Al-Farouk journal of sciences
Volume (2), Issue (3) (JUNE, 2026)
ISSN: 3135-2359

مجلة الفاروق للعلوم
المجلد (2) العدد 3 يونيو 2026
ردمد: 3135-2359

The gap between software developers and users in SRS

Aesam ammar al zmzam[1],hazem Ashor Amran Abolholl[2],Moamer Ehtash[3], and Taihirt Alhashan[4]

[1]Higher Institute Of Engineering techniques – Tripoli maz_12321@hotmail.com
[2]Golloge of Engineering Technologies –Yefren abolholl.hazem@gmail.com
[3]Golloge of Engineering Technologies –Yefren jone.doe@example.com
[4]Golloge of Engineering Technologies –Yefren tgrada@yahoo.com

تاريخ الاستلام: 2026/04/21 تاريخ المراجعة 2026 /05/23 تاريخ القبول:2026/06/07- تاريخ النشر: 2026 /06/23

Abstract

The gap between software developers and users remains a persistent challenge despite numerous attempts to address it. This study explores the various dimensions of this gap, including differences in mindset, culture, and communication. It categorizes the gap into nine types, ranging from perspective and ownership gaps to communication and credibility gaps. The study emphasizes the cultural divide as a significant contributor to this problem and highlights the power imbalances that can further hinder effective communication.

Efforts to narrow this divide frequently revolve around the promotion of active user involvement in the software development process. However, the effectiveness of these initiatives is contingent upon meticulous attention to a complex interplay of technical and political factors. The study also explores innovative approaches, such as video analysis and clear representations, to enhance understanding between stakeholders.

This paper underscores the ongoing challenges in communication and understanding between software developers and users and highlights the need for improved communication, cultural sensitivity, and user participation to address these issues successfully.

Introduction

The field of software development stands at the forefront of technological innovation, driving advancements that shape our digital landscape. However, amidst this rapid evolution, an enduring challenge persists - the profound gap that often exists between software developers and end-users (Grudin, 1991). This gap, characterized by disparities in mindset, culture, and communication, has long been recognized as a formidable barrier to the creation of software systems that align seamlessly with user needs and expectations.

Despite concerted efforts to bridge this divide, the gap remains a pervasive and complex issue, manifesting in a increasing of ways. It manifests during requirements gathering, where miscommunications often result in systems that fall short of meeting user needs. This fundamental issue underscores the critical importance of effective communication, a factor that has been explored and debated extensively in both practitioner and academic literature.

This paper delves into the depths of this complex challenge, seeking to dissect its various dimensions and shed light on the nuanced factors that contribute to its persistence. Through a thorough survey of existing literature and empirical insights, we categorize the gap between software developers and users into nine distinct types. These include the perspective gap,

ownership gap, cultural gap, foresight gap, communication gap, expectation gap, credibility gap, appreciation gap, and relationship gap. Each of these categories illuminates a facet of the discord that hampers productive collaboration between developers and users.

The cultural divide emerges as a recurring theme in this discourse, highlighting the profound influence of cultural disparities in exacerbating communication breakdowns. These disparities extend to power imbalances, further complicating efforts to foster effective dialogue and mutual understanding.

In light of these challenges, efforts to bridge the gap have centered on the promotion of user participation in the software development process. While this approach holds promise, its success hinges on a nuanced understanding of both technical and political factors. Consequently, we delve into the intricacies of user participation, emphasizing the need for thoughtful, two-way communication, negotiation, and mutual learning.

Furthermore, this paper explores innovative approaches that offer potential solutions to enhance understanding among stakeholders. Techniques such as video analysis and clear representations hold promise in mitigating communication challenges and facilitating a more cohesive understanding of software requirements.

In summation, this paper embarks on a comprehensive journey to unravel the intricacies of the persistent gap between software developers and users. By shedding light on the multifaceted nature of this challenge and proposing strategies to address it, we endeavor to contribute to the ongoing dialogue aimed at fostering improved collaboration and more successful software development outcomes.

Literature Review

The gap between software developers and users has long been recognized, addressed, and several solutions have been presented to reduce this gap. However, in spite of these efforts the problem seems as prevalent as ever.

A survey of both practitioner and academic literature has noted that there is a major difference between software developers and the users, especially in their mindset. The difference has resulted in a large number of problems. The major rift is when developers and users are not able to communicate productively during gathering requirements documents. This miscommunication leads to systems that do not meet users' needs. The major part of this gap is the misunderstanding or the poor communication between these parties. As Naomi Karten, emphasizes in her book, *Managing Expectations,* it is not only important to be an "information-gathering specialist", but also an "information-gathering skeptic", [p. 77]. This role is necessary because customers, like developers, sometimes do not convey what they mean due to language and cultural barriers, not that they intent to do it deliberately.

Goransson et al (1999) remarks that there is a need for a person who is able to specify the goals for usability, conduct user analysis and task analysis and then direct the design team. The next point they stress is the existence of direct communication and cooperation between users and developers.

Mann (2002) has studied on the IT-User gap and given its detailed characteristics. He has made an overview of the various types of gap both in academic and practitioner literature and categorized the gap into nine types as follows:

1. **The Perspective Gap**: when one stakeholder group incompletely understands the point of view of another group. Developers may be not able to see the value attached to the systems in business and users may overlook that technology is just a tool, and not a goal. (Mann, 2002).

2. **Ownership Gap**: when developers feel they are the owners of the infrastructure whereas users feel they are the masters of the business processes. This sense of ownership will end in distinction of thoughts and eventually to conflict. (Mann, 2002).

3. Cultural Gap: developers and users have different traits, values, working behaviours, and/or priorities. We can see developers as more introverted, analytical who use rational persuasion to influence others. On the other hand, users, mostly the business users, are more extroverted, intuitive and use more intellectual strategies to influence others. Both users and developers tend to keep their professionally related culture which result in a vast gap between them. (Mann, 2002).

4. **Foresight Gap:** when one party has greater ability to see the future of the system, but is not successful to convey that vision to the other party or to convince them. Developers usually think that users' suggestions cannot work from a technical point-of-view whereas users may determine that a solution given by a developer is not acceptable or even will negatively affect the operations of the system. (Mann, 2002).

5. **Communication Gap**: when one stakeholder group s fails to understand what the other is saying. Users often say that developers have a list of jargon not understandable for them. And to translate the user requirements into useful productive systems is not simple for developers because they do not understand the business processes and their bases. (Mann, 2002).

6. Expectation Gap: users and developers have different or sometimes unrealistic expectations about each other's capacity and ability. Nowadays users expect more from systems because they have more knowledge on computers. Meanwhile, developers sometimes offer bombastic claims because they think users are all computer naïve. (Mann, 2002).

7. Credibility Gap: it happens when developers' previous performances, especially when they failed in development projects or when there was poor customer service, cause them to lose their credits. From the developers’ perspective, they find users demand too much and are inflexible to change their opinions. (Mann, 2002).

8. Appreciation Gap: it normally happens from developers' part when they feel that their hard work, long hours and contributions to the organization are not appreciated but when something goes wrong the eyes are all on them. In some cases, even the developers are not invited to be involved in business planning. (Mann, 2002).

9. Relationship Gap: when the two parties do not interact frequently so they won’t be able to form a feasible and constructive relationship which is required for the development of the system (Mann, 2002).

Researchers such as Edstrom (1977), Gingras and McLean (1979), and Zmud and Cox (1979) have reported that software professionals usually have distinctive ways of thinking and in majority of cases the gap between the two parties is cultural. They cite that poor communication is the main result of this cultural gap.

According to Grindley (1991), developers and stakeholders have different motivation, goals, language, and problem-solving strategies. Through a survey in the context of business, he found

that 46% considered that the gap between software developers and users in business was the most important challenge they faced in system delivery. 56% believed that the culture gap is an obstacle to use the software effectively. The culture and mind-set differences cause difficulties in communication. In fact culture gap is manifested in a conflicting mind-sets which indicate different goal-making and problem-solution strategies.

Wang (1994) defines the culture gap as ‘a conflict, pervasive yet unnatural, that has mis-aligned the objectives of executive managers and technologists and that impairs or prevents organizations from obtaining a cost-effective return from their investment in information technology’. (p.1) It means that when the objectives of two parties are not adjusted or coordinated properly, it can cause a gap. He sees communication as a consistent factor. Good communication makes the interests and expectations of two parties reconciled and obviously poor communication makes the problem worse. According to Wang, good communication is necessary to make objectives and strategic plans. Therefore, the quality of communication between these two during the design and development of a system is critical in creating successful system (De Brabander & Thiers, 1984).

This notion is compatible with what Grindley discussed in his earlier study that the gap is caused because of different approaches to set goals, to solve problems and to use language which hinder an organization’s achievement and development.

Taylor-Cummings and Feeny (1997) also highlight that the ‘cultural gap’ between IT developers and users as the major cause for the failure of IT projects which has caused alarm among IT management. According to them, this cultural gap is known to all but poorly defined. By definition, they mean the signs and symptoms of the culture gap must be described. According to 1991 Price Waterhouse survey of IT directors in the UK, 47% of them consider culture gap as the biggest problem in their projects.

Following their findings, Taylor et al (1997) attempt to define the culture gap between IT developers and users through two metaphors: they compare organizations with cultural and political systems, (Morgan, 1986). In this comparison, they focus on terms of cultures and sub-cultures, different interests, conflict and power. This can be useful because it addresses the problem in terms of culture which is the main gap between developers and users. Other definitions have focused on the participants' different thought processes due to different psychological elements. They recognize that inside each organizational culture, there are sub-cultural elements that have competing interests and goals which lead to conflict. What determines which goal or priority can be realized is determined by power.

Similarly, DeBrander et al (1977 & 1984) show that power imbalance between developers and users, where developers had ‘sanctionary power’ over users. This power imbalance makes users less willing to participate and communicate. They argue that such power asymmetry disfigure the proper communication, and results in only ‘apparent agreement’ rather than true ‘mutual agreement.’ With developers' domination over users, users would agree on the system solution, but after the implementation, they may not comply to this agreement. In experimental conditions in which both developers and users have equal power, users' behaviour is in a more agreed-upon manner. DeBrabander & Thiers (1984) conclude that ineffective communication occurs when developers have authority or power over users, causing users’ behaviour diverge from what they have previously agreed to do.

Baster et al (2001) refers to Business-Technology gap (B-T gap) when technology specialists and business users lack understanding of each other's sphere of work. The former lacks the skill and knowledge that they need to make quick reactions at the time of changes in business circumstances, while business users lack the technology skills related to the system. This causes a gap in which the two parties cannot have enough recognition and understanding to meet each other's needs. Software specialists have knowledge of implementing technology, but do not have sufficient understanding of industrial and economic operations. In contrast, business users have well-developed understandings of the world of commerce and industry, but do not have adequate understanding of how to use technology to meet organizational objectives and to make important decisions..

In their paper, Nath and Kundu (2004) also mention that the culture gap is the most common cause of not getting the right requirements which result in poor or reserved communication between the stakeholders during the requirements gathering process. Therefore, developers cannot gather complete requirements from users. To solve this gap, they develop an innovative formal model to control and manage user requirements based on the evolutionary life cycle approach which can integrate the requirements development with a release-planning approach.

One major subject related to software developer and user communication is user participation in the process of software development which is considered beneficial for creating a chance for more frequent communication between two parties. Consequently, it has been undeniably accepted that user must participate in software development yet the issue is how and in which way. Bostrom et al (1977), Mumford et al (1979) and Schuler (1993) categorize some noticeable approaches to make customer more participative into "socio-technical systems theory (STS) and participatory design (PD)".

Keil and Carmel (1995) have classified customer participation into macro level and the micro level. The former acknowledges that user participation is beneficial and provides general approaches for achieving this objective (e.g., PD and STS). The second category focuses on specific requirements analysis techniques which have been documented by Byrd et al (1992). Keil and Carmel suggested the combinations of techniques and communication channels to bridge the gap of communication between customers and developers which they called “links”. As practical solutions, they suggested that managers should be cautious for providing more links and developers can best be served if they succeed in establishing numerous direct links through which they can exchange information with customers which strengthens their shared understanding.

On the other hand, Howcraft and Wilson (2003) show that user participation does not necessarily guarantee the success of the system. They address the recurring failure of information systems although user has been involved and participating. They argue about the rational and political motivations regarding user participation and believe that it must be consistent with both technical and social or political determination in the process of design and implementation. This view is coherent with what other writers in this literature have mentioned. (Franz and Robey, 1984; Markus,1983; Noble, 1984; Winner, 1980). Then developers can form a better communication with users by considering the instrumental and politically motivated explanations about user participation.

Daniel and Dana (1982) developed a standardized constructive conflict model for user participation. Constructive conflict exists in an environment where there are a multiple criteria for success, and where participants in the process have conflicting goals. Daniel and Dana found

that participation introduces conflict especially at the initiation and design phases of system development. This study adapts the models of user participation presented in (Daniel and Dana, 1982; Foster and Franz, 1998) with additional components from literature (Eisma et al., 2004; Wilson et al., 1997; Grudin, 1991) to reflect key factors that could potentially affect the participation of end-users in software development in a developing country context.

Newman & Sabherwal, (1996) , Hunton & Beeler (1997) consider user participation constructive because it can improve the requirements determination process and lead to greater buy-in. Moreover, it keeps users informed about progress which ends in higher levels of user satisfaction, system quality, and system usage (Hwang & Thorn, 1999).

However, a surprising lack of specificity in describing the user–developer has been noted in their communication process. Robey and colleagues started the early work on conflict models of participation (Robey & Farrow, 1982; Robey *et al*., 1989). Their studies identified the role of user influence (i.e. power) in software development and how this influence may lead to conflict in some cases, yet it can be managed properly.

Following them, Newman & Noble (1990) proposed a 'learning process model' to explain the events that occur during user participation. They highlight the importance of two-way communication as a forerunner to learning, and they argue that 'reciprocal learning' must occur for participation to add value (i.e. users learn from developers and vice versa), rather than just 'one-way learning.' More recently, Hunton & Beeler (1997) examined the role of users having 'instrumental voice' (i.e. influence) during system design and its influence on their use and their satisfaction with the resulting system. According to this study, users who provided input into the design, but who lacked instrumental voice, were less satisfied with the resulting system than those who had real influence throughout the design process.

Majchrzak & Beath (2000) proposed a detailed model suggesting that several processes must occur for user participation to lead to successful outcomes. They argue that for positive outcome, participation must provoke negotiation, collaborative learning, and "cognitive elaboration". In short, developers and users must communicate with and learn from each other, by elaborating their own cognitive schemes as they negotiate critical decisions related to the design and implementation of the system at hand.

Grey (1995) argues that ineffective communication may indicate different risk perceptions between the project managers and users. He shows the importance of a shared understanding of project risk between them in order to come up with more effective results. Their understanding of risk is incomplete without examining the risk perceptions of the other group. If users and developers have different perceptions of project risk factors, more conflict and miscommunication will arise.

Gallivan and Keil (2003) suggest that project managers and software developers must look beyond what users provide as the content of information. They should also investigate what information users have not offered. They suggest that the developer team should create an environment in which users feel free to discuss about their concerns, whether positive or negative. They acknowledge that user participation in system development has a critical role in successful design and implementation of systems whereas it does not mean that user participation necessarily leads to successful outcomes. One of the problems creating this gap is the power imbalance between developers and users. When the developers have more power and dominance on the product, the user will be less inclined to adopt a system that they had designed.

Creighton et al (2006) consider one related issue is the gap between the conceptual models of end-users and formal specification/analysis models of developers and present a new method for the video analysis of scenarios by using video-based requirements to process models of software development. It uses a knowledge model—an RDF graph— which uses a semiotic interpretation of film language, transforming conceptual into formal models. Video helps stakeholders to visualize the future system more clearly with definite expectations. Furthermore, in the case of technological revolution, a video-analysis allows people who are unfamiliar with the new technology to experience it. This new technique can bring the mindset of the two parties closer.

Maalej et al. (2009) describe the problem of communication gaps between developers and end-users and propose to consider user input as important information. Developers should provide examples of their generic framework for users and additionally compare specified and monitored user behavior.

To reduce the gap, Liao (2010), investigates a model of a requirement elicitation based on VCA (Value Chain Analysis), to tackle the problem of the cognitive limitations that cause analysts to gather inadequate and inaccurate requirements. He explains the procedural steps of requirement elicitation which help analysts obtain meaningful requirements.

Erfurth et al (2010) claim that for a successful software to meet customers' needs, there must be a complete understanding of stakeholders' problems which is possible by a concise and clear representation which is representation understandable for stakeholders. He suggests the participative cardgame CUTA (Collaborative User's Task Analysis) to bridge the gap between stakeholder and developer. In this process, developers can expand interview techniques and evaluate their adapted CUTA to transform them to formal models. This will help to provide a concise and clear representation for system developers.

To bridge the gap, especially in large IT projects, Abelein & Paech (2102) believe that software development is a critical step for user participation especially when user requirements are translated by developers into a technical specification (i.e. system requirements, architecture and models). In this step a lot of decisions are taken directly or indirectly. The point is some of these decisions should be shared with users. Therefore, they create an approach which enhances communication between users and developers during this step by identifying changes which occur on initial user requirements, and those levels on which communication is disintegrated. According to them, most discussions are performed on the domain level and in terms of representation, they suggest the reuse of existing documentation. To find the most adequate means of communication, they look into the MTR, which suggest using rich data channels in case of high equivocal content.

Prior to them, El-Ramly et al. (2004) developed an approach which was to recover use cases from monitored user actions and perform that knowledge in user interface reengineering. Similarly, Antonio et al. (2000) and Li et al. (2007) developed approaches to recover a use case model from runtime information by the use case detection. Unlike them, Abelein & Paech (2012) reused their work by comparing monitored user behavior to a specificied user behavior. By monitoring user actions and behaviour, they aimed to detect mis-matches between user behaviour and developer assumptions. These mismatches can serve as starting points to reduce the misunderstanding and miscommunication between users and developers. This approach can help developers to improve user comprehension and to maximize the support of user needs in their applications.

Discussion & Analysis

1. The Human Friction Behind the "Nine Gaps"

When we examine Mann's categorization of the nine gaps, we aren't just looking at technical hurdles; we are looking at a fundamental clash of worldviews. The **Perspective** and **Cultural Gaps** are arguably the most damaging because they involve how people define "success." To a developer, success is an elegant, bug-free architecture (the how). To a user, success is simply finishing their work ten minutes faster (the what). This creates a **Communication Gap** that goes beyond mere jargon—it's a failure to translate value. When developers use "rational persuasion" and users rely on "intuitive strategies," they aren't just using different words; they are essentially operating on different psychological "operating systems."

2. The Illusion of Agreement

A striking insight in this study is the concept of **"apparent agreement"** caused by power imbalances. We often assume that if a user signs off on an SRS, they are happy. However, as DeBrabander points out, if a developer dominates the conversation with technical authority, the user might simply surrender. This isn't true consent; it's a defense mechanism. The user agrees to the plan to end the meeting, but the **Credibility Gap** actually widens because they don't believe the system will solve their real-world problems. This explains why so many "technically perfect" projects are abandoned post-launch—the users never felt they had an "instrumental voice" in the design.

3. Participation as a Political Minefield

The industry often treats "User Participation" as a magic cure. However, the paper argues that participation is a complex political interplay. For it to be effective, it must move from "one-way learning" (where devs interview users) to **"reciprocal learning."** This is where the innovative tools mentioned—like **Video Analysis** and **CUTA (Card Games)**—become essential. They level the playing field. A video allows a user to "see" a future they cannot visualize from a dry text document, effectively bridging the **Foresight Gap**. It turns a technical negotiation into a shared, tangible experience.

4. Closing the Distance

Ultimately, the "Business-Technology Gap" remains because software development is a human endeavor prone to human ego. The **Appreciation Gap**—where developers feel like unthanked mechanics and users feel like ignored customers—creates a cycle of resentment. To bridge this, we have to move away from better documentation templates and toward better empathy tools. Whether it is monitoring real-time behavior to catch "mismatches" (Abelein & Paech) or using Value Chain Analysis to understand the user's economic pain, the goal is the same: closing the distance between the person writing the code and the person living with the results.

5. Deep Dive: Socio-Technical Misalignment

Beyond the nine gaps, the study hints at a deeper structural tension: the difference between **Formal Models** and **Conceptual Realities**. Developers are trained to think in binary constraints, while users operate in a world of social nuance and shifting business priorities.

When a developer asks for "requirements," they are looking for a frozen snapshot. When a user provides them, they are describing a fluid process.

This misalignment is exacerbated by **Risk Perception** (Grey, 1995). Developers view risk through the lens of technical debt and system crashes, while users view risk as a threat to their daily productivity or job security. Without a shared language for risk, the **Relationship Gap** becomes a barrier that no amount of technical documentation can overcome.

Comparative Analysis Tables

Table 1: Contrasting Mindsets and Values

Feature	Software Developer Perspective	End-User Perspective
Primary Goal	Technical integrity and system stability.	Task efficiency and business value.
Language	Jargon-heavy, logical, and structured.	Domain-specific, intuitive, and fluid.
View of Technology	The primary focus and solution.	A tool to achieve a separate goal.
Problem Solving	Analytical and rational persuasion.	Intuitive and experiential strategies.
Risk Focus	System failure or "bugs."	Operational disruption or loss of time.

Table 2: Impact of Power and Participation Dynamics

Dynamic Type	Developer Behavior	User Behavior	Resulting Outcome
Power Imbalance	Dominates design via "Sanctionary Power."	Passive acceptance; "Apparent Agreement."	Poor adoption; system fails in practice.
One-Way Learning	Extracts information like a "recorder."	Provides raw data without context.	Perspective Gap; misaligned objectives.
Reciprocal Learning	Explains constraints; learns business logic.	Understands tech limits; explains "why."	Mutual Agreement; high system quality.
Instrumental Voice	Facilitates and listens to concerns.	Actively influences critical decisions.	High user satisfaction and "buy-in."

Final Analytical Summary

The gap in SRS is not a "bug" to be fixed; it is a permanent condition of multidisciplinary work. The goal of a project manager or lead developer shouldn't be to eliminate the gap entirely that is impossible given the different personalities and goals involved. Instead, the goal is to manage the tension.

By using "rich data channels" (like prototypes and videos) and balancing power dynamics, the "apparent agreement" that haunts so many software projects can be turned into a genuine, shared vision. The most successful software isn't built by the smartest developers; it's built by the teams that are the most culturally sensitive to each other’s professional worlds.

Conclusion

In conclusion, this study has provided a comprehensive exploration of the enduring challenges in communication and understanding between software developers and end-users. Despite numerous efforts to bridge the gap, it remains a persistent and complex issue with multifaceted dimensions. The study categorized the gap into nine types, highlighting disparities in perspective, ownership, culture, foresight, communication, expectations, credibility, appreciation, and relationships. These categories underscored the diverse ways in which developers and users often fail to align their viewpoints and expectations, leading to miscommunication and suboptimal software outcomes.

A key takeaway from this study is the significance of the cultural divide between developers and users, as well as the power imbalances that can exacerbate communication difficulties. Addressing these cultural and power dynamics is essential for fostering more effective

collaboration. Efforts to bridge the gap have centered on promoting user participation in software development. However, this study emphasized that success in this endeavor requires careful consideration of technical and political factors. Effective communication, negotiation, and mutual learning are essential components of successful user participation.

The study also explored innovative approaches, such as video analysis and clear representations, to enhance mutual understanding among stakeholders. These methods offer potential solutions to mitigate the communication challenges.

Finally this paper has shed light on the persistent nature of the gap between software developers and users, driven by differences in mindset, culture, and communication. It underscores the need for improved communication, cultural sensitivity, and active user participation to address these issues effectively. Ultimately, closing this gap is crucial for achieving successful software development projects that truly meet the needs and expectations of end-users.

References:

Abelein, U., Peech, B. “A Proposal for Enhancing User-Developer Communication in Large IT Projects,” 2012 *IEEE CHASE 2012*, Zurich, Switzerland.

Antonio, G., Lucca, D., Fasolino, A.R. Carlini, D., Federico, N. and Claudio, V. “Recovering use case models from object-oriented code: a thread-based approach,” In *Proc. of the Seventh Working Conf. on Reverse Engineering*, pages 108–117. IEEE, 2000.

Baster , G., Konana , P., Scott, J. “Business Components: A Case Study of Bankers Trust Australia Limited,” *Communications of the ACM*, 44(5), 2001.

Byrd, T.A., Cossick, K.L., and Zmud, R.W. “A synthesis of research on requirements analysis and knowledge acquisition techniques,” *MIS Q. 16*, 1, 117–138, 1992.

Bostrom, R.P., and Heinen, J.S. “MIS problems and failures: A socio-technical perspective—part I: The causes,” *MIS Q. 1*, 3, pp 17–32, 1977.

Creiton, O., Otte, M., Bruegge, B. “Software Cinema—Video-based Requirements Engineering,” *14th IEEE International Requirements Engineering Conference*, 2006.

Daniel, R. and Dana, F. “User Participation in Information System Development: A Conflict Model and Empirical Test,” *Management Science*, 28, 1, 73-85, 1982.

DeBrabander, B. & Edstrom, A. “Successful information system development projects,” *Management Science*, 24, 191–199, 1977.

DeBrabander, B. & Thiers, G. “Successful information system development in relation to situational factors which affect effective communication between MIS users and EDP specialists,” *Management Science*, 30 (2), 137–155, 1984.

Edström, A., & Galbraith, J.R. “Transfer of managers as a co-ordination and control strategy in multinational organizations,” *Administrative Science Quarterly*, 22(2): 248-263, 1977.

Eisma, R, Dickson, A., Goodman, J., Syme, A., Tiwari, L. and Newell, A.F. “Early User Participation in the Development of Information Technology-Related Products for Older People,” *Universal Access in Information Society*, 3, 131-140, 2004.

El-Ramly, M. and Stroulia, E. “Mining software usage data,” In *Proc.of the 1st Int. Workshop on Mining Software Repositories,* MSR 2004.

Erfurth,I., Krishner, K. “Requirements Elicitation with adapted CUTA Cards: First Experiences with Business Process Analysis,” *15th IEEE International Conference on Engineering of Complex Computer Systems,*2010.

Franz, C. R. and D. Robey. “An Investigation of User-led Systems Design: Rational and Political Perspectives,” *Communications of the ACM*, 27, 12, 1202–1209, 1984.

Gallivan, M.J., Keil, M. “The User-to-Developer Communication Process: A Critical Case Study,” *Information Systems Journal 13,* 2003.

Gingras, L., & McLean, E.R. “A study of users and designers of information systems,” UCLA. California: Graduate School of Management, 1979.

Göransson, B., Sändback, T., “Usability Designers improve the user-centred design process”, *INTERACT*’99, Edinburgh, 1999.

Grey, S. *Practical Risk Assessment for Project Management*. John Wiley and Sons, Chichester, UK, 1995.

Grindley, K. *Managing IT at board level: The hidden agenda exposed*. London, Pitman, 1991.

Grudin, J. “The Development of Interactive Systems: Bridging the Gaps between Developers and Users.” *IEEE Computer*, 24, 4, 59-69, 1991.

Hornik, S., Chen, H-G, Klein, G., Jiang, J. “Communication Skills of IS Providers: An Expectation Gap Analysis From Three Stakeholder Perspectives,” *IEEE Transactions on Professional Communication,* Vol 46, No. 1. 17-29, 2003.

Howcraft, Debra and Wilson, Melanie. “Participation: ‘bounded freedom’ or hidden constraints on user involvement,” *New Technology, Work and Employment* 18:1, 1-19, 2003.

Hunton, J.E. & Beeler, J.D. “Effects of user participation in systems development: a longitudinal field experiment,” *MIS Quarterly*,21 (4), 359–388, 1977.

Hwang, M.I. & Thorn, R.G. “The effect of user engagement on system success: a meta-analytical integration of research findings,” *Information and Management,* 35 (4), 229–236, 1999.

Johnson, G., & Scholes, K. *Exploring Corporate Strategy,* 5th edition. London, Prentice Hall, 1999.

Karten, N., *Managing Expectations,* Dorset House Publishing, New York, NY, 1994.

Keil, Mark and Carmel, Erran. “Customer-Developer Links in Software Development,” *Communication of the ACM,* vol. 38, no. 5, pp. 33-44, 1995.

Liao, H. “Requirement Elicitation of Enterprise Informationization from view of VCA,” *6th International Conference on Networked Computing (INC),* 2010.

Li, S. Hu, P. Chen, L. Wu, and Chen, W. “Discovering and mining use case model in reverse engineering,” In *Proc. of the 4th Int. Conf. on Fuzzy Systems and Knowledge Discovery,* FSKD’07. *IEEE, 2007.*

Maalej, W.,Happel, H. and Rashid, A. “When users become collaborators: Towards continuous and context-aware user input,” *Proc. of the 24th ACM SIGPLAN Conf. Comp. on Object Oriented Programming Systems Languages and Applications*, pages 981–990, 2009.
Majchrzak, A. & Beath, C.M. “Beyond user participation. A Model of Learning and Negotiation During Systems Development.” *Workshop on Redefining the Organization Roles of Information Technology in the Information Age.* ZmudR.W. (ed.). 2000.
Mann, J. “IT Education’s Failure to Deliver Successful Information Systems: Now is the Time to Address the IT-User Gap,” *Journal of Information Technology Education*, Vol 1, No. 4, 255, 2002.
Markus, M. L. “Power, Politics, and MIS Implementation,” *Communications of the ACM*, 26, 6, 430–444, 1983.

Mumford, E., and Henshall, D. *A Participative Approach to Computer Systems Design*. Associated Business Press, London, 1979.

Nath, I., Kundu, S. “Is Software Requirements a Corporate Asset?” *IEEE International Engineering Management Conference PP 244-24, .2004.*

Newman, M. & Noble, F. “User involvement as an interaction process: a case study,” *Information Systems Research*, 1 (1),89–113, 1990.

Newman, M. & Robey, D. “A social process model of user-analyst relationships,” *MIS Quarterly*, 16 (2), 249–266, 1992.
Newman, M. & Sabherwal, R. “Determinants of commitment to information systems development: a longitudinal investigation,” *MIS Quarterly*, 20 (1), 23–54, 1996.

Noble, D. *Forces of Production: a Social History of Industrial Automation* (New York: Knopf), 1984.
Dalla, L. O. B., Essgaer, M., Jetlawei, S. S., EL-sseid, M., Alsharif, A., & Agila, A. A. A. (2026). Local Precision and Global Harmony: A Comparative Literature Review (LR) Framework for Taylor and Fourier Series in Engineering Modeling. *Al-Farooq Journal of Sciences*, *2*(1), 275-304.
Robey, D. “Modeling interpersonal processes during system development: further thoughts and some suggestions,” *Information Systems Research*, 5 (4), 439–445, 1994.
Robey, D. & Farrow, D. “User involvement in information system development: a conflict model and empirical test,” *Management Science*, 28 (1), 73–85, 1982.
Robey, D., Farrow, D.L. & Franz, C.R. “Group process and conflict in system development,” *Management Science*, **35** (10),1172–1191, 1989.

Schuler, D., and Namioka, A. *Participatory Design: Principles and Practice.* Erlbaum, Hillsdale, NJ, 1993.
ALfordaga, M., & Omer, A. I. (2026). Real-Time Requirements Analysis in Smart Healthcare Monitoring Systems: A Case Study of an ATmega328P-Based Health Monitoring Device. Al-Farooq Journal of Sciences, 2(3), 1216-1236.
Taylor-Cummings, A., & Feeny, D.F. *Managing IT as a strategic resource*, London: McGraw Hill, 171-198, 1997.
Wang, C.B. *Techno Vision*, New York, McGraw-Hill, 1994.
Wilson, A., Bekker, M., Johnson, P. and Johnson, H. “Helping and Hindering User Involvement: A Tale of Everyday Design,” *ACM Conference on Human Factors in Computing Systems*, Atlanta, 178-185, 1997.

Winner, L. “Do artifacts have politics?”, *Daedalus*, 109, 121–36, 1980.
Zmud, R.W. & Cox, J.F. “The implementation process: a change approach,” *MIS Quarterly*, 3(2), 35-43, 1979.