Indonesian Interdisciplinary Journal of Sharia Economics (IIJSE)
e-ISSN: 2621-606X
Vol. 9. No. 2 (2026)
Page: 11758-11769

EFFECTIVENESS OF SOFTWARE PROJECT MANAGEMENT BASED ON CAPABILITY MATURITY MODEL INTEGRATION IN THE IMPLEMENTATION OF SECURE SOFTWARE DEVELOPMENT LIFE CYCLE AT PT XYZ

Muhammad Hanifudin[1]
Universitas Islam Malang, Malang, Indonesia
mhanifudin87@yahoo.co.id

Ahmad Hussein Satrio Pambudi[2]
Universitas Terbuka, Malang, Indonesia
054949648@ecampus.ut.ac.id

Abstract

This study evaluates the implementation of the Secure Software Development Life Cycle (S-SDLC) and examines the effectiveness of software project management based on the Capability Maturity Model Integration (CMMI) version 3 at PT Xyz. A qualitative research approach was employed using in-depth interviews, observations, and documentation analysis. Empirical findings were triangulated and mapped against S-SDLC indicators and twelve CMMI v3 Practice Areas (PAs) grouped into Doing, Managing, and Enabling. The findings indicate that S-SDLC has been integrated throughout the software development lifecycle, positioning security as a built-in process rather than a reactive add-on. Security-relevant practices collectively enhance software quality, system stability, and preventive security governance. With regard to project management effectiveness, the study shows that most PAs reached Capability Level 3 (Defined), reflecting a mature level of process standardization at the project level. However, effectiveness was more prominent in terms of quality, security, and process control, while schedule performance faced pressure due to workload imbalances and resource capacity constraints. These findings highlight that software project effectiveness is inherently multidimensional and cannot be assessed solely through schedule adherence. The study contributes to theory by demonstrating that CMMI v3 can serve as an evaluative framework compatible with S-SDLC and enriches project management discourse by positioning security as an integral dimension of process effectiveness. Practically, the study provides a fit-for-purpose internal evaluation framework for software development organizations seeking to improve process maturity without formal appraisal.

Keywords: CMMI v3, Secure Software Development Life Cycle, Software Project Management, Process Maturity

INTRODUCTION

Software has now become the backbone of various industrial sectors, ranging from finance and healthcare to government. The successful implementation of software systems determines the effectiveness of business processes and the competitiveness of organizations. Nevertheless, the success rate of software projects remains relatively low. The Standish Group (2020), through the CHAOS Report, recorded that only 31% of projects are completed on time, within budget, and meeting specifications, while the rest experience delays, cost overruns, or failure. This condition indicates that software project management is a strategic necessity in supporting organizational digital transformation.

The complexity of software projects arises from the involvement of multiple stakeholders, dynamic business requirements, and technological uncertainty. Without structured project management, such complexity can lead to conflicts, miscommunication, and inefficient use of resources. Ian Sommerville (2016) emphasizes that software project failures are more often triggered by weaknesses in planning, coordination, and control rather than technical limitations. Therefore, the main issues in software projects are managerial in nature.

Software project management provides a framework for balancing the dimensions of time, cost, and quality. A mature methodology helps organizations improve team productivity and reduce rework caused by undocumented requirement changes. Harold Kerzner (2017) states that project management maturity enables organizations to optimize resource utilization and minimize waste, particularly in companies managing multiple projects simultaneously.

Beyond efficiency, risk management is a key factor in software project success. Risks may originate from technical aspects (bugs, integration issues, security vulnerabilities) as well as non-technical factors (changes in business requirements, limited human resources, or user resistance). The Project Management Institute (2021) emphasizes that risk management is a core process because it reduces the likelihood of project failure and increases the chances of success.

The urgency of project management is also influenced by increasing demands for system quality, reliability, and security. For instance, sectors such as healthcare and finance require strict compliance with data security regulations. Roger S. Pressman and Bruce Maxim (2020) argue that software quality depends on disciplined managerial processes from planning through maintenance. Therefore, project management not only controls cost and schedule but also ensures system reliability and sustainability.

As organizational dependence on software systems increases, security risks become an increasingly dominant challenge. Failure to anticipate security risks may lead to data breaches, service disruptions, and financial losses. The IBM (2022) reported that the global average cost of a data breach reached USD 4.35 million, with many vulnerabilities originating from weaknesses in the software development process. These findings confirm that security cannot be separated from software project management and must be considered from the planning stage.

Traditional software development models, particularly the Software Development Life Cycle (SDLC), generally treat security as an additional activity at the final stage, especially during testing. This approach has proven ineffective because vulnerabilities are often discovered when the system is nearly complete, requiring additional time and cost for remediation. Gary McGraw (2006) describes the practice of “security as an afterthought” as

one of the main reasons systems fail to meet security standards. This situation highlights the need to integrate security from the early stages of development.

The Secure Software Development Life Cycle (S-SDLC) emerged as a framework that integrates security practices into every phase of development, from requirements analysis to maintenance. As a result, S-SDLC improves software quality while functioning as a proactive risk management strategy. Microsoft (2018) reported that implementing S-SDLC can reduce up to 50% of vulnerabilities reaching the production stage, thereby lowering security risks and reducing remediation costs.

From a project management perspective, S-SDLC aligns with risk management principles that emphasize early identification, analysis, mitigation, and monitoring of risks. The National Institute of Standards and Technology (2020) states that integrating security into SDLC represents a best practice in organizational risk management because it links technical controls with managerial governance. This is particularly critical for software development companies serving high-risk sectors such as finance and healthcare.

Security integration through S-SDLC also provides efficiency benefits. Early detection of vulnerabilities reduces rework costs and accelerates release cycles. Barry Boehm and Richard Turner (2004) show that fixing vulnerabilities during the design phase costs only one-tenth of the cost required after implementation. From a managerial perspective, S-SDLC directly contributes to efficient resource allocation and project sustainability.

The urgency of implementing S-SDLC is further reinforced by the emergence of regulations and industry standards requiring application security as part of the development lifecycle. ISO/IEC 27034 emphasizes integrating security within SDLC, while the General Data Protection Regulation (GDPR) requires data protection by design. Compliance with such regulations can only be achieved if project management adopts S-SDLC as part of its operational policies and procedures.

The effectiveness of software project management should not only be evaluated based on whether projects are completed on time and within budget but also on the organization's ability to create repeatable, measurable, and improvable processes. Such evaluation requires standardized measurement tools to avoid subjective judgments. Kerzner (2017) emphasizes that organizations require formal measurement frameworks to assess project management maturity and make data-driven improvement decisions.

The Capability Maturity Model Integration (CMMI) provides a comprehensive framework to evaluate organizational process maturity through practice areas focusing on consistency, predictability, and process effectiveness. This approach aligns with the perspective of Mary Beth Chrissis, Mike Konrad, and Sandra Shrum (2011), who emphasize that CMMI systematically improves organizational capability through structured process evaluation. The latest version, CMMI Version 3.0, strengthens the relationship between management practices and business performance while offering greater flexibility in appraisal (CMMI Institute, 2018), allowing organizations not only to measure process compliance but also its impact on outcomes.

For PT Xyz, the use of CMMI v3 serves as an instrument to evaluate the maturity of S-SDLC implementation, identify gaps in risk management, and formulate measurable improvement strategies. Such evaluation is important for enhancing internal efficiency while strengthening the company's competitiveness in the technology services market. This research is relevant to the field of management because its primary focus is evaluating the

effectiveness of software project management through a process-based approach. Within the management context, software projects are viewed as structured activities requiring planning, organizing, and controlling to achieve business objectives. By using CMMI v3 as a measurement framework, this study contributes to project management by providing a systematic approach to assessing organizational capability in managing information technology projects.

Furthermore, the research is also relevant to risk management and quality management. The implementation of S-SDLC is examined not merely as a technical practice but as part of an organizational strategy for managing software security risks. Thus, product security and quality can be positioned as outcomes of effective managerial practices. This study demonstrates that a capability-based process approach through CMMI v3 can support managerial decision-making and encourage continuous improvement within technology organizations. Initial observations by the researcher indicate that the main challenge in software project management at PT Xyz lies in weak technical documentation related to change management. Whenever project requirements change or technical team members rotate, the company faces adaptation difficulties due to the lack of adequate documentation regarding technical scope, system architecture, module behavior, and algorithms developed over more than ten years.

This condition creates significant managerial implications. First, new technical staff require extended onboarding periods to understand complex systems, which reduces team productivity. Second, unclear documentation leads to delays in completing tasks, including new feature development, system enhancements, and bug fixes. These delays may disrupt project release schedules and the fulfillment of Service Level Agreements (SLA) with clients. Moreover, this issue affects client trust and corporate reputation. In the highly competitive software industry, inconsistent service quality and delays in delivering results may reduce user satisfaction and weaken company credibility. The Standish Group (2020) emphasizes that failures in documentation and communication management are among the main causes of software project delays. Therefore, the issues experienced at PT Xyz represent not only operational obstacles but also managerial gaps in project governance.

This study is positioned at the intersection of software project management, risk management, and organizational capability measurement. Previous research has widely examined software project management effectiveness through the CMMI framework (Chrissis, Konrad, & Shrum, 2011), while other studies discuss security implementation through S-SDLC (McGraw, 2006; NIST, 2020). However, there remain limited studies explicitly linking CMMI as an effectiveness evaluation model with S-SDLC as the object of security implementation within real organizational contexts. Therefore, this research fills that gap, particularly within the software industry in Southeast Asia.

PT Xyz was selected as the research object due to its high organizational complexity in cross-sector and cross-country software development (Indonesia, Singapore, and China). Empirically, the company faces challenges related to technical documentation, knowledge transfer among teams, delays in delivering features and bug fixes, and increasing security risks. These conditions indicate the need to evaluate project management effectiveness while integrating security practices into the software development lifecycle. The researcher's close access to the organization enables in-depth primary data collection through interviews, observation, and documentation.

Beyond academic and empirical considerations, the selection of the research object is also supported by practical reasons. The researcher is a technology practitioner with more than ten years of experience in the software industry and currently serves as General Lead Software Engineer at PT Xyz. This position provides contextual understanding and access to relevant data for comprehensively examining the phenomenon. Therefore, the research offers both theoretical contributions and practical benefits for improving internal organizational processes. The novelty of this study lies in its integrative approach. First, the research uses CMMI v3 as an effectiveness measurement framework linking management practices with business performance. Second, S-SDLC is positioned not merely as a technical practice but as a risk management strategy whose maturity can be assessed using CMMI indicators. Third, the research presents an empirical study within a software development company in Southeast Asia, providing real-world evidence of process-based evaluation models. This contribution expands the application of CMMI into the domain of software security and highlights the importance of digital risk management in modern project management.

RESEARCH METHOD

This study employs a qualitative approach because it aims to gain an in-depth understanding of the implementation process of the Secure Software Development Life Cycle (S-SDLC) within the organization, as well as to evaluate the effectiveness of software project management using the Capability Maturity Model Integration (CMMI v3) framework. Maxwell (2013) emphasizes that qualitative research focuses on exploring meaning, context, and processes, making it suitable for examining complex and situational phenomena. Creswell & Creswell (2018) also highlight that a qualitative approach is used when researchers seek an interpretive understanding of the experiences and perspectives of individuals or groups. The data in this study consist of primary and secondary sources. Primary data were obtained through in-depth interviews, direct observations, and review of internal company documentation related to the implementation of the Secure Software Development Life Cycle (S-SDLC) and software project management. Primary data aim to capture the perspectives of organizational actors involved in the software development process, including project managers, software engineers, quality assurance personnel, and other relevant stakeholders. Through primary data, this study maps practices, experiences, and empirical challenges in implementing S-SDLC while evaluating its effectiveness using the Capability Maturity Model Integration (CMMI v3) indicators. Meanwhile, secondary data were collected from company documents, standard operating procedures, internal presentation materials, project reports, as well as academic literature and industry policies related to S-SDLC, software project management, and CMMI. Secondary data serve as a complement to strengthen the interpretation of primary findings and provide conceptual and structural context for the analysis.

Data collection in this study was conducted using three main techniques: in-depth interviews, observation, and document review. The use of these three techniques aims to obtain a comprehensive understanding of the implementation of the Secure Software Development Life Cycle (S-SDLC) and the effectiveness of software project management at PT Xyz. This triangulative approach aligns with the perspectives of Creswell & Creswell (2018) and Maxwell (2013), who emphasize the importance of using multiple data sources in qualitative research to enhance the validity of interpretations. Data analysis in qualitative

research is an iterative process that occurs from the stage of data collection through to drawing conclusions. Creswell & Creswell (2018) explain that qualitative analysis involves activities such as organizing the data, reading through all the information, coding, developing themes, and presenting interpretations. Maxwell (2013) emphasizes that this process is not linear but rather cyclical and interconnected with data collection, allowing the researcher to make thematic or exploratory adjustments throughout the course of the study.

RESULTS AND DISCUSSION

This limitation of scope was not chosen as a weakness of the study, but rather as an academic strategy to maintain focus, analytical depth, and data validity. By measuring 12 relevant PAs, the study can provide a realistic and representative picture of software project management effectiveness based on CMMI v3 indicators. This approach ensures that the research results remain contextual and accurately address the research objectives.

Table 1.
List of 12 CMMI v3 Practice Areas

No	Practice Area (PA)	Group
1	Project Planning (PP)	Managing
2	Project Monitoring and Control (PMC)	Managing
3	Requirements Development (RD)	Doing
4	Requirements Management (REQM)	Enabling
5	Stakeholder Engagement (SE)	Enabling
6	Risk and Opportunity Management (RSK)	Managing
7	Decision Analysis and Resolution (DAR)	Managing
8	Technical Solution (TS)	Doing
9	Configuration Management (CM)	Enabling
10	Verification (VER)	Doing
11	Validation (VAL)	Doing
12	Incident Resolution and Prevention (IRP)	Doing

The twelve Practice Areas (PA) analyzed in this study are grouped into three main categories of CMMI Version 3.0, namely Doing, Managing, and Enabling. This classification is important so that the analysis of software project management effectiveness can be conducted in a more structured manner and aligned with the functional roles of each PA within the software development lifecycle. The Doing category contains PAs that are directly related to software engineering and production activities, ranging from requirements management to the verification and validation of outputs. In this study, the PAs included in the Doing category consist of Requirements Development (RD), Technical Solution (TS),

Verification (VER), Validation (VAL), and Incident Resolution and Prevention (IRP). These five PAs represent the core activities that produce technical outputs and therefore serve as operational performance indicators in system development management.The Managing category includes PAs that regulate, coordinate, and control the project to ensure it aligns with the agreed objectives, scope, schedule, cost, and risk parameters. The PAs in this category are Project Planning (PP), Project Monitoring and Control (PMC), Risk and Opportunity Management (RSK), and Decision Analysis and Resolution (DAR). These four PAs function as process control instruments that ensure the project runs efficiently, measurably, and is capable of anticipating uncertainties during development.

The Enabling category comprises PAs that support the effectiveness of Doing and Managing activities through supporting mechanisms, governance structures, communication processes, and configuration control. In this study, the PAs included in the Enabling category are Requirements Management (REQM), Stakeholder Engagement (SE), and Configuration Management (CM). These three PAs do not directly produce technical products; however, they provide the foundational stability for processes, such as requirements traceability, stakeholder coordination, and the control of technical artifacts and changes.

The Implementation of Secure Software Development Life Cycle (S-SDLC)

The study's findings indicate that the implementation of the Secure Software Development Life Cycle (S-SDLC) at PT Xyz has been comprehensive and integrated across all stages of software development. Security is not treated as an additional activity at the end of the project but is embedded from the early stages through planning, requirements management, analysis, design, implementation, testing, documentation, and deployment. This approach aligns with the security-by-design paradigm and the principle of building security in, as emphasized in the literature (McGraw, 2006; NIST, 2020). Practices such as risk-based planning, cross-role requirements validation, design referencing standards and best practices, secure coding with structured peer reviews, layered testing covering security aspects, continuous documentation, and controlled deployment via CI/CD pipelines and post-release monitoring demonstrate that the organization has systematically adopted S-SDLC. Integrating security during planning and requirements engineering shows that the organization positions security as a proactive risk mitigation mechanism rather than merely reactive control. These findings support the view that many security vulnerabilities stem from inadequate requirements and analysis (Sommerville, 2016; Pressman & Maxim, 2020). During the design and implementation stages, the use of internal standards, design walkthroughs, and layered code reviews act as technical controls to reduce implementation errors and system inconsistencies. Analytically, these practices strengthen the organization's internal control over high-impact technical decisions. However, the effectiveness of these controls heavily depends on the depth of security evaluation during the review process, not solely on formal compliance with standards, as critiqued by McConnell (2004) and McGraw (2006).

The testing, documentation, and deployment phases show that S-SDLC is supported by continuous quality control mechanisms. Layered testing, continuously updated technical documentation, and controlled deployment with rollback plans and post-release monitoring reflect the organization's awareness of operational and security risks after system release. This approach is consistent with NIST (2020) and PMI (2021) recommendations regarding the importance of continuous assurance and risk management in modern systems.

Overall, the findings suggest that S-SDLC implementation at PT Xyz has established a strong and integrated software security governance foundation. However, its long-term effectiveness depends on consistent implementation, thorough security risk evaluations, and the organization's capacity to adapt to dynamic cyber threats. Therefore, S-SDLC should be viewed not as a static condition but as an evolving process that requires ongoing technical, managerial, and organizational support.

The Effectiveness of CMMI-Based Software Project Management

The research findings indicate that the effectiveness of software project management based on CMMI v3 at PT Xyz is relatively good and controlled within the scope of the 12 analyzed Practice Areas (PA). The three PA groups – Doing, Managing, and Enabling – show integration between technical activities, managerial functions, and process support mechanisms. The Doing group reflects systematic technical execution through requirements management, solution design, verification, validation, and incident handling. The Managing group demonstrates planning, monitoring, risk mitigation, and analysis-based decision-making, while the Enabling group ensures process consistency through controlled requirements management, stakeholder engagement, and configuration management.

This approach shows that software project management practices at the project level align with CMMI v3 core principles, namely process quality improvement, control, and sustainable project performance. The study's focus on 12 feasible and contextual PAs aligns with the fit-for-purpose assessment principle, as CMMI v3 does not require full implementation of all PAs but allows adaptation to the organization's goals and scope (CMMI Institute, 2023). Limiting the scope to observable PAs strengthens the validity of findings because the analysis is based on practices actually implemented in daily project operations. From a technical execution perspective, the Doing group shows that software development processes have been carried out systematically and quality-oriented. Integration among Requirements Development, Technical Solution, Verification, Validation, and Incident Resolution and Prevention demonstrates continuity between requirement specifications, technical design, implementation, and quality control. These findings align with Pressman and Maxim (2020) and Sommerville (2016), who emphasize that integration of technical processes is a key determinant of software project success. However, effectiveness in the Doing group depends on consistent application across projects, as variations in technical standards at the organizational level can affect uniformity in project outcomes.

The Managing group shows that managerial functions have been implemented effectively through planning, monitoring, risk mitigation, and evaluation-based decision-making. Practice Areas such as Project Planning, Project Monitoring & Control, Risk and Opportunity Management, and Decision Analysis & Resolution indicate that the organization has shifted from reactive project management to a more proactive and adaptive approach. This aligns with modern project management principles that place continuous monitoring and risk management as strategic pillars of project control (Kerzner, 2017; PMI, 2021). Empirically, these mechanisms allow teams to navigate changing requirements and technical challenges without compromising schedule stability or quality.

The Enabling group serves as a foundation for process stability, ensuring consistent execution of Doing and Managing practices. Practice Areas like Requirements Management, Stakeholder Engagement, and Configuration Management strengthen project governance

through controlled requirements management, planned stakeholder involvement, and traceability of project artifacts throughout the development cycle. These findings indicate that Enabling plays a key role in preventing miscommunication, requirement mismatches, and system version inconsistencies. However, its effectiveness could be further enhanced with broader organizational practices, such as formal training, cross-project standardization, and organizational learning mechanisms, which were not the focus of this study. Overall, the effectiveness of CMMI v3-based software project management at PT Xyz can be positively assessed within the study's scope. Integration of Doing, Managing, and Enabling practices demonstrates project-level process maturity supporting quality, control, and sustainability. However, this effectiveness is contextual and cannot be generalized as a representation of overall organizational maturity. Therefore, the findings support the view that CMMI v3 is most effective as a continuous improvement framework, where success is measured not by the number of PAs implemented but by consistency, depth, and relevance to project objectives and organizational needs (Chrissis et al., 2011; CMMI Institute, 2023).

The implementation of the Plan–Do–Check–Act (PDCA) cycle provides a systematic framework to explain the relationship between excessive workload and product delivery delays. In the Plan phase, project planning that does not account for actual human resource capacity can result in unrealistic schedule targets. Even with well-documented planning aligned to process standards, mismatches between workload and team capacity create latent conditions leading to delays during implementation. Operations management studies indicate that planning ignoring effective work capacity tends to cause bottlenecks and reduce schedule reliability (Slack & Brandon-Jones, 2022). In the Do phase, excessive workload directly affects the team's ability to execute development activities consistently. Even with disciplined work procedures, quality standards, and security practices, workload pressure increases the risk of burnout, reduces productivity, and slows task completion. Software engineering research shows that sustained high-intensity work does not accelerate project completion but increases the likelihood of delays due to diminished individual and team performance (Meyer et al., 2019).

The Check phase of PDCA identifies gaps between process compliance and project performance. Delivery delays despite disciplined processes show that process adherence does not equate to performance. Process adherence indicates how well activities follow standards, while performance reflects actual outcomes in terms of timeliness, productivity, and work continuity. In process management, an organization may achieve high compliance yet still experience performance decline if human factors and operational capacity are not proportionally managed (Drucker, 2007). The Action phase emphasizes the importance of continuous improvement that focuses not only on refining procedures but also on managing workload and team capacity. PDCA uses evaluation results to adjust plans, including balancing resource allocation and revising schedule targets. Therefore, delivery delays are not seen merely as failures of individuals or processes but as systemic signals that project management effectiveness requires integration between process discipline and workload capacity management. This approach aligns with the principle of continuous improvement, emphasizing balance between process standards and human performance as the primary executor of those processes (Deming, 1986).

CONCLUSION

1. The implementation of the S-SDLC at PT Xyz has been integrated, systematic, and consistent throughout the software development lifecycle. Security is positioned as an inherent aspect of the process (built-in security), not a reactive, add-on element. Practices such as risk-based planning, cross-role requirements validation, standardized design, secure coding, layered code reviews, iterative testing, and controlled deployment demonstrate the internalization of security principles with shift-left security design and security within the organization's workflow. These findings indicate that the implementation of the S-SDLC not only contributes to improved software technical quality and security but also strengthens development governance through increased traceability, cross-role collaboration, and better process control.
2. The effectiveness of software project management at PT Xyz demonstrates positive achievements within the scope of the CMMI v3 Practice Areas analyzed. The integration of Practice Areas into three groups demonstrates the integration of technical activities, managerial functions, and support mechanisms, resulting in process consistency and improved quality of development outcomes. Most Practice Areas are at Capability Level 3 (Defined), indicating that processes have been standardized and replicated across projects through IT Development Documentation 2.0. However, effectiveness is multidimensional: security and quality demonstrate strong results, while time is under pressure due to team workload and trade-offs between security, quality, and schedule. Thus, CMMI v3 proves relevant as a process improvement framework, as long as it is adopted adaptively and fit-for-purpose to the project context and organizational capacity.

REFERENCES

Agile Alliance. (2001). *Manifesto for agile software development*. https://agilemanifesto.org

Asmy, Y. Y., & Hasugian, L. P. (2021). Penilaian maturity level perangkat lunak menggunakan CMMI-DEV 1.3 pada aplikasi Manans MINT. *Jurnal Manajemen Informatika dan Komputerisasi Akuntansi (JAMIKA), 21*(1), 35–42. https://ojs.unikom.ac.id/index.php/jamika/article/download/5523/2678

Beck, K., Beedle, M., van Bennekum, A., Cockburn, A., Cunningham, W., Fowler, M., … Thomas, D. (2001). *Manifesto for agile software development*. Agile Alliance. https://agilemanifesto.org

Behl, A., & Behl, K. (2017). *Cyberwar: The next threat to national security and what to do about it*. Oxford University Press.

Boehm, B. W. (1981). *Software engineering economics*. Prentice Hall.

Boehm, B. W. (2000). *Software cost estimation with COCOMO II*. Prentice Hall.

Boehm, B. W., & Basili, V. R. (2001). Software defect reduction top 10 list. *Computer, 34*(1), 135–137. https://doi.org/10.1109/2.962984

Boehm, B. W., & Turner, R. (2004). *Balancing agility and discipline: A guide for the perplexed*. Addison-Wesley.

Chrissis, M. B., Konrad, M., & Shrum, S. (2011). *CMMI for development: Guidelines for process integration and product improvement* (3rd ed.). Addison-Wesley.

CMMI Institute. (2018). *CMMI® development v3.0: Improving processes for better performance*. ISACA/CMMI Institute.

CMMI Institute. (2018). *Standard CMMI appraisal method for process improvement (SCAMPI), version 1.3*. CMMI Institute.

CMMI Institute. (2023). *CMMI model version 3.0: Improving performance and outcomes*. CMMI Institute.

Creswell, J. W., & Creswell, J. D. (2018). *Research design: Qualitative, quantitative, and mixed methods approaches* (5th ed.). SAGE Publications.

Creswell, J. W., & Poth, C. N. (2018). *Qualitative inquiry and research design: Choosing among five approaches* (4th ed.). Sage Publications.

De Win, B., Scandariato, R., Buyens, K., Grégoire, J., & Joosen, W. (2009). On the secure software development process: CLASP, SDL and Touchpoints compared. *Information and Software Technology, 51*(7), 1152–1171. https://doi.org/10.1016/j.infsof.2008.01.010

Gibson, D., Goldenson, D., & Kost, K. (2006). *Performance results of CMMI-based process improvement* (CMU/SEI-2006-TR-004). Software Engineering Institute. https://doi.org/10.1184/R1/6582011.v1

Goldenson, D. R., & Gibson, D. L. (2003). *Demonstrating the impact and benefits of CMMI* (CMU/SEI-2003-SR-009). Software Engineering Institute.

Goldratt, E. M. (1997). *Critical chain*. North River Press.

Hillson, D. (2017). *The risk management handbook: A practical guide to managing the multiple dimensions of risk*. Kogan Page.

Humble, J., & Farley, D. (2010). *Continuous delivery: Reliable software releases through build, test, and deployment automation*. Addison-Wesley.

IBM Security. (2022). *Cost of a data breach report 2022*. IBM Corporation.

ISACA. (2023). *CMMI V3: Appraisal method and practices*. ISACA.

ISACA. (2023). *CMMI V3: Building capability and improving performance*. ISACA.

ISACA. (2023). *CMMI V3: Model overview*. ISACA.

Jiang, J. J., Klein, G., & Chen, H. G. (2001). The relative influence of IS project implementation policies and project leadership on eventual outcomes. *Project Management Journal, 32*(3), 49–55. https://doi.org/10.1177/875697280103200308

Kerzner, H. (2017). *Project management: A systems approach to planning, scheduling, and controlling* (12th ed.). Wiley.

Lincoln, Y. S., & Guba, E. G. (1985). *Naturalistic inquiry*. Sage Publications.

Maxwell, J. A. (2013). *Qualitative research design: An interactive approach* (3rd ed.). SAGE Publications.

McConnell, S. (2004). *Code complete: A practical handbook of software construction* (2nd ed.). Microsoft Press.

McGraw, G. (2006). *Software security: Building security in*. Addison-Wesley Professional.

Microsoft. (2018). *Security development lifecycle (SDL) process guidance*. Microsoft Corporation.

Miles, M. B., Huberman, A. M., & Saldaña, J. (2018). *Qualitative data analysis: A methods sourcebook* (4th ed.). SAGE Publications.

National Institute of Standards and Technology. (2020). *Systems security engineering: Considerations for a multidisciplinary approach* (SP 800-160 Vol. 1). U.S. Department of Commerce.

Niazi, M., Babar, M. A., & Verner, J. M. (2010). Software process improvement barriers: A cross-cultural comparison. *Information and Software Technology, 52*(11), 1204–1216. https://doi.org/10.1016/j.infsof.2010.06.005

Paulk, M. C., Curtis, B., Chrissis, M. B., & Weber, C. V. (1993). *Capability maturity model for software, version 1.1*. Software Engineering Institute.

Pohl, K., & Hof, H. (2015). *Requirements engineering: Fundamentals, principles, and techniques*. Springer.

Pressman, R. S., & Maxim, B. R. (2020). *Software engineering: A practitioner's approach* (9th ed.). McGraw-Hill Education.

Project Management Institute. (2021). *A guide to the project management body of knowledge (PMBOK® Guide)* (7th ed.). PMI.

Ramasubbu, N., Krishnan, M. S., & Kompalli, P. (2005). Leveraging global resources: A process maturity framework. *IEEE Software, 22*(3), 80–86. https://doi.org/10.1109/MS.2005.69

Royce, W. W. (1970). Managing the development of large software systems. *Proceedings of IEEE WESCON*, 1–9.

Schwalbe, K. (2019). *Information technology project management* (9th ed.). Cengage Learning.

Shelat, A., Kumar, S., & Ganesh, R. (2025). Assessing CMMI Level 3 adoption. *International Journal of Software Engineering and Applications, 14*(2), 45–56.

Shostack, A. (2014). *Threat modeling: Designing for security*. Wiley.

Silva, F. S. C., Soares, F. S. F., França, A. C. C., & Monteiro, C. V. F. (2015). Using CMMI together with agile software development. *Information and Software Technology, 58*, 20–43. https://doi.org/10.1016/j.infsof.2014.09.002

Sommerville, I. (2016). *Software engineering* (10th ed.). Pearson Education.

Souppaya, M., Scarfone, K., & Dodson, D. (2022). *Secure software development framework (SSDF) version 1.1* (NIST SP 800-218). NIST. https://doi.org/10.6028/NIST.SP.800-218

Staples, M., & Niazi, M. (2008). Organizational motivations for adopting CMM-based SPI. *Information and Software Technology, 50*(7–8), 605–620. https://doi.org/10.1016/j.infsof.2007.07.003

Tsai, W. (2021). The impact of project teams on CMMI implementations. *Systems Research and Behavioral Science, 34*(2), 239–252. https://doi.org/10.1007/s11213-020-09531-y

Turner, J. R. (2016). *Gower handbook of project management* (5th ed.). Routledge.

Viega, J., & McGraw, G. (2011). *Building secure software: How to avoid security problems the right way*. Addison-Wesley Professional.

Wibisono, M. I. (2020). Penilaian kematangan proses pengembangan perangkat lunak. *Jurnal Teknologi Informasi dan Ilmu Komputer, 7*(5), 975–984. https://media.neliti.com/media/publications/338058

Yilmaz, M., & Ozcan, T. (2023). Mapping CMMI-DEV v2 with Scrum. *SSRN Electronic Journal*. https://doi.org/10.2139/ssrn.4310530