ISSN: 2755-0214

Journal of Economics & Management Research

SCIENTIFIC Research and Community

Review Article

Open Access

Mitigating Ambiguity in Requirements for Enhanced Precision in Payment Application Development within the Payments Industry

Sridhar Mooghala

Senior Advisor, Fiserv, United States

ABSTRACT

The adoption of new technologies and the growing complexity of payment transactions promote a rapid transformation of the payments sector, which is currently in the midst of this transformation. Consequently, the demand for increased precision in developing payment applications is increasing. Despite this, there is still a significant amount of uncertainty in the standards, which can result in misunderstandings, delays, and increased costs. In this research, we investigate the factors contributing to ambiguity in payment application requirements and suggest an approach for limiting the associated risks.

***Corresponding authors**

Sridhar Mooghala, Senior Advisor, Fiserv, United States.

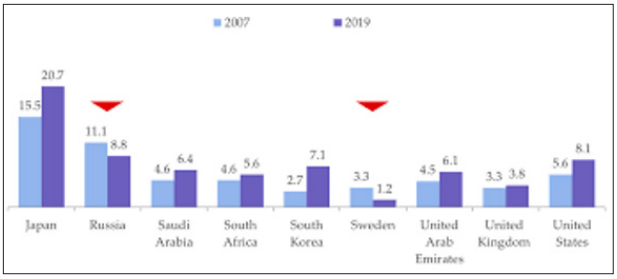
Received: November 08, 2022; **Accepted:** November 15, 2022; **Published:** November 22, 2022

Keywords: Ambiguity, Mobile, Mobile Payment Apps

Introduction

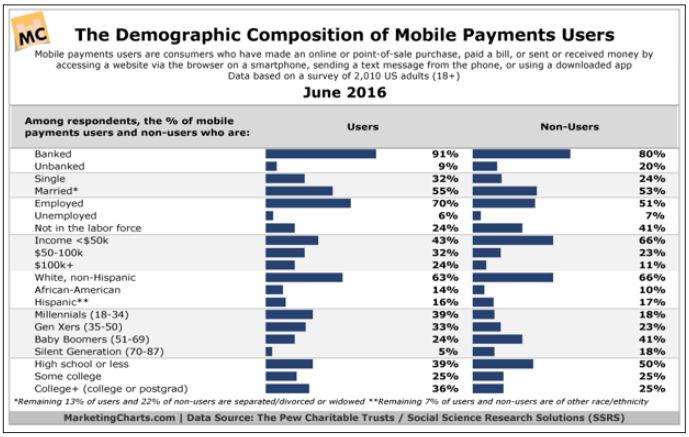
The payments business is at the cutting edge of how technology is changing things, and changes are happening quickly. Because of how quickly things change, the accuracy of making payment apps becomes very important. Because financial activities are getting more complicated, software developers must be cautious when making programs to ensure they are efficient, safe, and easy for users. But one big problem stopping this progress is that standards during development aren't always clear. This study looks into the critical problem of ambiguity and how it affects accuracy, trying to find good ways to deal with these problems. The study aims to provide insights that explain current problems and suggest workable solutions by examining the details of requirement specs in the payments industry.

To reach these goals, we need to learn more about the different kinds of uncertainty, how they affect the development of payment apps, and how to develop workable solutions based on established standards and best practices. This study aims to give helpful advice to professionals in the field who are trying to make payment apps more accurate and quicker by handling the complicated process of making them.



The present and projected non-cash transactions by region for the period of 2017- 2021, measured in billions of transactions, are represented by the symbols (A) for actuals and (F) for anticipated.

Non-cash transactions encompass several forms of electronic payments, including debit and credit card transactions, bank transfers, and the use of cheques.



Literature Review

Within the domain of payment application development, creating complex software systems that manage sensitive financial information and enable secure monetary transactions necessitates careful attention to a wide range of detailed specifications. Functional requirements encompass various critical aspects, including payment processing, account management, fraud prevention, regulatory compliance, reporting, and analytics [1]. These requirements are designed to ensure secure transaction initiation, processing, and completion, effective user account management, robust security measures, adherence to regulatory standards, and the ability to conduct insightful data analysis. Non-functional criteria encompass several aspects such as performance, security, usability, dependability, and scalability, which collectively contribute to the application's capacity to effectively manage transactions, safeguard sensitive data, offer user-friendly interfaces, ensure constant performance, and accommodate growing demands.

Efficiently eliciting and documenting requirements includes systematically gathering and analyzing information from stakeholders. This is achieved through various methodologies such as use cases, user stories, and data flow diagrams. The iterative process of continual evaluation and refinement guarantees that payment applications align with increasing corporate objectives and user expectations, promoting flexibility in response to changing requirements and emerging technology. In brief, it is imperative to thoroughly comprehend functional and non-functional requirements, alongside proficient elicitation and ongoing refinement processes, to guarantee payment apps' security, correctness, reliability, and usefulness.

Numerous research offer varied viewpoints on the difficulties presented by ambiguity in requirements and suggested approaches to mitigate these obstacles. The study [2] analysis highlighted the substantial influence of unclear requirements on software development projects' financial and qualitative aspects. According to research, agile software development approaches have been found to have a higher level of effectiveness in addressing ambiguity when compared to conventional methodology. In continuation of this, the research conducted by brought attention to the significant role played by requirements ambiguity in the occurrence of software faults, underscoring the utmost significance of achieving clarity during the initial phases of development [3-4]. Study conducted by focused on safety-critical systems [5]. This study's findings demonstrated that formal approaches can significantly mitigate ambiguity in requirements, especially when safety is of utmost importance. The study conducted by provided valuable insights into the dynamic nature of requirements engineering, focusing on the necessity for further investigation into requirements engineering methods designed explicitly for cloud-based software [6]. Collectively, these works provide intricate perspectives and diverse approaches, establishing a comprehensive basis for comprehending and resolving uncertainty within software development.

Ambiguity in Requirements in Payment Application Development

• Definition and Types of Ambiguity

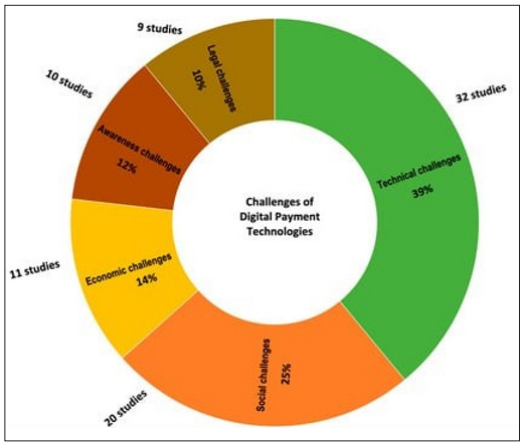
Regarding standards for developing payment apps, ambiguity covers many areas that need a deep understanding. One common type is syntactic ambiguity, which shows up as unclear or poorly organized standards that could cause development teams to get them wrong. Semantic ambiguity makes it unclear how something is supposed to work, leading to different meanings that are not the same as the original plan. When standards don't give a clear situational context, it can be hard to figure out how they apply or are relevant in certain situations. This is called contextual ambiguity. Recognizing and classifying these kinds of uncertainty are critical first steps in dealing with the problems they cause throughout the software development life cycle.

	Type of Ambiguity	Subtype
Language Ambiguity	Lexical Ambiguity	Homonym Ambiguity Polysemy Ambiguity
	Syntactic Ambiguity	Analytical Ambiguity, Attachment Ambiguity, Coordination Ambiguity, Elliptical Ambiguity
	Semantic Ambiguity	Scope Ambiguity
	Pragmatic Ambiguity	Referential Ambiguity, Deictic Ambiguity
	Vagueness, Language Error, Generality	
RE-Specific Ambiguity	Conceptual Translational Ambiguity	
	Requirements Document Ambiguity	
	Application Domain Ambiguity	
	System Domain Ambiguity	
	Development Domain Ambiguity	

• The Challenge of Ambiguity in Requirements

Despite the increasing demand for accuracy, ambiguity in requirements continues to be a substantial obstacle in the development of payment applications. This issue presents complex issues that have far- reaching implications throughout the development process. The presence of ambiguity promotes a range of interpretations of identical requirements among various stakeholders, resulting in confusion, ineffective communication, and the possibility of disputes arising. Misunderstandings can hinder the synchronization of development goals, user expectations, and corporate objectives. In addition, ambiguous requirements are a significant challenge for developers since they may have difficulties translating vague directives into actionable implementations. Insufficient clarity and precision impede progress, leading to delays in the development timetable and potentially compromising the timely deployment of payment apps.

The financial domain experiences the reverberating consequences of ambiguity, as indistinct specifications contribute to the need for more work and an expansion of project scope. The unforeseen modifications result in increased expenses, which impact the allocation of funds and may potentially burden the financial resources designated for developing the payment application. Mitigating issues arising from confusing requirements in payment application development necessitates a comprehensive approach that includes good communication, stakeholder participation, and rigorous documentation.

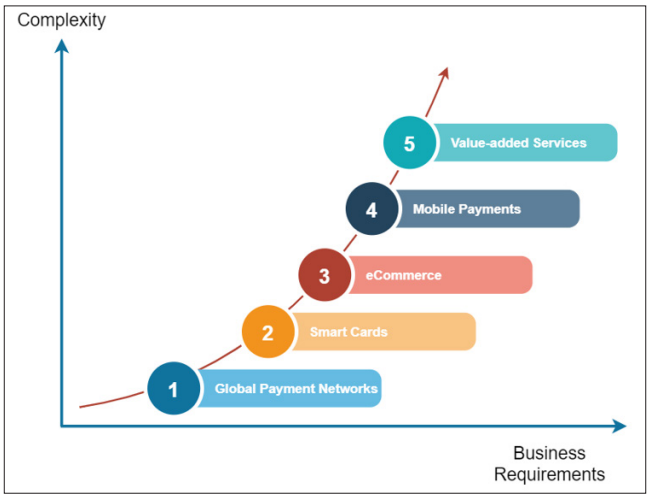


Exploring the Sources of Ambiguity

The complexities arising from uncertainty in payment application requirements stem from various origins, demanding a comprehensive investigation.

The occurrence of ambiguity frequently stems from requirements articulated using ambiguous or inaccurate terminology, creating opportunities for misinterpretations and diverse interpretations. Clear, succinct, and exact language is crucial in promoting a common understanding among all parties engaged in the development process. The absence of specified requirements introduces uncertainty, necessitating developers to make assumptions to address the gaps. Precise and comprehensive specifications play a crucial role in providing developers with guidance for precisely implementing functionalities, hence minimizing the need for speculative decision-making.

Ambiguity may also arise as a result of contradictory requirements, leading to a state of perplexity and doubt among software engineers. Resisting contradictory instructions is essential in optimizing the development process and maintaining the payment application's consistency. In brief, resolving ambiguity necessitates a thorough comprehension of the sources above, emphasizing clarity, precision, and consensus among relevant parties to optimize the efficacy of payment application development.



Impact of Ambiguity on the Development of Payment Applications

Uncertainty has effects that go far beyond the development phase and affect many parts of deploying and maintaining payment applications. As unclear requirements stay in place during execution, they cause problems that slow the application's operation. The mistakes made during development hurt the payment app's accuracy and made it less valuable overall. When it comes to financial transactions, accuracy is critical. Even the slightest deviation from the planned functions can have bad results, like sending money to the wrong account or failing transactions. These mistakes not only make people less likely to trust the service, but they could also cost both users and service companies money.

In addition, unclear standards have terrible effects on security. The payment app could be vulnerable if you add features that weren't meant to be there or forget to take necessary security steps [7]. In a time when online threats are constantly changing, any security breach caused by unclear rules could be a way for bad things to happen. Fraudulent transactions, unauthorized access, and data breaches are now more likely to happen. These problems affect the payment app's finances and users' privacy and trust. Security holes cause more damage than just immediate financial losses. They can also hurt the payment service provider's image and even get them in trouble with the law.

The iterative nature of software development worsens the problems caused by uncertainty after the development phase. As development teams deal with the need for more information, they have to keep doing work repeatedly. This rework adds to the time it takes to create and costs more, which cuts into the project's overall budget. The financial effects are twofold: the direct costs of the extra time and work needed for revisions and the possible secondary costs that come from taking longer to get the product to market. In the competitive world of payment services, getting apps and changes out there on time is essential. Uncertainty can cause delays that cause missed chances, giving competitors an edge in developing new and safe financial solutions.

Examples or Case Studies that Show What Happens When Requirements aren't Clear

This is because real-life case studies show what happens when there are unclear requirements in the payment business. Take, for example, a payment processing module that isn't clear about how to handle edge situations like transactions that aren't finished

or network failures. Because it's unclear, error-handling systems might not work correctly, making the payment app less reliable and more likely to be used fraudulently. If semantic ambiguity exists in the requirements for managing user accounts, security steps can be interpreted differently. When access control requirements aren't straightforward, user authentication methods that don't work well can be broken into, allowing hackers to get into user accounts and possibly causing data breaches. Also, regulatory compliance requirements that aren't clear in certain situations, like those related to the PCI DSS, could lead to noncompliance, which could mean fines from the government and damage to the app's image. It is essential to have clear, context- based advice on encryption standards. If this isn't done, people may use old or unsafe methods, which could compromise the safety of sensitive financial data. These examples show how important it is to have precise requirements right away to avoid the adverse effects of fuzzy requirements on the development of payment apps.

Strategies for Making Payment Application Requirements Less Vague

Businesses should use a complete plan that includes preventative and proactive steps to deal with problems when payment application requirements aren't precise. The following vital methods play a significant role:

- **Use Clear and Concise Language:** To effectively communicate objectives, you must use clear and concise language so that many stakeholders can understand it. Making sure that everyone can understand the message means staying away from expert jargon and vague phrases. Also, breaking complicated requirements into doable pieces makes things even more precise, making it easier for everyone to understand and lowering the chance of confusion.
- **Make Requirements More Specific:** To get accurate requirements, you have to be willing to give thorough descriptions that leave no room for confusion. This means ensuring that the parameters for input and output, how to handle errors and performance standards are clearly defined. Including examples and use cases is a great way to show how something works, which helps developers understand and meet specific standards.
- **Fix Requirements that are at Odds:** Finding and resolving

conflicting requirements as soon as possible is critical to keeping the development process running smoothly. Companies can ensure that the development team works toward strategic goals by putting different instructions in order of importance based on the overall business goals. When conflicts happen, trade- off analysis is a systematic way to find the best answers. This leads to a set of requirements that are consistent and logical.

- **Control What Stakeholders Expect:** A critical part of good requirement management is setting reasonable goals. It is essential to be clear about what it can't do to ensure everyone has an accurate idea of what the payment application can do. Actively involving stakeholders in the development process is vital to controlling expectations and giving people chances to clarify, make changes, and ensure the project fits the requirements.
- **Encourage participation from stakeholders:** Getting people involved in gathering requirements and ensuring is a key part of collaborative participation. Actively seeking feedback and facilitating talks through workshops, prototypes, and user testing improves clarity and aligns the development team with what end users and business representatives want and need.
- **Use formal requirement management:** Setting up a central store with version control is essential to make sure that everyone who needs to see the standards has access to the most up-to-date version and keeps a record of the past so that everyone can be held accountable. Setting up a clear and organized way for reviewing and agreeing on requirements is another way to make requirement management more organized and well-documented.
- **Make use of modeling and *Visual Aids*:** Using diagrams, flowcharts, wireframes, and other visual aids can help you see the needs. Graphical images of complicated ideas make them more concrete and easier to understand, which helps everyone involved understand how the payment application works. Using modeling tools improves this process even more by making representations that are more thorough and interactive [8]. This helps stakeholders understand better and leads to a more accurate Order 6618407 Mitigating Ambiguity in Requirements for Enhanced Precision in Payment Application Development within the Payments Industry interpretation of requirements.
- **It is Imperative to do Comprehensive Testing and Validation Procedures:** Implementing strict testing methods is necessary to ensure the standards are robust. Written test plans should be used as part of a complete validation strategy, and testing should continue throughout. A method like this ensures complete coverage, early detection of possible mistakes, and building a solid base for creating a top-notch payment application.

Conclusion

In summary, reducing ambiguity in payment application requirements reveals significant findings. The key findings of this study highlight the significance of utilizing precise and unambiguous language, increasing specificity in communication, and promoting the active participation of stakeholders to mitigate ambiguity. The research holds importance due to its ability to optimize payment application development, reduce delays and cost overruns, and improve stakeholder satisfaction. It is advisable for professionals working in the payments sector to give priority to effective communication, actively engage stakeholders, and employ visual aids to enhance clarity. The implementation of formal requirement management techniques and the execution of comprehensive testing procedures contribute to enhancing the resilience of payment application development. Implementing these ideas has the potential to enhance the efficiency and dependability of payment systems significantly.

References

1. Feyen E, Frost J, Gambacorta L, Natarajan H, Saal M (2021) Fintech and the digital transformation of financial services: implications for market structure and public policy. https://www.bis.org/publ/bppdf/bispap117.pdf.
2. Scholarworks@utep S, Rahad K (2021) Demystifying the Practices of Software Design and The Impact Demystifying the Practices of Software Design and The Impact on Codebase Quality and Sustainability on Codebase Quality and Sustainability. https://scholarworks.utep.edu/cgi/viewcontent.cgi?article=4442&context=open_etd.
3. Kakar A (2020) A Theory of Agile Software Development. SAIS 2020 Proceedings https://aisel.aisnet.org/sais2020/32/.
4. Lopez Tamara (2016) Error Detection and Recovery in Software Development. PhD thesis The Open University. https://doi.org/10.21954/ou.ro.0000bd61.
5. Kenner A, May R, Krüger J, Saake G, Leich T (2021) Safety, security, and configurable software systems. https://doi.org/10.1145/3461001.3471147.
6. Lee CH, Liu CL, Trappey AJC, Mo JPT, Desouza KC (2021) Understanding digital transformation in advanced manufacturing and engineering: A bibliometric analysis, topic modeling and research trend discovery. Advanced Engineering Informatics 50: 101428.
7. Weichbroth P, Łysik L (2020) Mobile Security: Threats and Best Practices. Mobile Information Systems 2020: 1-15.
8. Syahrina A, Kusumasari TF (2020) Designing User Experience and User Interface of a B2B Textile e-Commerce using Five Planes Framework. International Journal of Innovation in Enterprise System 4: 44-55.