. These challenges, identified in Chapter 5, include unclear feature definitions, complexities in data integration, resource-intensive data preparation, and the inherent variability in user behavior. Each of these obstacles has shaped the iterative development of the UAM, ensuring that the method offers targeted and actionable solutions.

The UAM is structured across four stages—feature identification, data source integration, data extraction, and analysis—each designed to systematically address the key challenges faced by software development teams. By focusing on these stages, the UAM ensures that features are clearly defined, data is seamlessly integrated, and user behaviors are analyzed effectively to support feature prioritization and system optimization.

A significant contribution of this research, detailed in this chapter, is the development of two generic processes that are integral to the UAM. First, the process for creating a feature-action map is a central component of the feature identification stage. This structured methodology defines and associates user actions with specific application features, ensuring clarity and precision in identifying features. Second, the process for designing a custom monitoring tool is crucial to the data source integration and data extraction stages. This process provides developers with a scalable and adaptable approach to collecting usage data tailored to their application's requirements. Together, these processes extend the utility of the UAM, enabling broader adoption and adaptability across various software platforms.

The versatility of the UAM is demonstrated through its application to diverse platforms, including IBM Academic Cloud, IBM Watson Workspace, and Odoo Notes. These case studies highlight the method's adaptability to varying environments and its ability to deliver actionable insights regardless of the platform's complexity or resource constraints. For instance, the IBM Academic Cloud exemplifies the UAM's scalability in structured and resource-rich contexts, while the Odoo Notes platform underscores its flexibility in more constrained and dynamic settings.

This chapter also acknowledges the limitations of the UAM, such as its dependency on high-quality usage data and a reliance on action-based metrics. These limitations provide opportunities for future research and refinement, including exploring additional metrics, enhancing data collection strategies, and extending the method's applicability to broader contexts. Despite these challenges, the UAM establishes itself as a robust framework capable of bridging gaps in feature prioritization and user behavior analysis.

By setting the stage for detailed discussions in subsequent sections, this chapter underscores the significance of the UAM in addressing critical gaps in software development practices. The discussions here lay a foundation for understanding how the method not only resolves immediate challenges but also provides a pathway for continuous improvement and innovation in feature prioritization and user engagement strategies.

6.1 Design of the Usage Analytics Method

The Usage Analytics Method (UAM) is a rigorously developed framework designed to address pivotal challenges encountered in software development, as detailed in Chapter 5. These challenges include unclear feature definitions, fragmented data sources, the labor-intensive nature of data preparation, and variability in user behavior. By aligning each of its stages with specific challenges, the UAM delivers actionable solutions for feature prioritization and user behavior analysis.

The UAM's systematic design ensures that software teams can efficiently identify critical features, integrate diverse datasets, process raw information into actionable insights, and analyze behavioral trends to inform development priorities. By addressing core obstacles and enhancing workflow integration, the UAM not only supports immediate software development needs but also establishes a foundation for scalable and adaptable analytical methodologies.

The Usage Analytics approach describes various components involving the analysis of software usage data to understand the impact of changes made to the application on the usage

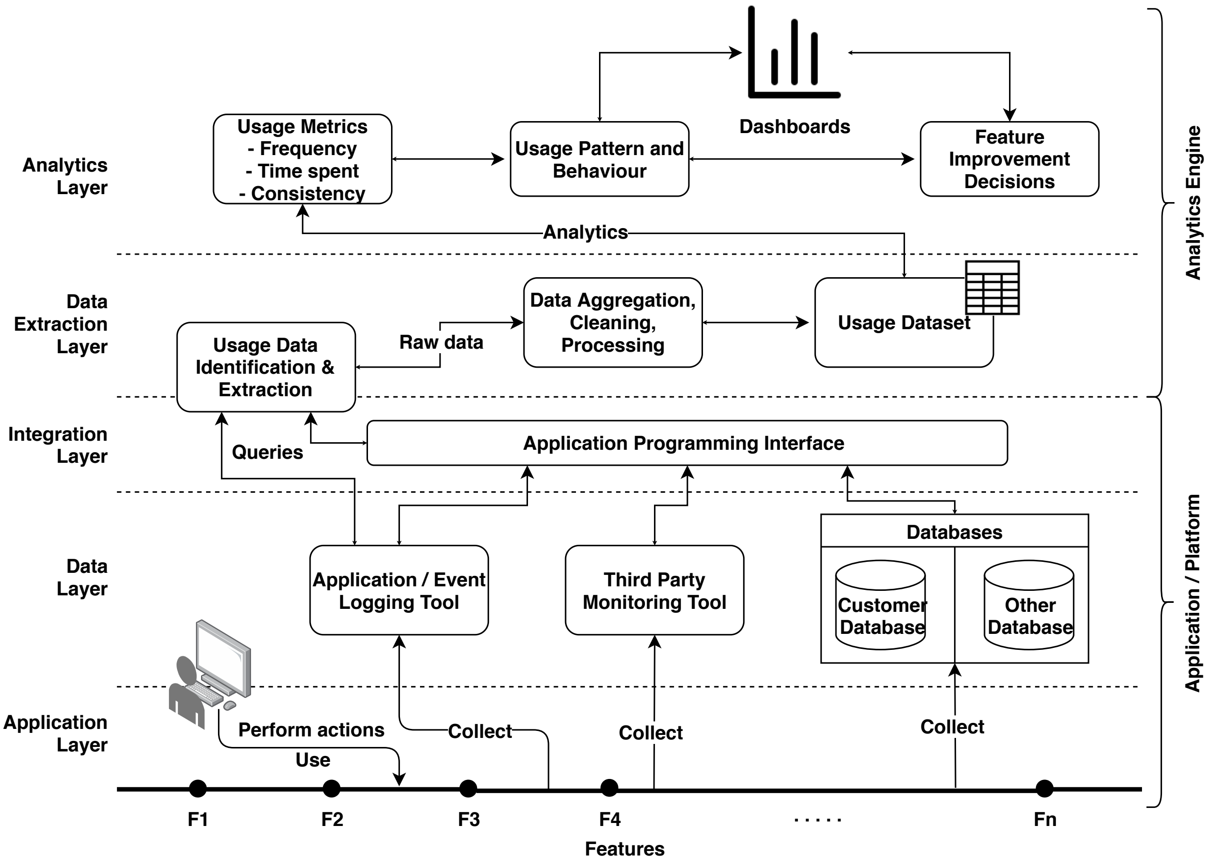


Figure 6.1: Usage Analytics architecture comprising core components of the analytics engine and application components used by the analytics engine

behaviour of the users. The high-level architecture of the Usage Analytics approach, as shown in Figure 6.1, groups different components necessary for the analytics into two categories: (1) Application/Platform specific components and (2) The Analytics Engine. The Application-specific components describe the necessary components from the target software application/platform in the Application, Data and Integration layers of the architecture. The application platform serves as the source of the usage data which may include application/event logging tools, third-party monitoring tools and the traditional databases storing the event data and the application logs. Usually, an application programming interface serves as the middle-man for the analytics engine to access the usage data sources.

The following sections provide an in-depth exploration of the UAM's four primary stages: feature identification, data source integration, data extraction, and analysis. Each stage is carefully detailed to demonstrate how it contributes to overcoming the identified challenges. References from earlier chapters are incorporated to connect theoretical underpinnings with practical applications. Insights from Chapter 7 validate the UAM through real-world implementations, including its adaptability to varied software environments and its role in fostering iterative improvements.

6.1.1 Feature Identification

The first stage of the UAM involves identifying and mapping application features to specific user actions, forming the foundation for subsequent analysis and prioritization efforts. This process tackles the challenge of unclear feature definitions, a critical issue that surfaced in the IBM Academic Cloud and IBM Watson Workspace case studies, where ambiguity in feature descriptions hindered effective user behavior analysis. By systematically defining features and linking them to user actions, this stage ensures clarity and consistency in understanding application functionalities.

Developers begin by cataloging features and actions using available documentation such as version logs, user manuals, and stakeholder inputs. This is complemented by data sources like application logs and telemetry tools, which capture user interactions in real-time. Through collaborative workshops, developers and stakeholders refine these definitions, ensuring they align with both technical capabilities and business objectives. These sessions also serve as an avenue for identifying dependencies and prioritizing high-impact features. The feature-action mapping process incorporates iterative validation steps to ensure robustness and alignment with application goals. Preliminary mappings are developed using historical usage data and are further refined through collaborative sessions involving users and cross-functional teams. These refinements enhance the shared understanding of application capabilities and ensure the mapping evolves in response to shifting requirements and new insights gained during development cycles. This dynamic process integrates stakeholder feedback and continuously adapts to support accurate and relevant feature identification.

This stage's outputs include a comprehensive feature-action map, which acts as a reference for subsequent stages, enabling seamless integration of user behavior data into analytical workflows. The map also facilitates alignment between development teams and stakeholders, ensuring that feature prioritization decisions are both data-driven and strategically aligned.

Creating a Feature-Action Map

An important result of the case study 1 was the creation of a new process to develop a feature-action map. This map provides a detailed visualization of user interactions with specific features and is supported by academic literature emphasizing the importance of understanding user behavior in software development. The concept of feature-action mapping is deeply embedded in user-centered design (UCD) principles, which emphasize the importance of designing software that meets the needs and preferences of its users. UCD advocates for an iterative design process that incorporates user feedback at every stage to ensure that the final product is both usable and useful (Norman, 2013).

User-Centered Design (UCD) (Abras et al., 2004) is a framework that places the user at the forefront of the design and development process. It involves understanding user needs, tasks, and environments to create products that provide a positive user experience. The feature-action map aligns with UCD by systematically capturing and analyzing how users interact with software features, ensuring that development efforts are focused on enhancing usability and functionality (Gulliksen and et al., 2003). Activity Theory (Nardi, 1996) provides a theoretical framework for analyzing human interactions with technology. It focuses on the actions and activities users perform to achieve their goals, considering the social and cultural context of these interactions (Kaptelinin and Nardi, 2006). The feature-action map applies activity theory by identifying and mapping user actions to specific features, thereby understanding the context and purpose of these interactions. Cognitive Load Theory (Sweller et al., 2011) explains how the human brain processes and stores information. It posits that reducing unnecessary cognitive load can enhance learning and performance. By mapping user actions to features, the feature-action map helps identify areas where the software may be causing cognitive overload, enabling designers to simplify and improve the user interface. Behavioral Analytics (Chaffey and Patron, 2012) involves the collection and analysis of data on user actions within a system. This approach provides insights into user behavior patterns, preferences, and pain points. The feature-action map leverages behavioral analytics to quantify how users interact with different features, guiding data-driven decision-making in the development process. Usability Engineering (Nielsen, 2012) focuses on improving the usability of software through systematic testing and evaluation. The feature-action map serves as a tool in usability engineering by providing a structured method for evaluating user interactions with features, identifying usability issues, and prioritizing improvements based on empirical data.

A generic process of creating a feature-action map involves several steps, which can be applied to various new and existing software platforms:

1. **Feature Identification:** For new applications, feature identification can begin with brainstorming sessions involving the development team to enumerate all potential features, as

suggested by (Ulrich and Eppinger, 2012). This process helps in gathering a comprehensive list of possible features. Another approach is user story mapping, which involves deriving features from user stories and user journey maps (Patton, 2014). This ensures that features are closely aligned with user needs and experiences. Additionally, stakeholder interviews and surveys can be conducted to capture user expectations and desired functionalities (Nuseibeh and Easterbrook, 2000), providing a broad perspective on what features should be included. For existing applications, reviewing action logs and tags within the application can help identify features based on user interactions and tagging systems already in place (Pazzani and Billsus, 2007). Conducting heuristic evaluations can identify key features and functionalities based on existing usability and design principles (Nielsen and Molich, 1990). Analyzing existing usage data to determine which features are actively used and which are not (Benyon, 2013), and performing audits of the current system to list all implemented features and their dependencies (Beyer and Holtzblatt, 1997), are also effective methods.

2. **Action Mapping:** Identifying specific user actions corresponding to each feature is crucial. This can be achieved through task analysis, which breaks down the steps users take to complete tasks associated with each feature (Hackos and Redish, 1998). Clickstream analysis tracks the sequence of user actions within the application (Bucklin and Sismeiro, 2009b), providing a detailed view of user navigation and interaction patterns. Event logging records user interactions at a granular level, providing comprehensive action data (Montero and Martínez-Ruiz, 2009). These techniques collectively ensure that the feature-action map accurately reflects how users interact with the software.

In Chapter 7, this process is validated through the IBM Academic Cloud case study, where the creation of a feature-action map clarified usage patterns and highlighted underutilized features. Developers found the map invaluable for linking user behavior to specific functionalities, which informed iterative updates. Additionally, collaborative workshops were employed to align technical definitions with business objectives, fostering a shared understanding of critical features.

This generic process provides a reusable methodology applicable across various software platforms, forming the foundation for precise user behavior analysis in subsequent stages. By creating a robust linkage between user interactions and software features, this process empowers teams to effectively prioritize enhancements and align development efforts with user needs. The mapping process also facilitated a feedback mechanism where users directly contributed insights, refining the accuracy of the mapping process over time.

6.1.2 Data Source Integration

Data Source Integration represents the second stage of the UAM, focusing on ensuring the collection and alignment of comprehensive and consistent user data from various sources. This stage directly addresses the challenge of fragmented and inconsistent data sources, as identified in the IBM Watson Workspace and Odoo Notes case studies. By systematically cataloging, standardizing, and validating these data sources, this stage establishes a reliable foundation for subsequent analysis.

Key Activities

1. **Cataloging Data Sources:** The integration process begins with cataloging all potential sources of user interaction data. Developers document key data repositories such as application logs, telemetry systems, event tracking databases, and third-party analytics tools. These repositories store critical information like timestamps, user actions, device details, and error logs. Collaborative workshops involving engineers and product managers ensure that all relevant sources are identified and cataloged for accessibility.

2. **Standardizing Data Formats:** One of the most pressing challenges of fragmented data is the inconsistency in format and structure across different sources. This activity involves creating a unified schema to standardize diverse datasets, ensuring compatibility during analysis. Custom scripts and tools convert logs and telemetry outputs into structured formats, such as JSON or CSV, that align with predefined data schemas.

3. **Identifying Data Gaps:** Comprehensive data integration also requires an analysis of existing gaps in data collection. Developers identify areas where critical information is missing, such as user action timestamps or session identifiers. These gaps are addressed by implementing enhancements in monitoring systems or designing custom data collection mechanisms tailored to the application's architecture.

4. **Designing Data Pipelines:** Efficient pipelines are established to manage the seamless flow of data from source systems to analytical repositories. These pipelines ensure data is

consistently collected, cleaned, and stored for further processing. Developers incorporate error-handling and recovery mechanisms to maintain the robustness of data flow systems.

Creating a Custom Monitoring Tool

To address the challenge of incomplete or insufficient monitoring systems, the UAM incorporates the design and deployment of custom monitoring tools. These tools provide tailored solutions for capturing detailed interaction data, offering flexibility and scalability for different application contexts.

1. **Requirement Analysis:** The initial step in developing a custom monitoring tool involves conducting a thorough requirement analysis. This includes clearly defining the objectives of the monitoring tool, such as identifying the specific data that needs to be collected, the metrics to be analyzed, and the desired outcomes. Developers must also determine the key features of the application that require monitoring and understand how users interact with these features. This comprehensive understanding will guide the design and development phases.

2. **Design Phase:** In the design phase, developers should create a high-level architectural design of the monitoring tool. This includes outlining the components such as the data collection module and data processing module. The design should specify the types of data to be collected, including timestamps, user IDs, action types, and other relevant metadata. Additionally, developers must identify the integration points within the application where the monitoring tool will capture data, such as UI elements, API calls, or backend processes. A detailed design ensures that all necessary data points are captured accurately and that the monitoring tool integrates seamlessly with the application.

3. **Development Phase:** The development phase involves building the various components of the monitoring tool. The data collection module is responsible for capturing user actions in real-time and storing them in a central database. Developers must implement mechanisms to ensure that user actions are recorded accurately and in a timely manner. The data processing module aggregates and pre-processes the collected data to prepare it for analysis. This involves several critical steps:

    - **Cleaning the Data to Remove Noise:** Noise in the data can result from irrelevant entries, duplicate logs, or system-generated events that do not contribute to meaningful insights. Cleaning the data ensures that only relevant and valid records are retained, reducing the complexity of subsequent analysis.

- **Handling Missing Values:** Missing values in the dataset can occur due to incomplete user actions, system errors, or network interruptions. These gaps can skew analysis results. Techniques such as imputation (e.g., using averages or medians), interpolation, or exclusion of incomplete records are applied to address missing data.
- **Ensuring Consistency:** Consistency involves standardizing data formats and ensuring alignment across all data points. For example, timestamps must follow the same format, and action types should use a unified naming convention. Consistent data enables accurate aggregation and comparison, ensuring that the metrics derived from the data are reliable and actionable.

These steps ensure that the data is of high quality and ready for further analysis and metric calculation.

4. **Implementation Phase:** During the implementation phase, the custom monitoring tool is integrated into the application. This integration should be seamless, ensuring minimal disruption to the user experience. Developers must establish a real-time data flow from the application to the monitoring tool, ensuring that data is captured and processed promptly. This phase also involves rigorous testing to verify that the monitoring tool functions as expected and captures the intended data accurately.

5. **Testing and Validation:** Testing and validation are critical steps to ensure the monitoring tool's functionality, reliability, and scalability. This phase involves multiple layers of testing to address different aspects of the tool's performance:

   - **Functional Testing:** Functional testing ensures that each component of the monitoring tool operates as intended. This includes verifying that the data collection module accurately captures all user interactions, such as clicks, navigations, and inputs, and that the data processing module correctly pre-processes the data by aggregating, cleaning, and formatting it for analysis. Functional testing also checks integration points within the application to ensure that data flows seamlessly from the application to the monitoring tool without omissions or errors.
   - **Performance Testing:** Performance testing assesses the monitoring tool's ability to

handle high volumes of data generated by real-world user interactions. This involves testing the tool under different load conditions to evaluate its responsiveness and stability. Key performance metrics include data capture latency, processing time, and storage efficiency. Ensuring that the tool can scale with increasing user activity is essential for its deployment in large-scale or high-traffic applications.

- **Usability Testing and Feedback Collection:** In addition to technical tests, usability testing gathers feedback from developers and stakeholders who interact with the tool. This process involves observing how users engage with the tool's interface and functionalities, identifying any usability barriers or inefficiencies. Feedback collection helps refine the tool's design and functionality, ensuring it aligns with the needs of its users and supports seamless integration into existing workflows.
- **Validation of Data Accuracy and Integrity:** Validation ensures that the data captured by the monitoring tool is accurate, complete, and consistent. This involves comparing the tool's output against expected results or known benchmarks. For instance, simulated user interactions can be used to verify that every action is logged correctly, with accurate timestamps and metadata.
- **Regression Testing:** If updates or modifications are made to the tool, regression testing ensures that new changes do not inadvertently disrupt existing functionalities. This step is crucial for maintaining reliability as the tool evolves.

These testing and validation efforts collectively ensure that the monitoring tool meets its performance requirements and operates reliably in real-world conditions. By addressing functional correctness, scalability, usability, and data accuracy, this phase guarantees the tool's effectiveness in supporting comprehensive usage analytics and informed decision-making.

6. **Deployment and Maintenance:** Once the monitoring tool has been tested and validated, it can be deployed in a live environment. Developers must ensure that the tool integrates smoothly with the production application and that it operates reliably under real-world conditions. Continuous monitoring of the tool's performance is essential to identify and address any issues promptly. Regular updates and enhancements are necessary to incorporate new features, address bugs, and improve performance, ensuring the monitoring tool remains effective and relevant as the application evolves.

By following these comprehensive steps, developers can create custom monitoring tools tailored to their specific applications. These tools will enable detailed data collection and analysis, providing valuable insights into user interactions and informing data-driven decisions for feature prioritization.

The outputs of this stage include a unified and comprehensive dataset ready for extraction and analysis. By integrating robust monitoring tools and creating seamless data pipelines, developers ensure that all user interactions are captured and stored in a consistent format. These efforts provide the analytical engine with high-quality inputs, supporting accurate and impactful analysis in subsequent stages. This stage also fosters organizational alignment by establishing clear protocols for data collection and management, ensuring long-term reliability and scalability of analytics efforts.

6.1.3 Data Extraction

The third stage of the Usage Analytics Method (UAM) focuses on data extraction, a critical process that transforms raw data into structured and actionable insights for further analysis. This stage addresses challenges such as fragmented data sources, inconsistencies in data quality, and the labor-intensive nature of preparing data for analytical workflows. Through adaptive practices validated in case studies, the data extraction stage ensures that datasets meet the analytical engine's requirements effectively and efficiently.

Key Activities

1. **Data Identification and Retrieval:** The initial step involves identifying and retrieving data from relevant sources, including application logs, telemetry systems, and monitoring tools. For example, in the IBM Watson Workspace case study, tools like New Relic were employed to access detailed logs capturing user actions, timestamps, and error events. Developers customized query mechanisms such as NRQL to extract specific data points efficiently, focusing on high-value metrics for analysis. This process ensures that only relevant and meaningful data is collected, minimizing noise and redundancy.

2. **Iterative Data Cleaning:** Data cleaning is an incremental process where duplicate entries, incomplete records, and erroneous values are systematically removed. In the IBM Academic Cloud case study, cleaning cycles were iteratively conducted to align data from

various sources, including telemetry and system logs. Time fields were standardized, categories were refined, and missing values were either estimated or flagged for exclusion. Iterative cleaning provided flexibility to adapt to evolving data needs, maintaining the integrity and relevance of the datasets.

3. **Data Preprocessing and Transformation:** Preprocessing involves structuring data to conform to the analytical engine's schema. Transformation steps include categorizing user actions, normalizing numerical fields, and merging complementary datasets. The Odoo Notes case study emphasized preprocessing, as raw log data required additional formatting to capture specific user interactions. Transformation pipelines also integrated metadata, such as user roles or session types, to enrich the analysis context.

4. **Validation and Consistency Checks:** Ensuring the accuracy and reliability of prepared datasets is essential for effective analysis. Validation steps were implemented to cross-check processed data against original logs, verifying consistency and completeness. Cross-functional teams reviewed the outputs to ensure alignment with analytical objectives. These checks reduced the risk of misinterpretation and improved confidence in the findings derived from the datasets.

The case studies highlighted practical and adaptable methods for data extraction. In the IBM Academic Cloud, challenges in integrating heterogeneous logs were addressed through manual alignment techniques. Scripts were employed to standardize formats and resolve inconsistencies. In the IBM Watson Workspace, query-based tools facilitated efficient data retrieval, enabling the extraction of detailed user interaction metrics while maintaining the flexibility to adapt to evolving analytical needs. In the Odoo Notes case study, custom logging configurations were developed to capture granular user interactions, ensuring that critical events were recorded. These configurations addressed gaps in existing telemetry systems and included data enrichment strategies to add contextual details, such as feature usage frequency, enhancing the value of the analysis.

The outputs of this stage include refined datasets free from redundancies and enriched with contextual information, supporting deeper insights into user behavior and feature performance. Validation processes ensure that data from multiple sources align with a unified schema, facilitating seamless integration into analytical workflows. By leveraging these datasets, the analytical engine derives actionable insights that support informed decisions on user engagement strategies and feature prioritization. This stage underscores the adaptability and effectiveness of the UAM in addressing real-world challenges and reinforces its scalability and relevance across diverse software environments.

6.1.4 Analysis

The core metrics used to understand the changes in usage behaviour of the software users are frequency, time spent and consistency. Frequency F of action A refers to the number of times an action was performed denoted by N by either an individual or a group of users denoted by u over a specific period of time t can be measured as shown in Equation 6.1. Typically, the period of time is measured in sessions as defined by the developer or software analyst (days, weeks or months) and ideally in a software development domain, the period of time is set to the amount of time a specific version of the application is available to the users. This enables the developers to gather the usage data for a specific version of the application and compare the usage data with the previous version of the application to understand the impact of changes made to the application on the usage behaviour of the users between the two versions. Furthermore, the value of t can be set to a larger value to include further versions of the application to generate a trend of change in the frequency of actions performed by the users over multiple versions of the application. For the purpose of this research, the value of t is set automatically for each participant of the experiment based on the amount of time the participant spent on the experiment session.

$$F_{A,u} = \frac{N_{A,u}}{t} \tag{6.1}$$

Time spent T of an action A refers to the amount of time spent on a feature by a user u, that is the sum of the amount of time spent on all the actions that belong to the feature where TS refers to the timestamp of the log entry for the action and it can be measured as shown in Equation 6.2. The term TS refers to a timestamp of the log entry for the action A. The term TS_{x+1} refers to the timestamp of the log entry for the next action performed by the user. The term TS_x refers to the timestamp of the log entry for the action performed by the user. The term k refers to the total number of actions performed by the user u. A crude assumption here is that the user only performed one action at a time. This assumption is made to simplify the calculation of the time spent on a feature. However, this assumption can be relaxed by considering the time spent on a feature as the sum of the time spent on each action performed by the user. In reality, the users of a software application may perform multiple actions at the same time. But, since the UA approach relies on the comparative score between versions of the application, the assumption does not affect the final result of the UA analysis.

$$T(A,u) = \sum_{x=1}^{k_u} \big(TS(A,u)_{x+1} - TS(A,u)_x\big) \tag{6.2}$$

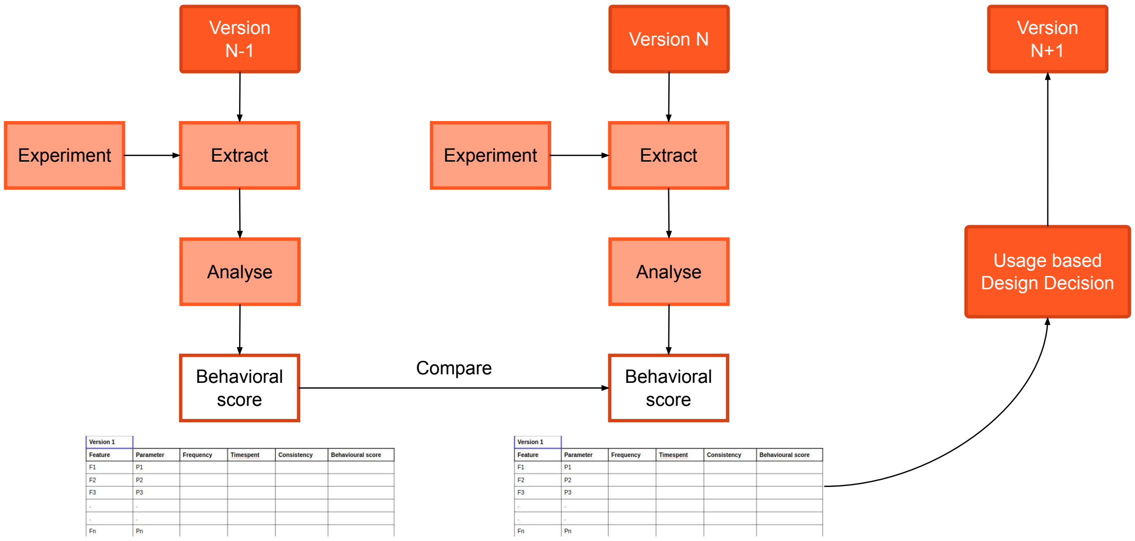
Consistency C of action A is a measure to calculate if the users exhibit similar consistency in accessing a feature when changes are implemented to the features by the developers. Based on the findings from version 2 of the Usage Analytics method, the Consistency measure is further divided into *Consistency of Frequency* and *Consistency of Timespent* and implemented in the final version of the Usage Analytics method (version 3). The Consistency of Frequency measure can be calculated by combining the change of Frequency metric for each action available with an application as shown in Equation 6.3. The Consistency of the Timespent measure can be calculated by combining the change of the Timespent metric for each action available with an application as shown in Equation 6.4.

$$CF(A,u) = F(A,u)_{(n+1)} - F(A,u)_n \tag{6.3}$$

$$CT(A,u) = T(A,u)_{(n+1)} - T(A,u)_n \tag{6.4}$$

This comparative analysis reveals the behavioural change exhibited by the user was either positive or negative. The behavioural change to be considered as either positive or negative is defined by the developer of the software application based on the context of usage of a feature as defined by the developers. For example, Microsoft Azure provides a tool for application developers to define expected user flows and user journeys[1]. To illustrate, spending more time on a feature could be considered better in a scenario where the users are expected to spend as much time as possible on a feature which in turn accumulates monetary benefits to the development team. For example, an application which provides news content for the users to read and advertisements are placed on the same page as the news content, in such a scenario, the users of the application are expected to spend as much time as possible. If the average time readers spend on a page decreases following changes made to the application, the change is considered negative. However,

[1] https://learn.microsoft.com/en-us/azure/active-directory-b2c/user-flow-overview



Version 1					
Feature	Parameter	Frequency	Timespent	Consistency	Behavioural score
F1	P1				
F2	P2				
F3	P3				
.	.				
.	.				
Fn	Pn				

Version 1					
Feature	Parameter	Frequency	Timespent	Consistency	Behavioural score
F1	P1				
F2	P2				
F3	P3				
.	.				
.	.				
Fn	Pn				

Figure 6.2: Compare behavioural scores between consecutive software versions

a payment page where the user enters the payment account details to pay for the membership requires the users to spend as little time as possible on that page, if the average time spent on that page is increased after some changes are implemented by the developer then that change is considered negative. As a result, the developers are responsible to define the positive or negative behavioural score for each feature of the application. This allows software developers to assess the impact of the changes made to the application by determining whether they positively or negatively influenced user behavior. This understanding is critical for identifying successful enhancements, addressing potential issues, and ensuring that the software evolves in alignment with user needs and expectations.

The analytics metrics used in the Usage Analytics Method include frequency, time spent, and consistency. Frequency refers to the number of times a feature is used within a given period. This metric helps identify which features are most popular and how often they are utilized by users. Time spent measures the duration of user engagement with each feature, providing insights into feature importance and user engagement levels. Consistency evaluates the regularity and reliability of feature usage over time, helping to identify features that are essential to the user experience versus those that are sporadically used.

The analysis phase concludes with actionable recommendations that align with organizational goals. These insights inform feature prioritization, resource allocation, and strategic planning, ensuring that development efforts are focused on delivering maximum value to users. By integrating data-driven decision-making into the development cycle, this stage enhances the overall effectiveness and adaptability of the UAM.

6.2 Experiment Design

The experiment design involves creating controlled environments to test the effectiveness of the Usage Analytics Method. This process begins with the selection of use cases, choosing diverse software applications to validate the method. A data collection framework is set up, with tools and protocols for consistent data collection across different platforms. For the purpose of both collection of the necessary data for the analysis and validation of the results, the following experiment setup in a controlled environment was designed and implemented using IBM Watson Workspace and Odoo Notes. The participants for the experiments designed for IBM Watson Workspace were the design and development team of the application, where, all participants are considered experts with the application. Since the IBM Watson Workspace application was only deployed internally in IBM for different development teams to use at the time of conducting this research, the expertise of the available users was only limited to the expert users. However, the participants for Odoo Notes were recruited randomly, including participants with varied levels of expertise and experience with the application. This varied level of expertise with the participants of the experiments with Odoo Notes eliminates any chance of bias related to familiarity with the type of application or the application itself. The varied level of expertise among participants, while eliminating familiarity bias, can also introduce challenges in interpreting the results. Participants with different levels of expertise may interact with the application differently, leading to variability in usage patterns that are not solely attributable to the application's design or updates. This variability can obscure the true impact of the software changes being analyzed, making it harder to derive consistent and generalizable insights. Additionally, inexperienced participants may require more time or support, potentially skewing metrics such as time spent or consistency. A more homogenous level of expertise among participants could provide a clearer understanding of how the changes affect typical users. The rest of this section describes the design of the experiments in detail.

The general structure of the experiment starts with a quick demonstration of the application features for the participant. Since the experiment is designed to include participants with various levels of experience with the application, some participants may never have any experience with the specific application or other similar applications. The demonstration step will introduce different features available with the software and how to use them. Once the participant is aware of the different interactions they can perform with the application, the participant is asked to perform a specific set of tasks. The tasks are designed in a way to explore most of the features of the software, however, some tasks which could not be completed by a single user alone such as the chat feature that requires some interaction from another user to complete the task are excluded from the list of tasks. A sample list of tasks the participants are asked to perform with Odoo Notes is shown below:

Task List (Odoo Notes):

1. Create a new column and name it “Temporary”, you can use this column to place any temporary notes you may create
2. Create a minimum of 7 notes (one for each new task)
3. Move the newly created notes to the correct column (if not done before); two ways of doing it:
   (a) Drag and drop
   (b) Click on the column name in the top right position after opening the note
4. *and so on ...*

The complete list of tasks is available in Appendix B. Each participant is instructed to perform all the tasks. Once the participant (user) starts interacting with the application, the data is automatically recorded by the monitoring system. Each of these tasks generates log entries in the application monitoring system. Each log entry refers to some *action* performed by the user. Each task completed in the list may result in one or more of these actions performed by the user. The list of actions a user can perform in reference to the application can be identified by exploring the application and feature design or exploring the source code of the application. An example of *actions list* using the Odoo Notes application is shown below in Log 6.1:

Log 6.1: List of actions performed by the user in Odoo Notes

```
{
    "Actions": [
    {"ActionName":"Create Stage", "ID": 1},
    {"ActionName":"Delete Stage", "ID": 2},
    {"ActionName":"Rename Stage", "ID": 3},
    {"ActionName":"Search with Note", "ID": 4},
    {"ActionName":"Search with Tag", "ID":5},
    ... ]
}
```

The complete action list is available in Appendix C. Once the user completes performing all tasks in the list. The user is asked to perform the same list of tasks with another version of the application. In the case of Odoo Notes, Odoo versions 10 and 11 were chosen for the experiments. With IBM Watson Workspace, since the deployment design followed a minor incremental update process with an option to revert back multiple stages, the users of the application may not see a major visual or interactive difference. This resulted in the Watson Workspace application not having specific different versions. In that case, the participants of the experiment were only asked to complete the tasks once using the current version of the application. This scenario, while effective for testing the UA method, is not entirely realistic in replicating natural user behavior. In real-world scenarios, users interact with software based on their specific needs, goals, and preferences, often performing tasks in varying sequences or skipping certain actions altogether. Real users may not complete all tasks, and the sequence provided in the experiment may not align with the organic flow of their interactions.

However, the structured approach used in this experiment is intentionally designed to maximize the coverage of user actions and interactions within the application. By ensuring that participants perform tasks across a broad range of features, the experiment allows for comprehensive data collection that reflects the full spectrum of possible user interactions. This is critical for evaluating the effectiveness of the UA method, as it provides a robust dataset for analyzing metrics like frequency, time spent, and consistency.

While this approach may not mirror realistic usage patterns perfectly, it is a controlled and systematic way to test the capabilities of the monitoring system and ensure that the analysis includes all potential user actions. Future iterations could consider incorporating more flexible task designs or real-world scenarios to further validate the method's applicability in practical settings.

6.2.1 Experiment Data and Application of UA Metrics

As discussed previously, the experiment data is collected in the form of log entries generated by the monitoring system. Some software applications use inbuilt monitoring systems whereas others rely on third-party application monitoring systems. IBM Watson Workspace application relies on GraphQL API [2] and Newrelic Application Performance Monitoring [3] to monitor the application. The GraphQL API is a query language for APIs and a runtime for fulfilling those queries with the application's existing data. The GraphQL API is used to collect the data from the application and store it in a database. The data collected by the GraphQL API is in the form of log entries. A typical query to extract the log entries from the database is shown below in Log 6.2:

Log 6.2: GraphQL query to extract log entries

```
{
 space(id: "5a32abb0e4b0e7caf246140d") {
   title
   updated
   conversation {
     createdBy {
       displayName
     }
     messages {
       items {
         contentType
         created
         updated
         createdBy {
           id
           customerId
           extId
           ibmUniqueID
           displayName
```

[2] https://graphql.org

[3] https://newrelic.com/lp/apm

```
     }
     updatedBy {
      displayName
     }
     content
     annotations
    }
   }
  }
 }
}
```

The space ID refers to the instance of the application used for the experiment. The query returns the log entries in the form of JSON objects. These JSON objects are further explored specifically with a focus on the *createdBy* and *annotations* fields. The createdBy field refers to the information that could be used to identify the user. The annotations field contains the log entry in the form of a complex string referring to the action performed by the user. Further exploration of the data referred to by the annotations field revealed that the GraphQL tool is primarily used to monitor the actions performed by the Watson AI rather than the users of the application. As a result, the data from GraphQL API are ignored. Instead, the Newrelic Application Performance Monitoring tool was used to collect the experiment data.

The Newrelic tool is a third-party application monitoring tool that collects the data from the application and stores it in a database of the developer's choice. The data collected by the Newrelic tool is in the form of log entries which can be extracted using the API by Newrelic. Each log entry contains information about the action performed by the user. The log entries are in the form of JSON objects. A sample log entry is shown below in Log 6.3:

Log 6.3: Sample log entry

```
{
  {
  "results": [
    {
      "events": [
        {
            "timeSinceLoad": 5552.808,
            "appVersion": "Beta",
            "teamSubscriptionIds": "[\"0\"]",
            "session": "7f8179dcbf74288c",
            "userAgentDevice": "Windows",
            "USER_ID": "e33a3950-58a8-414a-b300-e57e6912e3bb",
            "REQUEST_ID": "Web-c0eba120-0c1e-11e8-af9b-4d3b86be013f",
            "userAgentName": "Firefox",
            "screenOrientation": "landscape",
            "teamOfferings": "[\"PREVIEW\"]",
            "countryCode": "GB",
            "appId": 16466058,
            "customerId": "IBM-0000-0001",
            "userAgentOS": "Windows",
            "asnLatitude": "51.4964",
            "timestamp": 1518018688075,
            "currentUrl": "https://workspace.ibm.com/space/5a32abb0e4b0e7caf246140d",
            "browserLanguage": "en-US",
            "appName": "Toscana",
            "browserHeight": 920,
            "buildId": "20180207-083427",
            "SUBSCRIBER_ID": "com.ibm.web.toscana",
            "referrerUrl": "https://workspace.ibm.com/",
            "asnLongitude": "-0.122406006",
            "actionType": "InTeam",
            "userAgentVersion": "52.0",
            "name": "Unnamed Transaction",
            "pageUrl": "https://workspace.ibm.com/",
            "browserWidth": 946,
            "appLanguage": "en",
                  "actionName": "CHAT_MESSAGE_RECEIVED"
              }
       }
    }
}
```

A typical query to extract the log entries from the Newrelic database is shown below in Log 6.4:

Log 6.4: Newrelic queries to extract log entries

```
SELECT count(*) from PageAction, MobileAction facet actionName since 24 hours ago
    ↪ LIMIT 300
SELECT * from PageAction, MobileAction SINCE this quarter
```

The application of UA metrics is demonstrated through detailed case studies. In Case Study 1, IBM Academic Cloud, the initial design of the usage analytics method and feature prioritization challenges were identified. The focus was on understanding user needs and aligning development efforts accordingly. In Case Study 2, IBM Watson Workspace, the usage analytics method was improved and implemented in Version 2. This involved addressing implementation challenges and refining the method to better capture user engagement data. In Case Study 3, Odoo Notes, further improvement of the Usage Analytics Method was achieved in Version 3, with a custom monitoring tool to validate the method's effectiveness in a real-world setting.The experiment data collected from the application monitoring system is analyzed using the UA metrics: Frequency, Time Spent and Consistency as discussed in Section ??.

6.2.2 Post-experiment Survey

Post-experiment surveys were conducted to gather qualitative feedback from users. These surveys assessed overall satisfaction with the software features, understanding which features were found to be most useful, and identifying areas for further enhancement based on user feedback. After completing the tasks, the user is asked to complete a survey. In addition to the general questions related to the participant and his/her experience with the specific application (Odoo or IBM WW) and other applications of a similar type, the questions of the survey were primarily designed to validate the results of the analytics.

The complete survey form for experiments with IBM Watson Workspace is available in Appendix D.2. The survey consists of 10 questions. Question 1 gathers the Temporary user ID assigned to the participant. Question 2 asks the participant to briefly summarize the tasks performed by the participant during the freestyle session of the experiment. Question 3 asks the participant about their familiarity with IBM WW and other similar team collaboration applications. Question 4 asks the user if the scenario (set of tasks to perform with the application) of the experiment was understandable. Question 5 asks the participant to answer the level of difficulty in understanding the scenario of the experiment on a scale of 1 (very easy) to 5 (very hard). Question 6 asks the participant to rate the difficulty level of each task on a scale of 1 (very easy) to 5 (very hard). Question 7 is optional where the participant can provide additional feedback on the difficulty level of the tasks. Question 8 is designed to understand if capturing screenshots of participants' interaction with the experiment influences the way they interact with the application during the experiment. Question 9 is optional where the participant can provide additional feedback on the overall experiment. Question 10 is optional where the participant can include their name.

The complete survey form for experiments with Odoo Notes is available in Appendix D.1. The survey includes 8 questions in total. Questions 1 to 3 of the survey are designed to gather the user's experience and familiarity with Odoo Notes or with similar types of applications. Questions 4 and 5 are designed to understand which version of the application is preferred by the user and the reason for the preference. Question 6 is designed to understand the preference of the user specifically for each action. Question 7 is designed to understand the level of difficulty experienced by the user between the versions of the application. Specifically, a comparative analysis can be performed over each feature of the application, changes implemented to each feature and how these changes affected the participant of the application. Furthermore, the results of the comparative analysis are used to evaluate the results of the analytics performed on the experiment data.

Post-experiment surveys were conducted to gather qualitative feedback from users. These surveys assessed overall satisfaction with the software features, understanding which features were found to be most useful, and identifying areas for further enhancement based on user feedback. The Usage Analytics Method provides a systematic approach to prioritizing software features based on detailed analysis of user interactions. Through the use cases, the method has been validated and refined to offer actionable insights that enhance the efficiency and effectiveness of software development processes. By focusing on identifying features, usage data sources, data extraction, and analysis, the method ensures a comprehensive understanding of user behavior and feature performance. The usage metrics are applied to the usage data collected from the application monitoring system used in the experiments with the applications IBM Watson Workspace and Odoo Notes as described in Section 6.2.1. The results from the experiments and evaluation of the Usage Analytics method using the results of the surveys are presented and discussed in the next chapter.

6.3 Contributions of the Usage Analytics Method (UAM)

The Usage Analytics Method (UAM) represents a significant contribution to the field of software analytics, providing a structured and systematic framework for deriving actionable insights from usage data. By addressing critical challenges identified in earlier chapters, the UAM demonstrates its utility and adaptability across diverse platforms. This section outlines the core contributions of the UAM, emphasizing its impact on feature prioritization, user behavior analysis, and decision-making in software development.

Comprehensive Framework for Software Analytics

The UAM introduces a four-stage process—Feature Identification, Data Source Integration, Data Extraction, and Analysis—that provides a cohesive approach to understanding user interactions and the impact of software changes. This framework integrates theoretical insights with practical applications, ensuring that the method is both robust and adaptable to varying contexts. Each stage addresses specific challenges, such as unclear feature definitions, fragmented data sources, and the need for actionable metrics, creating a seamless workflow from raw data to strategic insights.

Addressing Key Challenges in Software Development

The UAM effectively resolves several critical challenges in software development:

1. **Unclear Feature Definitions:** By implementing the feature-action mapping process, the UAM ensures clarity and consistency in identifying application features and their associated user actions. This process was particularly impactful in the IBM Academic Cloud case study, where it clarified usage patterns and informed iterative updates.

2. **Fragmented Data Sources:** Through its data source integration stage, the UAM consolidates diverse data streams into a unified schema, enabling consistent and comprehensive analysis. This capability was validated in the IBM Watson Workspace case study, where diverse telemetry and log data were aligned for meaningful insights.

3. **Behavioral Variability:** The analysis stage of the UAM employs metrics such as frequency, time spent, and consistency to understand user behavior and measure the impact of updates. Comparative analysis across user cohorts, as demonstrated in the Odoo Notes case study, highlighted the effectiveness of the method in addressing behavioral variability.

Generic Processes for Broader Applicability

The UAM introduces two generic processes that extend its utility beyond the case studies:

1. **Feature-Action Mapping:** This process provides a structured approach to associating user actions with application features, aligning with user-centered design principles and ensuring usability improvements.

2. **Custom Monitoring Tools:** The method for designing lightweight and adaptable monitoring tools enables developers to capture detailed user interaction data, addressing gaps in traditional logging mechanisms. These tools were instrumental in Odoo Notes, where they provided critical insights into user workflows.

Adaptability Across Platforms

One of the most notable contributions of the UAM is its adaptability to varying software environments. By focusing on flexible processes rather than rigid frameworks, the method was successfully applied across the IBM Academic Cloud, IBM Watson Workspace, and Odoo Notes platforms. This adaptability underscores the method's relevance to both structured, enterprise-level systems and resource-constrained, lightweight applications.

Data-Driven Decision-Making

The UAM bridges the gap between data collection and actionable insights, fostering a culture of data-driven decision-making. The metrics and methodologies employed in the analysis stage empower stakeholders to prioritize features, allocate resources, and plan strategic updates based on empirical evidence. For instance, the frequency and time spent metrics guided critical interface redesigns and usability enhancements in the case studies, ensuring alignment with user needs and organizational objectives.

Contribution to Research Objectives

The UAM aligns closely with the research objectives and research questions posed in this thesis. Specifically, it addresses RQ2 by resolving challenges in identifying and utilizing usage data and metrics. Additionally, the structured activities in each stage directly support RQ3 by providing a systematic approach to feature prioritization and user behavior analysis. These contributions not only validate the method but also highlight its potential for further refinement and application in broader contexts.

In summary, the UAM provides a holistic and adaptable approach to understanding and leveraging usage data in software development. By addressing critical challenges, introducing reusable processes, and fostering data-driven decision-making, the UAM establishes itself as a valuable tool for improving user experiences and optimizing software systems.

Chapter 7

Results and Evaluation of the Usage Analytics Method

This chapter serves as a comprehensive integration of the findings and evaluations related to the Usage Analytics (UA) method, with a direct focus on addressing the research questions that underpin this study:

1. RQ2: "What are the challenges developers face in identifying and utilizing usage data for feature prioritization?"

2. RQ3: "How can activities be systematically structured to identify and utilize usage data for feature prioritization?"

The primary objective of this chapter is to construct a detailed narrative that connects the various challenges identified, the iterative processes of refining the UA method, and its practical application across a range of software platforms. Furthermore, the chapter provides a holistic perspective on the methodology's development, its utility, and its contributions to software development practices. The chapter evaluates the UA method's effectiveness in supporting feature prioritization through qualitative and quantitative assessments. It outlines specific outcomes, such as identifying high-impact features and improving decision-making processes, underscoring the method's practical value.

The integration of findings in this chapter is carefully structured to ensure clarity and transparency. By systematically linking research questions to outcomes, the narrative presents a clear progression from problem identification to solution development and evaluation. This approach not only highlights the UA method's robustness but also emphasizes its adaptability and relevance across varying software development environments. Moreover, this chapter reflects on the broader implications of the findings. The UA method's potential to bridge the gap between user behavior analysis and actionable feature prioritization decisions is underscored, demonstrating its significance in advancing software analytics. The discussion also addresses limitations and proposes pathways for further research, ensuring a balanced and forward-looking perspective on the study's contributions. By extending the discussion and incorporating comprehensive evaluations, this chapter aims to provide a nuanced understanding of how the UA method addresses the identified research questions. The detailed exploration of challenges, iterative refinements, and practical outcomes ensures that the study's findings are robust, transparent, and directly aligned with its objectives.

7.1 Evaluation of the Usage Analytics Method Through Case Studies

This section presents a comprehensive evaluation of the usage analytics methods implemented in three distinct case studies: IBM Academic Cloud, IBM Watson Workspace, and an additional use case in Odoo Notes. This chapter aims to provide a detailed assessment of the effectiveness and challenges associated with the application of advanced usage analytics within these software platforms. The evaluations conducted in this chapter employ a combination of qualitative and quantitative methods to provide a holistic understanding of the usage analytics methods. The qualitative evaluation involves structured interviews with developers, capturing their experiences and feedback regarding the implementation and impact of the usage analytics process. The quantitative evaluation leverages specific metrics such as frequency, timespent, and consistency of user actions to measure user interactions and derive insights into user behavior.

The purpose of these evaluations is multi-faceted:

1. **Understanding Developer Challenges:** To identify the key challenges developers face in leveraging usage data for feature prioritization and overall software improvement.

2. **Assessing Method Effectiveness:** To evaluate the effectiveness of the usage analytics methods in providing actionable insights that enhance the software development process.

3. **Refining Analytics Methods:** To gather feedback that can inform the continuous refinement and improvement of the usage analytics methods.

Evaluation of Case Study 1: IBM Academic Cloud

The primary goal of Case Study 1 was to investigate the challenges associated with the feature prioritization process within the IBM Academic Cloud project. This case study involved structured interviews with three developers, focusing on their experiences with the usage analytics process model. The evaluation highlighted significant themes such as the difficulty in analyzing user interactions due to unclear platform feature definitions and the time-consuming nature of data selection and preparation.

Evaluation of Case Study 2: IBM Watson Workspace

Building on insights from Case Study 1, Case Study 2 aimed to address the key challenges faced by developers in identifying and utilizing usage data effectively within the IBM Watson Workspace. Structured interviews with five participants, including an operational manager, an architect, a senior developer, and two developers, provided detailed feedback on the advanced usage analytics method. This evaluation focused on the application of analytics metrics, the development of feature-action maps, and the general implementation challenges encountered during the process.

Evaluation of Case Study 3: Odoo Notes

Case Study 3 extended the application of the usage analytics method to the Odoo Notes application, involving nine users interacting with two different versions of the application. The evaluation aimed to assess the changes in user behavior between versions 10 and 11 of Odoo Notes, using metrics such as frequency, timespent, and consistency. The results were compared with user survey responses to validate the effectiveness of the usage analytics method and identify the most impacted features.

In summary, Chapter 7 integrates qualitative and quantitative evaluations across three case studies to provide a comprehensive assessment of the usage analytics methods. The insights gained from these evaluations are intended to inform future improvements in usage analytics practices and enhance the overall software development process.

7.2 Evaluation of Case Study 1: IBM Academic Cloud

The descriptive analysis conducted in this study provided a comprehensive understanding of the challenges and effectiveness of the usage analytics process model implemented in the IBM Academic Cloud project. By systematically categorizing and interpreting the qualitative data from developer interviews, this analysis revealed significant themes and patterns, offering insights into the practical experiences and feedback from the developers. Descriptive Analysis (Lambert and Lambert, 2012) is a widely accepted qualitative research technique used to systematically describe and interpret the main features of a dataset. According to Vaismoradi et al. (2013), descriptive analysis helps in providing a rich and comprehensive account of the data by categorizing and interpreting responses to reveal underlying patterns and themes. This approach is particularly effective in qualitative studies where the aim is to understand and elucidate participant experiences and perspectives in a detailed manner.

7.2.1 Goal of the case study 1

The primary goal of Case Study 1 was to investigate the challenges associated with the feature prioritization process within the IBM Academic Cloud project. This case study aimed to identify specific obstacles faced by developers during the implementation of the usage analytics process model. Key objectives included understanding the difficulties in analyzing user interactions, identifying the impact of unclear platform feature definitions, and evaluating the overall effectiveness of the process model in achieving its intended outcomes.

7.2.2 Design of the case study 1

The descriptive analysis conducted for this qualitative analysis included several phases. The data collection phase involved conducting structured interviews with three developers who participated in the IBM Academic Cloud project, the demographics of the developers are shown in Table 7.1. These developers, selected for their substantial experience in data analytics, software architecture, and cloud computing, provided valuable insights into each stage of the usage analytics process model.

The interviews were recorded and transcribed to ensure accuracy and completeness. The next step in the analysis was data coding, where the transcribed interviews were systematically categorized into meaningful codes. Coding involved identifying segments of the transcripts that related to specific topics or issues. Codes were created based on the predefined stages of the usage analytics process model (Planning, Requirements Analysis, Designing, Building, Testing) and recurring themes identified in the feedback. This step ensured that the data was organized and prepared for deeper analysis. After coding the data, broader themes were developed to represent major recurring topics or issues discussed by the developers. These themes were developed inductively from the data and aligned with the predefined stages of the process model. Each theme accurately reflected the developers' experiences and challenges, providing a structured framework for interpreting the feedback. The coded data was organized under each theme and process stage, facilitating a structured and systematic interpretation. This organization helped identify patterns and trends within and across different stages of the process model. By organizing the data in this manner, it was possible to gain a comprehensive view of the developers' experiences and the specific challenges they encountered.

Table 7.1: Developer Demographics for Case Study 1: IBM Academic Cloud

Category	**Developer 1**	**Developer 2**	**Developer 3**
Experience Level	Senior (10+ years)	Senior (10+ years)	Mid-Level (5-10 years)
Technical Expertise	Python, Java Specialization: Data Analytics, Software Architecture	Python, JavaScript Specialization: Software Architecture, Machine Learning	Python, JavaScript Specialization: Full-Stack Development
Industry Experience	Enterprise Software Development, Cloud Computing	Data Analysis and Business Intelligence, Cloud Computing	Web Development, Enterprise Software Development
Roles in the Project	Lead Developer Responsibilities: Overseeing implementation	Software Architect Responsibilities: Designing software processes	Software Engineer Responsibilities: Developing and testing components
Tenure in the Company	2-5 years	4-5 years	Less than 2 years
Tenure in Current Role	1-2 years	3-4 years	Less than 1 year

Interview questions used in case study 1

1. Planning Stage:

   (a) How effective was the planning stage in setting clear goals for understanding user usage patterns, user behavior, and identifying critical features from the user's perspective?

   (b) Did the lack of clear definitions for platform features impact your ability to plan effectively? How did this affect your understanding of user interactions?

2. Requirements Analysis Stage:

(a) How comprehensive and effective was the requirements analysis stage in eliciting the necessary requirements for the usage analytics process model?

(b) How did the absence of a user interaction mapping to application features impact the requirements elicitation process?

(c) How challenging was it to classify usage data without clear feature definitions? Can you describe the difficulties encountered?

(d) How successful were you in identifying relevant usage data sources given the diverse types of data available? Were there any specific challenges?

(e) How much time and effort were required to decide on the specific types, sources, and formats of data to be used?

3. Designing Stage:

   (a) How effective was the design stage in developing the usage data extraction process, analytics algorithms, and evaluation process, given the complexity of the data and the absence of feature mapping?

   (b) What specific design challenges did you encounter due to the lack of clear feature definitions?

   (c) What design tools or techniques were used during this stage? How effective were they in achieving the design goals amidst the data selection and processing challenges?

   (d) How well did the collaboration between software architects and other team members work during the design stage, considering the data heterogeneity and processing bottlenecks?

4. Building Stage:

   (a) How efficient was the building stage in constructing the usage data extraction and analytics components, given the resource-intensive nature of data preparation?

   (b) Were there any significant challenges or roadblocks encountered due to the complexity of cleaning and transforming large volumes of data?

   (c) How well did the different components integrate, especially considering the inconsistencies and gaps in data sources?

   (d) Was the allocation of resources (time, personnel, tools) sufficient and effective during the building stage, given the time and effort required for data selection and preparation?

5. Testing Stage:

   (a) How thorough was the implementation and testing of the usage data extraction and analytics components, particularly in dealing with the absence of a clear mapping of user interactions to features?

   (b) Were there any major issues identified during testing due to the lack of clear feature definitions? How were these issues addressed?

   (c) How effective was the evaluation process in assessing the performance and accuracy of the usage analytics components, given the challenges in applying advanced analytical techniques?

   (d) How was user feedback incorporated during the testing stage? Were there multiple iterations to refine the components based on testing results, considering the initial data mapping challenges?

6. Overall Process Model:

   (a) How effective was the overall usage analytics process model in achieving its objectives, despite the significant challenges encountered in data collection and analysis?

   (b) What were the key strengths and weaknesses of the process model?

   (c) Based on your experience, what improvements would you suggest for the usage analytics process model, especially in terms of defining platform features and mapping user interactions?

   (d) How does the usage analytics process model compare with previous methods you have used for feature prioritization and user behavior analysis, particularly in addressing data selection and processing challenges?

   (e) Did the usage analytics process model have a noticeable impact on development time? How significant was this impact, considering the bottlenecks identified in data collection and processing?

7.2.3 Results of the case study 1

The interviews revealed several critical insights into the challenges and effectiveness of the usage analytics process model implemented in the IBM Academic Cloud project as shown in the Table 7.2.

One of the main challenges identified was the difficulty in analyzing user interactions due to unclear definitions of platform features. Developers emphasized that the lack of clear feature definitions made it difficult to map user interactions accurately and derive meaningful insights from the data. One developer noted, "Without clear definitions of platform features, it was nearly impossible to map user interactions accurately and derive meaningful insights from the data."

The absence of a clear mapping of user interactions to application features further complicated the requirements analysis stage. Developers struggled to understand which actions performed by users were related to specific features, leading to incomplete or unclear requirements. This sentiment was echoed by another developer who stated, "We spent a lot of time trying to figure out how user actions translated to specific features, which made the requirements gathering process much more complicated."

Data sources identification and selection posed significant challenges during the requirements analysis and designing stages. The diverse types of data available required developers to invest considerable time and effort in choosing the right data sources and formats. This created a bottleneck in the development process, as developers had to ensure data quality and relevance before proceeding. One developer remarked, "Choosing the right types, sources, and formats of data required a tremendous amount of time and effort, creating a bottleneck in our development process."

During the designing stage, developers faced difficulties due to the lack of clear feature definitions, which impacted the accuracy and effectiveness of the design efforts. Assumptions had to be made about feature definitions, which could potentially affect the overall design. Despite using standard design tools and techniques, the effectiveness was limited by the complexities in data selection and processing.

In the building stage, the development of usage data extraction and analytics components was resource-intensive and time-consuming. Developers highlighted the significant effort required for cleaning and transforming large volumes of data, which extended the development timeline. Integrating different components was also challenging due to inconsistencies and gaps in the

Table 7.2: Summary of results obtained for the evaluation of the Use Case 1: IBM Academic Cloud

Stage	**Theme**	**Key Findings**	**Developer Statements**
Planning	Lack of Clear Feature Definitions	The absence of well-defined platform features hindered effective planning and goal setting for understanding user behavior and interaction.	"Without clear definitions of platform features, it was nearly impossible to map user interactions accurately and derive meaningful insights from the data."
Requirements Analysis	Challenges in Mapping User Interactions to Features	Difficulty in correlating user actions with specific features led to incomplete or unclear requirements.	"We spent a lot of time trying to figure out how user actions translated to specific features, which made the requirements gathering process much more complicated."
Requirements Analysis	Time-Consuming Data Selection Process	Identifying and selecting appropriate data sources required significant time and effort, creating a bottleneck in the development process.	"Choosing the right types, sources, and formats of data required a tremendous amount of time and effort, creating a bottleneck in our development process."
Designing	Assumptions Due to Lack of Clear Feature Definitions	Assumptions made about feature definitions affected the accuracy and effectiveness of the design efforts.	"Designing the data extraction and analytics processes was challenging because we had to make assumptions about feature definitions, which could affect the accuracy of our designs."
Building	Resource-Intensive Data Preparation	Extensive effort was needed for data cleaning and transformation, prolonging the development timeline and requiring significant resources.	"The development phase was prolonged because cleaning and transforming the data was a massive task, requiring significant resources."
Testing	Difficulties in Validating Components	Validation was challenging without clear mappings of user interactions to features, necessitating thorough testing and multiple iterations.	"Testing was thorough, but without a clear mapping of user interactions to features, it was difficult to validate our components accurately."
Overall Process Model	Effectiveness and Areas for Improvement	The process model was effective but faced challenges in data collection and analysis. Developers highlighted the need for better feature definitions, automated data cleaning, and enhanced user interaction mapping.	"Better initial feature definitions and automated data cleaning processes would have made a big difference in our efficiency."

data sources, necessitating additional efforts to ensure seamless integration.

Testing and evaluation posed further challenges. The absence of a clear mapping of user interactions to features made it difficult to validate the accuracy of the components. Developers had to conduct thorough testing and multiple iterations to refine the components based on user feedback. One developer noted, "Testing was thorough, but without a clear mapping of user interactions to features, it was difficult to validate our components accurately."

Overall, the usage analytics process model was deemed effective in its objectives but was significantly hindered by challenges in data collection and analysis. Developers appreciated the structured approach and collaboration within the team, but the need for better-defined features, automated data cleaning processes, and enhanced mapping of user interactions was evident. Comparing the process model to previous methods, developers acknowledged its more structured framework but highlighted the increased initial setup and effort required for data preparation. The interviews provided valuable insights into the challenges and effectiveness of the usage analytics process model. Key challenges included unclear feature definitions, the absence of user interaction mapping, and time-consuming data selection and preparation processes. Despite these challenges, the structured approach and collaborative efforts were seen as strengths. Recommendations for improvement focused on better feature definitions, automated data cleaning, and enhanced user interaction mapping.

7.3 Evaluation of the Case Study 2: IBM Watson Workspace

The evaluation of Case Study 2 aimed to address the research question *(RQ2): "What are the key challenges faced by developers in identifying and utilizing usage data and key metrics related to feature prioritization effectively within the software platform?"* Building on insights from Case Study 1, this study explored the implementation of an advanced usage analytics method in IBM Watson Workspace, focusing on its impact on the software development process and the challenges encountered. Descriptive analysis was employed to systematically categorize and interpret the qualitative data obtained from interviews with the development team.

7.3.1 Goal of the case study 2

The primary goal of this case study was to identify and understand the challenges that developers face in leveraging usage data for effective feature prioritization within the IBM Watson

Workspace. Specifically, the study aimed to evaluate the application of advanced usage analytics and its impact on the software development process. This included examining how developers identify relevant usage data sources, apply analytics metrics, and develop tools like feature-action maps to enhance decision-making. By addressing these aspects, the case study sought to provide insights into the effectiveness of the usage analytics method and recommend improvements for its continuous refinement.

7.3.2 Design of case study 2

To gather detailed insights, structured interviews were conducted with the participants. The Table 7.3 provides a summary of the demographics of the participants involved in Case Study 2: IBM Watson Workspace. This information is crucial for understanding the context of the feedback provided during the structured interviews. The table includes details on the participants' roles, experience levels, technical expertise, tenure in the company, and tenure in their current roles. These demographics highlight the diversity and expertise of the team, which is essential for evaluating the implementation and effectiveness of the advanced usage analytics method.

Table 7.3: Demographics of Participants for Case Study 2: IBM Watson Workspace

Participant	**Role**	**Experience Level**	**Technical Expertise**	**Tenure in the Company**	**Tenure in Current Role**
Participant 1	Operational Manager	Senior (15+ years)	Project Management, Agile Methodologies	5-7 years	3-4 years
Participant 2	Architect	Senior (12+ years)	Software Architecture, Cloud Computing	3-5 years	2-3 years
Participant 3	Senior Developer	Senior (10+ years)	Python, Java, Data Analytics	5-7 years	4-5 years
Participant 4	Developer	Mid-Level (5-7 years)	JavaScript, Full-Stack Development	2-3 years	1-2 years
Participant 5	Developer	Mid-Level (5-7 years)	Java, Python, DevOps	2-3 years	1-2 years

A semi-structured interview format was chosen to balance guided questions with open-ended responses, allowing for comprehensive feedback while ensuring consistency across interviews. Questions were designed to explore specific aspects of the usage analytics method, including data identification, metrics application, feature-action mapping, and implementation challenges. Interviews were conducted with the participants using the semi-structured format. Each interview was recorded and transcribed to ensure accurate data collection. The questions focused on the following areas:

1. **Usage Data Identification and Extraction:**
   (a) What are the primary sources of usage data for IBM Watson Workspace?
   (b) How do you integrate data from multiple sources?
   (c) What challenges do you face in ensuring data consistency and accuracy?
2. **Application of Analytics Metrics:**
   (a) Which metrics do you use to analyze user interactions (e.g., frequency, timespent, consistency)?
   (b) How do you correlate these metrics with changes in the software?
   (c) What challenges do you encounter in applying these metrics effectively?
3. **Development of Feature-Action Maps:**
   (a) How do you identify and map user actions to specific features?
   (b) What processes do you follow to develop and maintain feature-action maps?
   (c) How do feature-action maps influence your development decisions?
4. **General Implementation Challenges:**
   (a) What are the main challenges you face in implementing usage analytics?
   (b) How do you address these challenges?
   (c) What improvements would you recommend for the usage analytics method?

All interview responses were meticulously documented and categorized to facilitate systematic analysis. This involved transcribing each interview in detail and organizing the transcripts into a manageable format for further analysis. Specific statements from participants were highlighted to provide direct evidence of their experiences and insights. This step was crucial to ensure that the qualitative data was accurately captured and ready for in-depth analysis.

A descriptive analysis approach was employed to systematically categorize and interpret the qualitative data obtained from the interviews. This involved the following steps:

1. **Coding:** Transcripts were thoroughly reviewed, and relevant segments were coded based on predefined stages of the UA method (Usage Data Identification and Extraction, Application of Analytics Metrics, Development of Feature-Action Map, and General Implementation Challenges) and recurring themes identified in the feedback. For instance, statements related to challenges in data integration were coded under "Data Integration Challenges," while insights on the effectiveness of specific metrics were coded under "Metrics Application."

2. **Theme Development:** Broader themes were developed to represent major recurring topics or issues discussed by the participants. These themes were developed inductively from the data, ensuring they accurately reflected the participants' experiences. For example, themes such as "Data Consistency Issues," "User Behavior Insights," and "Feature Prioritization Strategies" were identified.

3. **Data Organization:** Coded data was organized under each theme and process stage, facilitating a structured interpretation. This organization helped identify patterns and trends within and across different stages of the process model. For example, data under the theme "Metrics Application" was further broken down into sub-categories such as "Frequency Analysis," "Timespent Analysis," and "Consistency Analysis."

4. **Summarization:** Detailed summaries were written for each theme and stage, highlighting key points and including specific statements from participants to provide direct evidence. This step ensured that the analysis was grounded in the actual experiences and perspectives of the participants. Each summary included both qualitative insights and quantitative data where applicable, providing a comprehensive view of the findings.

Based on the feedback and insights from the interviews, the UA method was refined. This included updating the process model to Version 2, which incorporated new stages and improved techniques for data collection, metrics application, and feature-action mapping. For instance, a new stage "Identify Usage Data Sources" was added to systematically explore and document all potential sources of user interaction data. The changes to the UA method were documented in detail, including the rationale for each change and the expected benefits. This documentation provided a clear record of the iterative improvement process and the evolving understanding of usage analytics in software development. Each change was accompanied by examples and detailed explanations of how it improved the overall method.

7.3.3 Results of the case study 2

The implementation of the Usage Analytics (UA) method in IBM Watson Workspace aimed to identify key challenges developers face in leveraging usage data for effective feature prioritization and refine the UA method. This section presents the detailed findings from the case study, focusing on the application of the UA method, key observations, and the resultant improvements to the method. The insights were derived from structured interviews with the development team, analysis of usage data, and iterative refinement of the UA method. The evaluation of the case study was structured around the three main components of the UA method: Usage Data Identification and Extraction,

To provide a comprehensive overview of the interview responses, the following Table 7.4 presents a quantitative analysis of the key themes and the number of participants who highlighted each theme. The quantitative analysis helps to highlight the frequency of specific themes and challenges mentioned by participants, giving a clear picture of the key areas of focus and concern.

Table 7.4: Quantitative Analysis of Interview Responses for Case Study 2

Theme	**Number of Participants**	**Percentage of Total Participants**	**Phrases and Words Used**
Data Integration Challenges	3	60%	“complex integration”, “data source alignment”, “tool compatibility”
Effort Required for Comprehensive Data Collection	4	80%	“time-consuming”, “resource-intensive”, “manual effort needed”
Utility of Frequency Analysis	3	60%	“useful insights”, “action tracking”, “frequency metrics”
Utility of Timespent Analysis	4	80%	“valuable duration insights”, “time engagement”, “timespent metrics”
Utility of Consistency Analysis	2	40%	“behavior patterns”, “consistency tracking”, “repeated actions”
Need for Additional Context-Specific Metrics	5	100%	“additional metrics”, “contextual data”, “specific insights”
Importance of Ongoing Validation of Feature-Action Map	4	80%	“continuous validation”, “regular updates”, “evolving behaviors”
Challenges in Integrating Usage Analytics into Workflow	5	100%	“workflow integration”, “process adaptation”, “seamless incorporation”

Usage Data Identification and Extraction: Various sources of usage data were identified, including native and third-party monitoring tools, which provided a comprehensive view of user interactions with the application. This led to the addition of a new stage in the Usage Analytics Method version 2, "Identify Usage Data Sources." This stage involves systematically exploring the application architecture to pinpoint where relevant data is being collected, whether from in-built monitoring tools or third-party services. This ensures that all potential sources of user interaction data are considered, providing a more comprehensive understanding of user behavior. Three out of five participants (60%) mentioned the complexity of integrating different data sources, with phrases like "complex integration," "data source alignment," and "tool compatibility" being frequently used. Participant 2 stated, *"Integrating data from various sources was more challenging than anticipated, especially aligning the data formats and ensuring compatibility with our existing systems".* Additionally, four participants (80%) highlighted the effort required for comprehensive data collection, describing it as "time-consuming" and "resource-intensive." Participant 3 noted, *"The manual effort needed to ensure data accuracy and consistency was significant, taking up a considerable amount of our development time."*

Application of Analytics Metrics: The application of frequency and timespent analysis provided valuable insights into how often and how long users engaged with different features of the application. Participants found frequency analysis useful, with three 60% highlighting the insights it provided into action tracking. Phrases like "useful insights" and "frequency metrics" were common. Participant 4 commented, *"The frequency metrics gave us clear visibility into which features were being used the most, helping us prioritize those for further development".* Four participants (80%) emphasized the value of timespent analysis, using terms such as "valuable duration insights" and "time engagement." Participant 1 mentioned, *"Timespent metrics were crucial in understanding user engagement levels, particularly for features we suspected were underutilized".*

Development of Feature-Action Map: The integration of the process to develop a feature-action map was a significant result of the case study. This process is part of the "Identify Features of the Application" stage and involves feature identification, action mapping, data correlation, visualization, and validation, providing a structured approach to understanding user interactions with specific features. An important outcome was the creation of a feature-action map, which provides a detailed visualization of user interactions with specific features. This process is supported by academic literature emphasizing the importance of understanding user behavior in software development. Participant 3 emphasized, *"The feature-action map was instrumental in visualizing how users interact with different features, allowing us to identify and prioritize areas that needed improvement"*. Participant 2 added, *"By correlating user actions with specific features, we could better understand the context of user interactions and make more informed decisions"*. Participants stressed the need for additional context-specific metrics, with all five participants (100%) suggesting the incorporation of more nuanced metrics. Phrases like "additional metrics," "contextual data," and "specific insights" were frequently mentioned. Participant 4 stated, *"We realized that while the existing metrics were helpful, additional context-specific metrics could provide even deeper insights into user behavior and preferences"*. The importance of ongoing validation and refinement of the feature-action map was highlighted by four participants (80%), who emphasized "continuous validation" and "evolving behaviors." Participant 1 remarked, *"The feature-action map needs to be continuously updated and validated to ensure it reflects current user behavior accurately."*

Challenges in Implementation: Despite the clear benefits, implementing advanced usage analytics presented several challenges. Integrating with multiple data sources, including in-built monitoring tools and third-party services, required significant effort to ensure consistent and accurate data collection. Mapping user actions to specific features and correlating this data with usage metrics required careful planning and execution to ensure meaningful and actionable results. All five participants (100%) mentioned the difficulties of integrating these processes seamlessly into existing workflows, with phrases such as "workflow integration," "process adaptation," and "seamless incorporation." Participant 5 noted, *"Integrating usage analytics into our existing workflow was challenging, as it required adapting our processes and training the team to use new tools effectively"*. These challenges highlight the practical difficulties in embedding advanced analytics within established development practices. Participants also discussed the need for ongoing validation and refinement of the feature-action map. As user behaviors and preferences evolve, the development team must continuously monitor and update the map to ensure it remains accurate and relevant. This requires a commitment to iterative testing and feedback loops, as well as the ability to quickly adapt to changes in user behavior. Participant 3 stated, *"Maintaining an accurate and up-to-date feature-action map is a continuous effort that requires regular validation and adjustments based on user feedback."*

The structured interviews provided deep insights into the experiences and challenges faced by the development team when implementing the usage analytics method. The responses underscored the importance of comprehensive data integration, the utility of specific analytics metrics, and the need for a flexible and evolving feature-action map. By systematically analyzing these responses, the case study highlighted critical areas for improvement in the usage analytics method and provided actionable recommendations for its refinement and future application. Participants emphasized the importance of ongoing validation and the integration of additional context-specific metrics to enhance the understanding of user behavior. The challenges of integrating usage analytics into existing workflows were also highlighted, necessitating adaptations and training to effectively use the new tools. Overall, the feedback from the development team was instrumental in refining the usage analytics method and ensuring its practical applicability in real-world software development scenarios.

7.4 Evaluation of Key Metrics - Case Study 2 and Case Study 3

A total of 5 users participated in the experiments with IBM Watson Workspace and since each user was the developer of the Workspace Application, no specific tasks were provided to interact with the application. Each user was asked to interact with the application in a free-style manner and try to simulate how an actual end-user of the application would use the application. The primary aim of the observations with the IBM Watson Workspace application is to explore the usage data, devise a procedure to collect the identified usage data in an efficient manner and test the analysis process of the collected usage data using the UA analysis metrics as discussed in Chapter 6 Section 6.2.1. A total of 9 users participated in the experiments with Odoo Notes. The users were asked to interact first with Odoo version 10 and then with Odoo version 11. The users were asked to perform the tasks given in the experiment task list. The usage data from both experiments were collected and analyzed.

Table 7.5: Summary of the experiments conducted

Application name	**Application Version**	**Number of users**	**Total number of actions performed**	**UA method version applied**
IBM Watson Workspace	Beta 52.0	5	113	1
Odoo Notes	10	9	1175	1, 2
Odoo Notes	11	9	1193	2, 3

Table 7.5 summarizes the experiments conducted in the following order: UA Method and

version 1 of the experiment design as shown in Figure 7.1 and UA Method and version 2 of the experiment design as shown in Figure 7.2 are applied over IBM Watson Workspace application. The UA Method and versions 2 and 3 of the experiment design as shown in Figure 7.2 and Figure 7.3 are applied over the Odoo Notes application. The results of the experiments are presented and discussed in this chapter.

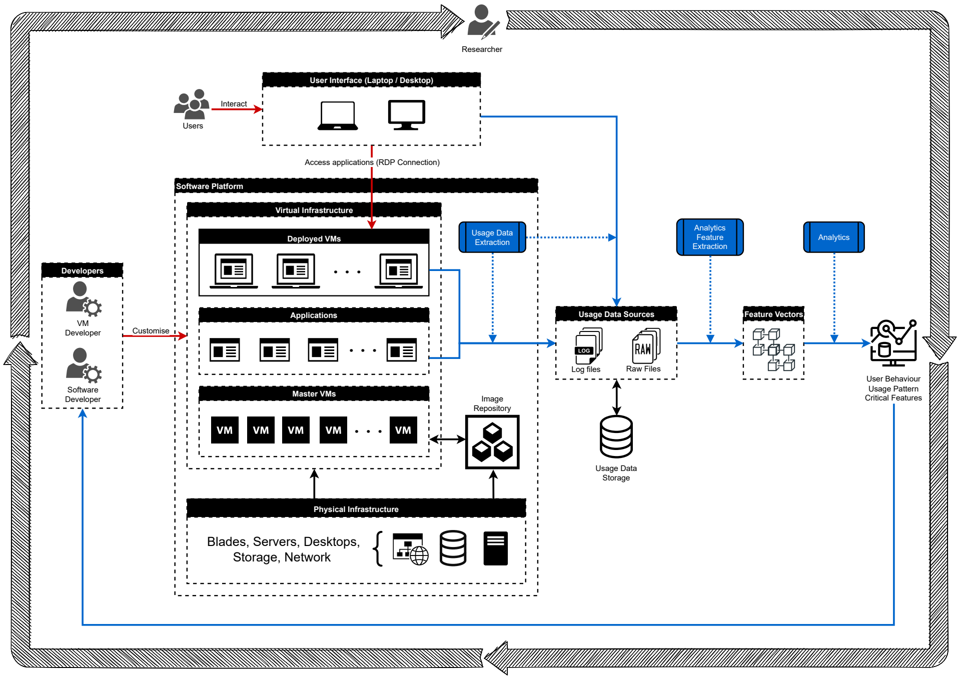


Figure 7.1: Usage Analytics Method Design - Version 1 resulted from the exploration of IBM Academic Cloud Application.

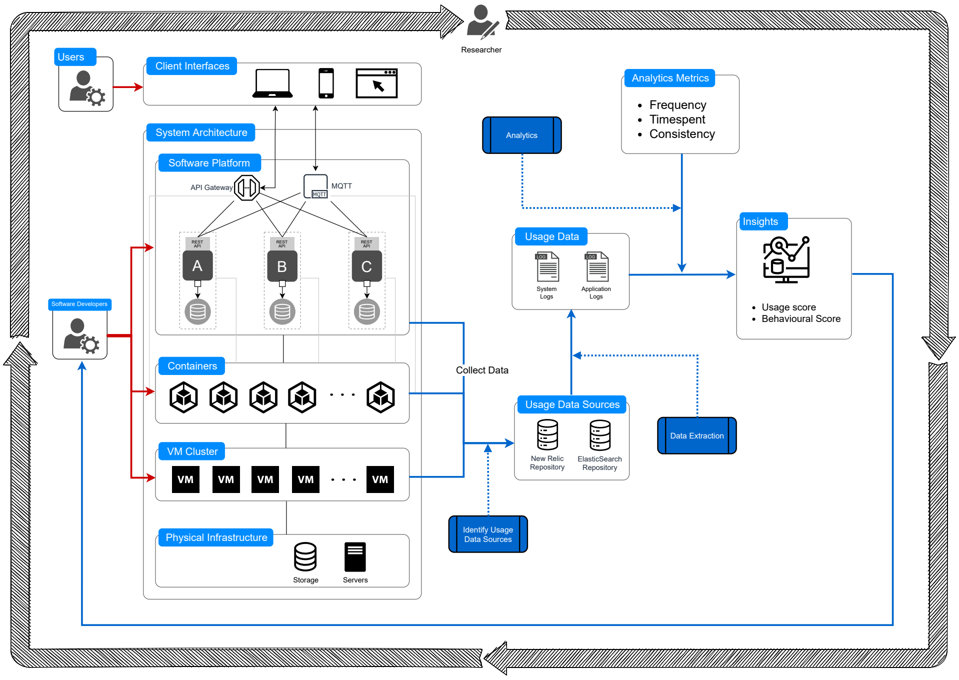


Figure 7.2: Usage Analytics method - Version 2 resulting from the work with IBM Watson Workspace Application

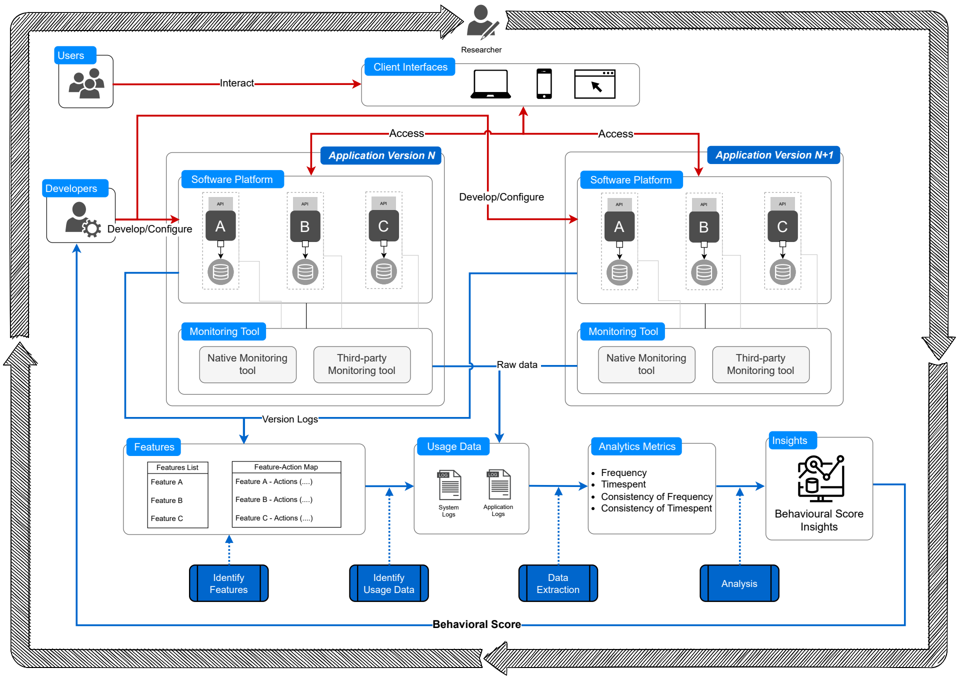


Figure 7.3: Usage Analytics method - Version 3 as a result of work with Odoo Notes

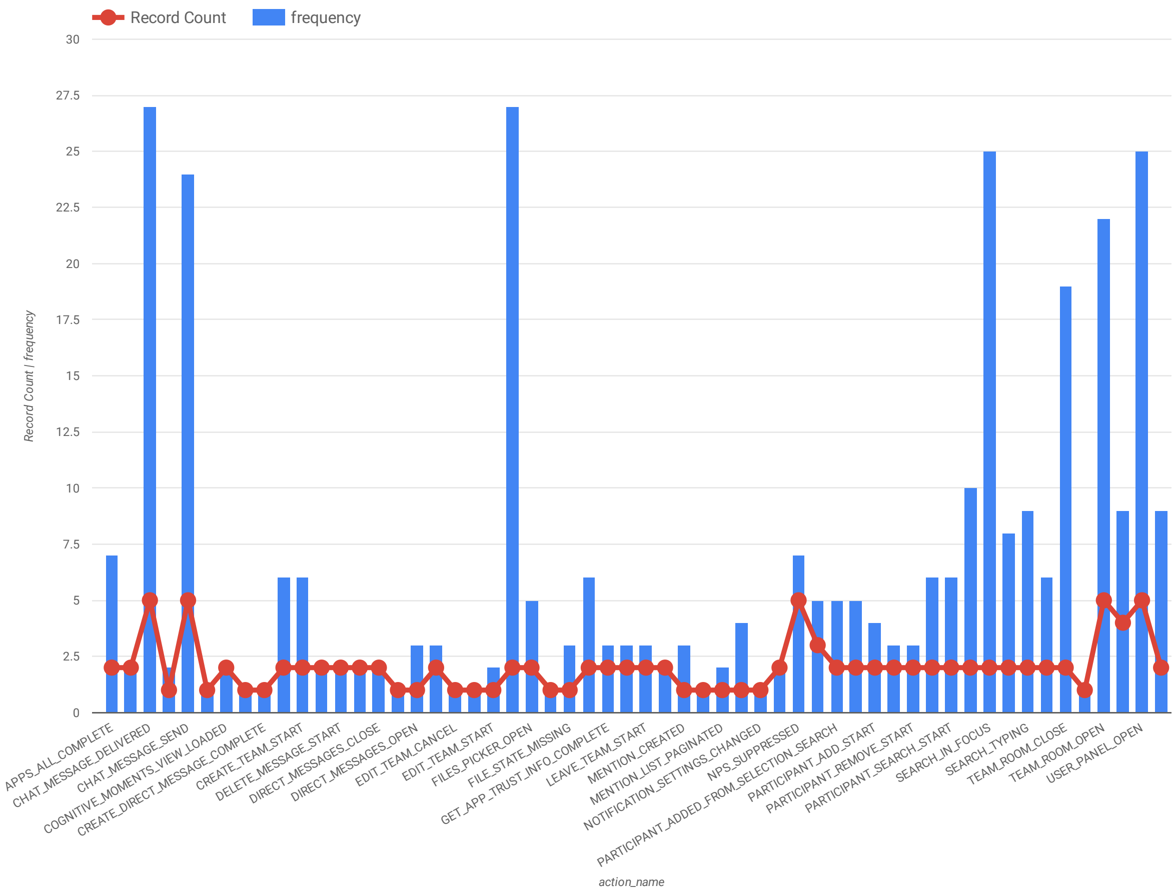


Figure 7.4: Illustration of Frequency of Actions performed by the participants in IBM Watson Workspace experiment

7.4.1 Results from the experiment with IBM Watson Workspace

Figure 7.4 denotes the frequency measure of all actions performed by all users of the experiments with the IBM Watson Workspace application. A total of 5 users participated in the observation activity and a total of 113 actions were performed by the participants. This includes some actions which are a result of direct interaction of the users with the application and some actions that are performed by the application on behalf of the user. The *frequency* measure of the actions is represented by the blue vertical bars and the red dots represent the *record count*. The value *Frequency* is a measure of the number of times an action is performed by the user over a period of time. The value *Record Count* is a measure of how many users performed an action. This analysis forms the first step in the development of the analysis process for the Usage Analytics method. As shown in Figure 7.5, each action when hovered upon shows the detailed specific value of the frequency and record count metrics. For example, the actions

CHAT_MESSAGE_SEND, EXIT_SEARCH, TEAM_ROOM_OPEN, TEAM_ROOM_CLOSE are highlighted in the Figure 7.5 are associated with the following values:

1. CHAT_MESSAGE_SEND: Frequency = 24, Record Count = 5
2. EXIT_SEARCH: Frequency = 27, Record Count = 2
3. TEAM_ROOM_OPEN: Frequency = 22, Record Count = 5
4. TEAM_ROOM_CLOSE: Frequency = 19, Record Count = 2

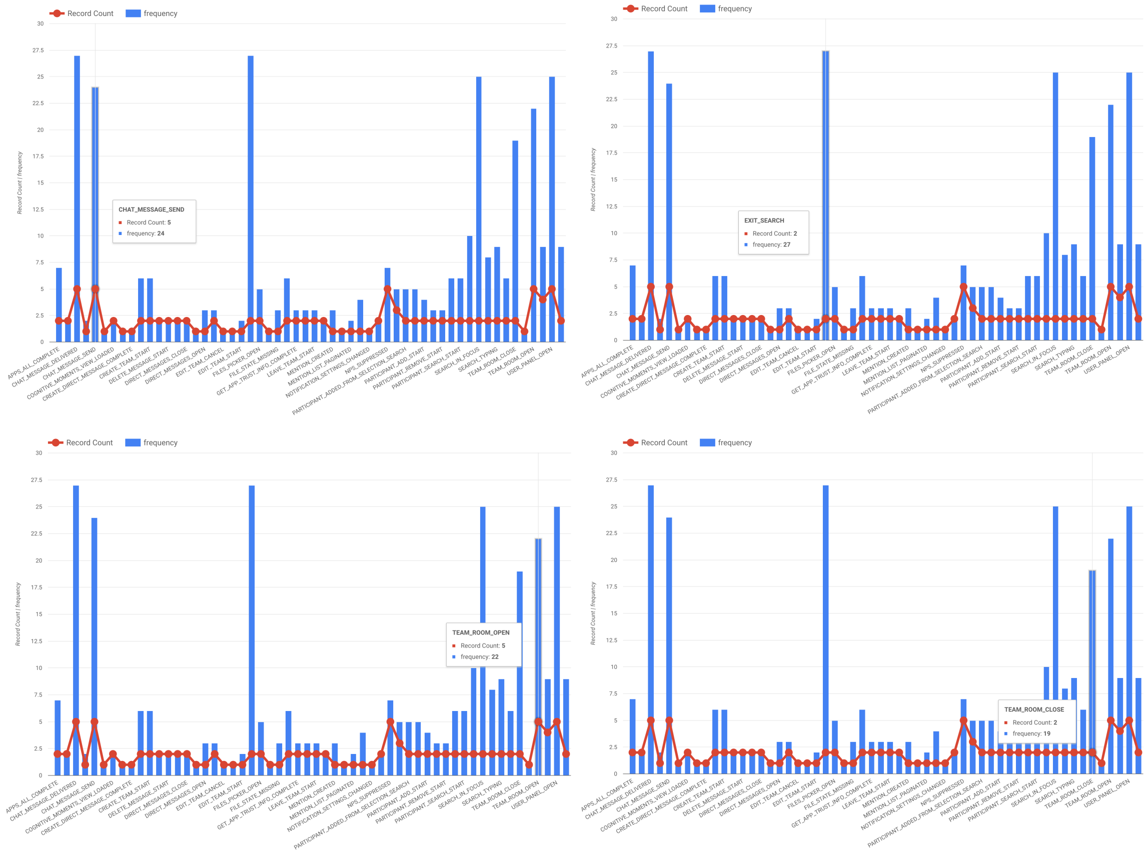


Figure 7.5: Illustration of Frequency of Actions performed by the participants in IBM Watson Workspace experiment. Some example actions are highlighted: CHAT_MESSAGE_SEND, EXIT_SEARCH, TEAM_ROOM_OPEN, TEAM_ROOM_CLOSE.

From the above example, all 5 users performed the action CHAT_MESSAGE_SEND 24 times each. The action EXIT_SEARCH was performed by 2 users 27 times. The action TEAM_ROOM_OPEN was performed by 5 users 22 times. The action TEAM_ROOM_CLOSE was performed by 2 users 19 times. Considering the 5 actions in the example, the actions CHAT_MESSAGE_SEND and TEAM_ROOM_OPEN are the most frequently performed actions by the users. Each action is performed once minimum by each user and the following 10 actions were only performed once. The actions EXIT_SEARCH and TEAM_ROOM_CLOSE are the least frequently performed actions by the users. The following insights can be obtained from this measure:

1. Any change made to the Workspace application which affects the actions CHAT_MESSAGE_SEND and TEAM_ROOM_OPEN would have a significantly higher impact on the user behaviour. The reason for this conclusion is that these actions are performed by all 5 users who participated in the experiment as denoted by the Record Count score.

2. Any change made to the Workspace application which affects the following actions EXIT_SEARCH and TEAM_ROOM_CLOSE would have a significantly lower impact on the user behaviour compared to the actions CHAT_MESSAGE_SEND and TEAM_ROOM_OPEN. The reason for this conclusion is that these actions are performed by only 2 users who participated in the experiment as denoted by the Record Count score.

3. As a result of the above two observations, the actions CHAT_MESSAGE_SEND and TEAM_ROOM_OPEN can be assigned a higher priority for the analysis of the user behaviour compared to the actions EXIT_SEARCH and TEAM_ROOM_CLOSE.

4. The actions CHAT_MESSAGE_SEND and TEAM_ROOM_OPEN are the most frequently performed actions by the users. This means that the users are more likely to perform these actions compared to other actions. This can be used as a starting point for the analysis of user behaviour.

5. Exploring the frequency score of the actions can be used to further prioritise the actions for the analysis of the user behaviour. For example, the actions CHAT_MESSAGE_SEND and TEAM_ROOM_OPEN are performed by all 5 users. However, the action CHAT_MESSAGE_SEND is performed 24 times by all 5 users whereas the action TEAM_ROOM_OPEN is performed 22 times by all 5 users. A priority rank can be assigned to the actions based on the frequency score and record count. The action CHAT_MESSAGE_SEND can be assigned a higher priority compared to the action TEAM_ROOM_OPEN for the analysis of the user behaviour.

6. Based on the analysis so far, the following priority scores can be assigned to the actions:

   (a) CHAT_MESSAGE_SEND: Frequency = 24, Record Count = 5, Priority = 1

   (b) TEAM_ROOM_OPEN: Frequency = 22, Record Count = 5, Priority = 2

   (c) EXIT_SEARCH: Frequency = 27, Record Count = 2, Priority = 3

(d) `TEAM_ROOM_CLOSE`: Frequency = 19, Record Count = 2, Priority = 4

Analysing these measures on their own reveals a small part of user behaviour. However, the updated version of the Usage Analytics method, as discussed in Chapter 5 Section 5.3.3 and Section 5.4.3, provides a way to analyse the user behaviour in a more meaningful way by combining frequency scores with other metrics.

The updated version 2 of the Usage Analytics method included two additional metrics to the frequency and record count metrics. The two additional metrics are *Timespent* and *consistency.* The timespent metric is a measure of the time spent by the user on performing an action. The consistency metric is a measure of how consistent the user is in performing an action. The Usage Analytics method was further updated in version 3 to analyse the frequency, timespent and consistency metrics between two consecutive versions of the application and a new analysis metric called *Behavioural Score* is included. The updated version 3 of the Usage Analytics method is discussed in Chapter 5 Section 5.4.3. The consistency metric is calculated by calculating the change in frequency and change in timespent between two consecutive versions of the application. The Behavioural Score metric is a measure of how much the user behaviour has changed between two consecutive versions of the application. The Behavioural Score metric is calculated by combining the frequency, timespent and consistency metrics. The Behavioural Score metric is calculated for each user and each action and compared between two consecutive versions of the application. Such comparative analysis would help us understand how the behaviour of each user changed between the two versions of the application. Ideally, the score would be allocated to each feature of the application by aggregating the scores of all actions associated with that feature.

7.4.2 Results from the experiment with Odoo Notes

The results of the experiment with the Odoo Notes application are presented in this section. The results are presented in the form of the updated version 3 of the Usage Analytics method. A total of 9 users participated in the experiments. A total of 1175 actions were performed by the participants with Odoo Notes version 10 and a total of 1193 actions were performed by the participants with Odoo Notes version 11. These actions are now analysed in order to understand the change in usage patterns of users between Odoo Notes version 10 and 11. The results of the experiments with Odoo Notes are presented in the form of the following graphs:

Figure 7.6 shows the total number of occurrences of all actions performed by each user

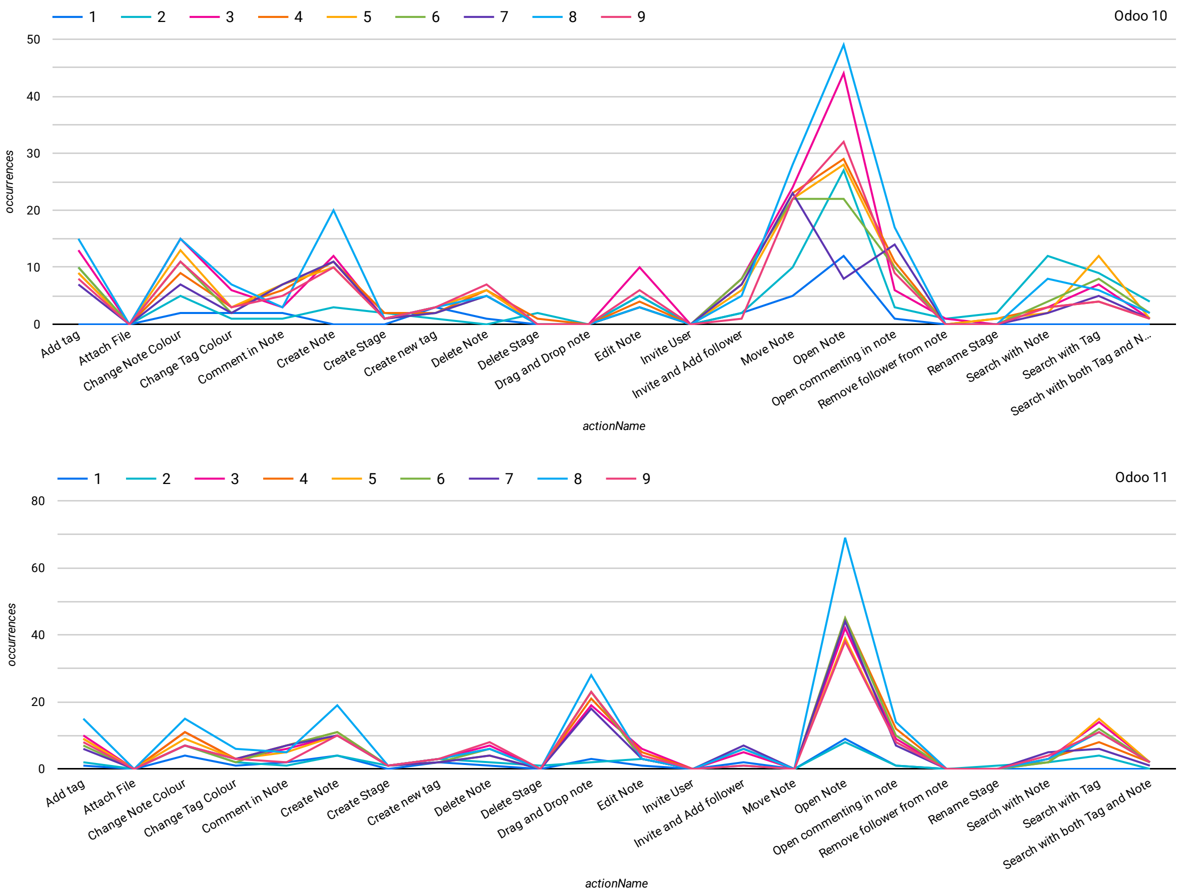
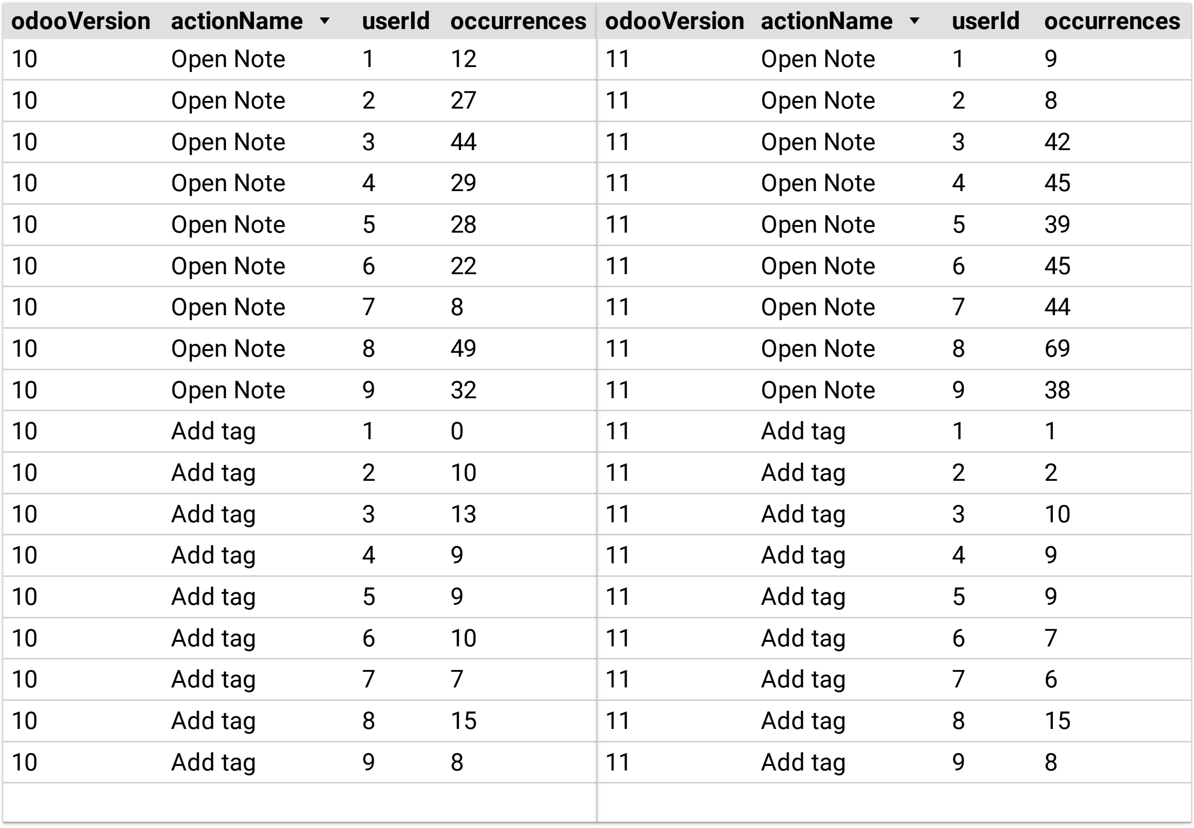


Figure 7.6: Total number of occurrences of all actions performed by each user shown separately (Top: Odoo Notes version 10, Bottom: Odoo Notes version 11)

shown separately. The top graph in Figure 7.6 shows the total number of occurrences of all actions performed by each user of the experiments with Odoo Notes version 10. The bottom graph in Figure 7.6 shows the total number of occurrences of all actions performed by each user of the experiments with Odoo Notes version 11. The total number of occurrences of all actions performed by each user of the experiments with Odoo Notes version 10 is 1175. The total number of occurrences of all actions performed by each user of the experiments with Odoo Notes version 11 is 1193. These results show that the number of occurrences of all actions performed by each user of the experiments with Odoo Notes version 11 is slightly higher than the number of occurrences of all actions performed by each user of the experiments with Odoo Notes version 10. Given the same set of tasks to perform, users needed to perform more actions with Odoo Notes version 11 compared to Odoo Notes version 10.

Considering the individual occurrences scores of the user, for example, User 8 performed the action `Add tag` 15 times each with both Odoo Notes version 10 and Odoo Notes version 11. However, the action `Open Note` was performed 49 times with Odoo Notes version 10 and

69 times with Odoo Notes version 11. This means that user 8 took 20 tries more to perform a set of tasks where the action of opening a note had to be performed with Odoo Notes version 11 compared to Odoo Notes version 10. An assumption could be made here that any changes implemented to the action `Add Tag` in version 11 of the Odoo Notes application did not affect the user behaviour. However, the changes implemented to the action `Open Note` in version 11 of the Odoo Notes application affected the user behaviour. The validity of this assumption can be tested by considering the occurrences scores of the actions `Add Tag` and `Open Note` performed by other users who participated in the experiment. The Occurrence scores of the actions `Add Tag` and `Open Note` performed by all users are as shown in Figure 7.7. The following comparative study can be made by exploring these scores:



odooVersion	actionName ▾	userId	occurrences	odooVersion	actionName ▾	userId	occurrences
10	Open Note	1	12	11	Open Note	1	9
10	Open Note	2	27	11	Open Note	2	8
10	Open Note	3	44	11	Open Note	3	42
10	Open Note	4	29	11	Open Note	4	45
10	Open Note	5	28	11	Open Note	5	39
10	Open Note	6	22	11	Open Note	6	45
10	Open Note	7	8	11	Open Note	7	44
10	Open Note	8	49	11	Open Note	8	69
10	Open Note	9	32	11	Open Note	9	38
10	Add tag	1	0	11	Add tag	1	1
10	Add tag	2	10	11	Add tag	2	2
10	Add tag	3	13	11	Add tag	3	10
10	Add tag	4	9	11	Add tag	4	9
10	Add tag	5	9	11	Add tag	5	9
10	Add tag	6	10	11	Add tag	6	7
10	Add tag	7	7	11	Add tag	7	6
10	Add tag	8	15	11	Add tag	8	15
10	Add tag	9	8	11	Add tag	9	8

Figure 7.7: Comparative chart for the Occurrence scores of the actions `Add Tag` and `Open Note` performed by all users. The table on the left shows the scores for Odoo Notes version 10 and the table on the right shows the scores for Odoo Notes version 11.

1. The assumption holds true for users 4, 5, 6, 7, 8 and 9 for the action `Open Note`, that is 6 out of 9 users and users 4, 5, 8 and 9 for the action `Add Tag`, that is 4 out of 9 users. However, user 7 performed the action `Add Tag` one less time with Odoo Notes version 11

compared to Odoo Notes version 10 and user 1 performed the action only once with Odoo Notes version 11 and did not perform the action with Odoo Notes version 10. Allowing for the possibility of human error, the assumption holds true for 7 out of 9 users.

2. Some outliers can be observed in the scores of the action `Open Note` performed by users 3 and 7 between Odoo Notes versions 10 and 11 with a difference of -19 actions and 36 actions respectively. Similarly, some outliers are observed in the scores of the action `Add tag` performed by the users 2, 3 and 6 with a difference of -8, -3 and -3 actions respectively. Although these irregularities could be the result of human error, these outliers are further evaluated using the post-experiment survey results.

Combining the values of the action occurrence scores of all users, the sum of action occurrences is as shown in Figure 7.8. These scores show the overall impact of the changes implemented to the Odoo Notes application between version 10 and version 11. The following insights can be drawn from the scores:

1. The action `Drag and Drop note` was not performed by any of the users with Odoo Notes version 10. However, the action `Drag and Drop note` was performed 160 times in total by all users with Odoo Notes version 11.

2. The action `Move Note` was performed 179 times in total by all users with Odoo Notes version 10. However, the action `Move Note` was not performed by any of the users with Odoo Notes version 11.

3. Considering the context of the usage of the feature in Odoo Notes where users have two options to move a note from one stage (column) to another, either by dragging and dropping the note or by clicking on a dedicated button, this feature is named **Kanban View** as discussed in Section 4.5.2. The action `Drag and Drop note` was performed 160 times in total by all users with Odoo Notes version 11. This means that the users preferred to drag and drop the note to move it from one stage to another rather than clicking on the `Move Note` button. On the other hand, the action `Move Note` was performed 179 times in total by all users with Odoo Notes version 10. This means that the users preferred to click on the `Move Note` button to move the note from one stage to another rather than dragging and dropping the note with Odoo Notes 10.

4. Based on the above observations, the changes made to the application in version 11 of Odoo Notes application to the feature **Kanban View** affected the user behaviour. Specifically, changes implemented to the action `Drag and Drop note` would have a positive impact on the user behaviour and changes implemented to the action `Move Note` would have a negative impact on the user behaviour.

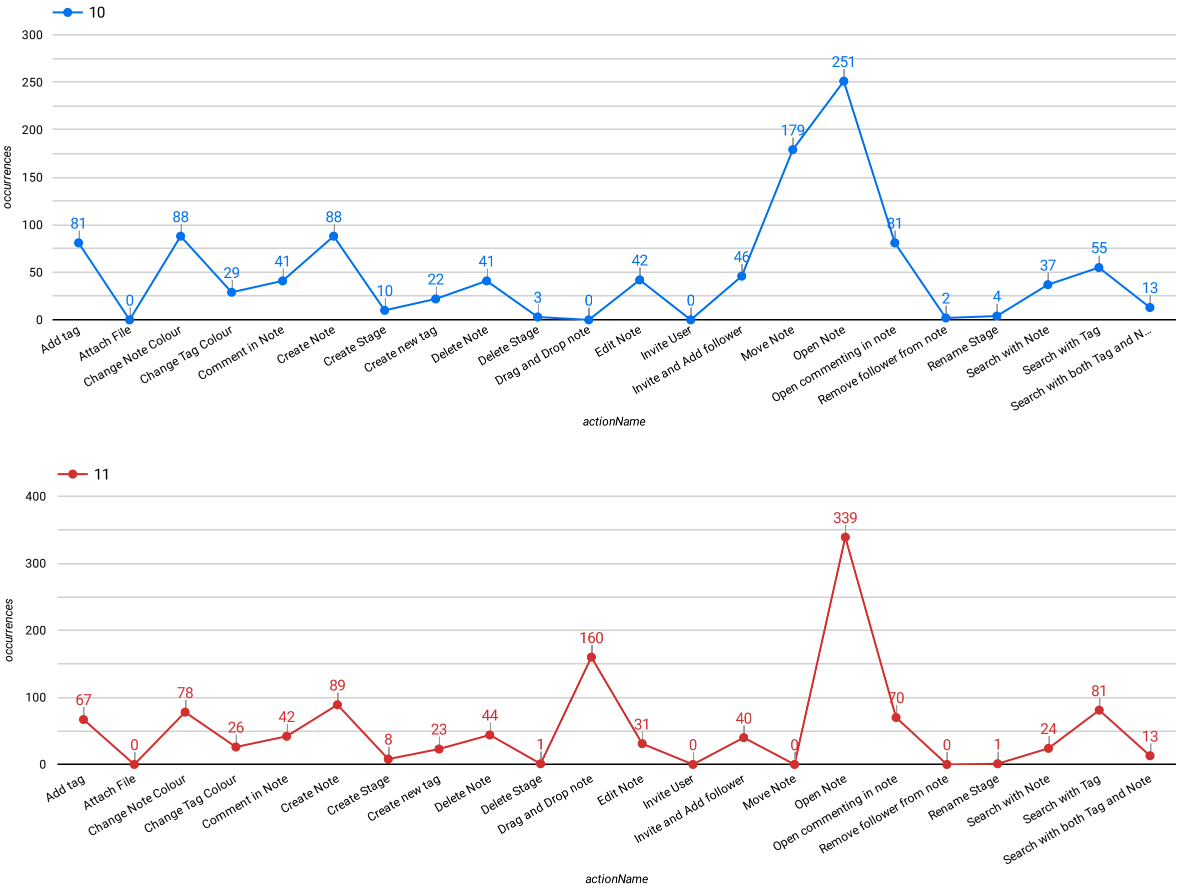


Figure 7.8: Sum of occurrences of all actions performed by all users shown (Top: Odoo Notes version 10, Bottom: Odoo Notes version 11)

Similarly, comparing the action occurrence scores of all users for all actions performed with Odoo Notes version 10 and Odoo Notes version 11 as shown in Figure 7.9, the impact of changes implemented by the developers on the user behaviour can be deduced. Furthermore, applying the metrics Frequency, Timespent and Consistency to the action occurrence scores discussed above would provide further insights into the impact of the changes implemented by the developers on user behaviour. For example, we can create user personas by identifying the users who would be most and least affected by the changes implemented by the developers. Future changes could be tested separately for separate user personas to ensure that the changes do not affect the user behaviour of the other user personas. Additionally, the intensity of the changes could be tested for different user personas based on the impact of the changes on user behaviour.

Addressing the research questions, the UA method can be used to answer the following questions:

1. *RQ3: Can the developed feature usage analysis method be applied to quantify the users' behavioural changes?* - The developed Usage Analytics method provides a novel way to use the usage metrics Frequency, Timespent and Consistency to assign numerical scores to the changes in the way different features of the software are used by the users. These scores can be used to compare the changes in user behaviour between two versions of the same application.

2. *RQ3(a) Does the developed method meet the requirements identified as defined in RQ1?* - The Usage Analytics method includes a process to identify the usage data and the sources of the identified usage data addressing the research question *RQ1(a) Which data types can be used to identify the usage of features in software applications?*. The Usage Analytics method also provides a process to extract the usage data from the identified usage data sources and how to analyse them using the analytics metrics addressing the research question *RQ1(b) What are the challenges to extracting and analysing the usage data according to the scientific literature?*.

Frequency is the number of times an action is performed by a user over a period of time. It is calculated by dividing an action's occurrence value by the user's total time over an action. Based on the context of usage of a feature, a higher frequency value could indicate that the user is more engaged and familiar with the feature. However, if a change in the frequency value is observed, it could indicate that the user behaviour has changed. Figure 7.10 shows the sum of frequency measures of all actions performed by each user of the experiments with the Odoo Notes application. The top graph in Figure 7.10 shows the sum of the frequency measures of all actions performed by each user of the experiments with Odoo Notes version 10. The bottom graph in Figure 7.10 shows the sum of the frequency measure of all actions performed by each user of the experiments with Odoo Notes version 11.

Since the frequency metric is calculated by dividing the occurrence value of an action by the total time spent by the user over an action and each user participated in the experiment with each version of the application only once. The total time period remains the same. As a consequence, the sum of the frequency measure of all actions performed by each user of the

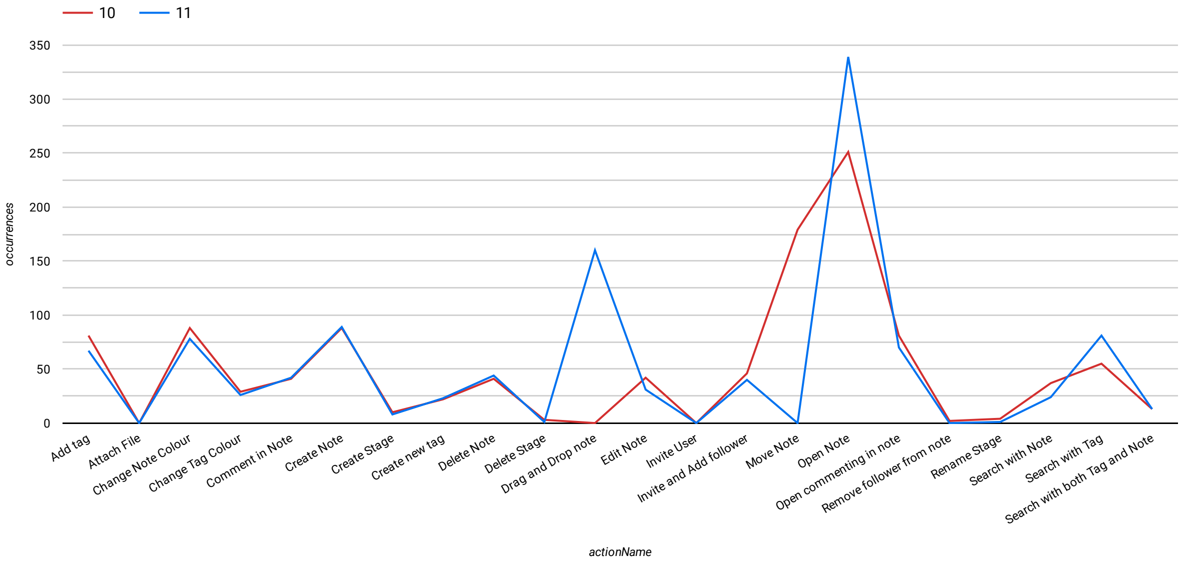


Figure 7.9: Sum of occurrences of all actions performed by all users compared between Odoo Notes version 10 and Odoo Notes version 11

experiments with Odoo Notes version 10 and Odoo Notes version 11 would lead to similar insights as the sum of occurrence scores of all actions performed by each user of the experiments with Odoo Notes version 10 and Odoo Notes version 11.

In addition to the insights gathered above, calculating the sum of all frequencies of all users with each action performed with Odoo Notes version 10 and Odoo Notes version 11 would provide further insights into the impact of the changes implemented by the developers on the user behaviour. Specifically, the changes in the combined measure of frequencies represent the overall change in usage of all users of the application. Figure 7.11 shows the sum of all frequencies of all users with each action performed with Odoo Notes version 10 and Odoo Notes version 11.

The frequency and time spent metrics are computed for specific user actions in both the previous and current versions of the application. These metrics capture two essential dimensions: how often a feature is used and the time invested by users while interacting with it. Together, they provide a holistic view of feature engagement. By subtracting the previous version's metric values from the current version's values, a comparative baseline is established. This difference

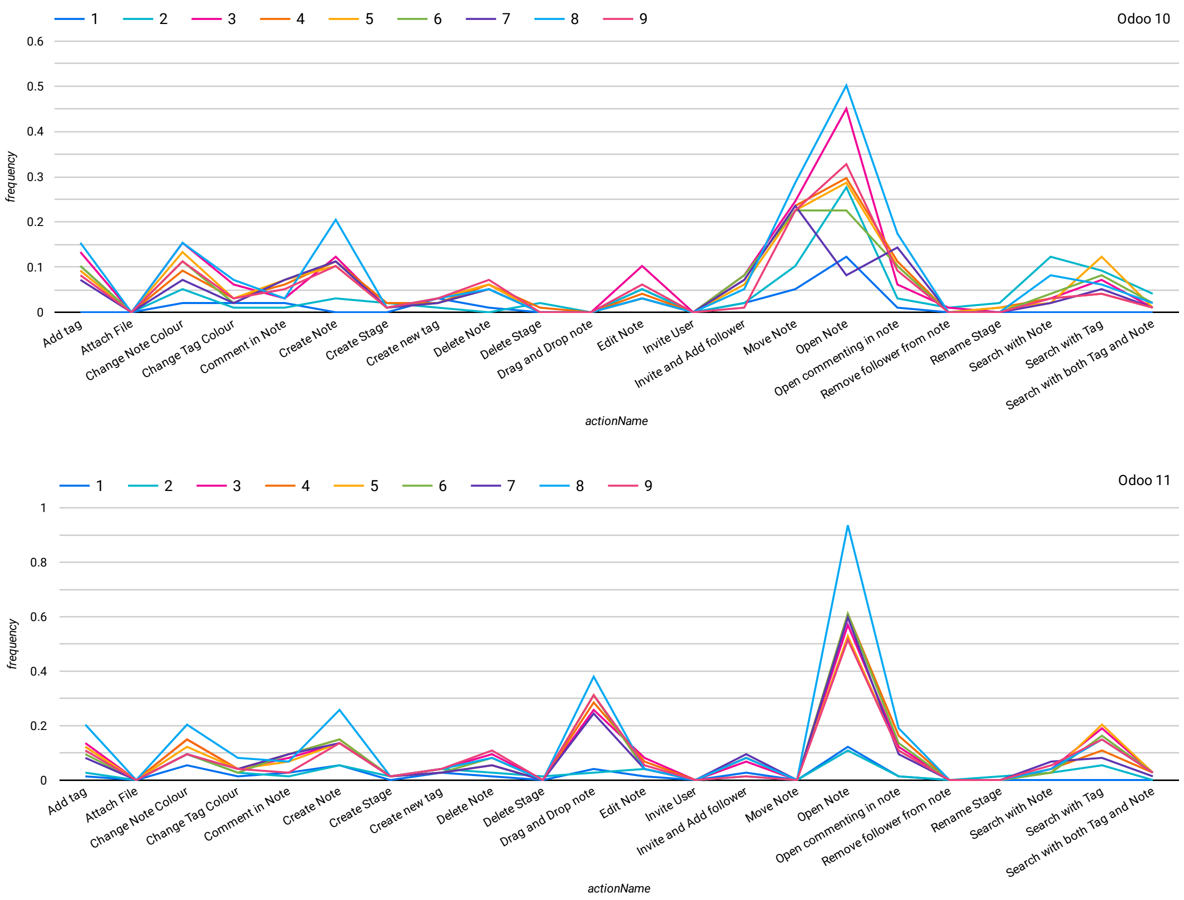


Figure 7.10: Sum of frequency measure of all actions performed by each user shown (Top: Odoo Notes version 10, Bottom: Odoo Notes version 11)

reflects whether user engagement with the feature has increased or decreased in the newer version. A negative result (lower engagement in the current version) indicates that users preferred the feature implementation or behavior in the previous version.

Consistency of Frequency refers to the difference in the frequency metric for a specific user action between the current and previous versions of the application. It quantifies how consistently users engage with a feature in terms of usage count after a software update. **Consistency of Time Spent** represents the difference in the time spent by users on a specific action across versions. It reflects how the time investment per interaction changes, highlighting shifts in user experience or efficiency.

A low or negative consistency metric in the context of this research suggests dissatisfaction or reduced efficiency in performing the action with the updated version. It does not necessarily indicate failure but rather points to user preference trends or potential usability issues. Conversely, a positive difference indicates improved engagement and likely user satisfaction with the newer version.

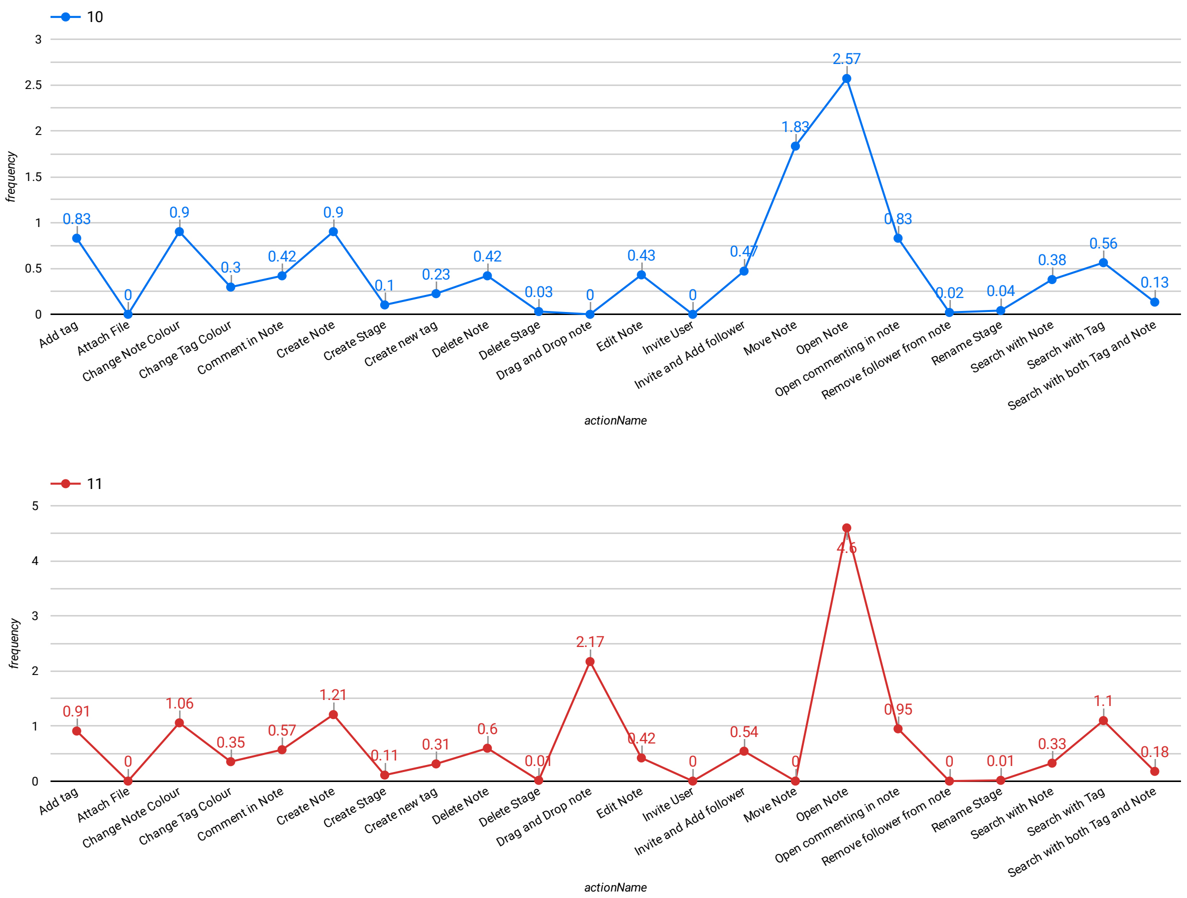


Figure 7.11: Sum of frequency measure of all actions performed by all users shown (Top: Odoo Notes version 10, Bottom: Odoo Notes version 11)

Based on these scores, the following observations can be made:

1. Based on the Consistency of Frequency scores between the versions of the application and the range of the scores, a numerical ranking of the actions can be deduced. The action with the lowest Consistency of Frequency score (greatest negative delta value of the frequency scores between the two consecutive versions) is ranked as 1, the action with the second-lowest score is ranked as 2 and so on.

2. Based on the above observations, as an example for rankings of the actions, action `Move Note` is seen to have the lowest Consistency of Frequency score between the versions of the application, that is 0 (Odoo Notes version 11) and 1.83 (Odoo Notes version 10), 0-1.83 = -1.83 Consistency of Frequency value. Therefore, action `Move Note` is ranked as 1. The action `Search with Note` is seen to have the second lowest Consistency of Frequency score between the versions of the application, that is 0.37 (Odoo Notes version 11) and 0.32 (Odoo Notes version 10), 0.37-0.32 = -0.05 Consistency of Frequency value. Therefore,

action `Search with Note` is ranked as 2 and so on. The rankings of the actions based on the Consistency of Frequency scores are shown in Table 7.6.

actionName	frequency (10)	frequency (11)	Consistency of Frequency	Rank
Move Note	1.833306508	0	-1.833306508	1
Search with Note	0.3789516246	0.3256457623	-0.05330586239	2
Rename Stage	0.0409677432	0.01356857343	-0.02739916978	3
Remove follower from note	0.0204838716	0	-0.0204838716	4
Delete Stage	0.0307258074	0.01356857343	-0.01715723398	5
Edit Note	0.4301613036	0.4206257762	-0.009535527403	6
Attach File	0	0	0	-
Invite User	0	0	0	-
Create Stage	0.102419358	0.1085485874	0.006129229407	7
Search with both Tag and Note	0.1331451654	0.1763914546	0.04324628914	8
Change Tag Colour	0.2970161382	0.3527829091	0.05576677088	9
Invite and Add follower	0.4711290469	0.5427429371	0.07161389024	10
Add tag	0.8295967999	0.9090944196	0.07949761973	11
Create new tag	0.2253225876	0.3120771888	0.0867546012	12
Open commenting in note	0.8295967999	0.9498001399	0.12020334	13
Comment in Note	0.4199193678	0.5698800839	0.1499607161	14
Change Note Colour	0.9012903505	1.058348727	0.1570583768	15
Delete Note	0.4199193678	0.5970172308	0.177097863	16
Create Note	0.9012903505	1.207603035	0.3063126845	17
Search with Tag	0.5633064691	1.099054448	0.5357479785	18
Open Note	2.570725886	4.599746392	2.029020506	19
Drag and Drop note	0	2.170971748	2.170971748	20

Table 7.6: The Consistency of Frequency scores between Odoo versions 10 and 11, and the rankings of the actions based on the Consistency of Frequency scores

The individual Timespent scores of each user for each action performed with both versions of the Odoo Notes application are represented in Figure 7.12 and the Timespent scores of all users combined for each action performed with both versions of the Odoo Notes application are represented in the Figure 7.13.

A similar approach to the *Timespent* analysis is followed to rank the actions based on the Consistency of the Timespent measure between Odoo Notes versions 10 and 11. The rankings of the actions based on the Consistency of Timespent are shown in Table 7.7. The analysis shows that the action with the lowest value of Consistency of Timespent score is ranked as 1, the action with the second lowest value of Consistency of Timespent score is ranked as 2 and so on. The action `Move Note` is seen to have the lowest Consistency of Timespent between the versions of the application, that is 0 (Odoo Notes version 11) and 1595.37 (Odoo Notes version 10), 0-1595.37 = -1595.37 Consistency of Timespent value. Therefore, action `Move Note` is ranked as 1. The action `Create Note` is seen to have the second lowest Consistency of Timespent between the versions of the application, that is 131.71 (Odoo Notes version 11) and 1392.06 (Odoo Notes version 10), 131.71-1392.06 = -1260.35 Consistency of Timespent. Therefore, action `Create`

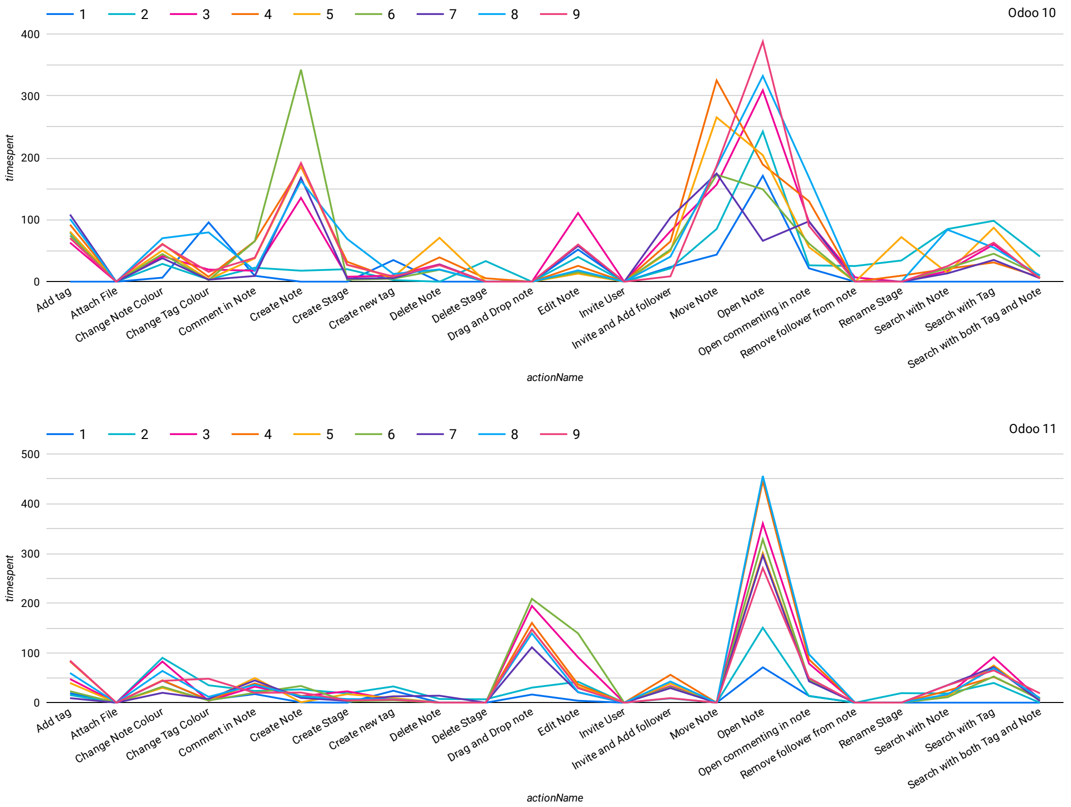


Figure 7.12: Individual Timespent scores of each user for each action performed with both versions of the Odoo Notes application

`Note` is ranked as 2. The action `Drag and Drop note` is seen to have the highest Consistency of Timespent between the versions of the application, that is 2.17 (Odoo Notes version 11) and 0 (Odoo Notes version 10), 2.17-0 = 2.17 Consistency of Timespent value. Therefore, action `Drag and Drop note` is ranked as 20.

Figure 7.14 shows the scores Consistency of Frequency and Consistency of Timespent compared with each other for all actions. Each action of the application is ranked based on the analysis using the metrics Frequency, Consistency of Frequency, Timespent and Consistency of Timespent. The rankings represent the extent of changes in the way users interacted with each action between the two versions of the application. Positive scores are in favour of the actions in version 11 of the Odoo Notes application and negative scores are in favour of the actions in version 10 of the application. Positive scores of the Consistency of Frequency metric of action indicate that the users interacted more with the action in version 11 of the application than in version 10. Similarly, positive scores of the Consistency of Timespent metric of action indicate that the users spent more time interacting with the action in version 11 of the application than in version 10. The rankings are assigned based on the scores of the actions. The actions with the

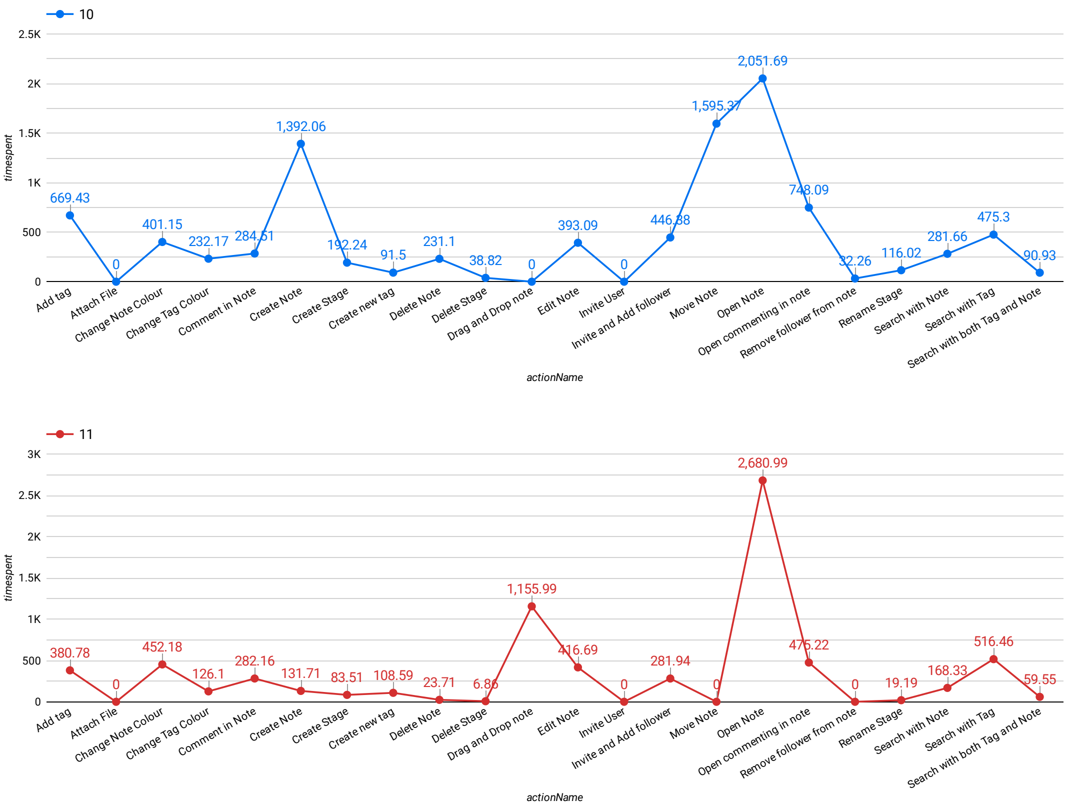


Figure 7.13: Sum of Timespent scores of all users for each action performed with both versions of the Odoo Notes application

lowest scores are ranked first and the actions with the highest scores are ranked last, transcribing the scores into rankings.

Following modifications made by the developers to version 10 of the application, version 11 was released. If the actions in version 10 are preferred by the user, such actions are ranked lower (close to 1, ranks start from 1). Note, from the context of usage of the application, higher frequency and higher timespent values with actions are treated as positive preferences of the user. If for any application, the opposite is true, an inverse of the scores can be calculated easily.

The rankings differ for each action when considered in isolation since the metrics Frequency and Timespent follows separate scales of measurement. That is, the ranking of an action based on the Consistency of the Frequency metric may not be the same as the ranking of the same action based on the Consistency of the Timespent metric. Therefore, the rankings of the actions based on the Consistency of Frequency and Consistency of Timespent metrics are combined by taking the average of the ranks of the actions based on the two metrics as shown in Figure 7.8.

Based on the average rankings calculated, users of the experiments with the Odoo Notes application demonstrate a negative impact on their behaviour with action `Move Note` as the

actionName	timespent (10)	timespent (11)	Consistency of Timespent	Rank
Move Note	1595.374366	0	-1595.374366	1
Create Note	1392.056177	131.709223	-1260.346954	2
Add tag	669.4324243	380.7813053	-288.651119	3
Open commenting in note	748.0859563	475.2221642	-272.8637922	4
Delete Note	231.0988841	23.712363	-207.3865211	5
Invite and Add follower	446.3797917	281.940506	-164.4392858	6
Search with Note	281.6616054	168.3294818	-113.3321235	7
Create Stage	192.2435629	83.51401258	-108.7295504	8
Change Tag Colour	232.1729901	126.1033735	-106.0696166	9
Rename Stage	116.018924	19.19365382	-96.82527018	10
Remove follower from note	32.26249504	0	-32.26249504	11
Delete Stage	38.8183043	6.86315918	-31.95514512	12
Search with both Tag and Note	90.92868209	59.54830289	-31.3803792	13
Comment in Note	284.5105414	282.1575766	-2.352964878	14
Attach File	0	0	0	-
Invite User	0	0	0	-
Create new tag	91.49551892	108.5865567	17.09103775	15
Edit Note	393.0921113	416.6926537	23.60054231	16
Search with Tag	475.3014901	516.4567277	41.15523767	17
Change Note Colour	401.1518185	452.1825666	51.03074813	18
Open Note	2051.693581	2680.988213	629.2946324	19
Drag and Drop note	0	1155.9897	1155.9897	20

Table 7.7: The difference in timespent measures between Odoo versions 10 and 11, and the rankings of the actions based on the difference in timespent measure

most impacted action of the application. The action `Search with Tag` is the least impacted action. The average rankings of the actions are normalised on a scale of 1 to 20. The normalised average rankings are shown in the last column of Table 7.8. The normalised average rankings are used to determine the impact of the actions on the users' behaviour.

1. Identify High-Priority Actions: Actions like “Move Note” (Rank 1), “Attach File” (Rank 2), and “Add Tag” (Rank 3) are top priorities as they show the highest consistency in frequency and time spent metrics. These actions are critical to user workflow and should be the focus of initial development efforts.

2. Analyze the Context of Each Action: Understanding the context in which these actions are performed can provide deeper insights. For instance, “Move Note” involves organizing notes, which is essential for maintaining an efficient workflow. Enhancements in this area can significantly improve user productivity.

3. Feature Enhancements and Bug Fixes: High-priority actions identified can be further scrutinized for potential enhancements or existing issues. For example, if “Attach File” is highly ranked, developers should ensure that this feature is robust, user-friendly, and free of bugs.

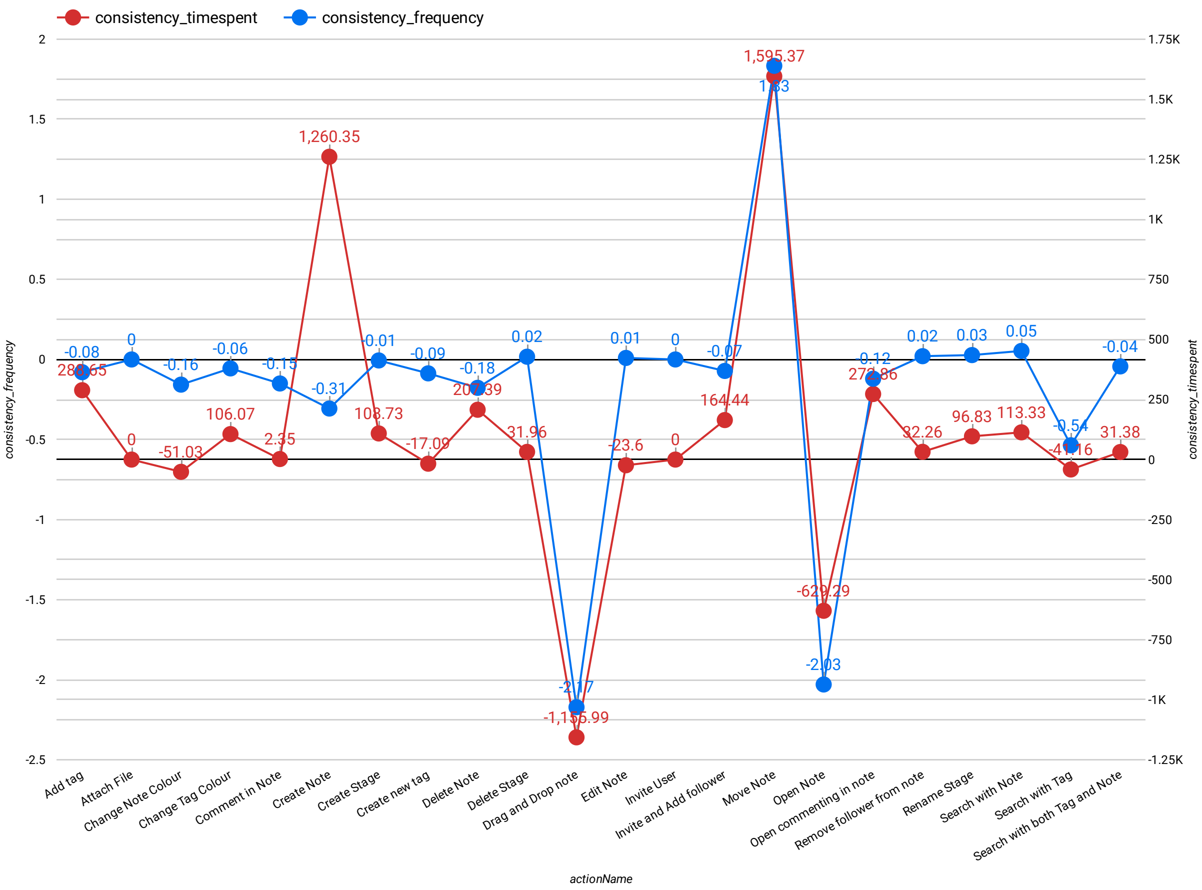


Figure 7.14: The scores of the actions based on the Consistency of Frequency and Consistency of Timespent metrics compared against each other

4. Design User-Centric Improvements: For actions like “Change Tag Colour” and “Create Stage,” which have mid-level ranks, consider user feedback for making these features more intuitive or adding functionalities that enhance their usage.

5. Address Low-Priority Actions: Actions like “Search with Tag” and “Open Note,” which rank lower, indicate less frequent or less critical usage. These features can be deprioritized unless they are essential for specific user segments or align with strategic goals.

The Usage Analytics method is applied to IBM Watson Workspace application in the production environment and Odoo Notes version 10 and 11 as shown in Chapter 5 and obtained the insights and results as shown here. The results show that the method is applicable to any software product and can be used to identify the most impacted actions and features of the application.

actionName	CF	Rank (F)	CT	Rank (T)	Avg. Rank	Avg. Rank (Normalised)
Move Note	-1.833	1	0	-	1	1
Attach File	0.000	-	-1260.3	2	2	2
Add tag	0.079	11	-1595.4	1	6	3
Change Tag Colour	0.056	9	-272.9	4	6.5	4
Create Stage	0.006	7	-108.7	8	7.5	5
Delete Stage	-0.017	5	-96.8	10	7.5	5
Change Note Colour	0.157	15	-288.7	3	9	6
Create new tag	0.087	12	-164.4	6	9	6
Edit Note	-0.010	6	-32	12	9	6
Comment in Note	0.150	14	-207.4	5	9.5	7
Remove follower from note	-0.020	4	23.6	16	10	8
Rename Stage	-0.027	3	41.2	17	10	8
Search with Note	-0.053	2	629.3	19	10.5	9
Invite and Add follower	0.072	10	-31.4	13	11.5	10
Create Note	0.306	17	-113.3	7	12	11
Delete Note	0.177	16	-106.1	9	12.5	12
Open commenting in note	0.120	13	0	-	13	13
Search with both Tag and Note	0.043	8	51	18	13	13
Invite User	0.000	-	-2.4	14	14	14
Drag and Drop note	2.171	20	-32.3	11	15.5	15
Open Note	2.029	19	17.1	15	17	16
Search with Tag	0.536	18	1156	20	19	17

Table 7.8: Rankings of the actions based on the Consistency of Frequency (CF) and Consistency of Timespent (CT) metrics, and the average of the two rankings

7.5 Results of the Surveys Conducted

The purpose of design science evaluation is to determine how well the artefact under evaluation performs for achieving its intended purpose (Ghezzi et al., 2014). Applied to the current research context this means that the method resulting has to be evaluated and assessed against relevant criteria depending on its purpose. The method is designed and developed to address the research gap identified in the academic literature while also working closely with the developers of IBM Academic Cloud, IBM Watson Workspace and working with Odoo Notes application. Specifically, data collection, extraction and analysis activities of the proposed Usage Analytics method are evaluated to the applications following the approach of case study research (Pagano and Bruegge, 2013). In the first step, the method is applied in practice, usage data sources are identified, and existing application monitoring tools are exploited and adopted to extract the usage data. Thereafter, the analysis method designed and developed is applied to the Workspace application. Additionally, to verify the developed method is applicable not only to IBM applications but in general to any software product, the method is applied to the Odoo Notes application. Data collection is performed by conducting experiments where participants are recruited from the development team of the IBM Watson Workspace application to act as users of the application and simulate the usage of the application. For Odoo Notes, participants are recruited at Dublin City University from the postgraduate and doctoral student groups. The participants are asked to perform a set of tasks in the application and their actions are recorded. The data is then analysed using the developed method. Once the experiment is completed, each participant is asked to complete a survey. The results of the analysis are then compared with the survey results, thus evaluating against the research questions and the research gap identified in the literature review.

User ID	What is your experience with Odoo Notes?	What is your experience with note-taking (kanban-style) applications?	Expertise
1	Only heard of it but have not used it	Use it occasionally	Intermediate
2	Use it occasionally	Use it occasionally	Expert
3	Use it occasionally	Use it everyday	Expert
4	Only heard of it but have not used it	Use it everyday	Intermediate
5	Never heard of it	Used it just a few times	Novice
6	Only heard of it but have not used it	Only heard of it but have not used it	Novice
7	Only heard of it but have not used it	Use it everyday	Expert
8	Only heard of it but have not used it	Used it just a few times	Novice
9	Only heard of it but have not used it	Use it occasionally	Intermediate

Table 7.9: Familiarity of the users with Odoo Notes and similar types of applications

The results of the surveys conducted with Odoo Notes are presented in Appendix E.1. Regarding the familiarity of the Odoo Notes application with the users, only one user was completely unfamiliar with the application, and 2 users have previously used the application before the experiment. The remaining 7 users have heard about the application but never used it before. However, only one user never used any similar application before. 1 user a similar application just a few times, 3 users used a similar application a few times before the experiment, 3 users use a similar application occasionally and 3 users use a similar application every day. Table 7.9 shows the familiarity of the users with the Odoo Notes application and similar applications. Based on the answers to questions 2 and 3 of the survey, users 2, 3 and 7 are considered experts, users 1, 4, and 9 are considered intermediate and users 5, 6 and 8 are considered as novices. To ensure the results are not biased, the participants with different levels of expertise and experience are recruited and asked to complete the survey after the experiment. The survey results are then compared with the results of the analysis to evaluate the method.

User ID	Which version of Odoo would you prefer in general?	The reason for the above choice
1	11	easy to interact with the features; quick to find the required features; simpler than the other
2	11	easy to interact with the features; quick to find the required features;visually better; better response time
3	10	easy to interact with the features; better response time; simpler than the other
4	11	easy to interact with the features; quick to find the required features; simpler than the other
5	11	easy to interact with the features; visually better;better response time; simpler than the other
6	11	quick to find the required features; visually better; simpler than the other
7	11	quick to find the required features; better response time; simpler than the other
8	10	easy to interact with the features; usually better; better response time
9	11	easy to interact with the features; quick to find the required features
	10 - 22.2%, 11 - 77.78%	

Table 7.10: Preference of the users for the different versions of the application in general and the reason for their preference

The next two questions of the survey “Which version of Odoo would you prefer in general?” and “The reason for the above choice” are used to understand the preference of the users for the different versions of the application in general and the reason for their preference. The answers are presented in Table 7.10. The results show that the majority of the users (78%) prefer version 11 of the application. The reason for the preference is that version 11 is easier to interact with the features, quick to find the required features and is simpler than the other version.

actionName	**Odoo 10**	**Odoo 11**	**Either**	**None**
Move Note	100%	0%	0%	0%
Change Note Colour	88.88%	11.11%	0%	0%
Attach File	77.78%	11.11%	0%	11.11%
Edit Note	66.66%	22.22%	0%	0%
Comment in Note	66.66%	33.33%	0%	0%
Delete Stage	55.55%	0%	44.44%	0%
Create Stage	44.44%	0%	55.55%	0%
Add tag	44.44%	11.11%	33.33%	11.11%
Change Tag Colour	44.44%	22.22%	33.33%	0%
Create new tag	44.44%	22.22%	22.22%	11.11%
Remove follower from note	44.44%	33.33%	11.11%	11.11%
Open commenting in note	22.22%	55.55%	22.22%	0%
Delete Note	22.22%	66.66%	11.11%	0%
Invite and Add follower	22.22%	77.77%	0%	0%
Invite User	11.11%	55.55%	22.22%	11.11%
Rename Stage	11.11%	66.66%	22.22%	0%
Create Note	11.11%	66.66%	11.11%	0%
Open Note	11.11%	66.66%	22.22%	0%
Search with Note	11.11%	77.77%	11.11%	0%
Drag and Drop note	11.11%	88.88%	0%	0%
Search with Tag	11.11%	88.88%	0%	0%
Search with both Tag and Note	0%	88.88%	11.11%	0%

Table 7.11: Preference of the users for the different versions of the application for each action

The next question of the survey “Which version of Odoo would you prefer for each action?” is used to understand the preference of the users for the different versions of the application for each action. The results are presented in Table 7.11. The results show that the majority of the users prefer version 11 of the application for all the actions. The reason for the preference is that version 11 is easier to interact with the features, quick to find the required features and is simpler than the other version. Specifically, 100% of the users prefer the Odoo version 10 with the action `Move Note`, 78% of the users prefer the Odoo version 11 with the action `Search with Note`, 89% of the users prefer the Odoo version 11 with the action `Search with both Tag and Note`, 89% of the users prefer the Odoo version 11 with the action `Search with Tag`, 89% of the users prefer the Odoo version 11 with the action `Drag and Drop note` and 89% of the users prefer the Odoo version 11 with the action `Search with Tag`. Translating these preferences, for the action `Move Note` the changes implemented by the developers in version 11 of the application are not liked by the users. For the action `Search with Note`, `Search with both Tag and Note`, `Search with Tag` and `Drag and Drop note`, the changes implemented by the developers in version 11 of the application are liked by the users.

actionName / User ID	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**Avg. Difficulty**
Move Note	4	5	3	5	5	4	5	4	3	**4.22**
Change Tag Colour	3	3	4	4	4	5	5	4	5	**4.11**
Attach File	5	5	2	4	4	3	5	4	4	**4.00**
Add tag	3	3	4	4	5	4	5	3	4	**3.89**
Create new tag	4	2	3	4	3	4	4	2	4	**3.33**
Change Note Colour	3	3	2	3	5	3	4	4	2	**3.22**
Delete Stage	4	4	1	2	3	4	3	4	3	**3.11**
Edit Note	3	4	2	5	2	4	2	1	3	**2.89**
Comment in Note	3	4	1	3	4	5	1	1	2	**2.67**
Create Stage	3	1	2	3	4	2	2	2	4	**2.56**
Remove follower from note	3	5	2	2	2	2	2	1	2	**2.33**
Invite and Add follower	2	2	3	1	2	2	1	2	4	**2.11**
Search with both Tag and Note	2	2	1	1	3	4	2	1	2	**2.00**
Open commenting in note	1	1	5	2	1	2	3	1	1	**1.89**
Rename Stage	2	3	3	2	1	2	1	1	1	**1.78**
Create Note	2	1	4	1	3	2	1	1	1	**1.78**
Search with Note	1	2	4	1	2	1	1	1	2	**1.67**
Delete Note	2	1	2	2	1	1	2	2	2	**1.67**
Invite User	2	2	2	1	2	2	1	1	2	**1.67**
Search with Tag	1	1	4	2	1	1	1	2	1	**1.56**
Open Note	2	2	1	1	1	1	2	2	1	**1.44**
Drag and Drop note	1	1	1	1	1	1	2	2	2	**1.33**

Table 7.12: Difficulty of each action as experienced by the user in Odoo 11 compared to Odoo 10

The last question of the survey “Compared to Odoo 10, how easy was it to perform the following actions in Odoo 11? 1: Very Easy, 2: Easy, 3: Neutral, 4: Difficult, 5: Very Difficult” is used to understand the ease of performing the each of the actions in the different versions of the application. The results are presented in Table 7.12. A higher score means the user found the action more difficult to perform in Odoo version 11 than compared to Odoo version 10. On the other hand, a lower score means the user found the action easier to perform in Odoo version 11 than compared to Odoo version 10. The results show that the action `Move Note` was the most difficult action to perform in Odoo version 11 compared to Odoo version 10. The action `Drag and Drop note` was the easiest action to perform in Odoo version 11 compared to Odoo version 10. Comparing the results of the Usage Analytics method as shown in Table 7.8 and the results from the survey presented in Table 7.11 and Table 7.12, The Usage Analytics method has successfully identified the actions that most affected the user, that is, `Move Note` and by grouping the actions based scores for the usage metrics, we can identify a group of actions that could be prioritised for improvement. If we consider the top 10 actions identified by the Usage Analytics method, we can see that 7 of the 10 actions are also confirmed by the users through the survey. The actions `Move Note`, `Drag and Drop note`, `Change Note Colour`, `Change Tag Colour`, `Attach File`, `Add tag`, `Create new tag`, `Delete Stage`, `Edit Note`, `Comment in Note` are all identified correctly by the method.

Application of the final version of the Usage Analytics method to the Odoo Notes application and evaluating the results of the analysis with the survey results presented with the following conclusions:

1. 7 out of the top 10 actions performed by the user were correctly identified by the Usage Analytics method as the actions impacted them the most by the changes made to the application by the developers. However, it is essential to note that while the accuracy of identifying user actions is a significant achievement, it is not the primary focus of this research. The main objective is to demonstrate the capability of the Usage Analytics method in systematically analyzing and prioritizing software features based on user interactions. The accuracy of identifying user actions could potentially be improved by incorporating more advanced, state-of-the-art metrics. Techniques such as machine learning algorithms for pattern recognition, natural language processing for understanding user feedback, and advanced statistical models for behavior prediction could enhance the precision of identifying user actions and their impacts. Nevertheless, the current metrics have proven sufficient to demonstrate the core functionality of the Usage Analytics method—showcasing its ability to detect significant user interactions and the corresponding impact of software changes.

2. The developed feature-action map can be used to identify the priority list of features based on the ranked user actions. This priority list provides a structured approach for developers to identify which features should be focused on in future development cycles. By analyzing user interactions and the resulting changes, the method offers a data-driven basis for decision-making, ensuring that the most critical features are prioritized. The ability to

generate a priority list is crucial for streamlining the development process. Developers can use this list to quickly determine which features require immediate attention, thereby reducing the time and effort needed for the decision-making process.

3. The priority list of features can be used by the developers for the future development of the application in the following ways:
   (a) The priority list of features can be used to include the features with the highest priority in the next development cycle, reducing the time and effort required for the decision-making process.
   (b) The priority list of features can be used to identify the users who are most affected by the changes made to the application, the reviews and feedback from these users can be filtered out and prioritised for the developers to address them first.

The research emphasizes the importance of identifying and analyzing user interactions rather than focusing solely on the accuracy of the metrics used. The primary goal is to demonstrate the utility of the Usage Analytics method in understanding user behavior and its impact on software development. By proving the method's capability to identify significant user interactions and generate actionable insights, the research lays the groundwork for further refinement and adoption in diverse software development contexts.

Future research can explore integrating more sophisticated metrics to enhance accuracy. However, the current study successfully showcases the core functionality of the Usage Analytics method, proving its potential as a valuable tool for developers to prioritize features and improve user satisfaction.

Chapter 8

Conclusion and Future Work

8.1 Conclusion

Research in past has established the importance of involving the users in the process of development of software applications (Chen et al., 2011; Hess et al., 2013; Pagano and Bruegge, 2013; Yaman et al., 2016) and the advantages of including feedback from users in the decision-making process (Fabijan et al., 2015; Krusche and Bruegge, 2014). Many user feedback-gathering tools and mechanisms exist and are actively used in the software development industry as discussed in Chapter 2. The current automatic feedback-gathering tools still predominantly focus on fixing bugs and crashes and maintaining the systems by gathering performance metrics. However, the ability to understand how users of a software application use different features based on the existing usage data is unclear and currently performed in an ad-hoc manner (Fitzgerald and Stol, 2017b; Tizard et al., 2022; Fabijan et al., 2016b). Software developers and/or software analysts still have to spend additional time and effort combing through enormous amounts of data from application logs to understand how users are affected by the recent changes implemented to the software. This thesis presents the usage analytics method that could be used by the developers of a software application to identify the impact of changes made to an application on the usage behaviour of the users. Specifically, changes in the usage behaviour exhibited by the user as a result of changes made to the application by software developers regardless of the magnitude of the change can be captured using the Usage Analytics method. Furthermore, the method when used in conjunction with traditional software feedback mechanisms could reduce the time delay caused by obtaining manual feedback from users, translating and identifying the specific legitimate issues. The analytics approach discussed could also be further used to prioritise the features as candidates for the future development cycle by identifying the most affected features based on the observed behavioural changes from the users.

This thesis presents the Usage Analytics method, designed to help developers identify the impact of changes on user behavior. The method captures changes in user behavior resulting from software modifications, regardless of the magnitude of these changes. When used alongside traditional feedback mechanisms, the method can reduce the time delay associated with obtaining manual feedback, translating and identifying specific issues. Additionally, it can prioritize features for future development cycles by identifying those most affected based on observed user behavior. The proposed Usage Analytics method includes processes for identifying data sources, extracting usage data, and analyzing this data using specific metrics. The research problem was identified through a systematic literature review and an analysis of current practices in industry and academia. The method was developed in collaboration with developers from IBM Academic Cloud and IBM Watson Workspace and applied to another application, Odoo Notes, for evaluation. The method was tested through designed experiments where volunteers performed predefined tasks using the applications. Usage data related to user actions was extracted and analyzed. Findings from these experiments were used to refine the method, resulting in multiple iterations. Version 2 was evaluated with IBM Watson Workspace, and version 3 was evaluated with Odoo Notes, demonstrating the method's applicability across different types of software applications with varying data availability.

This thesis presented a systematic approach to addressing challenges in usage analytics and feature prioritization through the iterative development of the Usage Analytics (UA) method. The research aimed to bridge the gap between user behavior analysis and actionable insights for developers, aligning findings with Research Questions 2 and 3. The study identified critical challenges developers face, such as data integration, feature-action mapping, and time-intensive data selection processes. Through iterative refinements across three case studies—IBM Academic Cloud, IBM Watson Workspace, and Odoo Notes—the research demonstrated how these challenges could be systematically addressed. The UA method introduced metrics such as frequency, timespent, and consistency to provide developers with clear insights into user behavior. Each iteration of the method incorporated feedback from previous implementations, refining its processes and ensuring its adaptability across diverse platforms. The first version, applied to IBM Academic Cloud, focused on identifying core challenges and laying the groundwork for data collection and analysis. The second version, developed for IBM Watson Workspace, introduced feature-action mapping and improved metrics to capture user interactions more effectively. The final version, applied to Odoo Notes, validated the method's generalizability and enhanced its usability in a non-IBM platform. The method's adaptability was demonstrated across platforms, addressing the varying challenges in data availability and feature identification. These contributions highlight the method's potential to support developers in prioritizing features and optimizing software development cycles based on actionable insights.

In Case Study 1, conducted within the IBM Academic Cloud project, the goal was to investigate the challenges associated with the feature prioritization process. The evaluation involved structured interviews with developers who provided insights into the challenges and effectiveness of the usage analytics process model. The main challenges identified were the lack of clear feature definitions and the difficulty in mapping user interactions to specific features, which impacted the effectiveness of planning and requirements analysis stages. The descriptive analysis revealed that substantial time and effort were required to select and prepare data, creating bottlenecks in the development process. Despite these challenges, the structured approach and collaboration within the team were seen as strengths. Recommendations for improvement included better feature definitions, automated data cleaning, and enhanced user interaction mapping. One of the key findings from Case Study 1 was that the absence of well-defined platform features significantly hindered the developers' ability to plan effectively and understand user behavior. The developers highlighted the necessity for clear feature definitions to accurately map user interactions and derive meaningful insights from the data. Additionally, the interviews revealed that the lack of a clear mapping of user interactions to application features led to incomplete or unclear requirements, making the requirements elicitation process more complicated and time-consuming. The data selection and preparation process also posed significant challenges, requiring considerable time and effort from the developers. The diverse types of data available made it difficult to choose the right data sources and formats, creating a bottleneck in the development process. Despite these challenges, the developers acknowledged that the usage analytics process model provided a structured approach that facilitated collaboration and helped identify critical areas for improvement.

Case Study 2 aimed to address the research question: "What are the key challenges faced by developers in identifying and utilizing usage data and key metrics related to feature prioritization effectively within the software platform?" This study, conducted with IBM Watson Workspace, involved structured interviews with an operational manager, an architect, a senior developer, and two developers. The evaluation focused on the implementation of an advanced usage analytics method. The main challenges identified were related to the identification and extraction of usage data, application of analytics metrics, and development of feature-action maps. Feedback from the development team led to refinements in the method, resulting in Version 2. Descriptive analysis highlighted the importance of systematic data collection, comprehensive feature-action mapping, and iterative testing for continuous improvement. The findings emphasized the need for integrating multiple data sources and ongoing validation of analytical processes. In Case Study 2, developers faced challenges in identifying relevant usage data sources and applying analytics metrics to extract meaningful insights. The diverse sources of usage data, including native and third-party monitoring tools, provided a comprehensive view of user interactions with the application. However, integrating these data sources and ensuring data quality were major challenges, requiring significant effort to collect consistent and accurate data across different platforms. The development of feature-action maps was another significant outcome of Case Study 2. This process involved systematically mapping user actions to specific features, providing a structured approach to understanding user interactions. The integration of feature-action mapping into the usage analytics method helped the development team visualize user interactions and identify critical areas for improvement. Despite these advancements, the developers emphasized the need for ongoing validation and refinement of the feature-action map to ensure its accuracy and relevance.

Case Study 3 involved the application of the final version of the Usage Analytics method to the Odoo Notes application. The goal was to evaluate the method's applicability across different application domains. Structured experiments with nine users provided insights into the impact of changes on user behavior. The results showed that the method successfully identified actions that most affected user behavior, and a priority list of features was generated based on the extent of impact. The evaluation highlighted that the method could be used to filter user reviews and feedback, prioritize critical issues, and optimize future development cycles. The analysis combined frequency, timespent, and consistency metrics to provide a comprehensive view of user interactions and behavioral changes. In Case Study 3, the Usage Analytics method was applied to the Odoo Notes application, involving experiments with nine users. The experiments aimed to understand the changes in user behavior between different versions of the application. The results demonstrated that the method could effectively identify the most impacted actions and features based on the usage metrics. The frequency, timespent, and consistency metrics provided a detailed view of user interactions, helping developers prioritize features for future development. The evaluation also revealed that the Usage Analytics method could be used to filter user reviews and feedback, focusing on the most critical issues. By prioritizing features based on the observed behavioral changes, developers could optimize the development process and ensure that the most significant issues were addressed first. The findings from Case Study 3 confirmed the method's applicability across different application domains and its potential to enhance the efficiency of software development processes.

Overall, the evaluations conducted in these case studies demonstrate that the Usage Analytics method is effective in identifying critical features and understanding user behavior. The method's systematic approach to data collection, analysis, and feature prioritization offers significant advantages over traditional feedback mechanisms. The findings from these evaluations support the method's applicability across different types of software applications and its potential to optimize development processes.

The research systematically derived a super-set of data types and metrics through a rigorous literature review. This super-set served as the foundation for developing and evaluating the Usage Analytics (UA) method. The key metrics—frequency, time spent, and consistency—were selected based on their prominence in the literature and validated through iterative applications across case studies. By leveraging this super-set, the research ensured a comprehensive framework that balances theoretical rigor with practical applicability. The super-set provided a structured approach to understanding and analyzing user behavior, ensuring that the UA method addressed diverse software environments effectively. Metrics such as engagement rates and behavioral consistency, grounded in the super-set, proved essential for feature prioritization and user behavior analysis in the case studies.

The research is subject to limitations on the software applications where human users interact directly with the application. Applications, where the users interact with the application through a third-party application or web service, are not considered in the scope of this research. The research is also subject to limitations on the availability of usage data sources. The data sources considered in the scope of this research are the ones that are available to the developers of the application. If the developers do not choose to include the proposed usage analytics in the design of the software application, the method could still be used post-development of the application. A direction to develop a custom monitoring tool to collect the usage data from the application is also demonstrated for future researchers to explore the possibility of collecting the usage data from the applications where the data sources are not available to the developers of the application.

8.2 Contributions

A systematic literature review was conducted to identify the research gap by exploring current techniques to collect and analyse the usage of application features in a software development domain. Taking into account that identifying the importance of the features and selecting the features as candidates for development is currently performed in an ad-hoc manner. Although there are techniques available for collecting user feedback to understand which features are useful to the users, yet rarely answer why those features are important for them. As a result, the selection and prioritization of features become far from optimal and the product deviates from what the users need. This research addresses this gap by proposing a method to systematically analyse how features are used by end-users and the usage patterns exhibited by the users. The research has resulted in thirteen academic papers so far, published in various conferences, workshops and an article in a book chapter, and one patent documenting the various achievements accomplished as a part of this research.

This research directly addressed RQ1 by identifying and validating a comprehensive super-set of data types and metrics for feature prioritization in software analytics. The iterative refinement process, involving developers and stakeholders, demonstrated the practical relevance of the selected metrics. The alignment between the theoretical super-set and industrial applications highlighted the adaptability and scalability of the UA method. The iterative application of the UA method across diverse platforms—IBM Academic Cloud, Watson Workspace, and Odoo Notes—validated the super-set's utility, reinforcing its role as a foundational framework for usage analytics. A critical aspect of this research was the inclusion of participant feedback from developers and stakeholders in refining the UA method. Participant feedback highlighted the importance of actionable metrics like frequency, time spent, and consistency, which were refined through iterative discussions and evaluations. This collaborative approach ensured that the selected metrics aligned with real-world needs, addressing gaps identified during the initial stages of the research. The iterative refinement process not only validated the selected metrics but also provided insights into improving their applicability across platforms. For example, feedback from IBM developers emphasized the need for engagement metrics, which were integrated and fine-tuned to enhance the UA method's effectiveness.

This research has also resulted in an opportunity in terms of an internship to work closely with the development team of IBM Watson Workspace, which helped the research to incorporate the industrial perspective and knowledge about the research problem. The developed Usage Analytics method resulted in an Intellectual Property article publication and prior art for a patent application available on IP.com (`https://priorart.ip.com/IPCOM/000261887`). The Usage Analytics project won several awards: DCU Commercialisation Award 2017; Tech Ireland Awards 2018 - Finalists for the Outstanding Academic Achievement of the Year; Tech Excellence Awards 2019 - Finalist for the Project of the Year award (Private sector); DCU Invent Commercialisation Award 2022. The Watson Workspace application is considered one of the case studies for the research to evaluate the practical application and usefulness of the developed artefact in the industry. This study contributes to research and the problem domain in three aspects: advancing practice, improving and creating methodology, and extending theory. Specifically, the following contributions are made in academia and practice:

1. **Developers of an application can identify important features and optimize application development:** The Usage Analytics method provides a systematic approach for developers to identify the most critical features based on user behavior. By understanding how users interact with different features, developers can prioritize the most impactful features for future development, ensuring that the application aligns with user needs and preferences.

2. **Architects of an application can use the method to incorporate usage data extraction and analysis into the design of the application:** The method includes processes for identifying data sources, extracting usage data, and analyzing this data using specific metrics. By integrating these processes into the application design, architects can ensure that the necessary usage data is collected and analyzed effectively, providing valuable insights into user behavior.

3. **Impact on the knowledge of usage data in the cloud-based application domain, how to identify, extract and analyze them:** The research advances knowledge in the cloud-based application domain by providing a comprehensive method for identifying, extracting, and analyzing usage data. The findings contribute to a deeper understanding of how usage data can be leveraged to improve software development processes and

enhance user satisfaction. This research bridges the gap between theoretical frameworks and practical applications in software analytics. By systematically deriving and validating a super-set of data types and metrics, it contributes a robust framework for analyzing user behavior and prioritizing features. The iterative refinement process, grounded in participant feedback, highlights the dynamic interplay between research and industry needs, ensuring that the UA method is both adaptable and impactful.

4. **A novel method was designed to identify, extract, and analyze the usage data of the features of an application:** The Usage Analytics method includes a detailed process for identifying, extracting, and analyzing usage data. The method provides a structured approach for understanding user interactions and identifying the most critical features based on observed behavior. The findings from this research open avenues for further exploration, particularly in expanding the super-set to incorporate emerging data types and metrics, and validating the UA method in additional industrial contexts.

5. **A novel analytics approach was designed to analyze the usage data of the features of an application by demonstrating the usage metrics such as Frequency, Timespent, and Consistency:** The research introduced a novel analytics approach that combines multiple usage metrics to provide a comprehensive view of user interactions. The frequency, timespent, and consistency metrics help developers understand how users interact with different features and identify changes in behavior resulting from software modifications.

6. **A new understanding of the usage behavior of the users of the application and its correlation with the changes implemented to the application by the developers of the application:** The research provides new insights into how user behavior changes in response to software modifications. By correlating user behavior with specific changes implemented by developers, the research helps identify the most impactful modifications and prioritize features for future development.

8.2.1 Design Knowledge Contributions

This research introduced a robust framework, the Usage Analytics Method (UAM), designed to link user interaction data with feature prioritization decisions. A key conceptual advancement lies in its generalizable structure, validated across diverse platforms including IBM Academic

Cloud, IBM Watson Workspace, and Odoo Notes. This adaptability underscores UAM's applicability in varying software development environments.

The UAM framework relies on evidence-based decision-making, leveraging user interaction metrics to inform feature prioritization. Developed iteratively in collaboration with industry practitioners, it balances theoretical robustness with practical relevance.

Methodological Contributions

The UAM framework was developed and refined through a systematic, rigorous methodology. Key methodological contributions include:

1. super-set derivation

   (a) A comprehensive super-set of metrics and data points critical for analyzing user behavior and guiding feature prioritization was identified through a systematic literature review.

   (b) Metrics were categorized into foundational types (e.g., frequency, duration) and advanced types (e.g., engagement depth), providing a scalable baseline for research and industry use.

2. collaborative refinement

   (a) Iterative brainstorming sessions with industry practitioners refined the super-set, ensuring alignment with real-world software development challenges.

3. iterative validation

   (a) Case studies were employed to validate the framework's design and applicability, enabling continuous refinement.

This approach bridges the gap between theoretical constructs and practical applications, offering a replicable model for future research in software analytics.

Practical Contributions

The UAM framework demonstrated its value in practical applications by providing actionable insights and enhancing software development decision-making. Specifically:

1. Feature Prioritization: UAM's structured approach to user behavior analysis enabled developers to prioritize features based on observed user interactions, enhancing development efficiency.

2. Data-Driven Development: UAM's reliance on usage metrics facilitated evidence-based decision-making, reducing reliance on subjective feedback and improving feature selection.

3. Cross-Platform Validation: UAM's adaptability across diverse platforms, including IBM and non-IBM applications, underscored its generalizability and scalability.

4. Developer Empowerment: UAM equipped developers with a systematic approach to connect user behavior data with design decisions, enhancing data-driven development practices.

5. Real-World Applicability: UAM's successful application in real-world settings validated its utility and relevance in modern software development environments.

6. Continuous Improvement: UAM's iterative refinement process ensured ongoing enhancements, aligning the framework with evolving industry needs and challenges.

These practical contributions underscore UAM's value in enhancing software development practices and empowering developers with actionable insights.

Extensibility and Adaptability

UAM is inherently extensible, allowing for the integration of additional metrics as software systems evolve. While this research focused on three core metrics, the framework supports:

1. Engagement Metrics: Session depth, navigation patterns, and user flow.

2. Performance Metrics: Task completion rates and error analysis.

3. Advanced Behavioral Analytics: AI-driven insights and predictive modeling.

The stages of UAM—feature identification, identification of usage data sources, and data extraction—remain pivotal when incorporating additional metrics. Specifically:

1. **Feature Identification:** Each new metric aligns with specific software features and user goals.

2. **Identification of Usage Data Sources:** Relevant sources, such as clickstreams, heatmaps, or user logs etc., are identified to provide actionable data.

3. **Data Extraction and Processing:** Tailored extraction methods ensure consistency and compatibility across all metrics.

These stages maintain UAM's robustness while enhancing its scalability and relevance in evolving analytical scenarios.

8.3 Limitations of the Study

Empirical studies are fundamental to assessing the effectiveness, adaptability, and limitations of the UAM. This section delves into an exhaustive analysis of the case studies presented in this research, identifying critical limitations, exploring their implications, and proposing actionable recommendations for improvement. By addressing these shortcomings, the research can contribute to a more robust and universally applicable framework for UAM. The case studies reveal multiple limitations that inform the research findings and highlight areas for refinement. Each study's unique context offers valuable lessons, yet the collective insights point to broader challenges in applying UAM across diverse environments. Below is an expanded discussion of these limitations, their implications, and how they can be mitigated in future research.

8.3.1 Case Study 1: IBM Academic Cloud

1. **Participant Diversity:** The study primarily involved IBM developers with extensive technical expertise, limiting the generalizability of findings to broader user groups. The lack of diverse user profiles, including novice users or those from non-technical backgrounds, restricts the ecological validity of the findings. Research by Nielsen (2012) underscores the importance of including varied participant demographics to uncover usability challenges overlooked by expert users.

2. **Controlled Environment:** The controlled settings of the study ensured consistent testing conditions but failed to replicate the complexities of real-world environments. Real-world scenarios often include unpredictable user behaviors, system errors, and varying user goals that cannot be captured in a strictly controlled study.

3. **Data Access:** Access to comprehensive datasets within the IBM ecosystem enabled smooth implementation of UAM. However, this level of access is atypical in external settings where data availability is limited or where privacy concerns restrict data collection. This discrepancy raises questions about the adaptability of UAM in more constrained environments.

Impact on Findings:

1. The limited participant diversity restricts the applicability of findings to broader user groups.

2. Controlled conditions may lead to overestimation of UAM's effectiveness in less predictable settings.

3. Reliance on rich datasets does not address potential challenges in environments with limited instrumentation.

Recommendations for Future Work:

1. Recruit participants from a wide range of technical and non-technical backgrounds, including end-users from different industries.

2. Design studies that incorporate real-world variability, such as unexpected system interruptions or evolving user goals.

3. Test UAM in external environments with data access constraints to evaluate its adaptability and robustness.

8.3.2 Case Study 2: IBM Watson Workspace

1. **Narrow Task Scope:** The study emphasized tasks related to collaboration and user engagement, excluding other critical dimensions of user behavior such as error recovery, task prioritization, and cognitive load during multitasking. Norman and Nielsen

2. **Platform-Specific Constraints:** The study's findings are inherently tied to the unique architecture and features of Watson Workspace. Consequently, insights derived from this platform may not generalize to systems with different interaction paradigms or architectural designs.

3. **Limited Longitudinal Analysis:** The study primarily analyzed short-term interactions, missing the opportunity to observe how user behaviors evolve over time. This oversight limits the ability to assess UAM's long-term utility and adaptability.

Impact on Findings:

1. The narrow focus on collaboration tasks restricts the generalizability of findings to other aspects of user interaction.
2. Platform-specific results hinder broader applicability across diverse systems.
3. Short-term evaluations fail to capture changes in user behavior or long-term trends.

Recommendations for Future Work:

1. Include tasks that test non-collaboration features, such as adapting to new system updates or managing system configurations.
2. Conduct cross-platform evaluations to ensure UAM's adaptability to a variety of system architectures.
3. Perform longitudinal studies to observe behavioral trends and refine UAM metrics over time.

8.3.3 Case Study 3: Odoo Notes

1. **Single Non-IBM Platform:** The inclusion of only one non-IBM platform limits the study's ability to demonstrate UAM's generalizability across diverse software ecosystems. Although Odoo Notes' open-source architecture offers valuable contrast to proprietary IBM systems, it does not provide sufficient evidence for cross-industry applicability. Kitchenham et al. (2004) highlight the necessity of validating tools in varied contexts to establish external validity.
2. **Small Participant Pool:** The limited number of participants reduces the statistical significance and general reliability of the study's findings. A broader participant base would enhance the robustness of conclusions drawn from this case study.
3. **Restricted Metric Set:** Metrics focused on frequency, time spent, and consistency, excluding potentially insightful metrics such as user satisfaction, error rates, and task-switching behaviors. Expanding the metric set would provide a more holistic understanding of user interactions.

Impact on Findings:

1. Reliance on a single non-IBM platform provides insufficient evidence for broad applicability.

2. A small participant pool undermines the statistical strength of the findings.

3. Limited metrics result in a narrow evaluation of user behavior, overlooking critical interaction patterns.

Recommendations for Future Work:

1. Platform Diversity: Validate UAM on a wider variety of non-IBM platforms, including industry-specific and microservices-based systems (Balalaie et al., 2016), to establish generalizability.

2. Increased Participant Pool: Recruit participants from diverse professional and cultural backgrounds to improve the study's reliability.

3. Expanded Metrics: Incorporate additional metrics such as user satisfaction scores, cognitive load, and task-switching rates to enrich the evaluation.

The case studies conducted in this research highlight the potential of the UAM while revealing critical areas for improvement. Addressing the limitations through diverse participant recruitment, expanded task designs, broader platform evaluations, and enriched metric inclusion will significantly enhance the method's applicability and credibility. Future research should aim to validate UAM in real-world and constrained environments, thereby cementing its role as a versatile and generalizable tool for modern software analytics.

8.4 Future Work

Usage analytics can reveal the engagement level of customers during the development and evaluation process of a software analytic project. It is well recognised that engaging customers is a challenging task, especially in the context of software engineering tools. Customers always tend to keep their existing way of carrying out a task or the way of using a service. Furthermore, it is usually lacking in investing time to understand the pros and cons of the proposed tools due to the tight development schedule. Thus, understanding customer engagement has a significant impact on the development of applications and services.

In future, the following aspects of this research are recommended to be explored:

1. **Generalizability to Other Platforms and Domains:** While the UAM framework was validated using IBM Academic Cloud, IBM Watson Workspace, and Odoo Notes, future studies should explore its applicability across a wider array of platforms, including:

   (a) IoT Applications: Understanding how usage data from interconnected devices can be analyzed to improve real-time functionality and usability.

   (b) Virtual and Augmented Reality (VR/AR): Investigating how user interactions in immersive environments can provide insights for optimizing experiences, such as gesture accuracy or spatial navigation patterns.

   (c) Cloud-Based Business Platforms: Assessing the UAM's capability in handling multi-tenancy and diverse user groups in large-scale enterprise systems. Future research should focus on comparing the similarities and differences in user interaction patterns across these platforms to better understand UAM's adaptability.

2. **Integration with Situational and Emotional Analytics:** Expanding UAM to include situational analytics and emotion recognition could provide deeper insights into user behavior. For instance:

   (a) Replacing Biometric Tools: Future studies could investigate how usage patterns (e.g., navigation paths, error rates) can serve as proxies for biometric data like eye-tracking and facial recognition.

   (b) Context-Aware Analytics: Incorporating situational factors (e.g., user location, device type, or time of use) into UAM to enhance its effectiveness in dynamic and mobile environments. This integration could bridge the gap between human factors and software analytics, improving both usability and user satisfaction.

3. **Extensibility to Additional Metrics and Feature Sets:** The current study focused on frequency, time spent, and consistency metrics. Future work could expand on this by including:

   (a) User Retention Rates: Analyzing how frequently users return to the application and what features contribute to retention.

   (b) Feature Abandonment Metrics: Understanding why users discontinue using specific features.

   (c) Clickstream Data Analysis: Examining the sequential nature of user actions to identify common workflows and pain points. These additional metrics could help developers better understand user preferences and prioritize features more effectively.

4. **Development of Real-Time Monitoring Dashboards:** To enhance decision-making processes, future research should focus on building a real-time dashboard for UAM. Potential features include:

   (a) Anomaly Detection: Identifying unusual usage patterns that may indicate bugs or usability issues.

   (b) Predictive Analytics: Forecasting user behavior and feature adoption based on historical data.

   (c) Customizable Views: Allowing developers to tailor dashboards for specific roles or goals (e.g., project managers vs. UX designers). Such dashboards could significantly enhance the practicality of UAM in fast-paced development cycles.

5. **Integration of AI and Machine Learning:** Future research could explore the integration of advanced AI and machine learning techniques to:

   (a) Cluster User Behavior: Group users based on shared patterns to develop targeted features.

   (b) Predict User Needs: Use historical data to recommend features or design changes proactively.

   (c) Automate Feedback Analysis: Enable real-time analysis of qualitative feedback to identify actionable insights. This would make UAM more adaptive and capable of handling the complexities of large-scale, dynamic environments.

6. **Validation in Longitudinal and Large-Scale Studies:** Long-term and large-scale validation is crucial for assessing UAM's robustness. Future studies could:

   (a) Conduct Multi-Year Studies: Analyze usage trends and feature adoption over extended periods.

   (b) Involve Diverse User Groups: Test UAM in geographically dispersed user bases to ensure inclusivity and reliability. These studies could highlight domain-specific limitations and provide a stronger evidence base for UAM's generalizability.

7. **Iterative Refinement and Feedback Integration:** UAM's iterative nature should be further leveraged to refine its metrics and methodologies. Suggested areas include:

(a) Continuous Feedback Loops: Regularly integrating user feedback into metric refinement processes.

(b) Transparent Documentation: Recording each refinement iteration to improve reproducibility and transparency. This would ensure that UAM remains aligned with the evolving needs of the software development industry.

8. **Scalability and Cloud Integration:** Future work should explore how to optimize UAM for scalability. Key directions include:

    (a) Leveraging Distributed Cloud Systems: Using cloud infrastructure to handle high-throughput data processing in real-time.

    (b) Supporting Multi-Tenancy: Adapting UAM to manage and analyze data for multiple user groups within a single application. Such enhancements would make UAM a viable solution for large-scale enterprise applications.

9. **Cross-Domain Applications:** The framework's applicability to hybrid systems where users interact via third-party applications or services should be explored. Future research could investigate:

    (a) Web Services Integration: Applying UAM to APIs and third-party integrations to understand their impact on user experience.

    (b) Hybrid Environments: Testing UAM in systems that combine multiple platforms (e.g., a desktop application with mobile extensions). This exploration could refine the framework to accommodate more diverse interaction models.

By addressing these areas, future research has the potential to not only enhance the utility and applicability of the UAM framework but also significantly advance the field of software development practices. Such investigations could lead to the development of methodologies that are robust, scalable, and user-centric, effectively bridging the gap between theoretical insights and practical applications. These advancements would not only support developers in creating more adaptable and efficient software systems but also drive the evolution of user-focused analytics as a core component of modern software engineering. Furthermore, a more refined UAM framework could become a critical tool for improving decision-making processes in diverse domains, ranging from enterprise applications to consumer-facing platforms. Ultimately, this work will pave the way for a future where data-driven analytics seamlessly integrates with agile, iterative development models, offering actionable insights that empower organizations to respond dynamically to user needs and market demands.

Appendix A

URL Links for Repositories and other Documents

1. **Generating Feature-Action map for Odoo Notes:** `https://github.com/chintu0019/Usage-Analytics/blob/master/Documents/Feature-Action%20Map.json`

2. **Feature-Action map for IBM Watson Workspace:** `https://github.com/chintu0019/Usage-Analytics-IBM/blob/master/Files/Workspace-Features-List/Feature-Action%20Map.json`

3. **Experiment data extracted from IBM Watson Workspace:** `https://github.com/chintu0019/Usage-Analytics-IBM/blob/e3d7cf2bb024d39a5f699b3c9437b3703e4348af/Logs/Experiment-Logs`

Appendix B

Experiment tasks - Odoo

1. Create a new column and name it “Temporary”, you can use this column to place any temporary notes you may create
2. create a minimum of 7 notes (one for each new task)
3. Move the newly created notes to the correct column (if not done before); two ways of doing it:
   (a) Drag and drop
   (b) Click on the column name on top right position after opening the note
4. Assign a different colour to the note to specify the type of task
   (a) RED for personal task
   (b) PURPLE for a work-related task
   (c) BLUE for others
5. Assign a corresponding tag (create a new tag if necessary) to the note
   (a) “personal” for personal task
   (b) “work” for work-related task
   (c) “other” for other tasks
6. check for typos in all the notes and fix them
7. Comment the name of the day you want to assign the task

8. Invite and add the follower you want to work with to the task at hand

9. Find all the notes that contain the word “hello” (search with note)

10. count the number of personal and work-related notes (use search), use a temporary note to mention the count

11. attach the file named “dummy file” to all the work-related notes

Appendix C

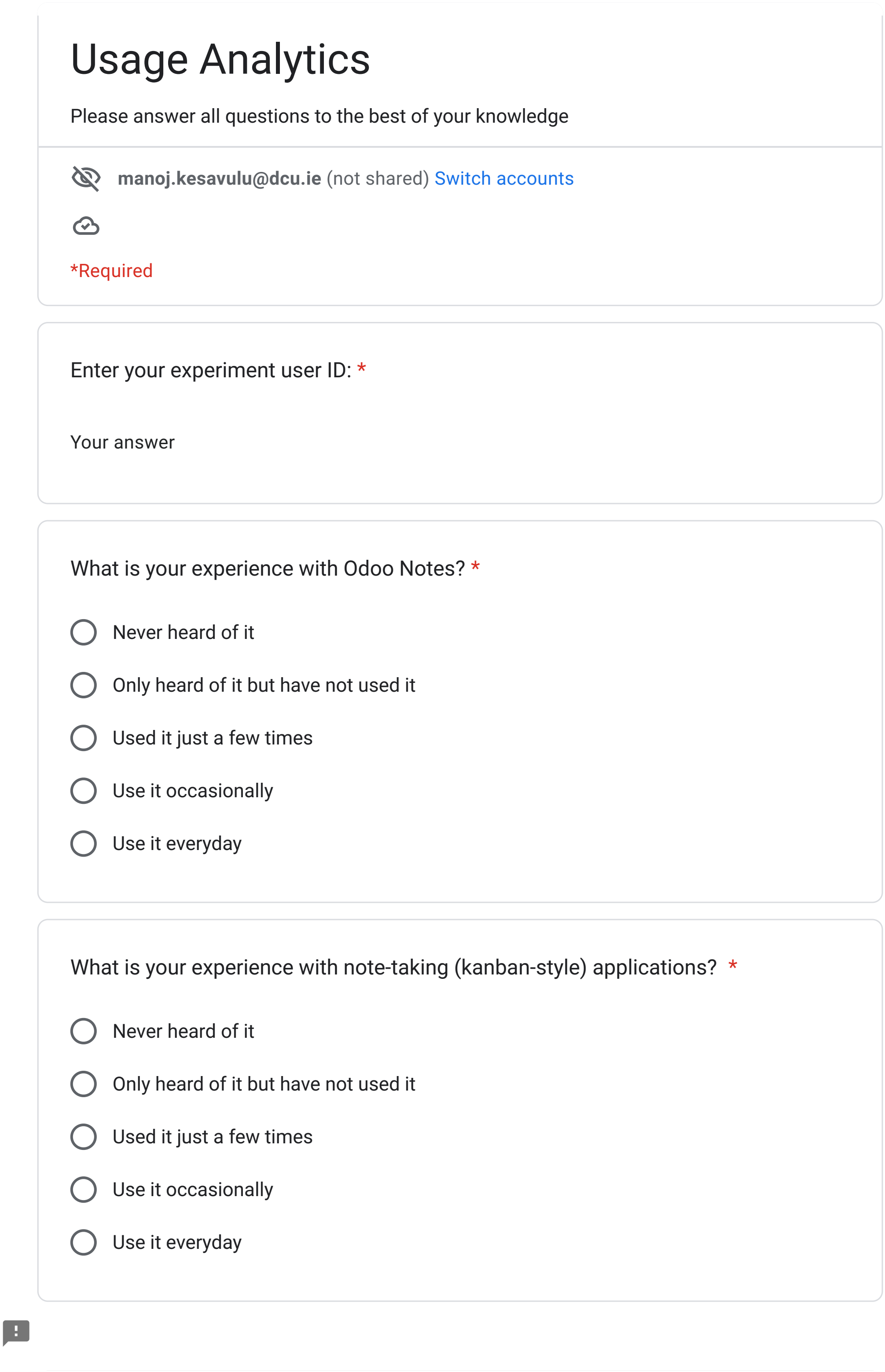
Feature-Action map - Odoo

```
{
"Actions": [
{"ActionName":"Create Stage", "ID": 1},
{"ActionName":"Delete Stage", "ID": 2},
{"ActionName":"Rename Stage", "ID": 3},
{"ActionName":"Search with Note", "ID": 4},
{"ActionName":"Search with Tag", "ID":5},
{"ActionName":"Search with both Tag and Note", "ID": 6},
{"ActionName":"Create Note", "ID": 7},
{"ActionName":"Delete Note", "ID": 8},
{"ActionName":"Open Note", "ID": 9},
{"ActionName":"Edit Note", "ID": 10},
{"ActionName":"Move Note", "ID": 11},
{"ActionName":"Drag and Drop note", "ID": 12},
{"ActionName":"Open commenting in note", "ID": 13},
{"ActionName":"Attach File", "ID": 14},
{"ActionName":"Comment in Note", "ID": 15},
{"ActionName":"Invite and Add follower", "ID": 16},
{"ActionName":"Invite User", "ID": 17},
{"ActionName":"Remove follower from note", "ID": 18},
{"ActionName":"Create new tag", "ID": 19},
{"ActionName":"Add tag", "ID": 20},
{"ActionName":"Change Tag Colour", "ID": 21},
{"ActionName":"Change Note Colour", "ID": 22}
]
}
```

Appendix D

Survey Forms

D.1 Survey form - Odoo Notes



Usage Analytics

Please answer all questions to the best of your knowledge

manoj.kesavulu@dcu.ie (not shared) Switch accounts

*Required

Enter your experiment user ID: *

Your answer

What is your experience with Odoo Notes? *

○ Never heard of it

○ Only heard of it but have not used it

○ Used it just a few times

○ Use it occasionally

○ Use it everyday

What is your experience with note-taking (kanban-style) applications? *

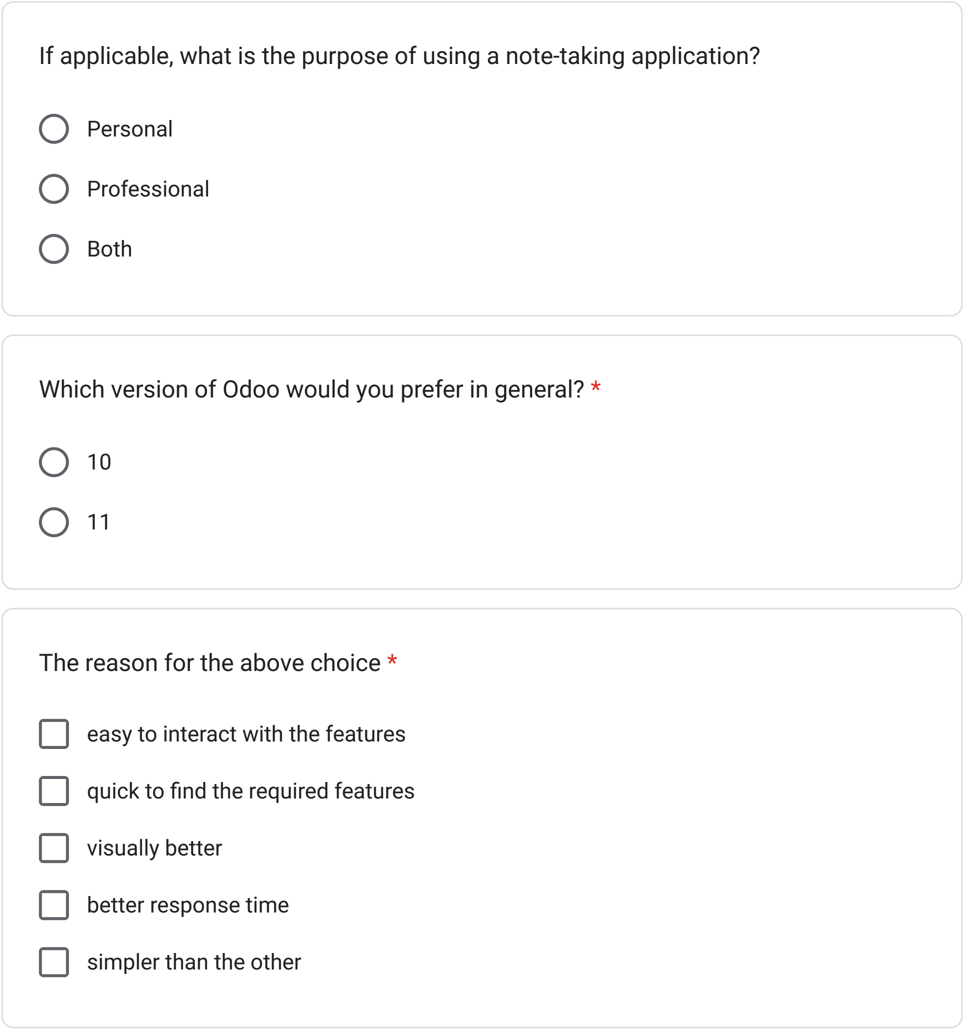
○ Never heard of it

○ Only heard of it but have not used it

○ Used it just a few times

○ Use it occasionally

○ Use it everyday



If applicable, what is the purpose of using a note-taking application?

○ Personal

○ Professional

○ Both

Which version of Odoo would you prefer in general? *

○ 10

○ 11

The reason for the above choice *

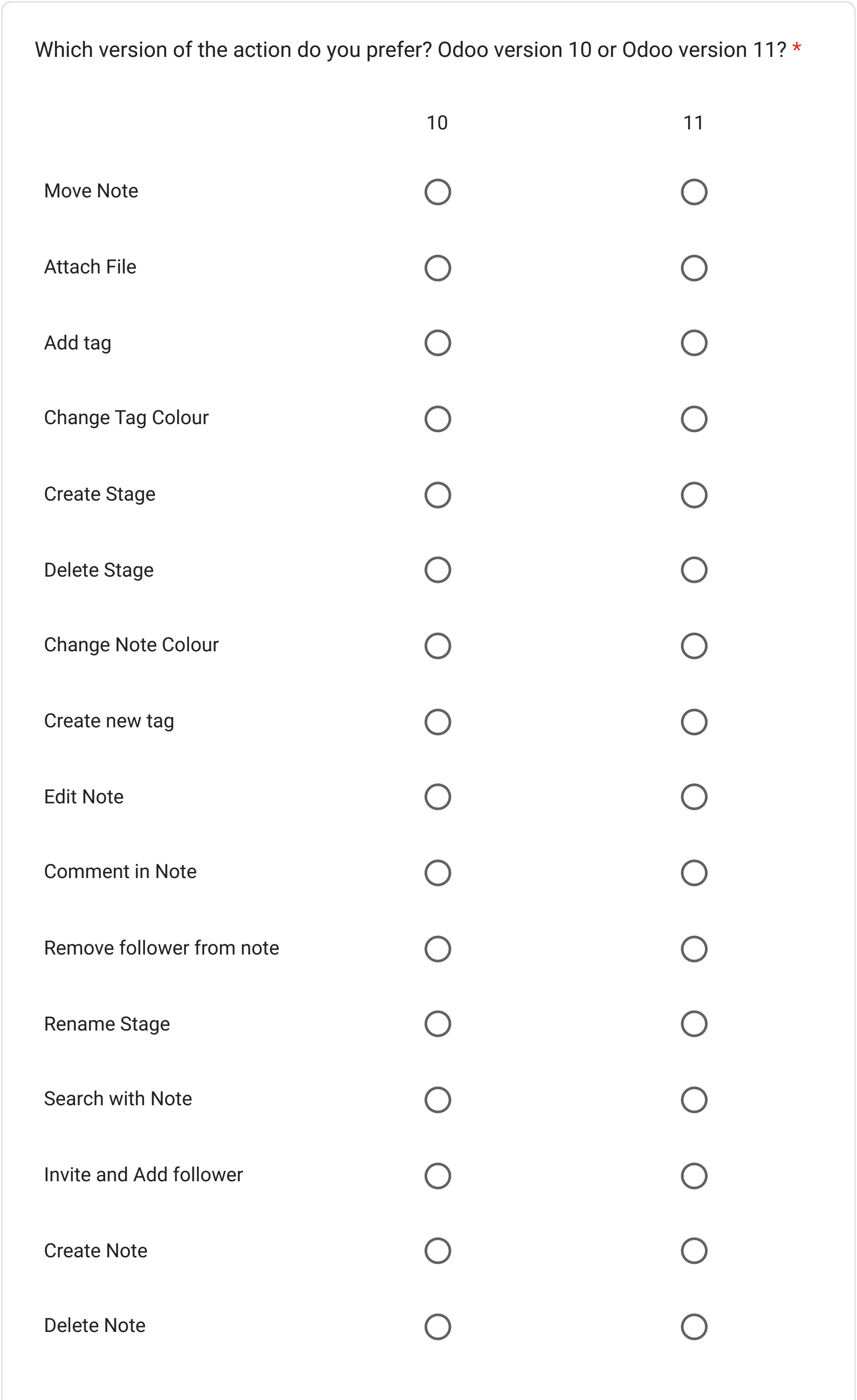
□ easy to interact with the features

□ quick to find the required features

□ visually better

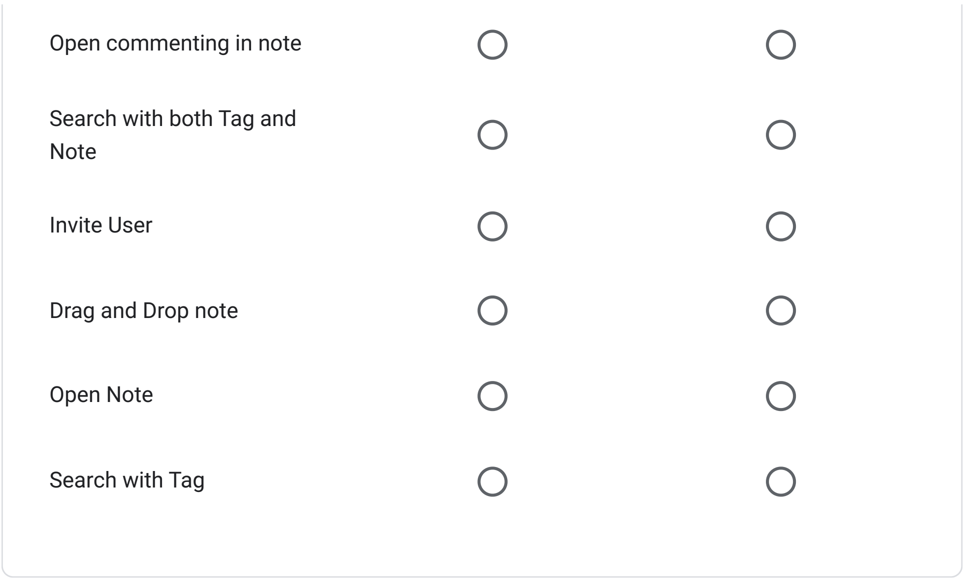
□ better response time

□ simpler than the other

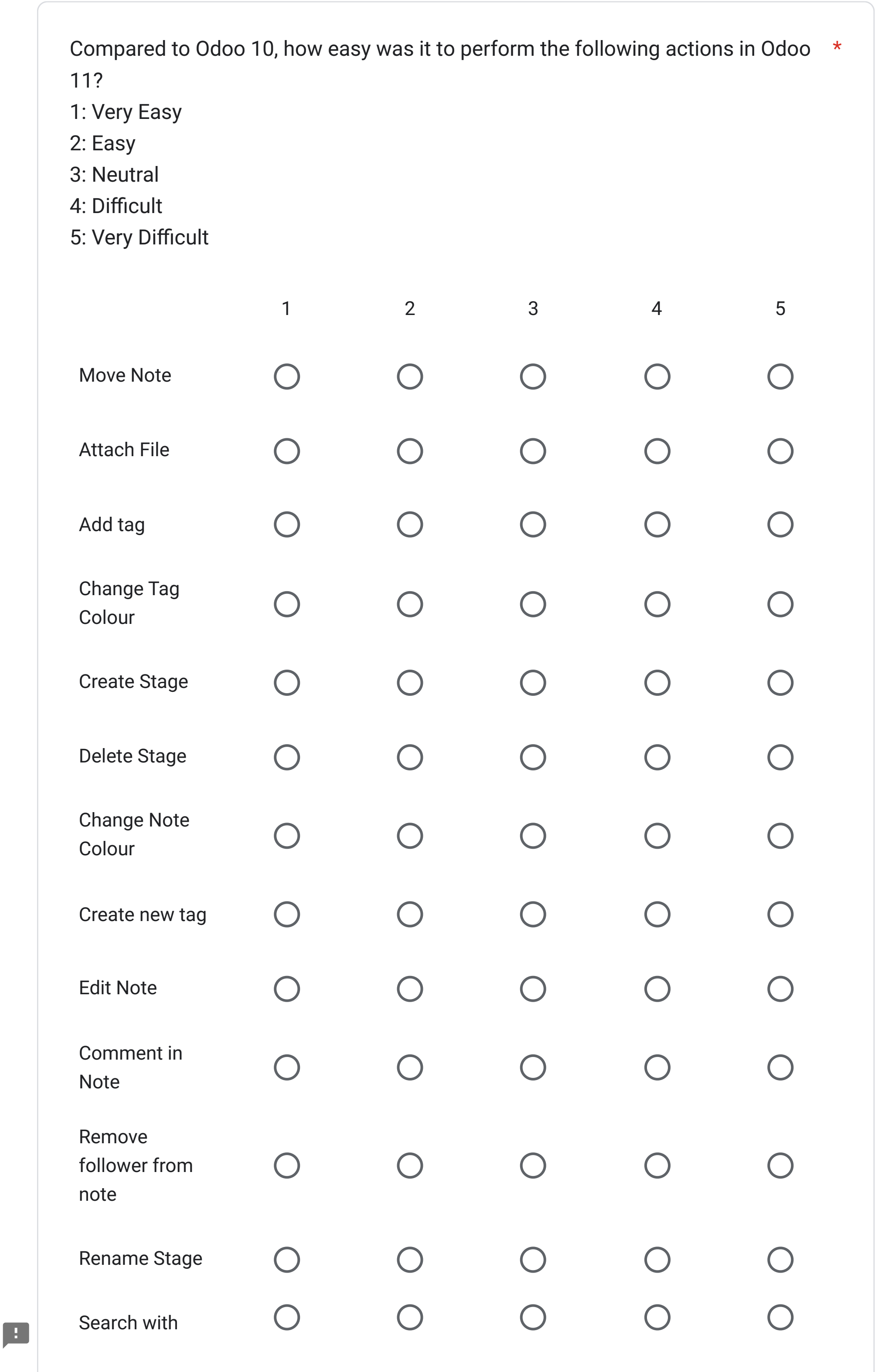


Which version of the action do you prefer? Odoo version 10 or Odoo version 11? *

	10	11
Move Note	○	○
Attach File	○	○
Add tag	○	○
Change Tag Colour	○	○
Create Stage	○	○
Delete Stage	○	○
Change Note Colour	○	○
Create new tag	○	○
Edit Note	○	○
Comment in Note	○	○
Remove follower from note	○	○
Rename Stage	○	○
Search with Note	○	○
Invite and Add follower	○	○
Create Note	○	○
Delete Note	○	○



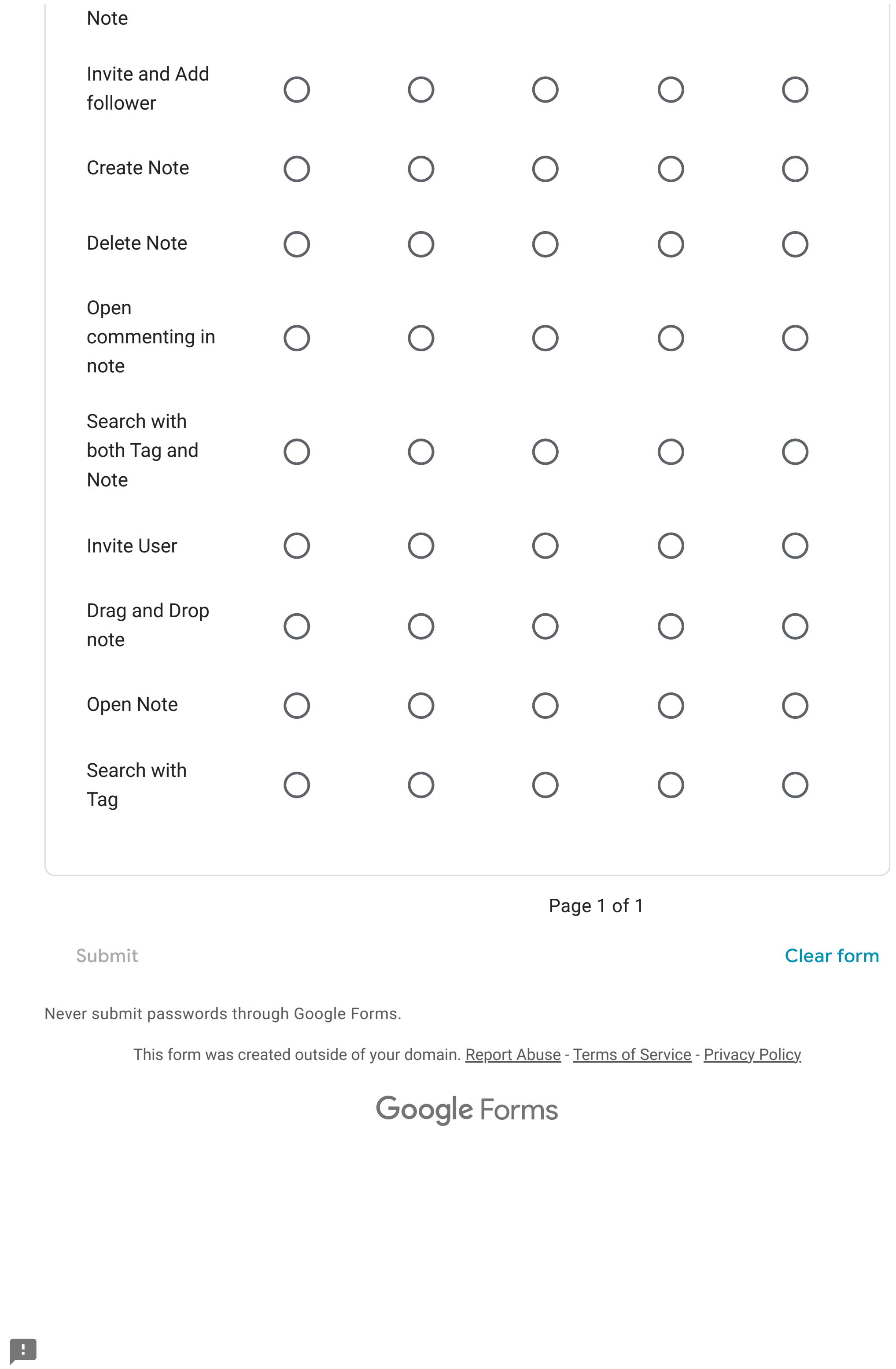
Open commenting in note	○	○
Search with both Tag and Note	○	○
Invite User	○	○
Drag and Drop note	○	○
Open Note	○	○
Search with Tag	○	○



Compared to Odoo 10, how easy was it to perform the following actions in Odoo 11? *

1: Very Easy
2: Easy
3: Neutral
4: Difficult
5: Very Difficult

	1	2	3	4	5
Move Note	○	○	○	○	○
Attach File	○	○	○	○	○
Add tag	○	○	○	○	○
Change Tag Colour	○	○	○	○	○
Create Stage	○	○	○	○	○
Delete Stage	○	○	○	○	○
Change Note Colour	○	○	○	○	○
Create new tag	○	○	○	○	○
Edit Note	○	○	○	○	○
Comment in Note	○	○	○	○	○
Remove follower from note	○	○	○	○	○
Rename Stage	○	○	○	○	○
Search with	○	○	○	○	○



Note

Invite and Add follower	○	○	○	○	○
Create Note	○	○	○	○	○
Delete Note	○	○	○	○	○
Open commenting in note	○	○	○	○	○
Search with both Tag and Note	○	○	○	○	○
Invite User	○	○	○	○	○
Drag and Drop note	○	○	○	○	○
Open Note	○	○	○	○	○
Search with Tag	○	○	○	○	○

Page 1 of 1

Submit

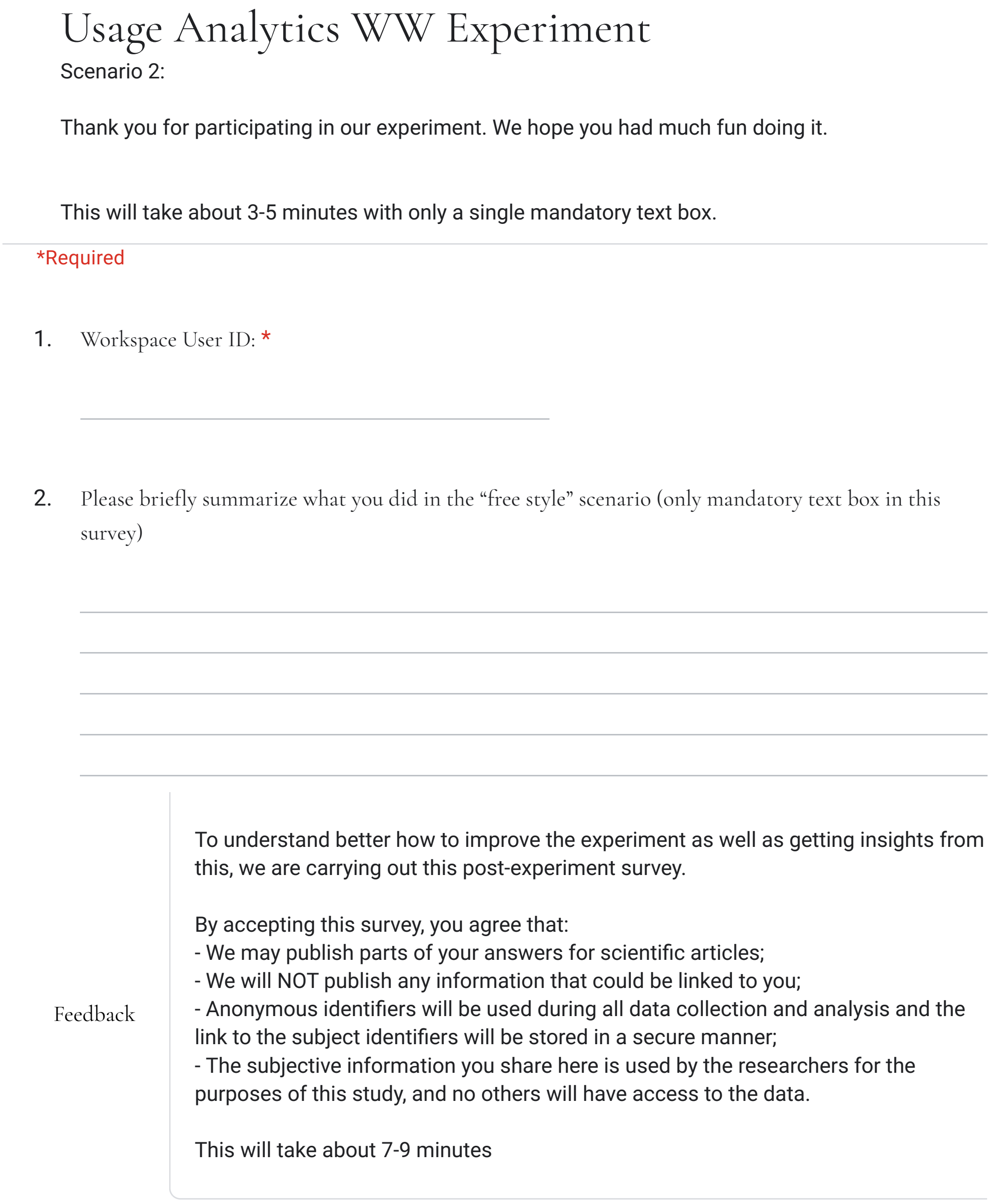
Clear form

Never submit passwords through Google Forms.

This form was created outside of your domain. Report Abuse - Terms of Service - Privacy Policy

Google Forms

D.2 Survey form - IBM Watson Workspace



Usage Analytics WW Experiment

Scenario 2:

Thank you for participating in our experiment. We hope you had much fun doing it.

This will take about 3-5 minutes with only a single mandatory text box.

*Required

1. Workspace User ID: *

____

2. Please briefly summarize what you did in the “free style” scenario (only mandatory text box in this survey)

____

____

____

____

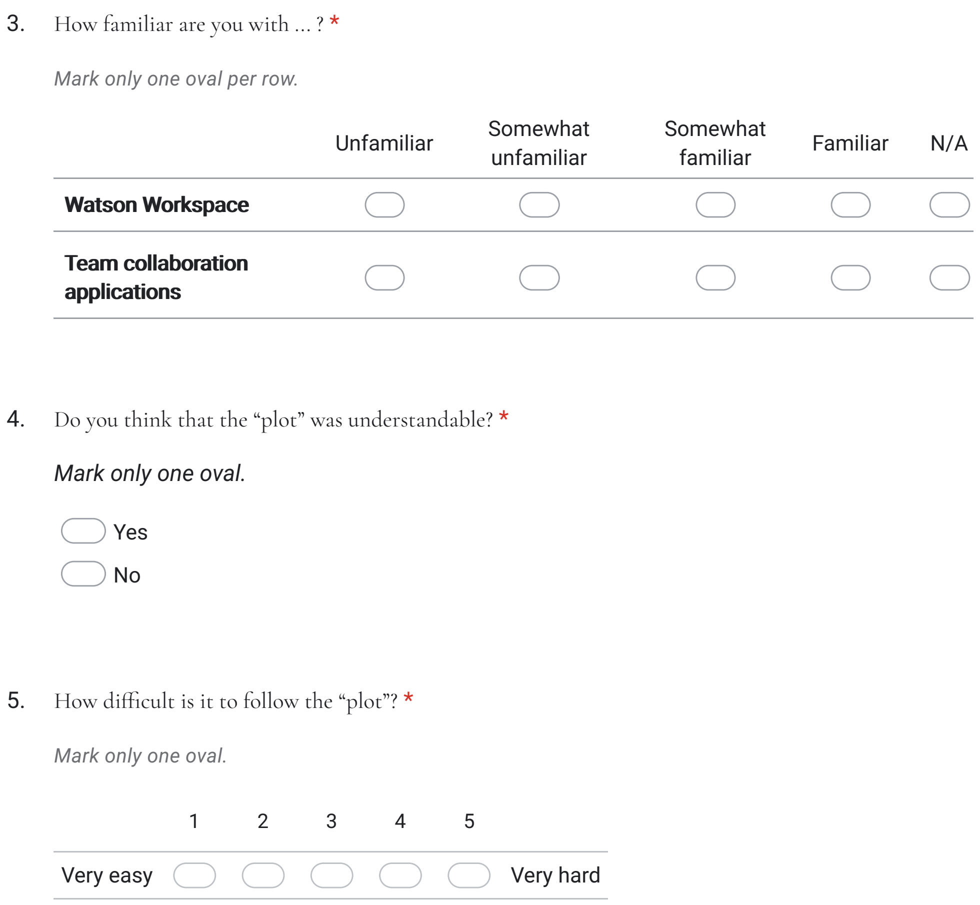
____

Feedback

To understand better how to improve the experiment as well as getting insights from this, we are carrying out this post-experiment survey.

By accepting this survey, you agree that:
- We may publish parts of your answers for scientific articles;
- We will NOT publish any information that could be linked to you;
- Anonymous identifiers will be used during all data collection and analysis and the link to the subject identifiers will be stored in a secure manner;
- The subjective information you share here is used by the researchers for the purposes of this study, and no others will have access to the data.

This will take about 7-9 minutes



3. How familiar are you with ... ? *

Mark only one oval per row.

	Unfamiliar	Somewhat unfamiliar	Somewhat familiar	Familiar	N/A
Watson Workspace	○	○	○	○	○
Team collaboration applications	○	○	○	○	○

4. Do you think that the “plot” was understandable? *

Mark only one oval.

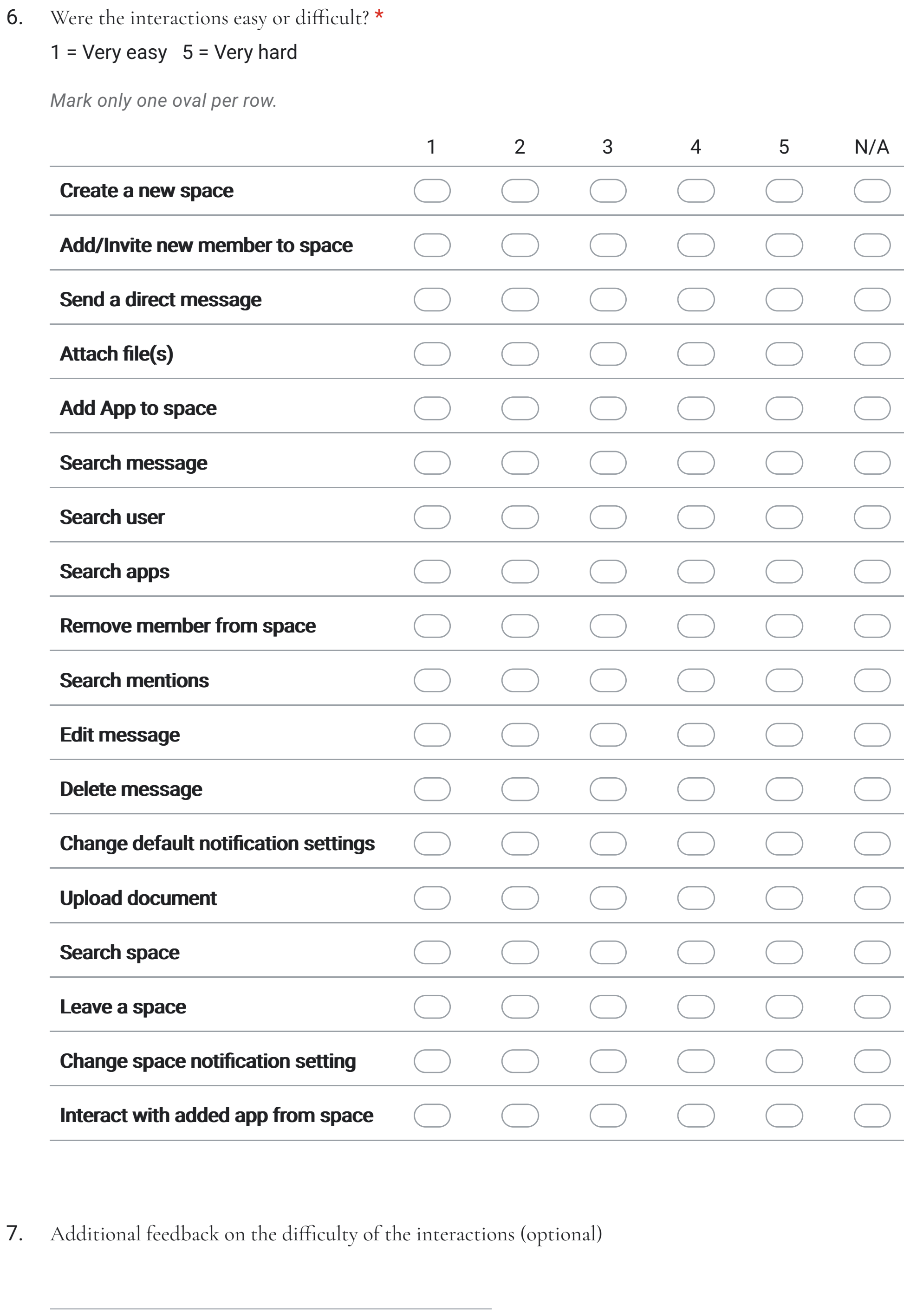
○ Yes

○ No

5. How difficult is it to follow the “plot”? *

Mark only one oval.

	1	2	3	4	5	
Very easy	○	○	○	○	○	Very hard



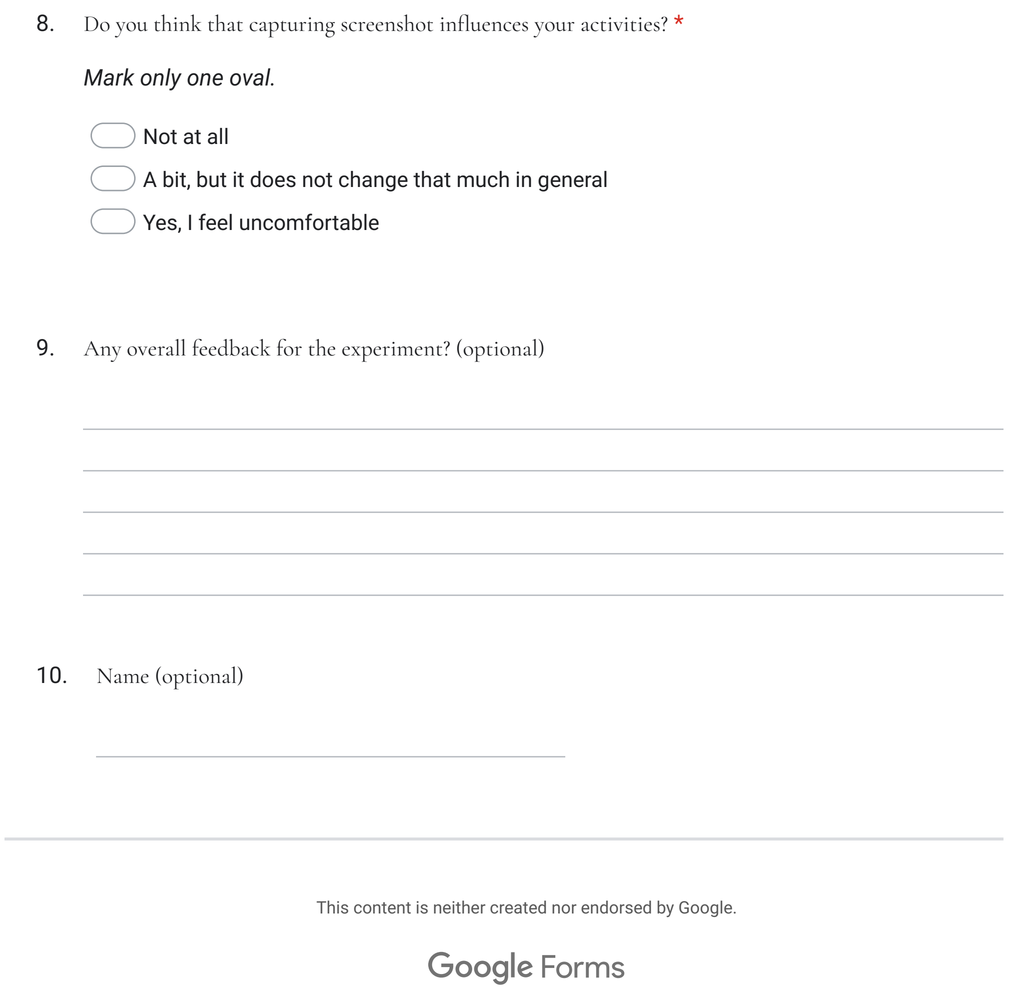
6. Were the interactions easy or difficult? *

1 = Very easy   5 = Very hard

Mark only one oval per row.

	1	2	3	4	5	N/A
Create a new space	○	○	○	○	○	○
Add/Invite new member to space	○	○	○	○	○	○
Send a direct message	○	○	○	○	○	○
Attach file(s)	○	○	○	○	○	○
Add App to space	○	○	○	○	○	○
Search message	○	○	○	○	○	○
Search user	○	○	○	○	○	○
Search apps	○	○	○	○	○	○
Remove member from space	○	○	○	○	○	○
Search mentions	○	○	○	○	○	○
Edit message	○	○	○	○	○	○
Delete message	○	○	○	○	○	○
Change default notification settings	○	○	○	○	○	○
Upload document	○	○	○	○	○	○
Search space	○	○	○	○	○	○
Leave a space	○	○	○	○	○	○
Change space notification setting	○	○	○	○	○	○
Interact with added app from space	○	○	○	○	○	○

7. Additional feedback on the difficulty of the interactions (optional)



8. Do you think that capturing screenshot influences your activities? *

Mark only one oval.

○ Not at all

○ A bit, but it does not change that much in general

○ Yes, I feel uncomfortable

9. Any overall feedback for the experiment? (optional)

10. Name (optional)

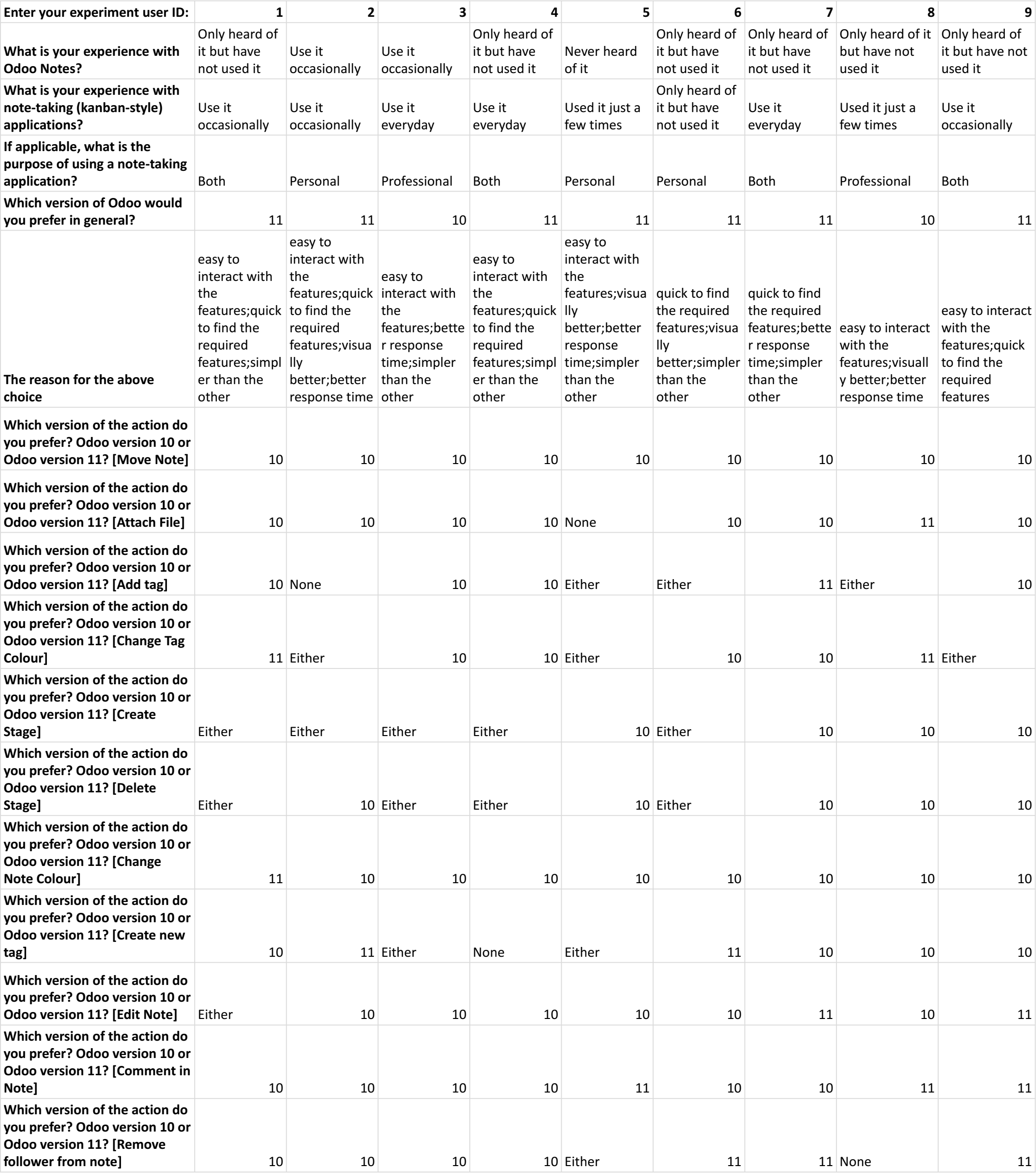
This content is neither created nor endorsed by Google.

Google Forms

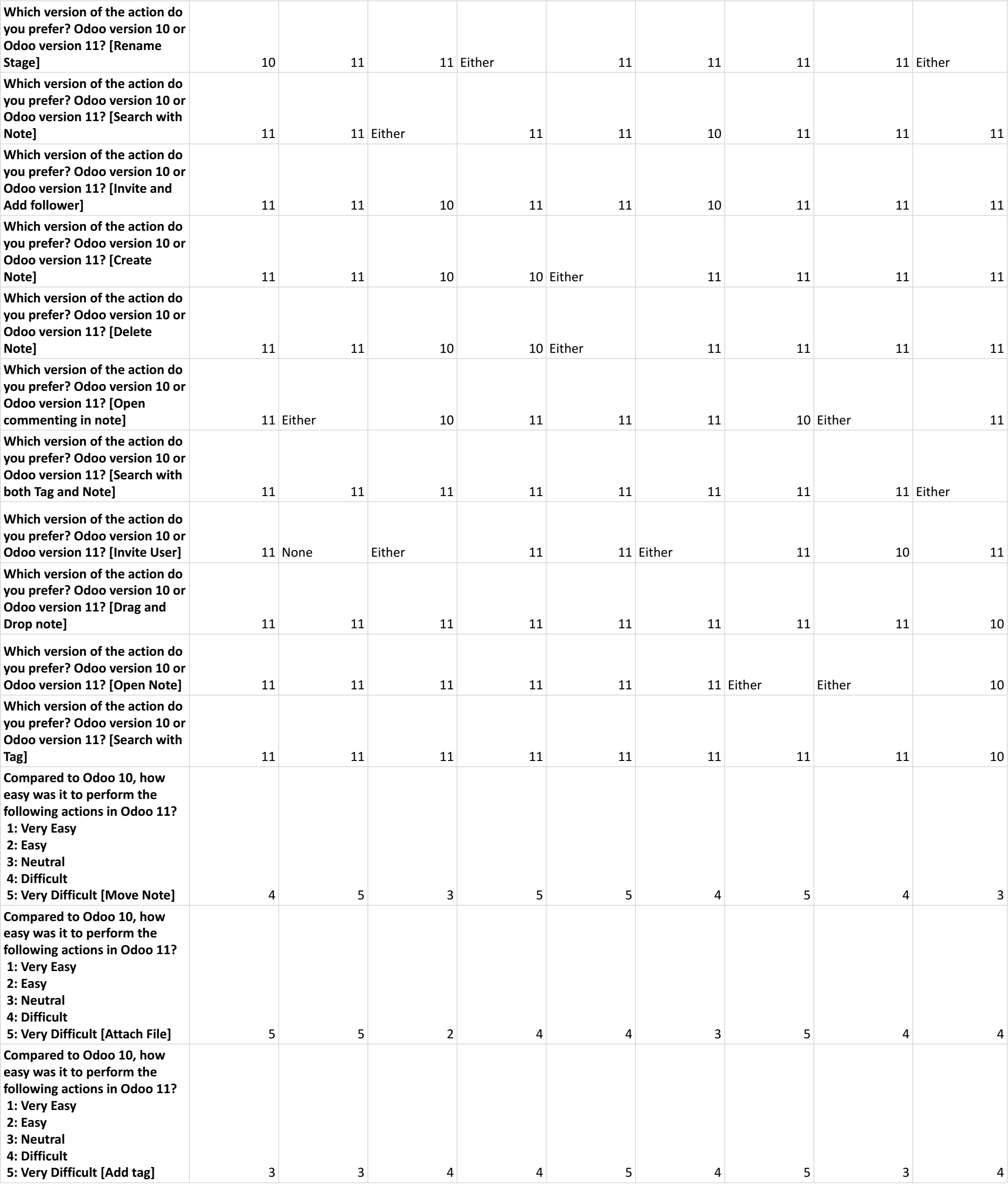
Appendix E

Survey Results

E.1 Survey Results - Odoo Notes

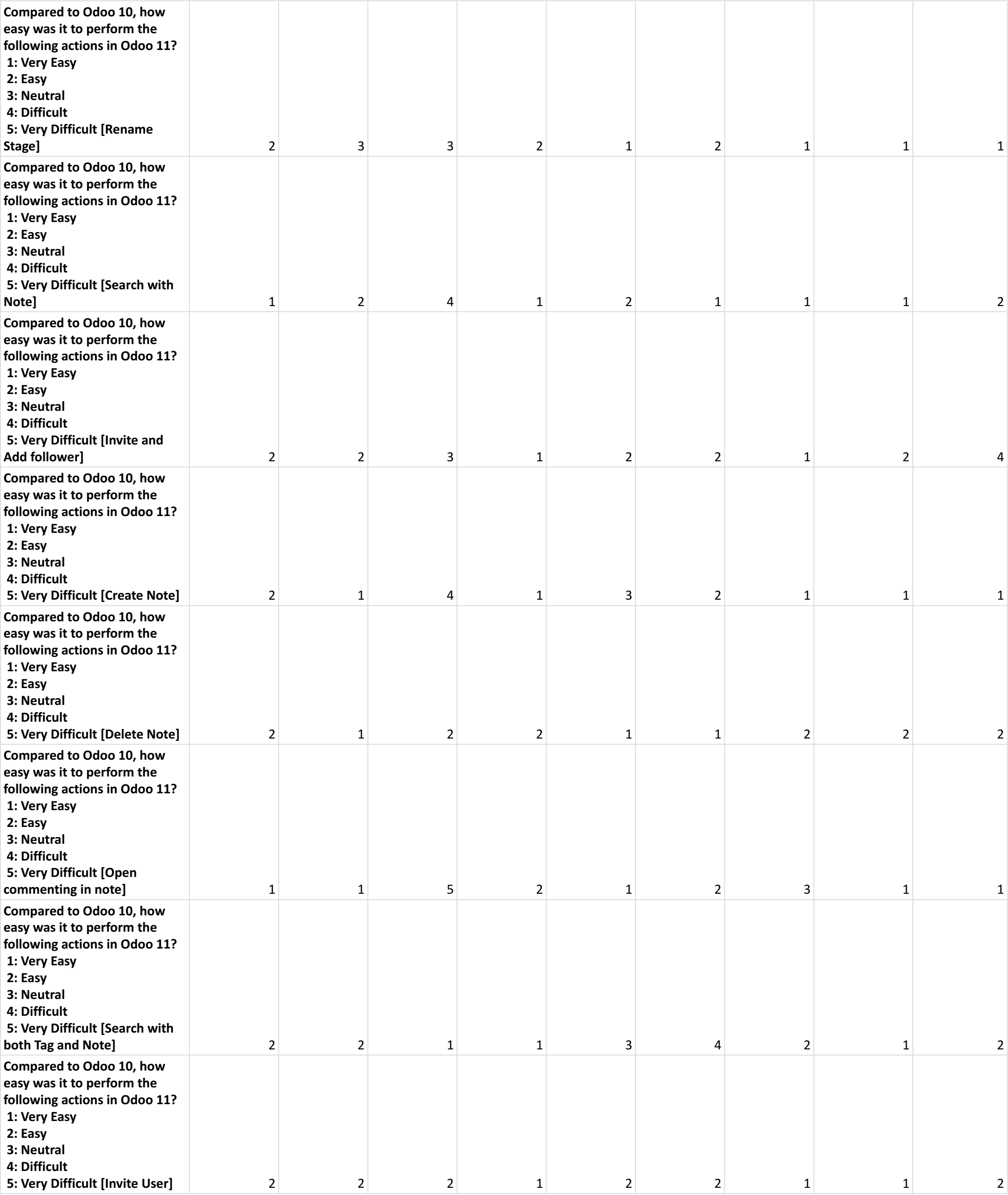


Enter your experiment user ID:	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**
What is your experience with Odoo Notes?	Only heard of it but have not used it	Use it occasionally	Use it occasionally	Only heard of it but have not used it	Never heard of it	Only heard of it but have not used it	Only heard of it but have not used it	Only heard of it but have not used it	Only heard of it but have not used it
What is your experience with note-taking (kanban-style) applications?	Use it occasionally	Use it occasionally	Use it everyday	Use it everyday	Used it just a few times	Only heard of it but have not used it	Use it everyday	Used it just a few times	Use it occasionally
If applicable, what is the purpose of using a note-taking application?	Both	Personal	Professional	Both	Personal	Personal	Both	Professional	Both
Which version of Odoo would you prefer in general?	11	11	10	11	11	11	11	10	11
The reason for the above choice	easy to interact with the features;quick to find the required features;simpler than the other	easy to interact with the features;quick to find the required features;visually better;better response time	easy to interact with the features;better response time;simpler than the other	easy to interact with the features;quick to find the required features;simpler than the other	easy to interact with the features;visually better;better response time;simpler than the other	quick to find the required features;visually better;simpler than the other	quick to find the required features;better response time;simpler than the other	easy to interact with the features;visually better;better response time	easy to interact with the features;quick to find the required features
Which version of the action do you prefer? Odoo version 10 or Odoo version 11? [Move Note]	10	10	10	10	10	10	10	10	10
Which version of the action do you prefer? Odoo version 10 or Odoo version 11? [Attach File]	10	10	10	10	None	10	10	11	10
Which version of the action do you prefer? Odoo version 10 or Odoo version 11? [Add tag]	10	None	10	10	Either	Either	11	Either	10
Which version of the action do you prefer? Odoo version 10 or Odoo version 11? [Change Tag Colour]	11	Either	10	10	Either	10	10	11	Either
Which version of the action do you prefer? Odoo version 10 or Odoo version 11? [Create Stage]	Either	Either	Either	Either	10	Either	10	10	10
Which version of the action do you prefer? Odoo version 10 or Odoo version 11? [Delete Stage]	Either	10	Either	Either	10	Either	10	10	10
Which version of the action do you prefer? Odoo version 10 or Odoo version 11? [Change Note Colour]	11	10	10	10	10	10	10	10	10
Which version of the action do you prefer? Odoo version 10 or Odoo version 11? [Create new tag]	10	11	Either	None	Either	11	10	10	10
Which version of the action do you prefer? Odoo version 10 or Odoo version 11? [Edit Note]	Either	10	10	10	10	10	11	10	11
Which version of the action do you prefer? Odoo version 10 or Odoo version 11? [Comment in Note]	10	10	10	10	11	10	10	11	11
Which version of the action do you prefer? Odoo version 10 or Odoo version 11? [Remove follower from note]	10	10	10	10	Either	11	11	None	11

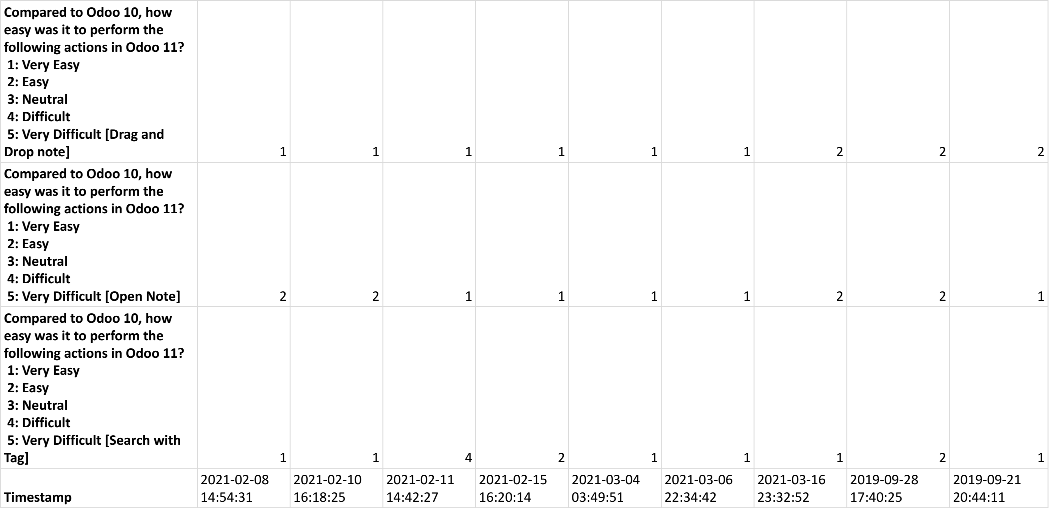


Which version of the action do you prefer? Odoo version 10 or Odoo version 11? [Rename Stage]	10	11	11	Either	11	11	11	11	Either
Which version of the action do you prefer? Odoo version 10 or Odoo version 11? [Search with Note]	11	11	Either	11	11	10	11	11	11
Which version of the action do you prefer? Odoo version 10 or Odoo version 11? [Invite and Add follower]	11	11	10	11	11	10	11	11	11
Which version of the action do you prefer? Odoo version 10 or Odoo version 11? [Create Note]	11	11	10	10	Either	11	11	11	11
Which version of the action do you prefer? Odoo version 10 or Odoo version 11? [Delete Note]	11	11	10	10	Either	11	11	11	11
Which version of the action do you prefer? Odoo version 10 or Odoo version 11? [Open commenting in note]	11	Either	10	11	11	11	10	Either	11
Which version of the action do you prefer? Odoo version 10 or Odoo version 11? [Search with both Tag and Note]	11	11	11	11	11	11	11	11	Either
Which version of the action do you prefer? Odoo version 10 or Odoo version 11? [Invite User]	11	None	Either	11	11	Either	11	10	11
Which version of the action do you prefer? Odoo version 10 or Odoo version 11? [Drag and Drop note]	11	11	11	11	11	11	11	11	10
Which version of the action do you prefer? Odoo version 10 or Odoo version 11? [Open Note]	11	11	11	11	11	11	Either	Either	10
Which version of the action do you prefer? Odoo version 10 or Odoo version 11? [Search with Tag]	11	11	11	11	11	11	11	11	10
Compared to Odoo 10, how easy was it to perform the following actions in Odoo 11? 1: Very Easy 2: Easy 3: Neutral 4: Difficult 5: Very Difficult [Move Note]	4	5	3	5	5	4	5	4	3
Compared to Odoo 10, how easy was it to perform the following actions in Odoo 11? 1: Very Easy 2: Easy 3: Neutral 4: Difficult 5: Very Difficult [Attach File]	5	5	2	4	4	3	5	4	4
Compared to Odoo 10, how easy was it to perform the following actions in Odoo 11? 1: Very Easy 2: Easy 3: Neutral 4: Difficult 5: Very Difficult [Add tag]	3	3	4	4	5	4	5	3	4

Compared to Odoo 10, how easy was it to perform the following actions in Odoo 11? 1: Very Easy 2: Easy 3: Neutral 4: Difficult 5: Very Difficult [Change Tag Colour]	3	3	4	4	4	5	5	4	5
Compared to Odoo 10, how easy was it to perform the following actions in Odoo 11? 1: Very Easy 2: Easy 3: Neutral 4: Difficult 5: Very Difficult [Create Stage]	3	1	2	3	4	2	2	2	4
Compared to Odoo 10, how easy was it to perform the following actions in Odoo 11? 1: Very Easy 2: Easy 3: Neutral 4: Difficult 5: Very Difficult [Delete Stage]	4	4	1	2	3	4	3	4	3
Compared to Odoo 10, how easy was it to perform the following actions in Odoo 11? 1: Very Easy 2: Easy 3: Neutral 4: Difficult 5: Very Difficult [Change Note Colour]	3	3	2	3	5	3	4	4	2
Compared to Odoo 10, how easy was it to perform the following actions in Odoo 11? 1: Very Easy 2: Easy 3: Neutral 4: Difficult 5: Very Difficult [Create new tag]	4	2	3	4	3	4	4	2	4
Compared to Odoo 10, how easy was it to perform the following actions in Odoo 11? 1: Very Easy 2: Easy 3: Neutral 4: Difficult 5: Very Difficult [Edit Note]	3	4	2	5	2	4	2	1	3
Compared to Odoo 10, how easy was it to perform the following actions in Odoo 11? 1: Very Easy 2: Easy 3: Neutral 4: Difficult 5: Very Difficult [Comment in Note]	3	4	1	3	4	5	1	1	2
Compared to Odoo 10, how easy was it to perform the following actions in Odoo 11? 1: Very Easy 2: Easy 3: Neutral 4: Difficult 5: Very Difficult [Remove follower from note]	3	5	2	2	2	2	2	1	2



Compared to Odoo 10, how easy was it to perform the following actions in Odoo 11? 1: Very Easy 2: Easy 3: Neutral 4: Difficult 5: Very Difficult [Rename Stage]	2	3	3	2	1	2	1	1	1
Compared to Odoo 10, how easy was it to perform the following actions in Odoo 11? 1: Very Easy 2: Easy 3: Neutral 4: Difficult 5: Very Difficult [Search with Note]	1	2	4	1	2	1	1	1	2
Compared to Odoo 10, how easy was it to perform the following actions in Odoo 11? 1: Very Easy 2: Easy 3: Neutral 4: Difficult 5: Very Difficult [Invite and Add follower]	2	2	3	1	2	2	1	2	4
Compared to Odoo 10, how easy was it to perform the following actions in Odoo 11? 1: Very Easy 2: Easy 3: Neutral 4: Difficult 5: Very Difficult [Create Note]	2	1	4	1	3	2	1	1	1
Compared to Odoo 10, how easy was it to perform the following actions in Odoo 11? 1: Very Easy 2: Easy 3: Neutral 4: Difficult 5: Very Difficult [Delete Note]	2	1	2	2	1	1	2	2	2
Compared to Odoo 10, how easy was it to perform the following actions in Odoo 11? 1: Very Easy 2: Easy 3: Neutral 4: Difficult 5: Very Difficult [Open commenting in note]	1	1	5	2	1	2	3	1	1
Compared to Odoo 10, how easy was it to perform the following actions in Odoo 11? 1: Very Easy 2: Easy 3: Neutral 4: Difficult 5: Very Difficult [Search with both Tag and Note]	2	2	1	1	3	4	2	1	2
Compared to Odoo 10, how easy was it to perform the following actions in Odoo 11? 1: Very Easy 2: Easy 3: Neutral 4: Difficult 5: Very Difficult [Invite User]	2	2	2	1	2	2	1	1	2



Compared to Odoo 10, how easy was it to perform the following actions in Odoo 11? 1: Very Easy 2: Easy 3: Neutral 4: Difficult 5: Very Difficult [Drag and Drop note]	1	1	1	1	1	1	2	2	2
Compared to Odoo 10, how easy was it to perform the following actions in Odoo 11? 1: Very Easy 2: Easy 3: Neutral 4: Difficult 5: Very Difficult [Open Note]	2	2	1	1	1	1	2	2	1
Compared to Odoo 10, how easy was it to perform the following actions in Odoo 11? 1: Very Easy 2: Easy 3: Neutral 4: Difficult 5: Very Difficult [Search with Tag]	1	1	4	2	1	1	1	2	1
Timestamp	2021-02-08 14:54:31	2021-02-10 16:18:25	2021-02-11 14:42:27	2021-02-15 16:20:14	2021-03-04 03:49:51	2021-03-06 22:34:42	2021-03-16 23:32:52	2019-09-28 17:40:25	2019-09-21 20:44:11

Appendix F

Consistency Scores

The consistency scores for the experiment tasks with the Odoo Notes application are shown in the following table:

userId	actionName	consistency_frequency	consistency_timespent
1	Add tag	-0.01356857343	-18.90453815
1	Attach File	0	0
1	Change Note Colour	-0.03379042211	-37.85884213
1	Change Tag Colour	0.006915298175	90.36656308
1	Comment in Note	-0.006653275252	-7.796715021
1	Create Note	-0.05427429371	-0.7291219234
1	Create Stage	0	0
1	Create new tag	0.003588660549	10.89690995
1	Delete Note	-0.003326637626	-0.1371440887
1	Delete Stage	0	0
1	Drag and Drop note	-0.04070572028	-16.39874506
1	Edit Note	0.03764110558	47.99980903
1	Invite User	0	0
1	Invite and Add follower	-0.006653275252	-8.825518131
1	Move Note	0.05120967901	43.70661092
1	Open Note	0.0007860687685	99.99578142
1	Open commenting in note	-0.003326637626	7.930943012
1	Remove follower from note	0	0

1	Rename Stage	0	0
1	Search with Note	0	0
1	Search with Tag	0	0
1	Search with both Tag and Note	0	0
2	Add tag	0.07528221116	63.52208877
2	Attach File	0	0
2	Change Note Colour	-0.04377033498	-61.4874022
2	Change Tag Colour	-0.01689521105	-31.42152095
2	Comment in Note	-0.003326637626	-1.036033154
2	Create Note	-0.02354848631	-8.900658846
2	Create Stage	0.006915298175	1.421996832
2	Create new tag	-0.03046378448	-30.02299309
2	Delete Note	-0.02713714685	-7.428654909
2	Delete Stage	0.006915298175	26.44088101
2	Drag and Drop note	-0.02713714685	-30.29785299
2	Edit Note	0.01050395872	-2.140885592
2	Invite User	0	0
2	Invite and Add follower	0.006915298175	12.91068697
2	Move Note	0.102419358	85.08754301
2	Open Note	0.1679836792	91.71218753
2	Open commenting in note	0.01715723398	13.15757418
2	Remove follower from note	0.0102419358	25.33299494
2	Rename Stage	0.006915298175	15.07517123
2	Search with Note	0.09576608276	66.49722314
2	Search with Tag	0.0379031285	58.69153404
2	Search with both Tag and Note	0.0409677432	40.74921799
3	Add tag	-0.002540568858	15.52653718
3	Attach File	0	0
3	Change Note Colour	0.004374729317	-41.48875117
3	Change Tag Colour	0.02074589453	15.88193941
3	Comment in Note	-0.05068563316	-14.43976116
3	Create Note	-0.01278250466	122.0679915

3	Create Stage	-0.003326637626	-14.81203485
3	Create new tag	-0.009979912878	3.511626244
3	Delete Note	-0.03352839918	27.75095606
3	Delete Stage	0	0
3	Drag and Drop note	-0.2578028951	-194.6696591
3	Edit Note	0.02100791745	19.69087315
3	Invite User	0	0
3	Invite and Add follower	0.01409261927	48.23670697
3	Move Note	0.2458064592	156.8133934
3	Open Note	-0.1192349087	-51.23088598
3	Open commenting in note	-0.06066554604	18.04998851
3	Remove follower from note	0.0102419358	6.929500103
3	Rename Stage	0	0
3	Search with Note	-0.009979912878	2.328601837
3	Search with Tag	-0.1182664774	-31.38717341
3	Search with both Tag and Note	-0.01689521105	-0.9250850677
4	Add tag	-0.02993973863	9.114121437
4	Attach File	0	0
4	Change Note Colour	-0.05707688549	16.08052063
4	Change Tag Colour	-0.009979912878	2.598886967
4	Comment in Note	-0.03352839918	26.95632482
4	Create Note	-0.03659301389	172.0924573
4	Create Stage	0.006915298175	25.70051575
4	Create new tag	-0.006653275252	-0.8949689865
4	Delete Note	0.007177321098	38.86104465
4	Delete Stage	0.0102419358	5.514264107
4	Drag and Drop note	-0.284940042	-160.4548419
4	Edit Note	-0.02687512393	-11.10823894
4	Invite User	0	0
4	Invite and Add follower	-0.009717889955	8.926298141
4	Move Note	0.2355645234	325.1020932
4	Open Note	-0.313569666	-256.7997551

4	Open commenting in note	-0.05016158731	43.01380348
4	Remove follower from note	0	0
4	Rename Stage	0.0102419358	9.752995014
4	Search with Note	-0.009979912878	-3.317331791
4	Search with Tag	-0.06758084421	-20.53127217
4	Search with both Tag and Note	-0.01689521105	-1.294410944
5	Add tag	-0.02993973863	41.21458507
5	Attach File	0	0
5	Change Note Colour	0.01102800457	20.57898116
5	Change Tag Colour	-0.009979912878	-0.9950740337
5	Comment in Note	0.003850683472	-11.88265896
5	Create Note	-0.03326637626	187.5616546
5	Create Stage	-0.003326637626	9.961589098
5	Create new tag	-0.009979912878	-1.648380041
5	Delete Note	-0.01995982576	70.44967103
5	Delete Stage	0	0
5	Drag and Drop note	-0.3120771888	-146.8916204
5	Edit Note	-0.009979912878	-18.70308185
5	Invite User	0	0
5	Invite and Add follower	-0.01995982576	11.15162659
5	Move Note	0.2253225876	265.4299929
5	Open Note	-0.2424001612	-96.77872944
5	Open commenting in note	-0.006129229407	9.717289925
5	Remove follower from note	0	0
5	Rename Stage	0.0102419358	71.99710393
5	Search with Note	-0.006653275252	1.376310825
5	Search with Tag	-0.08062537179	12.32108426
5	Search with both Tag and Note	-0.01689521105	-0.6063678265
6	Add tag	0.007439344021	51.55554485
6	Attach File	0	0
6	Change Note Colour	0.01768127982	12.38658571
6	Change Tag Colour	-0.006653275252	-1.682683945

6	Comment in Note	-0.02328646338	45.40091372
6	Create Note	-0.03659301389	308.8317707
6	Create Stage	-0.003326637626	0.5205657482
6	Create new tag	-0.006653275252	0.1369864941
6	Delete Note	-0.03020176156	19.00878811
6	Delete Stage	0	0
6	Drag and Drop note	-0.3120771888	-208.8755009
6	Edit Note	-0.009979912878	-123.8971648
6	Invite User	0	0
6	Invite and Add follower	0.0005240458456	20.51992488
6	Move Note	0.2253225876	172.8433774
6	Open Note	-0.3852632166	-179.1894655
6	Open commenting in note	-0.03326637626	15.08556271
6	Remove follower from note	0	0
6	Rename Stage	0	0
6	Search with Note	0.01383059635	10.86717153
6	Search with Tag	-0.08088739472	-7.483486414
6	Search with both Tag and Note	-0.006653275252	3.947545767
7	Add tag	-0.009717889955	99.28732777
7	Attach File	0	0
7	Change Note Colour	-0.02328646338	18.10694098
7	Change Tag Colour	-0.02022184868	-4.183352232
7	Comment in Note	-0.02328646338	-35.5171926
7	Create Note	-0.02302444046	158.0554397
7	Create Stage	-0.003326637626	1.292878866
7	Create new tag	-0.006653275252	-6.967528582
7	Delete Note	-0.003064614703	13.17255497
7	Delete Stage	0	0
7	Drag and Drop note	-0.2442343217	-111.4076192
7	Edit Note	-0.009979912878	36.80671501
7	Invite User	0	0
7	Invite and Add follower	-0.02328646338	74.21465635

7	Move Note	0.2355645234	174.7639158
7	Open Note	-0.5150817444	-230.2650464
7	Open commenting in note	0.04840708723	54.56279135
7	Remove follower from note	0	0
7	Rename Stage	0	0
7	Search with Note	-0.04735899553	-21.80365825
7	Search with Tag	-0.03020176156	-36.52036524
7	Search with both Tag and Note	-0.003326637626	2.1539042
8	Add tag	-0.04989956439	41.59093618
8	Attach File	0	0
8	Change Note Colour	-0.04989956439	6.386523008
8	Change Tag Colour	-0.009717889955	67.86098933
8	Comment in Note	-0.03711705973	-18.18979216
8	Create Note	-0.05296417909	150.0753648
8	Create Stage	-0.003326637626	61.84516883
8	Create new tag	-0.009979912878	5.674592972
8	Delete Note	-0.03020176156	19.402704
8	Delete Stage	0	0
8	Drag and Drop note	-0.379920056	-139.2924292
8	Edit Note	-0.009979912878	-3.017534018
8	Invite User	0	0
8	Invite and Add follower	-0.03020176156	-2.150266886
8	Move Note	0.2867742024	184.6075647
8	Open Note	-0.4343767122	-123.7036674
8	Open commenting in note	-0.01584711936	70.94940376
8	Remove follower from note	0	0
8	Rename Stage	0	0
8	Search with Note	0.04122976613	67.23090315
8	Search with Tag	-0.08780269289	-14.26381278
8	Search with both Tag and Note	-0.006653275252	0.7595999241
9	Add tag	-0.02661310101	-14.2554841
9	Attach File	0	0

9	Change Note Colour	0.01768127982	16.26469588
9	Change Tag Colour	-0.009979912878	-32.35613108
9	Comment in Note	0.02407253215	18.8578794
9	Create Note	-0.03326637626	171.2920566
9	Create Stage	-0.003326637626	22.79887009
9	Create new tag	-0.009979912878	2.222717285
9	Delete Note	-0.03685503681	26.30660129
9	Delete Stage	0	0
9	Drag and Drop note	-0.3120771888	-147.7014315
9	Edit Note	0.007177321098	30.76896572
9	Invite User	0	0
9	Invite and Add follower	-0.003326637626	-0.5448291302
9	Move Note	0.2253225876	187.019875
9	Open Note	-0.1878638446	116.9649484
9	Open commenting in note	-0.01637116521	40.39643526
9	Remove follower from note	0	0
9	Rename Stage	0	0
9	Search with Note	-0.02354848631	-9.84709692
9	Search with Tag	-0.1082865645	-1.981745958
9	Search with both Tag and Note	-0.01689521105	-13.40402484

Bibliography

Abrahamsson, P., Salo, O., Ronkainen, J., and Warsta, J. (2002). Agile software development methods: Review and analysis. *Espoo, Finl. Tech. Res. Cent. Finland, VTT Publ.*, page 478.

Abras, C., Maloney-Krichmar, D., Preece, J., et al. (2004). User-centered design. *Bainbridge, W. Encyclopedia of Human-Computer Interaction. Thousand Oaks: Sage Publications*, 37(4):445–456.

Aggarwal, C. C. and Yu, P. S. (2009). *Data mining: The textbook.* Springer.

Al-Bayati, B., Clarke, N., and Dowland, P. (2016). Adaptive behavioral profiling for identity verification in cloud computing: A model and preliminary analysis. *GSTF Journal on Computing (JoC)*, 5(1):21.

Alahyari, H., Svensson, R., and Gorschek, T. (2017). Hybrid approaches in software engineering: Combining qualitative and quantitative methods. *Journal of Software: Evolution and Process*, 29(8):e1905.

Aldhaheri, R. and Abdullah, R. (2020). A multi-metric approach to software feature prioritization using usage analytics. *Journal of Software: Evolution and Process*, 32(10):e2256.

Amininiaki, S. and Saidi, M. (2024). Advanced log data analysis techniques for software development. *International Journal of Software Engineering and Knowledge Engineering*, 34(2):145–162.

Apel, S., Lengauer, C., Möller, B., and Kästner, C. (2008). An algebra for features and feature composition. In *International Conference on Algebraic Methodology and Software Technology*, pages 36–50. Springer.

Appsero (2023). 7 steps to perform user feedback analysis for software development.

Arias, G., Vilches, D., Banchoff, C., Harari, I., Harari, V., and Iuliano, P. (2012). The 7 key factors to get successful results in the IT Development projects. *Procedia Technol.*, 5:199–207.

Arora, G. and Malik, R. (2017). Evaluating consistency in user interactions for enhancing software usability. *Empirical Software Engineering*, 22(6):3023–3050.

Atterer, R., Wnuk, M., and Schmidt, A. (2006). Knowing the user's every move. *Proceedings of the 15th international conference on World Wide Web - WWW '06*, page 203.

Ballandies, M. C., Holzwarth, V., Sunderland, B., Pournaras, E., and Brocke, J. v. (2022). Constructing effective customer feedback systems–a design science study leveraging blockchain technology. *arXiv preprint arXiv:2203.15254*.

Banerjee, A. and Ghosh, J. (2001). Clickstream clustering using weighted longest common subsequences. In *Proceedings of the web mining workshop at the 1st SIAM conference on data mining*, volume 143, page 144.

Batory, D. (2006). Feature modularity for product-lines. *Tutorial at: OOPSLA*, 6:9.

Batory, D., Benavides Cuevas, D. F., and Ruiz Cortés, A. (2006). Automated analysis of feature models: challenges ahead. *Communications of the ACM-Software product line, 49 (12), 45-47.*

Batory, D., Sarvela, J. N., and Rauschmayer, A. (2004). Scaling step-wise refinement. *IEEE Transactions on Software Engineering*, 30(6):355–371.

Bauer, M., Sergieieva, K., and Meixner, G. (2017). Enabling Focused Software Quality Assurance in Agile Software Development Processes for Mobile Applications using Text and Usage Mining Methods. In *Proc. 12th Int. Jt. Conf. Comput. Vision, Imaging Comput. Graph. Theory Appl.*, pages 128–132. SCITEPRESS - Science and Technology Publications.

Beck, K., Beedle, M., Van Bennekum, A., Cockburn, A., Cunningham, W., Fowler, M., Grenning, J., Highsmith, J., Hunt, A., Jeffries, R., et al. (2001). Manifesto for agile software development.

Benyon, D. (2013). *Designing Interactive Systems: A Comprehensive Guide to HCI, UX and Interaction Design.* Pearson Education.

Berndtsson, M., Forsberg, D., Stein, D., and Svahn, T. (2018). BECOMING A DATA-DRIVEN

ORGANISATION. In *Proceedings of the 26th European Conference on Information Systems (ECIS2018)*, Portsmouth, United Kingdom. AIS.

Beyer, H. and Holtzblatt, K. (1997). *Contextual Design: Defining Customer-Centered Systems.* Morgan Kaufmann.

Bezemer, C. P., Zaidman, A., Platzbeecker, B., Hurkmans, T., and Hart, A. (2010). Enabling multi-tenancy: An industrial experience report. *IEEE International Conference on Software Maintenance, ICSM.*

Boehm, B. W. (1988). A spiral model of software development and enhancement. *Computer*, 21(5):61–72.

Bosch, J. (2000). *Design and use of software architectures: adopting and evolving a product-line approach.* Pearson Education.

Bosch, J. (2012). Building products as innovation experiment systems. *Lect. Notes Bus. Inf. Process.*, 114 LNBIP(Icsob):27–39.

Bosch, J. and Olsson, H. (2014). Continuous software engineering: An update on the state of the art. *Journal of Systems and Software*, 95:68–86.

Bosch, J. and Olsson, H. (2016). The real-time feedback imperative in software development. In *2016 24th ACM SIGSOFT International Symposium on Foundations of Software Engineering*, pages 847–850. ACM.

Bosch, J., Olsson, H. H., Björk, J., and Crnkovic, I. (2014). The early stage software startup development model: A framework for operationalizing lean principles in software startups. In *Agile Conference*, pages 40–50. IEEE.

Bucklin, R. E. and Sismeiro, C. (2009a). Click Here for Interact Insight: Advances in Clickstream Data Analysis in Marketing. *Journal of Interactive Marketing*, 23(1):35–48.

Bucklin, R. E. and Sismeiro, C. (2009b). Modeling the clickstream: Implications for web-based advertising efforts. *Marketing Science.*

Buse, R. P. and Zimmermann, T. (2012a). Data quality in software engineering. In *Proceedings of the 34th International Conference on Software Engineering*, pages 1257–1267.

Buse, R. P. and Zimmermann, T. (2012b). Information needs for software development analytics. *Proceedings of the 34th International Conference on Software Engineering (ICSE)*, pages 987–996.

Buse, R. P. L. and Zimmermann, T. (2012c). Analytics for software development. In *Proceedings of the 34th International Conference on Software Engineering (ICSE)*, pages 921–930. IEEE.

CGI (2023). The future of agile: trends and predictions for 2023 and beyond.

Chaffey, D. and Patron, D. (2012). *Digital Business and E-commerce Management: Strategy, Implementation and Practice*. Pearson Education.

Chandra, D. G. and Malaya, D. B. (2012). Role of cloud computing in education. In *Computing, Electronics and Electrical Technologies (ICCEET), 2012 International Conference on*, pages 832–836. IEEE.

Chen, C. C., Liu, J. Y. C., and Chen, H. G. (2011). Discriminative effect of user influence and user responsibility on information system development processes and project management. *Inf. Softw. Technol.*, 53(2):149–158.

Chen, C.-H., Wu, C.-H., and Shen, C.-L. (2013). Data mining for the online retail industry: A case study of rfm model-based customer segmentation using data mining. *Journal of Database Marketing & Customer Strategy Management*, 19(3):197–208.

Chen, J. and Jin, Y. (2016a). A survey on feature location techniques. *Journal of Software: Evolution and Process*, 28(10):826–852.

Chen, L. and Jin, L. (2016b). A scalable feedback mechanism for modern software development. *Journal of Software: Evolution and Process*, 28(8):660–671.

Chen, L. and Liu, Y. (2017). User-centric feature prioritization in software development: A case study. *Journal of Systems and Software*, 123:159–171.

Cito, J., Leitner, P., Gall, H. C., Dadashi, A., Keller, A., and Roth, A. (2015a). Runtime metric meets developer: building better cloud applications using feedback. In *2015 ACM International Symposium on New Ideas, New Paradigms, and Reflections on Programming and Software (Onward!)*, pages 14–27. ACM.

Cito, J., Leitner, P., Gall, H. C., Dadashi, A., Keller, A., and Roth, A. (2015b). Runtime Metric Meets Developer: Building Better Cloud Applications Using Feedback. In *2015 ACM Int. Symp. New Ideas, New Paradig. Reflections Program. Softw.*, pages 14–27, New York, New York, USA. ACM Press.

Claps, G. G., Berntsson Svensson, R., and Aurum, A. (2015). On the journey to continuous deployment: Technical and social challenges along the way. *Inf. Softw. Technol.*, 57(1):21–31.

Classen, A., Heymans, P., and Schobbens, P.-Y. (2008). What's in a feature: A requirements engineering perspective. In *International Conference on Fundamental Approaches to Software Engineering*, pages 16–30. Springer.

Claypool, M., Brown, P. L., Le, T., and Waseda, M. (2001a). Inferring user interest from navigation patterns. *Proceedings of the International Conference on Intelligent User Interfaces*, pages 33–40.

Claypool, M. et al. (2001b). Implicit interest indicators. In *Proceedings of the 6th International Conference on Intelligent User Interfaces*, pages 33–40.

Claypool, M., Le, P., Wased, M., and Brown, D. (2001c). Implicit interest indicators. *Proceedings of the 6th International Conference on Intelligent User Interfaces*, pages 33–40.

Creswell, J. W. (2002). *Educational research: Planning, conducting, and evaluating quantitative.* Prentice Hall Upper Saddle River, NJ.

Crowston, K. and Kammerer, E. (2003). Analyzing the reliability of user behavior patterns in software usage data. *Journal of Software Maintenance and Evolution: Research and Practice*, 15(5):345–367.

Czarnecki, K., Østerbye, K., and Völter, M. (2002). Generative programming. In *European Conference on Object-Oriented Programming*, pages 15–29. Springer.

De Chaves, S. A., Uriarte, R. B., and Westphall, C. B. (2011). Toward an architecture for monitoring private clouds. *IEEE Communications Magazine*, 49(12):130–137.

Deka, G. and Vemuru, S. (2021). Best practices for implementing usage analytics in modern software development environments. *Journal of Systems and Software*, 173:110878.

Duvall, P. M., Matyas, S., and Glover, A. (2007). *Continuous Integration: Improving Software Quality and Reducing Risk.* Addison-Wesley Professional.

Dyckhoff, A. L., Zielke, D., Bültmann, M., Chatti, M. A., and Schroeder, U. (2012). Design and implementation of a learning analytics toolkit for teachers. *Educational Technology and Society*, 15(3):58–76.

Fabijan, A., Olsson, H. H., and Bosch, J. (2015). Customer Feedback and Data Collection Techniques in Software R&D: A Literature Review. In *2014 40th EUROMICRO Conf. Softw. Eng. Adv. Appl.*, number 210, pages 139–153. IEEE.

Fabijan, A., Olsson, H. H., and Bosch, J. (2016a). The Lack of Sharing of Customer Data in Large Software Organizations: Challenges and Implications. In *Int. Conf. Agil. Softw. Dev. Springer, Cham*, volume 251, pages 39–52.

Fabijan, A., Olsson, H. H., and Bosch, J. (2016b). Time to Say 'Good Bye': Feature Lifecycle. *Proc. - 42nd Euromicro Conf. Softw. Eng. Adv. Appl. SEAA 2016*, pages 9–16.

Fagerholm, F., Guinea, A. S., Mäenpää, H., and Münch, J. (2014). Building blocks for continuous experimentation. In *Proc. 1st Int. Work. Rapid Contin. Softw. Eng. - RCoSE 2014*, pages 26–35, New York, New York, USA. ACM Press.

Féris, M. A. A., Zwikael, O., and Gregor, S. (2017). Qplan: Decision support for evaluating planning quality in software development projects. *Decision Support Systems*, 96:92–102.

Ferreira, M. and Costa, P. (2024). Integrating surveys with user feedback mechanisms in software development. *Software: Practice and Experience*, 54(4):789–804.

Fitzgerald, B. and Stol, K. J. (2017a). Continuous software engineering: A roadmap and agenda. *Journal of Systems and Software*, 123:176–189.

Fitzgerald, B. and Stol, K. J. (2017b). Continuous software engineering: A roadmap and agenda. *J. Syst. Softw.*, 123:176–189.

Fitzgerald, B. and Stol, K.-J. (2017c). Data-driven software engineering: Analytics, metrics, and beyond. *Journal of Systems and Software*, 125:1–10.

Fu, Q., Lou, J.-G., Lin, Q., Ding, R., Zhang, D., and Xie, T. (2013). Contextual analysis of program logs for understanding system behaviors. *Proceedings of the 10th Working Conference on Mining Software Repositories*, pages 397–400.

FullScale (2023). The power of integrating data analysis in software development.

Ganter, B. and Wille, R. (1997). *Formal Concept Analysis: Mathematical Foundations*. Springer-Verlag New York, Inc., Secaucus, NJ, USA, 1st edition.

Gasparetti, F. (2017). Modeling user interests from web browsing activities. *Data mining and knowledge discovery*, 31(2):502–547.

Ghezzi, C., Pezzè, M., Sama, M., and Tamburrelli, G. (2014). Mining behavior models from user-intensive web applications. In *Proceedings of the 36th International Conference on Software Engineering*, pages 277–287. ACM.

Gorschek, T., Fricker, S. A., and Palm, K. (2017). A lightweight innovation process for software-intensive product development. *IEEE software*, 24(1):57–64.

Graf, S., Ives, C., Rahman, N., and Ferri, A. (2011). AAT: a tool for accessing and analysing students' behaviour data in learning systems. *1st International Conference on Learning Analytics and Knowledge*, pages 174–179.

Gulliksen, J. and et al. (2003). *User-Centered Design - In Search of Common Ground*. Human-Computer Interaction, Lawrence Erlbaum Associates.

Guo, Y. and Barnes, S. (2012). Enhancing e-learning systems through user behavior analysis. *Computers & Education*, 59(4):1377–1393.

Gupta, A., Mahajan, A., and Venkatesh, K. (2021). Clustering algorithms for analyzing user behavior in software applications. *Information and Software Technology*, 132:106488.

Gurp, J., Bosch, J., and Selic, B. (2009). Evolving software systems through incremental improvement. *IEEE Software*, 26(4):72–79.

Hackos, J. T. and Redish, J. C. (1998). *User and Task Analysis for Interface Design*. Wiley.

Hamiot, N. and Verlaine, D. (2024). Integrating automated feedback with social media analysis in software development. *Journal of Software Engineering*, 28(3):234–250.

Han, S., Dang, Y., Ge, S., Zhang, D., and Xie, T. (2012). Performance debugging in the large via mining millions of stack traces. In *2012 34th International Conference on Software Engineering (ICSE)*, pages 145–155. IEEE.

Harker, S. D. and Eason, K. D. (1983). Data integration in software engineering. *International Journal of Man-Machine Studies*, 18(2):233–248.

Helfert, M., Donnellan, B., and Ostrowski, L. (2012). The case for design science utility and quality-evaluation of design science artifact within the. *Systems, Signs & Actions*, 6(1):46–66.

Hess, J., Randall, D., Pipek, V., and Wulf, V. (2013). Involving users in the wild—Participatory product development in and with online communities. *Int. J. Hum. Comput. Stud.*, 71(5):570–589.

Hevner, A. and Chatterjee, S. (2010). *Design research in information systems: theory and practice*, volume 22. Springer Science & Business Media.

Hevner, A. R. (2007). A three cycle view of design science research. *Scandinavian journal of information systems*, 19(2):4.

Holmström Olsson, H. and Bosch, J. (2013). Towards Data-Driven Product Development: A Multiple Case Study on Post-deployment Data Usage in Software-Intensive Embedded Systems. In *Lect. Notes Bus. Inf. Process. B. Ser.*, volume 167, pages 152–164. Springer.

Huang, M.-H. and Rust, R. T. (2018). Data-driven service innovation: The role of big data and analytics in predicting user satisfaction. *Journal of Service Research*, 21(2):181–193.

Iivari, J., Hirschheim, R., and Klein, H. K. (2000). A dynamic framework for classifying information systems development methodologies and approaches. *Journal of management information systems*, 17(3):179–218.

Iivari, J. and Venable, J. (2009). Action research and design science research-seemingly similar but decisively dissimilar. In *ECIS*, pages 1642–1653.

Iqbal, M. and Ahmed, S. (2024). A/b testing combined with online forum feedback for software improvement. *Software Testing, Verification & Reliability*, 34(2):98–115.

Jakobi, T. and Stevens, G. (2013). Always beta: cooperative design in the smart home. In *Proc.*

2013 ACM Conf. Pervasive ubiquitous Comput. Adjun. Publ. - UbiComp '13 Adjun., pages 837–844, New York, New York, USA. ACM Press.

Jensen, C. and Scacchi, W. (2005). Data integration in software development environments. In *Proceedings of the 2005 International Symposium on Empirical Software Engineering*, pages 144–153. IEEE.

Jiang, L. and Naudé, P. (2011). User involvement and market orientation in the software industry. *Journal of Systems and Software*, 84(1):109–124.

Jin, H. and Lee, K. (2013). Analyzing implicit feedback for improving user satisfaction in software systems. In *Proceedings of the 22nd International Conference on World Wide Web*, pages 785–796.

Johnson, R. and Smith, J. (2019). Continuous user feedback in software development: A systematic review. *Information and Software Technology*, 105:82–94.

Joshi, N. (2024). The role of user experience (ux) in custom software development.

Kakar, A. (2020). Mitigating consensus bias in feature prioritization: Insights from social influence theory. *Journal of Software: Evolution and Process*, 32(1):e2234.

Kang, K. C., Cohen, S. G., Hess, J. A., Novak, W. E., and Peterson, A. S. (1990). Feature-oriented domain analysis (foda) feasibility study. Technical report, Carnegie-Mellon Univ Pittsburgh Pa Software Engineering Inst.

Kang, K. C., Kim, S., Lee, J., Kim, K., Shin, E., and Huh, M. (1998). Form: A feature-; oriented reuse method with domain-; specific reference architectures. *Annals of Software Engineering*, 5(1):143.

Kapoor, A. et al. (2021). Implicit feedback for improving user experience in software systems. *ACM Transactions on Computer-Human Interaction (TOCHI)*, 28(2):1–22.

Kaptelinin, V. and Nardi, B. A. (2006). *Acting with Technology: Activity Theory and Interaction Design*. MIT Press.

Karlsson, L. et al. (2007a). Prioritization of features in agile development using the kano model. *IEEE Software*, 24(3):28–35.

Karlsson, L., Thelin, T., and Regnell, B. (2007b). Requirements prioritization: An experiment on exhaustive pair-wise comparisons versus planning game partitioning. *Empirical Software Engineering*, 12(1):3–33.

Kesavulu, M., Bezbradica, M., and Helfert, M. (2017a). Generic refactoring methodology for cloud migration-position paper. In *International Conference on Cloud Computing and Services Science*, volume 2, pages 692–695. SCITEPRESS.

Kesavulu, M., Dang-Nguyen, D.-T., Bezbradica, M., and Helfert, M. (2018a). An Overview of User-level Usage Monitoring in Cloud Environment. In *The UK Academy for Information Systems (UKAIS).*

Kesavulu, M., Dang-Nguyen, D.-T., Bezbradica, M., and Helfert, M. (2018b). A usage analytics model for analysing user behaviour in ibm academic cloud.

Kesavulu, M., Helfert, M., and Bezbradica, M. (2017b). A Usage-based Data Extraction Framework for Cloud-Based Application - An Human-Computer Interaction approach. In *International Conference on Computer-Human Interaction Research and Applications (CHIRA)*, Madeira, Portugal.

Kim, D. and Park, S. (2020). Actionable insights from user behavior analysis in software development. *Empirical Software Engineering*, 25(3):1879–1905.

Kim, G., Humble, J., Debois, P., and Willis, J. (2016). *The DevOps Handbook: How to Create World-Class Agility, Reliability, & Security in Technology Organizations.* IT Revolution Press.

Kim, J. W. et al. (2011). Understanding and improving information architecture of large-scale analytics projects. In *Proceedings of the 20th ACM International Conference on Information and Knowledge Management*, pages 1597–1602.

Kitchenham, B. A. and Charters, S. (2004). Procedures for performing systematic reviews. *Keele University Technical Report*, 33(2004):1–26.

Kitchenham, B. A. et al. (2004). Empirical research methods in software engineering. *Journal of Empirical Software Engineering*, 9(3):311–324.

Kittur, A., Chi, E. H., and Suh, B. (2008). Crowdsourcing user studies with mechanical turk. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, pages 453–456.

Krusche, S. and Alperowitz, L. (2014). Introduction of continuous delivery in multi-customer project courses. In *Companion Proc. 36th Int. Conf. Softw. Eng. - ICSE Companion 2014*, pages 335–343, New York, New York, USA. ACM Press.

Krusche, S. and Bruegge, B. (2014). User Feedback in Mobile Development. *Proc. 2nd Int. Work. Mob. Dev. Lifecycle - MobileDeLi '14*, pages 25–26.

Kumar, A., Sung, M., Xu, J. J., and Wang, J. (2004). Data streaming algorithms for efficient and accurate estimation of flow size distribution. *SIGMETRICS Perform. Eval. Rev.*, 32(1):177–188.

Labib, C., Hasanein, E., and Hegazy, O. (2010). Early development of Graphical User Interface (GUI) in agile methodoligies. *INFOS2010 - 2010 7th Int. Conf. Informatics Syst.*

Lambert, V. A. and Lambert, C. E. (2012). Qualitative descriptive research: An acceptable design. *Pacific Rim international journal of nursing research*, 16(4):255–256.

Lee, C., Myrick, R., Asai, D., Coughlin, J. F., and Weck, O. L. D. E. (2013). Learning From a Design Experience : Continuous User Involvement in Development of Aging-in-Place Solution for Older Adults. *ICED13 19th Int. Conf. Eng. Des.*, (August):1–10.

Lee, J. et al. (2020). User-centric development driven by usage analytics. *Empirical Software Engineering*, 25(1):244–265.

Lee, K. and Taylor, R. (2021). Understanding the relationship between software features and user actions for better software design. *Journal of Software: Evolution and Process*, 33(4):e2358.

Lethbridge, T. C. et al. (2005). An investigation of the limitations of traditional feedback methods in software engineering education. In *Proceedings of the 27th international conference on Software engineering*, pages 191–200.

Lindgren, E. and Münch, J. (2016). Raising the odds of success: the current state of experimentation in product development. *Inf. Softw. Technol.*, 77:80–91.

Litoiu, M. et al. (2010). Real-time feedback in adaptive systems. In *2010 32nd ACM/IEEE International Conference on Software Engineering*, pages 455–464. IEEE.

Liu, X., Zhang, Z., and Li, M. (2020). Advancements in real-time analytics platforms for software development. *Journal of Systems and Software*, 162:110566.

Lo, D. and Nagappan, N. (2015). Towards understanding user behavior consistency for software feature improvement. In *Proceedings of the 37th International Conference on Software Engineering (ICSE)*, pages 314–324. IEEE.

Lo, D. and Zimmermann, T. (2013). Mining metrics to predict component failures. In *Proceedings of the 35th International Conference on Software Engineering (ICSE)*, pages 589–598. IEEE.

Lou, J., Lin, Q., Ding, R., and Fu, Q. (2013). Software analytics for incident management of online services: An experience report. *Automated Software . . .*, pages 475–485.

Maalej, W., Happel, H.-J., and Rashid, A. (2009). When users become collaborators: towards continuous and context-aware user input. In *Proceedings of the 24th ACM SIGPLAN conference companion on Object oriented programming systems languages and applications*, pages 981–990.

Maalej, W., Kurtanović, Z., Nabil, H., and Stanik, C. (2016). Bug report, feature request, or simply praise? on automatically classifying app reviews. *2016 IEEE 24th International Requirements Engineering Conference (RE)*, pages 283–293.

March, S. T. and Smith, G. F. (1995). Design and natural science research on information technology. *Decision support systems*, 15(4):251–266.

Martin, C. et al. (2018). Enhancing user satisfaction through user-centric development practices. *Journal of Systems and Software*, 137:289–299.

Masoudi, R. and Ghassemi, H. (2024). Enhanced clickstream analysis techniques for better understanding user context. *Journal of Web Engineering*, 23(1):54–70.

Meijer, E. and Kapoor, V. (2014). The Responsive Enterprise: Embracing the Hacker Way. *Queue*, 12(10):10–18.

Menzies, T. and Williams, L. (2011). Automated software engineering: An introduction. *Automated Software Engineering*, 18:105–107.

Menzies, T. and Zimmermann, T. (2013). Software analytics: so what? *IEEE Software*, 30(4):31–37.

Moløkken-Østvold, K. and Jørgensen, M. (2003). A review of surveys on software effort estimation. *Proceedings of the 2003 International Symposium on Empirical Software Engineering (ISESE)*, pages 223–230.

Montero, C. S. and Martínez-Ruiz, F. J. (2009). Event logging in usability testing: Advantages and disadvantages. In *Human-Computer Interaction. New Trends*, pages 196–199. Springer.

Montes, J., Sánchez, A., Memishi, B., Pérez, M. S., and Antoniu, G. (2013). Gmone: A complete approach to cloud monitoring. *Future Generation Computer Systems*, 29(8):2026–2040.

Mougouei, D. and Powers, D. M. W. (2017). Dependency-Aware Software Release Planning through Mining User Preferences.

Muller, J. and Stein, L. (2019). Behavioral sequence analysis for software feature prioritization. *IEEE Transactions on Software Engineering*, 45(6):1285–1297.

Müller, M., Hämäläinen, T., and Järvenpää, S. (2019). Sequence mining for analyzing user behavior patterns. *Journal of Systems and Software*, 150:75–88.

Murphy, G. and Weiss, D. (2013). Prioritizing software features using usage data and user feedback. *Empirical Software Engineering*, 18(5):877–904.

Muthitacharoen, A. M. and Saeed, K. A. (2009). Examining user involvement in continuous software development. *Commun. ACM*, 52(9):113.

Nardi, B. A. (1996). Activity theory and human-computer interaction. *Context and consciousness: Activity theory and human-computer interaction*, 436:7–16.

Newman, W. M. (2010). *Heuristic Evaluation Methods: A Review.* Academic Press.

Nguyen, T. and Wu, M. (2018). Behavioral analysis for user-centric feature prioritization in software development. *IEEE Transactions on Software Engineering*, 44(11):1045–1061.

Nielsen, J. (1994). *Usability Engineering.* Morgan Kaufmann.

Nielsen, J. (2012). Usability 101: Introduction to usability. *Nielsen Norman Group.*

Nielsen, J. and Molich, R. (1990). Heuristic evaluation of user interfaces. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, pages 249–256. ACM.

Norman, D. (2013). *The Design of Everyday Things: Revised and Expanded Edition.* Basic Books.

Norman, D. A. and Draper, S. W. (1986). *User Centered System Design; New Perspectives on Human-Computer Interaction.* CRC Press.

Nuseibeh, B. and Easterbrook, S. (2000). Requirements engineering: A roadmap. In *Proceedings of the Conference on The Future of Software Engineering*, pages 35–46. ACM.

Office, I. C. A. (2020). Leveraging ai to manage and prioritize feature requests. `https://www.ibm.com/blogs/research/2020/08/ai-feature-requests`. Accessed: 2024-05-26.

Ogonowski, C., Ley, B., Hess, J., Wan, L., and Wulf, V. (2013). Designing for the living room: long-term user involvement in a living lab. In *Proc. SIGCHI Conf. Hum. Factors Comput. Syst. - CHI '13*, number April, page 1539, New York, New York, USA. ACM Press.

Olsson, H. and Bosch, J. (2014a). From opinions to data-driven software r&d: A multi-case study on how to close the 'open loop' problem. *Journal of Systems and Software*, 95:122–134.

Olsson, H. and Bosch, J. (2014b). A systematic approach to feature prioritization in agile software development. In *Proceedings of the 2014 International Conference on Agile Software Development*, pages 47–56.

Olsson, H. H., Alahyari, H., and Bosch, J. (2012). Climbing the" stairway to heaven"–a mulitiple-case study exploring barriers in the transition from agile development towards continuous deployment of software. In *Software Engineering and Advanced Applications (SEAA)*, pages 392–399. IEEE.

Olsson, H. H. and Bosch, J. (2014c). From Opinions to Data-Driven Software R&D: A Multi-case Study on How to Close the 'Open Loop' Problem. In *2014 40th EUROMICRO Conf. Softw. Eng. Adv. Appl.*, pages 9–16. IEEE.

Olsson, H. H. and Bosch, J. (2015). Towards Continuous Customer Validation: A Conceptual Model for Combining Qualitative Customer Feedback with Quantitative Customer Observation. In *Lect. Notes Bus. Inf. Process.*, volume 210, pages 154–166.

Pachidi, S., Spruit, M., and Van De Weerd, I. (2014). Understanding users' behavior with software operation data mining. *Computers in Human Behavior*, 30(January):583–594.

Pagano, D. and Bruegge, B. (2013). User involvement in software evolution practice: A case study. In *2013 35th Int. Conf. Softw. Eng.*, pages 953–962. IEEE.

Pagano, D. and Maalej, W. (2013). User involvement in software evolution practice: A case study. In *2013 35th International Conference on Software Engineering (ICSE)*, pages 953–962. IEEE.

Patton, J. (2014). *User Story Mapping: Discover the Whole Story, Build the Right Product.* O'Reilly Media, Inc.

Pazzani, M. J. and Billsus, D. (2007). Content-based recommendation systems. *The Adaptive Web*, pages 325–341.

Peffers, K., Tuunanen, T., Rothenberger, M. A., and Chatterjee, S. (2007). A design science research methodology for information systems research. *Journal of management information systems*, 24(3):45–77.

Pereira, J. and Silveira, M. (2018). Integrating machine learning with traditional metrics for feature prioritization in software development. *Journal of Systems and Software*, 145:85–101.

Petruch, K., Tamm, G., and Stantchev, V. (2012). Deriving in-depth knowledge from it-performance data simulations. *International Journal of Knowledge Society Research (IJKSR)*, 3(2):13–29.

Piccoli, G., Rodriguez, J., Palese, B., and Bartosiak, M. L. (2020). Feedback at scale: designing for accurate and timely practical digital skills evaluation. *European Journal of Information Systems*, 29(2):114–133.

Poppendieck, M. and Cusumano, M. A. (2012). Lean Software Development: A Tutorial. *IEEE Softw.*, 29(5):26–32.

Preece, J., Rogers, Y., and Sharp, H. (2015). *Interaction Design: Beyond Human-Computer Interaction.* John Wiley & Sons.

ProductHQ (2023). Leveraging user feedback for software growth.

Puthal, D., Sahoo, B. P., Mishra, S., and Swain, S. (2015). Cloud computing features, issues, and challenges: a big picture. In *2015 International Conference on Computational Intelligence and Networks*, pages 116–123. IEEE.

Qu, L., Wang, Y., and Orgun, M. a. (2013). Cloud Service Selection Based on the Aggregation of User Feedback and Quantitative Performance Assessment. *IEEE Int. Conf. Serv. Comput.*, pages 152–159.

Racheva, Z., Daneva, M., and Buglione, L. (2010). A comparative study of feature prioritization methods in agile software development. In *2010 36th EUROMICRO Conference on Software Engineering and Advanced Applications*, pages 367–374. IEEE.

Rahimi, A. and Jahanian, F. (2023). Enhancing in-app feedback with social media insights. *IEEE Transactions on Software Engineering*, 49(1):120–135.

Rangel, C. and Martinez, L. (2024). Combining prototypes with user interviews for early design validation. *Design Studies*, 79:101070.

Revelle, M., Dit, B., and Poshyvanyk, D. (2010). Using data fusion and web mining to support feature location in software. In *2010 IEEE 18th International Conference on Program Comprehension*, pages 14–23. IEEE.

Reyes, J. A. (2015). The Skinny on Big Data in Education. *TechTrends*, 59(2):75–80.

Riebisch, M. (2003). Towards a more precise definition of feature models. *Modelling Variability for Object-Oriented Product Lines*, pages 64–76.

Ries, E. (2011). *The Lean Startup - How Today's Entrepreneurs Use Continuous Innovation to Create Radically Successful Businesses.* Crown books.

Rigby, D. K., Sutherland, J., and Takeuchi, H. (2016a). Embracing agile. *Harvard Business Review*, 94(5):40–50.

Rigby, P. C. et al. (2016b). Validation of software engineering practices: Case studies and lessons learned. *Harvard Business Review.*

Rindos, A., Vouk, M., and Jararweh, Y. (2014). The virtual computing lab (vcl): an open source cloud computing solution designed specifically for education and research. *International Journal of Service Science, Management, Engineering, and Technology (IJSSMET)*, 5(2):51–63.

Rissanen, O. and Munch, J. (2015). Continuous Experimentation in the B2B Domain: A

Case Study. In *2015 IEEE/ACM 2nd International Workshop on Rapid Continuous Software Engineering*, pages 12–18. IEEE.

Rodden, K., Hutchinson, H., and Fu, X. (2010a). Measuring the impact of real-time feedback in software development. *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, pages 2203–2212.

Rodden, T. et al. (2010b). The importance of real-time feedback in adaptive software systems. In *Proceedings of the 5th International Workshop on Software Engineering for Adaptive and Self-Managing Systems*, pages 74–83.

Rouse, M. (2018). Ibm's data governance strategy. *IBM Systems Journal*, 57(3):17–29.

Ruhe, G. and Greer, D. (2002). The art and science of software release planning. *IEEE software*, 19(4):47–53.

Sabbagh, Y. and Khalil, R. (2024). Integrating bug tracking with usage data and online forums. *Empirical Software Engineering*, 29(3):345–360.

Schellong, D., Kemper, J., and Brettel, M. (2017). Generating Consumer Insights from Big Data Clickstream Information and the Link with Transaction-Related Shopping Behavior. *Proceedings of the 25th European Conference on Information Systems (ECIS)*, 2017:1–15.

Schneider, K., Meyer, S., Peters, M., Schliephacke, F., Mörschbach, J., and Aguirre, L. (2010). Feedback in Context: Supporting the Evolution of IT-Ecosystems. In Ali Babar, M., Vierimaa, M., and Oivo, M., editors, *Prod. Softw. Process Improv.*, pages 191–205, Berlin, Heidelberg. Springer Berlin Heidelberg.

Schwaber, K. and Sutherland, J. (2017). *The Definitive Guide to Scrum: The Rules of the Game*. Retrospectiva del Sprint de Nexus. Scrum.org.

Science, T. and Organization, I. (2023). Scrum: A systematic literature review. *International Journal of Advanced Computer Science and Applications*, 14(4):200–210.

Shams, I. and Hashemi, R. (2018). Anomaly detection in user access patterns for enhancing application security. *Journal of Network and Computer Applications*, 109:102–113.

Shams, I., Hashemi, R., and Safari, M. (2020). Enhancing feature prioritization with user engagement and feature abandonment metrics. *Journal of Systems and Software*, 165:110573.

Shams, T. and Whittaker, I. (2020). Usage analytics for user-centered design and decision-making in software development. *Journal of Systems and Software*, page 110763.

Sjoberg, D. I. et al. (2005). The importance of empirical validation in software engineering research. *Journal of Empirical Software Engineering*, 10(1):117–137.

Sjøberg, D. I. K., Dyba, T., and Jørgensen, M. (2005). The future of empirical methods in software engineering research. *Proceedings of the 2005 International Symposium on Empirical Software Engineering (ISESE)*, pages 148–157.

Smith, L. and Brown, A. (2019). Mapping user actions to software features: Insights for feature prioritization. *Software Quality Journal*, 27(2):289–309.

Soltani, S. and Moussavi, Z. (2021). Refining feature prioritization through integrated usage metrics and user feedback. *Empirical Software Engineering*, 26(3):3056–3080.

SoluteLabs (2023). Top agile trends to watch out for in 2023 - adapting to change.

Sun, X. (2016). Virtual Machine Optimizations Using Markov Chain Data Analytics in Heterogeneous Cloud Computing. In *International Conference on Smart Cloud*, pages 248–253.

Sun, Y. et al. (2019). Leveraging implicit feedback to enhance user experience in software applications. *IEEE Transactions on Software Engineering*, 45(4):333–345.

Sweller, J., Ayres, P., and Kalyuga, S. (2011). *Cognitive Load Theory*. Springer.

Tang, A. and Zhang, H. (2011). Feature usage analysis for improving software systems. *Journal of Systems and Software*, 84(3):481–497.

Tarmuji, F. and Abdullah, Z. (2024). Real-time dashboards with communication tools for improved feedback interpretation. *Journal of Systems and Software*, 177:110938.

Tizard, J., Rietz, T., Liu, X., and Blincoe, K. (2022). Voice of the users: an extended study of software feedback engagement. *Requirements Engineering*, 27(3):293–315.

Ulrich, K. T. and Eppinger, S. D. (2012). *Product Design and Development*. McGraw-Hill.

Vaismoradi, M., Turunen, H., and Bondas, T. (2013). Content analysis and thematic analysis: Implications for conducting a qualitative descriptive study. *Nursing & Health Sciences*, 15(3):398–405.

Vozniuk, A., Holzer, A., and Gillet, D. (2016). Increasing the efficiency of feedback collection in software development. In *2016 IEEE 16th International Conference on Advanced Learning Technologies (ICALT)*, pages 368–370. IEEE.

Waller, M. A. and Fawcett, S. E. (2020). Data governance and quality in software development: Best practices and case studies. *Journal of Business Logistics*, 41(2):145–160.

Wang, C., Schwan, K., Talwar, V., Eisenhauer, G., Hu, L., and Wolf, M. (2011). A flexible architecture integrating monitoring and analytics for managing large-scale data centers. *ACM International Conference on Autonomic Computing*, page 141.

Wang, G., Zhang, X., Tang, S., Zheng, H., and Zhao, B. Y. (2016). Unsupervised clickstream clustering for user behavior analysis. In *Proceedings of the 2016 CHI Conference on Human Factors in Computing Systems*, pages 225–236. ACM.

Wang, Z., Zhu, X., Adeli, E., Zhu, Y., Nie, F., Munsell, B., and Wu, G. (2017). Multi-modal classification of neurodegenerative disease by progressive graph-based transductive learning. *Medical Image Analysis*, 39:218–230.

Webster, J. and Watson, R. T. (2002). Analyzing the past to prepare for the future: Writing a literature review. *MIS quarterly*, pages xiii–xxiii.

Wei, L., Liu, J., Wang, T., and Yu, P. S. (2019). Hybrid anomaly detection for large-scale systems. *IBM Journal of Research and Development*, 63(2):69–83.

Wilcox, E., Nusser, S., Schoudt, J., Cerruti, J., and Badenes, H. (2010). Agile Development Meets Strategic Design in the Enterprise. In *Agil. Process. Softw. Eng. Extrem. Program.*, number July 2015, pages 208–212. Springer Berlin Heidelberg, Berlin, Heidelberg.

Witten, I. H., Frank, E., Hall, M. A., and Pal, C. J. (2016). *Data mining: Practical machine learning tools and techniques*. Morgan Kaufmann.

Xia, F., Liu, J., and Liu, J. (2019). Improving cost-value analysis using advanced machine learning techniques. *Information and Software Technology*, 110:82–94.

Xu, G., Zhang, Y., and Yi, X. (2008). Modelling user behaviour for Web recommendation using LDA model. *Proceedings - 2008 IEEE/WIC/ACM International Conference on Web Intelligence and Intelligent Agent Technology - Workshops, WI-IAT Workshops 2008*, (September 2015):529–532.

Yaman, S. G., Sauvola, T., Riungu-Kalliosaari, L., Hokkanen, L., Kuvaja, P., Oivo, M., and Männistö, T. (2016). Customer Involvement in Continuous Deployment: A Systematic Literature Review. In *Lect. Notes Comput. Sci. (including Subser. Lect. Notes Artif. Intell. Lect. Notes Bioinformatics)*, volume 9619, pages 249–265.

Yang, J., Wang, H., Lv, Z., Wei, W., Song, H., Erol-Kantarci, M., Kantarci, B., and He, S. (2017). Multimedia recommendation and transmission system based on cloud platform. *Future Generation Computer Systems*, 70:94–103.

Zave, P. (2003). An experiment in feature engineering. In *Programming methodology*, pages 353–377. Springer.

Zhang, D., Dang, Y., Lou, J.-G., Han, S., Zhang, H., and Xie, T. (2011). Software analytics as a learning case in practice. *Proceedings of the International Workshop on Machine Learning Technologies in Software Engineering - MALETS '11*, pages 55–58.

Zhang, H., Liu, Z., and Wang, X. (2018). Cost-value optimization of software product line engineering. *Journal of Systems and Software*, 136:112–126.

Zhang, J. and Wang, X. (2018). Learning from usage data to improve software quality. *Empirical Software Engineering*, 23(2):640–666.

Zhou, M. and Chen, L. (2017). Machine learning techniques for user behavior analysis in software development. *IEEE Transactions on Knowledge and Data Engineering*, 29(5):1101–1114.

Zimmermann, T. and Nagappan, N. (2010). Towards data-driven feature prioritization in software development. *IEEE Software*, 27(6):26–33.

Zimmermann, T. and Nagappan, N. (2019). Consistency metrics for evaluating user engagement in evolving software systems. *Journal of Systems and Software*, 150:68–84.

Çökeli, H. (2024). A guide to product usage analytics and measuring software usage.