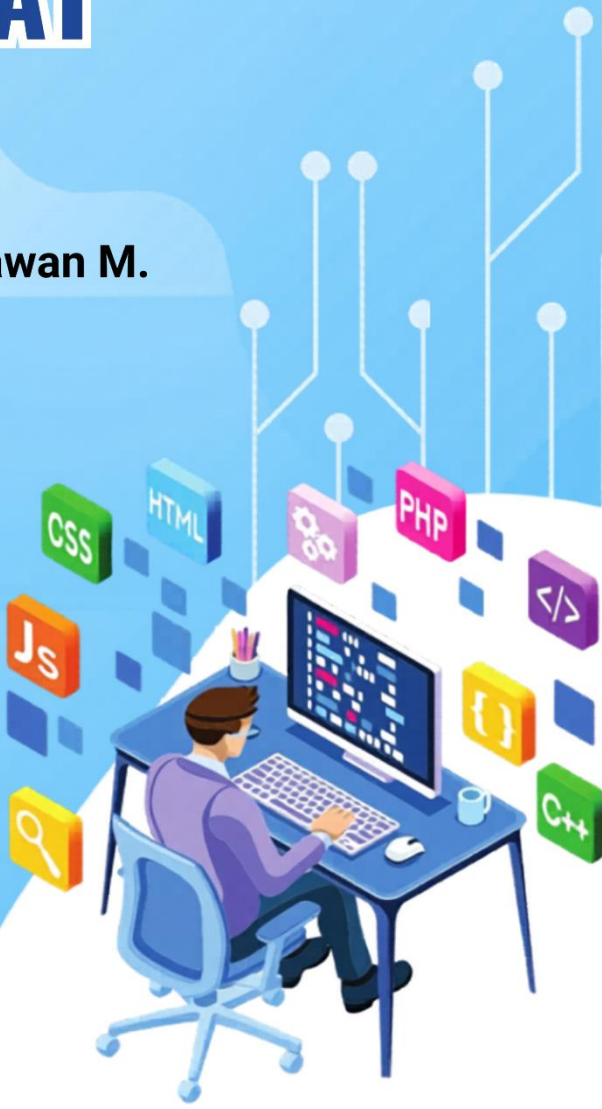
UNIVERSITAS BUDDHI DHARMA

REKAYASA PERANGKAT LUNAK

Ardiane Rossi Kurniawan M.



REKAYASA PERANGKAT LUNAK

Ardiane Rossi Kurniawan Maranto

Universitas Buddhi Dharma

REKAYASA PERANGKAT LUNAK

ISBN:

Penulis:
Ardiane Rossi Kurniawan Maranto, M.Kom.

Editor:
Aditiya Hermawan, M.Kom., M.M.

Layout
Lidya Lunardi, S.Kom.

Desain sampul:
Rio Permana, S.Kom.

Penerbit:
UNIVERSITAS BUDDHI DHARMA
Gedung Vipassi Lt. 1 Universitas Buddhi Dharma
Jl. Imam Bonjol No 41 Karawaci Ilir, Tangerang 15115
Telp. (021) 5517853
E-Mail: lp3m@buddhidharma.ac.id

Cetakan I, _____ 2026

KATA PENGANTAR

Puji syukur kehadirat Tuhan Yang Maha Esa atas rahmat dan karunia-Nya, sehingga buku referensi Rekayasa Perangkat Lunak ini dapat terselesaikan. Penulisan buku ini dilatarbelakangi oleh kebutuhan akan sumber belajar yang komprehensif dan relevan dengan perkembangan terkini di bidang rekayasa perangkat lunak, khususnya bagi mahasiswa yang mengambil mata kuliah Rekayasa Perangkat Lunak. Tujuan utama buku ini adalah untuk membekali pembaca dengan pemahaman mendalam mengenai konsep, metodologi, dan praktik terbaik dalam pengembangan perangkat lunak.

Ruang lingkup materi dalam buku ini mencakup berbagai aspek penting, mulai dari konsep dasar, model proses, metodologi Agile dan DevOps, analisis kebutuhan, perancangan sistem, manajemen proyek, pengujian, keamanan, hingga tren masa depan seperti AI dalam RPL dan komputasi awan. Saya berharap buku ini dapat menjadi panduan yang efektif bagi mahasiswa dalam memahami kompleksitas rekayasa perangkat lunak dan mempersiapkan mereka untuk tantangan di dunia industri.

Saya mengucapkan terima kasih yang sebesar-besarnya kepada semua pihak yang telah memberikan dukungan dan inspirasi selama proses penulisan buku ini. Semoga buku ini dapat memberikan kontribusi positif bagi kemajuan ilmu pengetahuan dan teknologi, serta bermanfaat bagi para pembaca dalam mengembangkan kompetensi di bidang rekayasa perangkat lunak.

DAFTAR ISI

DAFTAR PUSTAKA

Abdellatif, A., Badran, K., Costa, D. E., & Shihab, E. (2022). A Comparison of Natural Language Understanding Platforms for Chatbots in Software Engineering. *IEEE Transactions on Software Engineering*, *48*(8), 3087–3102. https://doi.org/10.1109/TSE.2021.3078384

Adger, W. N., Brown, K., & Hulme, M. (2005). Redefining global environmental change. *Global Environmental Change*, *15*(1), 1–4. https://doi.org/10.1016/j.gloenvcha.2004.12.002

Alsaqqa, S., Sawalha, S., & Abdel-Nabi, H. (2020). Agile Software Development: Methodologies and Trends. *International Journal of Interactive Mobile Technologies (IJIM)*, *14*(11), 246. https://doi.org/10.3991/ijim.v14i11.13269

Amershi, S., Begel, A., Bird, C., DeLine, R., Gall, H., Kamar, E., Nagappan, N., Nushi, B., & Zimmermann, T. (2019). Software Engineering for Machine Learning: A Case Study. *2019 IEEE/ACM 41st International Conference on Software Engineering: Software Engineering in Practice (ICSE-SEIP)*, 291–300. https://doi.org/10.1109/ICSE-SEIP.2019.00042

Bass, L., Clements, P., & Kazman, R. (2021). *Software Architecture in Practice*. Pearson Education.

Bergin, S., & Keating, J. (2003). A case study on the adaptive maintenance of an Internet application. *Journal of Software Maintenance and Evolution: Research and Practice*, *15*(4), 245–264. https://doi.org/10.1002/smr.275

Blischak, J. D., Davenport, E. R., & Wilson, G. (2016). A Quick Introduction to Version Control with Git and GitHub. *PLOS Computational Biology*, *12*(1), e1004668. https://doi.org/10.1371/journal.pcbi.1004668

Booch, G. (2005). *The Unified Modeling Language User Guide*. Pearson Education.

Brooks. (1987). No Silver Bullet Essence and Accidents of Software Engineering. *Computer*, *20*(4), 10–19. https://doi.org/10.1109/MC.1987.1663532

Burns, B., Grant, B., Oppenheimer, D., Brewer, E., & Wilkes, J. (2016).

Borg, Omega, and Kubernetes. *Communications of the ACM*, *59*(5), 50–57. https://doi.org/10.1145/2890784

Chandramouli, R., & Butcher, Z. (2025). *Guidelines for API protection for cloud-native systems*. https://doi.org/10.6028/NIST.SP.800-228

Chapin, N., Hale, J. E., Khan, K. M., Ramil, J. F., & Tan, W. (2001). Types of software evolution and software maintenance. *Journal of Software Maintenance and Evolution: Research and Practice*, *13*(1), 3–30. https://doi.org/10.1002/smr.220

Dongmo, C. (2024). A Review of Non-Functional Requirements Analysis Throughout the SDLC. *Computers*, *13*(12), 308. https://doi.org/10.3390/computers13120308

dos Santos, P. S. M., Beltrão, A. C., de Souza, B. P., & Travassos, G. H. (2018). On the benefits and challenges of using kanban in software engineering: a structured synthesis study. *Journal of Software Engineering Research and Development*, *6*(1), 13. https://doi.org/10.1186/s40411-018-0057-1

Emam, K. El, & Koru, A. G. (2008). A Replicated Survey of IT Software Project Failures. *IEEE Software*, *25*(5), 84–90. https://doi.org/10.1109/MS.2008.107

Fagarasan, C., Popa, O., Pisla, A., & Cristea, C. (2021). Agile, waterfall and iterative approach in information technology projects. *IOP Conference Series: Materials Science and Engineering*, *1169*(1), 012025. https://doi.org/10.1088/1757-899X/1169/1/012025

Fan, Q., Yu, Y., Wang, T., Yin, G., & Wang, H. (2021). Why API documentation is insufficient for developers: an empirical study. *Science China Information Sciences*, *64*(1), 119102. https://doi.org/10.1007/s11432-019-9880-8

Ferreira Martins, H., Carvalho de Oliveira Junior, A., Dias Canedo, E., Dias Kosloski, R. A., Ávila Paldês, R., & Costa Oliveira, E. (2019). Design Thinking: Challenges for Software Requirements Elicitation. *Information*, *10*(12), 371. https://doi.org/10.3390/info10120371

Ferris, C., & Farrell, J. (2003). What are Web services? *Communications of the ACM*, *46*(6), 31. https://doi.org/10.1145/777313.777335

Fielding, R. T., & Taylor, R. N. (2002). Principled design of the modern Web architecture. *ACM Transactions on Internet Technology*, *2*(2), 115–150. https://doi.org/10.1145/514183.514185

Fitzgerald, B., & Stol, K.-J. (2017). Continuous software engineering: A roadmap and agenda. *Journal of Systems and Software*, *123*, 176–189. https://doi.org/10.1016/j.jss.2015.06.063

Folke, C., Hahn, T., Olsson, P., & Norberg, J. (2005). ADAPTIVE GOVERNANCE OF SOCIAL-ECOLOGICAL SYSTEMS. *Annual Review of Environment and Resources*, *30*(1), 441–473. https://doi.org/10.1146/annurev.energy.30.050504.144511

Gamma, E., Helm, R., Johnson, R., & Vlissides, J. (1994). *Design Patterns: Elements of Reusable Object-Oriented Software*. Addison-Wesley Professional.

Garousi, G., Garousi, V., Moussavi, M., Ruhe, G., & Smith, B. (2013). Evaluating usage and quality of technical software documentation. *Proceedings of the 17th International Conference on Evaluation and Assessment in Software Engineering*, 24–35. https://doi.org/10.1145/2460999.2461003

Gousios, G., Pinzger, M., & Deursen, A. van. (2014). An exploratory study of the pull-based software development model. *Proceedings of the 36th International Conference on Software Engineering*, 345–355. https://doi.org/10.1145/2568225.2568260

Gupta, A., Jain, P., Khandelwal, S., & Nawal, M. (2026). *Comparative Analysis of Software Development Models: Evaluating Effectiveness Across the SDLC* (pp. 220–233). https://doi.org/10.1007/978-3-031-95540-2_20

Hassan, H. B., Barakat, S. A., & Sarhan, Q. I. (2021). Survey on serverless computing. *Journal of Cloud Computing*, *10*(1), 39. https://doi.org/10.1186/s13677-021-00253-7

IEEE. (2014). *IEEE Standard for Software Quality Assurance Processes*. IEEE. https://doi.org/10.1109/IEEESTD.2014.6835311

ISO/IEC/IEEE. (2021). *ISO/IEC/IEEE 24774:2021 Systems and software engineering — Life cycle management — Specification for process description*. IEEE.

Jobin, A., Ienca, M., & Vayena, E. (2019). The global landscape of AI

ethics guidelines. *Nature Machine Intelligence*, *1*(9), 389–399. https://doi.org/10.1038/s42256-019-0088-2

Keung, J. W., Mariani, L., Xiang, J., & Yu, X. (2023). ISSRE 2021 special section. *Information and Software Technology*, *159*, 107227. https://doi.org/10.1016/j.infsof.2023.107227

Kratzke, N. (2022). Cloud-Native Observability: The Many-Faceted Benefits of Structured and Unified Logging—A Multi-Case Study. *Future Internet*, *14*(10), 274. https://doi.org/10.3390/fi14100274

Larman, C., & Basili, V. R. (2003). Iterative and incremental developments. a brief history. *Computer*, *36*(6), 47–56. https://doi.org/10.1109/MC.2003.1204375

Lewis, J., & Fowler, M. (2014). *Microservices*. https://martinfowler.com/articles/microservices.html

Li, Z., Avgeriou, P., & Liang, P. (2015). A systematic mapping study on technical debt and its management. *Journal of Systems and Software*, *101*, 193–220. https://doi.org/10.1016/j.jss.2014.12.027

Masso, J., Pino, F. J., Pardo, C., García, F., & Piattini, M. (2020). Risk management in the software life cycle: A systematic literature review. *Computer Standards & Interfaces*, *71*, 103431. https://doi.org/10.1016/j.csi.2020.103431

McCabe, T. J. (1976). A Complexity Measure. *IEEE Transactions on Software Engineering*, *SE-2*(4), 308–320. https://doi.org/10.1109/TSE.1976.233837

McGraw, G. (2004). Software security. *IEEE Security & Privacy Magazine*, *2*(2), 80–83. https://doi.org/10.1109/MSECP.2004.1281254

Mell, P. M., & Grance, T. (2011). *The NIST definition of cloud computing*. https://doi.org/10.6028/NIST.SP.800-145

Meng, M., Steinhardt, S. M., & Schubert, A. (2020). Optimizing API Documentation. *Proceedings of the 38th ACM International Conference on Design of Communication*, 1–11. https://doi.org/10.1145/3380851.3416759

Mens, T., & Tourwe, T. (2004). A survey of software refactoring. *IEEE Transactions on Software Engineering*, *30*(2), 126–139. https://doi.org/10.1109/TSE.2004.1265817

Merkel, D. (2014). Docker: lightweight Linux containers for consistent development and deployment. *Linux J.*, *2014*(239).

Mesoudi, A. (2011). *Cultural Evolution*. University of Chicago Press. https://doi.org/10.7208/chicago/9780226520452.001.0001

Moradi Dakhel, A., Majdinasab, V., Nikanjam, A., Khomh, F., Desmarais, M. C., & Jiang, Z. M. (Jack). (2023). GitHub Copilot AI pair programmer: Asset or Liability? *Journal of Systems and Software*, *203*, 111734. https://doi.org/10.1016/j.jss.2023.111734

Murgia, A., Concas, G., Tonelli, R., Ortu, M., Demeyer, S., & Marchesi, M. (2014). On the influence of maintenance activity types on the issue resolution time. *Proceedings of the 10th International Conference on Predictive Models in Software Engineering*, 12–21. https://doi.org/10.1145/2639490.2639506

Myers, B. A., & Stylos, J. (2016). Improving API usability. *Communications of the ACM*, *59*(6), 62–69. https://doi.org/10.1145/2896587

Newman, S. (2015). *Building Microservices: Designing Fine-Grained Systems*. O'Reilly Media.

Niswar, M., Arisandy Safruddin, R., Bustamin, A., & Aswad, I. (2024). Performance evaluation of microservices communication with REST, GraphQL, and gRPC. *International Journal of Electronics and Telecommunications*, 429–436. https://doi.org/10.24425/ijet.2024.149562

Ostrom, E. (1990). *Governing the Commons*. Cambridge University Press. https://doi.org/10.1017/CBO9780511807763

Putnam, R. D. (2000). Bowling alone. *Proceedings of the 2000 ACM Conference on Computer Supported Cooperative Work*, 357. https://doi.org/10.1145/358916.361990

Quiña-Mera, A., Fernandez, P., García, J. M., & Ruiz-Cortés, A. (2023). GraphQL: A Systematic Mapping Study. *ACM Computing Surveys*, *55*(10), 1–35. https://doi.org/10.1145/3561818

Rafique, Y., & Misic, V. B. (2013). The Effects of Test-Driven Development on External Quality and Productivity: A Meta-Analysis. *IEEE Transactions on Software Engineering*, *39*(6), 835–856. https://doi.org/10.1109/TSE.2012.28

Ramadha, F. N., Wahyuni, E. D., & Vannes, D. D. (2024). SDLC Big Bang

dan Waterfall : Perbandingan Pendekatan dalam Pengembangan Perangkat Lunak. *NUANSA INFORMATIKA*, *18*(2), 41–45. https://doi.org/10.25134/ilkom.v18i2.158

Shepperd, M., & Jorgensen, M. (2007). A Systematic Review of Software Development Cost Estimation Studies. *IEEE Transactions on Software Engineering*, *33*(01), 33–53. https://doi.org/10.1109/TSE.2007.3

Tahir, T., Jahankhani, H., Tasleem, K., & Hassan, B. (2025). Cross-Project Multiclass Classification of EARS-Based Functional Requirements Utilizing Natural Language Processing, Machine Learning, and Deep Learning. *Systems*, *13*(7), 567. https://doi.org/10.3390/systems13070567

Trautsch, A., Herbold, S., & Grabowski, J. (2020). Static source code metrics and static analysis warnings for fine-grained just-in-time defect prediction. *2020 IEEE International Conference on Software Maintenance and Evolution (ICSME)*, 127–138. https://doi.org/10.1109/ICSME46990.2020.00022

Wang, Y., Mäntylä, M. V., Liu, Z., & Markkula, J. (2022). Test automation maturity improves product quality—Quantitative study of open source projects using continuous integration. *Journal of Systems and Software*, *188*, 111259. https://doi.org/10.1016/j.jss.2022.111259

Wen, M., Chen, J., Wu, R., Hao, D., & Cheung, S.-C. (2018). Context-aware patch generation for better automated program repair. *Proceedings of the 40th International Conference on Software Engineering*, 1–11. https://doi.org/10.1145/3180155.3180233

Wonohardjo, E. P., Sunaryo, R. F., & Sudiyono, Y. (2019). A Systematic Review of SCRUM in Software Development. *JOIV : International Journal on Informatics Visualization*, *3*(2), 108–112. https://doi.org/10.30630/joiv.3.2.167

Xiao, L., Zhao, G., Wang, X., Li, K., Lim, E., Wei, C., Yu, T., & Wang, X. (2024). An empirical study on the usage of mocking frameworks in Apache software foundation. *Empirical Software Engineering*, *29*(2), 39. https://doi.org/10.1007/s10664-023-10410-y

Yalçıner, A., Dikici, A., & Gökalp, E. (2024). *Data-Driven Software Engineering: A Systematic Literature Review* (pp. 19–32). https://doi.org/10.1007/978-3-031-71139-8_2

Zhao, Y., Serebrenik, A., Zhou, Y., Filkov, V., & Vasilescu, B. (2017). The impact of continuous integration on other software development practices: A large-scale empirical study. *2017 32nd IEEE/ACM International Conference on Automated Software Engineering (ASE)*, 60–71. https://doi.org/10.1109/ASE.2017.8115619

Zowghi, D., & Coulin, C. (2005). Requirements Elicitation: A Survey of Techniques, Approaches, and Tools. In *Engineering and Managing Software Requirements* (pp. 19–46). Springer-Verlag. https://doi.org/10.1007/3-540-28244-0_2

GLOSARIUM

Adaptive Maintenance	:	Pemeliharaan perangkat lunak untuk menyesuaikan sistem dengan perubahan lingkungan atau teknologi.
Agile Methodology	:	Metode pengembangan perangkat lunak yang fleksibel, iteratif, dan kolaboratif.
Automated Testing	:	Pengujian perangkat lunak secara otomatis menggunakan tools tertentu.
Backend	:	Bagian sistem yang menangani logika bisnis dan pengolahan data.
Branching Strategy	:	Strategi pengelolaan cabang kode dalam version control.
Bug	:	Kesalahan atau cacat pada perangkat lunak.
Chatbot	:	Program yang dapat berinteraksi dengan pengguna melalui percakapan otomatis.
CI/CD	:	Praktik integrasi dan distribusi kode secara otomatis dan berkelanjutan.
Cloud Computing	:	Penyediaan layanan komputasi melalui internet.
Code Review	:	Pemeriksaan kode program oleh pengembang lain untuk menjaga kualitas kode.
Containerization	:	Teknik pengemasan aplikasi beserta dependensinya agar berjalan konsisten.
Corrective Maintenance	:	Pemeliharaan untuk memperbaiki bug atau kesalahan sistem.
Database	:	Kumpulan data yang tersimpan dan terorganisasi secara sistematis.

Debugging	:	Proses menemukan dan memperbaiki kesalahan program.
DevOps	:	Kolaborasi antara tim development dan operations untuk mempercepat pengembangan perangkat lunak.
Docker	:	Platform containerization untuk menjalankan aplikasi dalam container.
Functional Requirement	:	Kebutuhan sistem yang menjelaskan fungsi yang harus disediakan.
Git	:	Sistem version control untuk mengelola perubahan kode program.
Incremental Model	:	Model pengembangan perangkat lunak secara bertahap.
Integration Testing	:	Pengujian gabungan antar modul sistem.
Iteration	:	Siklus pengembangan berulang untuk penyempurnaan sistem.
Kanban	:	Metode Agile berbasis visual untuk mengelola pekerjaan.
Kubernetes	:	Platform untuk mengelola container secara otomatis.
Machine Learning Pipeline	:	Tahapan pengembangan sistem machine learning.
Maintenance	:	Aktivitas pemeliharaan perangkat lunak setelah digunakan.
Microservices	:	Arsitektur aplikasi berbasis layanan kecil yang independen.

Model Spiral	:	Model pengembangan perangkat lunak berbasis iterasi dan manajemen risiko.
Non-Functional Requirement	:	Kebutuhan terkait kualitas sistem seperti keamanan dan performa.
Observability	:	Kemampuan memantau kondisi dan performa sistem.
Perfective Maintenance	:	Pemeliharaan untuk meningkatkan performa atau fitur sistem.
Quality Assurance (QA)	:	Proses memastikan kualitas perangkat lunak sesuai standar.
Refactoring	:	Perbaikan struktur kode tanpa mengubah fungsi utama program.
Repository	:	Tempat penyimpanan kode sumber proyek.
Requirement Elicitation	:	Proses pengumpulan kebutuhan sistem dari pengguna.
Risk Management	:	Proses identifikasi dan pengelolaan risiko proyek.
Scrum	:	Framework Agile berbasis sprint dan kolaborasi tim.
Secure Coding	:	Praktik penulisan kode yang aman dari ancaman keamanan.
Serverless	:	Model cloud computing tanpa pengelolaan server langsung.
Software Crisis	:	Krisis kegagalan proyek perangkat lunak akibat kompleksitas dan lemahnya manajemen.
Software Engineer	:	Profesional yang merancang dan mengembangkan perangkat lunak.

Software Metrics	:	Ukuran kuantitatif untuk menilai kualitas perangkat lunak.
Software Process	:	Serangkaian aktivitas terstruktur dalam pengembangan perangkat lunak.
System Testing	:	Pengujian keseluruhan sistem untuk memastikan semua fungsi berjalan baik.
Technical Debt	:	Konsekuensi jangka panjang akibat keputusan pengembangan yang kurang optimal.
Test-Driven Development (TDD)	:	Metode pengembangan dengan menulis pengujian sebelum kode program.
Testing	:	Proses pengujian perangkat lunak untuk menemukan kesalahan.
UML	:	Bahasa pemodelan standar untuk desain sistem perangkat lunak.
Unit Testing	:	Pengujian unit terkecil dalam program secara terpisah.
Use Case Diagram	:	Diagram UML yang menggambarkan interaksi pengguna dengan sistem.
Validation	:	Proses memastikan sistem sesuai kebutuhan pengguna.
Version Control	:	Sistem pengelolaan perubahan kode program.
Waterfall Model	:	Model pengembangan perangkat lunak yang berjalan secara berurutan.

INDEKS

E

F

G

I

K

M

O

HASIL SCANNING SIMILARITY

Berdasarkan penilaian teks pada Buku Ajar yang diajukan di bawah ini:

Penulis	:	Ardiane Rossi Kurniawan Maranto, M.Kom.
Fakultas	:	Sains dan Teknologi/Sistem Informasi
Judul	:	Rekayasa Perangkat Lunak
Tipe	:	Buku

Turnitin mencatat adanya tingkat kesamaan antara dokumen ini dengan dokumen yang ada dalam aplikasi seperti yang tertera di bawah ini:

Word Count	:	71,325
Character Count	:	474,969
Similarity Index	**:**	**14%**
Internet Source	:	13%
Publication	:	4%
Student paper	:	5%
Exclude Quotes	:	Off
Exclude Bibliography	:	On
Exclude Matches	:	Off

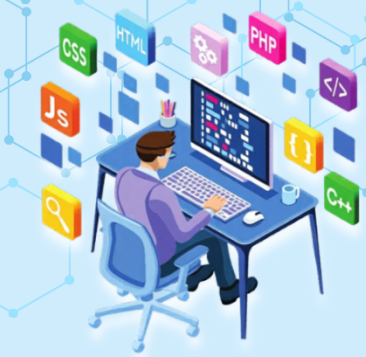
BIOGRAFI PENULIS

Ardiane Rossi Kurniawan Maranto, M.Kom. adalah akademisi dan praktisi di bidang ilmu komputer, khususnya pada kajian rekayasa perangkat lunak, pengembangan sistem informasi, pemrograman, dan metodologi pengembangan perangkat lunak. Ia memiliki perhatian pada penerapan prinsip-prinsip rekayasa perangkat lunak dalam membangun sistem yang terstruktur, efisien, andal, mudah dipelihara, serta sesuai dengan kebutuhan pengguna.

Melalui buku Rekayasa Perangkat Lunak, penulis berupaya menyajikan konsep-konsep dasar dan terapan dalam pengembangan perangkat lunak secara sistematis, mulai dari analisis kebutuhan, perancangan sistem, implementasi, pengujian, pemeliharaan, hingga manajemen proyek perangkat lunak. Buku ini disusun untuk membantu mahasiswa, dosen, dan praktisi memahami proses pengembangan perangkat lunak secara komprehensif, baik dari sisi teori maupun praktik.

Dengan latar belakang keilmuan di bidang komputer, penulis berharap buku ini dapat menjadi referensi pembelajaran yang relevan dalam meningkatkan pemahaman mengenai pentingnya rekayasa perangkat lunak dalam menghasilkan produk perangkat lunak yang berkualitas, berkelanjutan, dan mampu menjawab kebutuhan dunia industri maupun pendidikan.

REKAYASA PERANGKAT LUNAK



Buku ajar Rekayasa Perangkat Lunak ini membahas konsep, proses, dan praktik pengembangan perangkat lunak secara lengkap, mulai dari dasar RPL, analisis kebutuhan, perancangan sistem, pengujian, hingga pemeliharaan perangkat lunak. Materi disusun sistematis dan dilengkapi latihan untuk membantu mahasiswa memahami penerapan RPL dalam dunia industri.

Selain membahas metode modern seperti Agile, DevOps, CI/CD, dan cloud computing, buku ini juga mengangkat teknologi terkini seperti containerization dan kecerdasan buatan dalam pengembangan perangkat lunak. Dengan bahasa yang ringkas dan aplikatif, buku ini menjadi panduan pembelajaran yang relevan bagi mahasiswa dan calon praktisi teknologi.

UNIVERSITAS BUDDHI DHARMA
Gedung Vipassi Lt. 1
Jl. Imam Bonjol No. 41 Karawaci Ilir
Tangerang 15115
Telp. (021) 5517853
E-mail: lp3m@buddhidharma.ac.id

0 123456 787902

REKAYASA PERANGKAT LUNAK

Ardiane Rossi K. M.

KETEBALAN BUKU = 1,5 CM

REKAYASA PERANGKAT LUNAK

Buku ajar Rekayasa Perangkat Lunak ini membahas konsep, proses, dan praktik pengembangan perangkat lunak secara lengkap, mulai dari dasar RPL, analisis kebutuhan, perancangan sistem, pengujian, hingga pemeliharaan perangkat lunak. Materi disusun sistematis dan dilengkapi latihan untuk membantu mahasiswa memahami penerapan RPL dalam dunia industri.

Selain membahas metode modern seperti Agile, DevOps, CI/CD, dan cloud computing, buku ini juga mengangkat teknologi terkini seperti containerization dan kecerdasan buatan dalam pengembangan perangkat lunak. Dengan bahasa yang ringkas dan aplikatif, buku ini menjadi panduan pembelajaran yang relevan bagi mahasiswa dan calon praktisi teknologi.

UNIVERSITAS BUDDHI DHARMA
Gedung Vipassi Lt. 1
Jl. Imam Bonjol No. 41 Karawaci Ilir
Tangerang 15115
Telp. (021) 5517853
E-mail: lp3m@buddhidharma.ac.id

0 123456 787902

REKAYASA PERANGKAT LUNAK

Ardiane Rossi K. M.

UNIVERSITAS BUDDHI DHARMA

REKAYASA PERANGKAT LUNAK

Ardiane Rossi Kurniawan M.