

# REKAYASA PERANGKAT LUNAK

Ardiane Rossi Kurniawan M.



# REKAYASA PERANGKAT LUNAK

Ardiane Rossi Kurniawan Maranto



Universitas Buddhi Dharma

# REKAYASA PERANGKAT LUNAK

ISBN:

Hak Cipta 2026 pada Penulis

Hak penerbitan pada UNIVERSITAS BUDDHI DHARMA. Bagi mereka yang ingin memperbanyak sebagian isi buku ini dalam bentuk atau cara apapun harus mendapatkan izin tertulis dari penulis dan penerbit UNIVERSITAS BUDDHI DHARMA.

**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

Hak Cipta dilindungi Undang-undang

All Right Reserved

Cetakan I, \_\_\_\_\_ 2026

## KATA PENGANTAR

Puji syukur kehadiran 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

KATA PENGANTAR.....	3
DAFTAR ISI.....	4
BAB 1 KONSEP DASAR REKAYASA PERANGKAT LUNAK.....	9
A. Tujuan Pembelajaran.....	9
B. Pendahuluan.....	10
C. Definisi dan Ruang Lingkup .....	11
D. Karakteristik Perangkat Lunak.....	13
E. Software Crisis .....	15
F. <i>Software Process</i> .....	16
G. Peran <i>Software Engineer</i> .....	18
H. Rangkuman .....	19
I. Latihan Mahasiswa .....	20
BAB 2 MODEL PROSES PERANGKAT LUNAK.....	28
A. Tujuan Pembelajaran.....	28
B. Pendahuluan.....	28
C. <i>Waterfall Model</i> .....	30
D. <i>Spiral Model</i> .....	33
E. <i>Prototyping Model</i> .....	35
F. <i>Incremental Model</i> .....	37
G. <i>Agile Methodology</i> .....	39
H. Rangkuman .....	42
I. Latihan Mahasiswa .....	43
BAB 3 AGILE DAN DEVOPS.....	51
A. Tujuan Pembelajaran.....	51
B. Pendahuluan.....	52
C. <i>Scrum Framework</i> .....	54
D. Kanban .....	57
E. DevOps Culture.....	59
F. CI/CD.....	60

G.	Rangkuman .....	62
H.	Latihan Mahasiswa .....	64
BAB 4	ANALISIS KEBUTUHAN.....	79
A.	Tujuan Pembelajaran.....	79
B.	Pendahuluan.....	79
C.	<i>Requirement Elicitation</i> .....	81
D.	<i>Functional Requirement</i> .....	83
E.	Non-Functional Requirement.....	85
F.	<i>Use Case Modeling</i> .....	86
G.	<i>Requirement Validation</i> .....	88
H.	Rangkuman .....	90
I.	Latihan Mahasiswa .....	92
BAB 5	PERANCANGAN SISTEM.....	99
A.	Tujuan Pembelajaran.....	99
B.	Pendahuluan.....	100
C.	Desain Arsitektur.....	102
D.	UML Diagram.....	103
E.	<i>Design Pattern</i> .....	105
F.	<i>Microservices Architecture</i> .....	106
G.	<i>API Design</i> .....	108
H.	Rangkuman .....	112
I.	Latihan Mahasiswa .....	114
BAB 6	MANAJEMEN PROYEK PERANGKAT LUNAK.....	117
A.	Tujuan Pembelajaran.....	117
B.	Pendahuluan.....	117
C.	Project Planning.....	119
D.	Estimasi Biaya.....	123
E.	Risk Management.....	126
F.	<i>Quality Assurance</i> .....	129
G.	Software Metrics.....	131
H.	Rangkuman .....	134

I.	Latihan Mahasiswa .....	136
<b>BAB 7</b>	<b>PENGUJIAN PERANGKAT LUNAK .....</b>	<b>139</b>
A.	Tujuan Pembelajaran .....	139
B.	Pendahuluan .....	139
C.	<i>Unit Testing</i> .....	141
D.	Integration Testing .....	143
E.	<i>System Testing</i> .....	145
F.	<i>Automated Testing</i> .....	148
G.	<i>Test-Driven Development (TDD)</i> .....	151
H.	Rangkuman .....	154
I.	Latihan Mahasiswa .....	156
<b>BAB 8</b>	<b>KEAMANAN PERANGKAT LUNAK .....</b>	<b>160</b>
A.	Tujuan Pembelajaran .....	160
B.	Pendahuluan .....	161
C.	<i>Secure Coding</i> .....	162
D.	Praktik <i>Secure Coding</i> .....	163
E.	Rangkuman .....	164
F.	Latihan Mahasiswa .....	166
<b>BAB 9</b>	<b>VERSION CONTROL DAN KOLABORASI .....</b>	<b>168</b>
A.	Tujuan Pembelajaran .....	168
B.	Pendahuluan .....	169
C.	Git dan Repository .....	170
D.	<i>Branching Strategy</i> .....	173
E.	<i>Code Review</i> .....	176
F.	<i>Continuous Integration</i> .....	178
G.	Documentation .....	180
H.	Rangkuman .....	183
I.	Latihan Mahasiswa .....	184
<b>BAB 10</b>	<b>PEMELIHARAAN DAN EVOLUSI .....</b>	<b>187</b>
A.	Tujuan Pembelajaran .....	187
B.	Pendahuluan .....	187

C.	<i>Corrective Maintenance</i> .....	189
D.	<i>Adaptive Maintenance</i> .....	191
E.	<i>Perfective Maintenance</i> .....	194
F.	<i>Refactoring</i> .....	196
G.	<i>Technical Debt</i> .....	198
H.	Rangkuman .....	201
I.	Latihan Mahasiswa .....	203
BAB 11 CLOUD DAN CONTAINERIZATION .....		205
A.	Tujuan Pembelajaran .....	205
B.	Pendahuluan .....	205
C.	<i>Cloud Computing</i> .....	207
D.	Docker .....	211
E.	<i>Kubernetes</i> .....	213
F.	<i>Serverless</i> .....	217
G.	<i>Observability</i> .....	219
H.	Rangkuman .....	223
I.	Latihan Mahasiswa .....	225
BAB 12 ARTIFICIAL INTELLIGENCE DALAM RPL .....		228
A.	Tujuan Pembelajaran .....	228
B.	Pendahuluan .....	228
C.	<i>AI-assisted Coding</i> .....	230
D.	<i>Machine Learning Pipeline</i> .....	232
E.	<i>Software Analytics</i> .....	235
F.	<i>Chatbot Development</i> .....	238
G.	<i>Ethical AI</i> .....	242
H.	Rangkuman .....	245
I.	Latihan Mahasiswa .....	246
DAFTAR PUSTAKA .....		249
GLOSARIUM .....		256
INDEKS .....		260
HASIL SCANNING SIMILARITY .....		264



## 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](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>
- Ramadhya, 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>
- Yalçın, 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](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](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.

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

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

## INDEKS

---

---

---

### A

*Agile* · 17, 20, 21, 24, 25, 27, 28, 30, 39, 40, 41, 42, 43, 44, 45, 47, 48, 52, 53, 54, 55, 56, 58, 63, 66, 72, 74, 151  
*Analisis* · 12, 18, 22, 27, 31, 34, 37, 44, 50, 82, 108, 131, 242  
*API* · 78, 102, 104, 110, 111, 112, 113, 114, 115, 116, 117, 118, 119, 144, 152, 153, 158, 161, 184, 186, 188, 194, 196, 197, 219, 223, 246  
*Aplikasi* · 25, 46, 47, 70, 71, 217, 242  
*Arsitektur* · 39, 41, 103, 104, 105, 109, 111, 116, 219, 220, 229, 244, 251  
*Assurance* · 120, 121, 132, 133, 139

---

### B

*Branch* · 174, 176, 177

---

### C

*Chatbot* · 243, 244, 245, 246, 247, 251  
*Cloud* · 213, 215, 223, 226, 228, 229  
*Code* · 136, 179, 180, 187  
*Coding* · 165, 166, 235, 236, 250  
*Container* · 211, 216, 218, 220, 229  
*Continuous* · 52, 54, 61, 62, 64, 66, 71, 145, 151, 154, 161, 171, 173, 177, 178, 181, 182, 186, 187, 218, 241

---

### D

*Deployment* · 38, 52, 54, 62, 64, 107, 110, 215, 218, 220, 229, 241  
*Design* · 12, 22, 27, 31, 35, 37, 104, 108, 109, 111, 116, 117  
*Development* · 12, 16, 24, 34, 56, 59, 63, 72, 74, 85, 122, 142, 143, 144, 154, 155,

156, 158, 162, 163, 165,  
236, 243, 246, 250, 251, 253  
*DevOps* · 52, 53, 54, 60, 61, 64,  
65, 69, 70, 72, 73, 77, 79,  
104, 151, 182, 184  
*Diagram* · 90, 94, 99, 106, 107,  
116  
*Docker* · 70, 78, 210, 211, 216,  
217, 218, 220, 228, 229

---

## **E**

*Engineer* · 9, 11, 18, 19, 20, 21,  
23, 25, 26, 27

---

## **F**

*Framework* · 55, 63, 246

---

## **G**

*Git* · 18, 62, 171, 172, 173, 174,  
175, 176, 177, 178, 186, 241

---

## **I**

*Interface* · 102, 104, 111, 117,  
118, 184, 238, 239, 245

---

## **K**

*Kanban* · 17, 39, 52, 53, 58, 59,  
64, 65, 66, 67

*Kubernetes* · 70, 78, 210, 212,  
218, 219, 220, 221, 222,  
228, 229

---

## **M**

*Machine Learning* · 236, 237,  
238, 240, 245, 251

*Maintenance* · 13, 23, 27, 32,  
193, 196, 197, 198, 199, 206

*Metrics* · 120, 122, 135, 136,  
139, 231

*Microservices* · 104, 109, 110,  
117

*Model* · 17, 21, 25, 26, 27, 28,  
29, 30, 31, 32, 33, 34, 35, 36,  
37, 38, 42, 43, 44, 45, 46, 47,  
48, 49, 50, 88, 89, 128, 213,  
214, 215, 216, 217, 222,  
224, 229, 239, 248

---

## **O**

*Observability* · 224, 225, 226,  
227, 228, 230, 231

---

**P**

Pengujian · 13, 18, 22, 27, 32, 38, 44, 62, 78, 134, 143, 146, 148, 149, 152, 158, 195, 205, 246  
Perancangan · 12, 18, 22  
Perangkat Lunak · 9, 10, 11, 12, 13, 14, 16, 19, 20, 21, 23, 25, 26, 28, 39, 45, 47, 49, 66, 82, 103, 116, 121, 122, 138, 139, 163, 192, 194, 211, 217, 237, 243, 247, 250  
*Performance* · 87, 137, 149, 153, 158, 161  
*Pipeline* · 63, 64, 70, 78, 79, 237, 238, 240, 251  
*Planning* · 34, 37, 48, 57, 63, 66, 72, 76, 121, 122, 123, 124, 138

---

**R**

*Refactoring* · 155, 192, 199, 200, 201, 202, 207  
*Requirement* · 82, 83, 85, 86, 87, 90, 91, 93, 94, 95, 100

---

**S**

*Scrum* · 17, 39, 52, 53, 55, 56, 57, 58, 63, 64, 65, 66, 67, 68, 72, 74, 75, 76, 77  
*Security* · 66, 71, 73, 79, 87, 149, 158, 163, 164  
*Serverless* · 222, 223, 224, 228, 229  
*Sistem* · 18, 26, 31, 44, 47, 49, 50, 87, 88, 98, 99, 106, 197, 236, 247, 248  
*Software* · 9, 10, 11, 12, 13, 15, 16, 18, 19, 20, 21, 22, 23, 24, 25, 26, 27, 31, 79, 120, 122, 135, 136, 139, 140, 143, 163, 165, 211, 213, 214, 228, 240, 241, 242, 243, 250, 251, 253  
*Sprint* · 48, 55, 56, 57, 63, 66, 67, 68, 69, 72, 75, 76, 77

---

**T**

*Testing* · 13, 22, 32, 38, 49, 58, 59, 71, 79, 142, 143, 144, 145, 146, 147, 148, 149, 150, 151, 152, 153, 156, 157, 158, 159, 160, 161

---

**V**

*Validation* · 17, 24, 34, 83, 90,  
94, 95, 100

---

**W**

*Waterfall* · 17, 20, 21, 24, 25,  
28, 29, 30, 31, 32, 33, 34, 37,  
42, 43, 44, 45, 46, 72, 74, 77

## 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
<b>Similarity Index</b>	:	<b>14%</b>
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





# REKAYASA PERANGKAT LUNAK



Ardiane Rossi K. M.




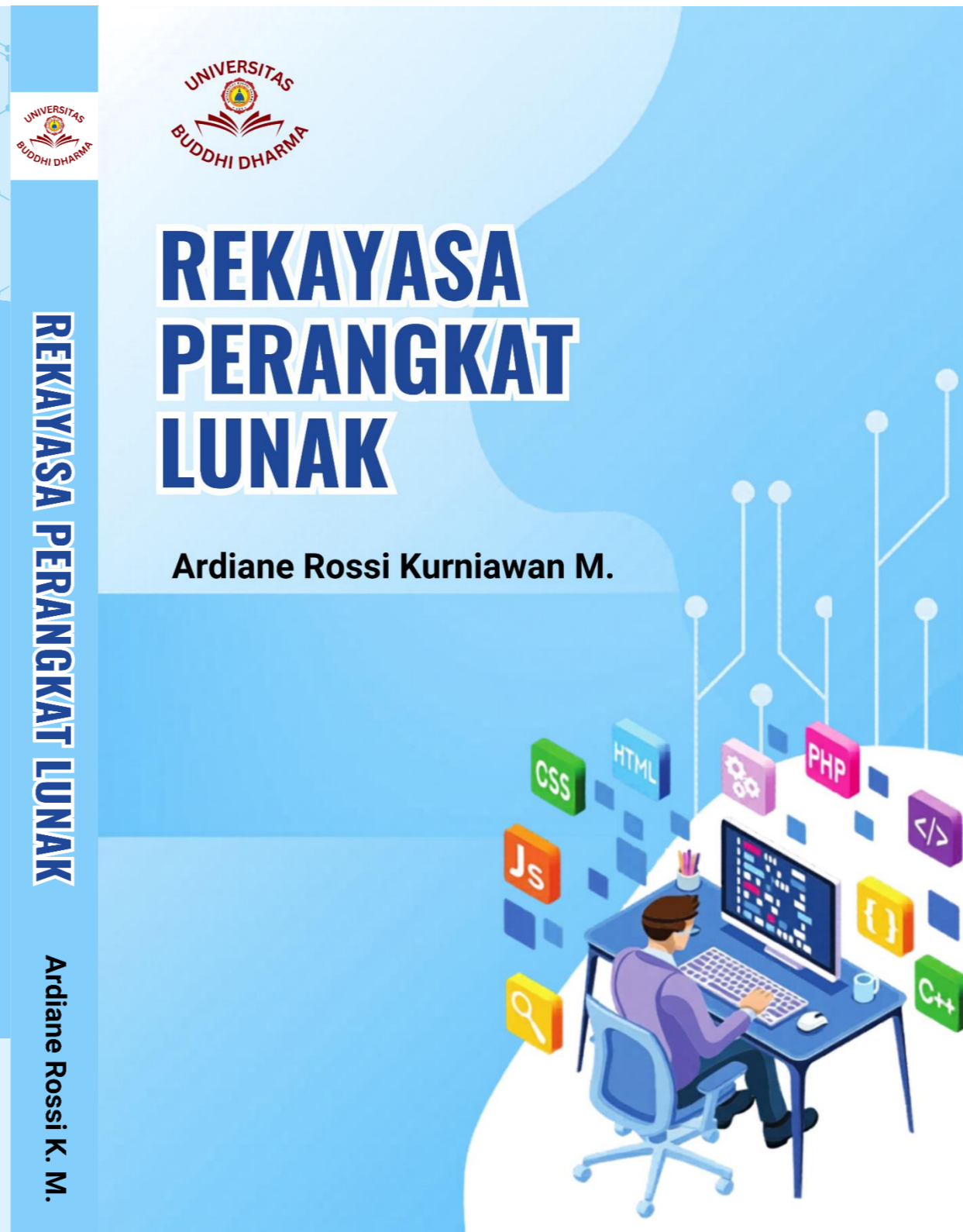

# REKAYASA PERANGKAT LUNAK

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



# REKAYASA PERANGKAT LUNAK

**Ardiane Rossi Kurniawan M.**

REKAYASA PERANGKAT LUNAK

Ardiane Rossi K. M.