METODE PENGEMBANGAN SISTEM

SECARA TRADISIONAL DAN TERBARU

Raditia Vindua, S.Si., M.Kom.

METODE PENGEMBANGAN SISTEM

SECARA TRADISIONAL DAN TERBARU

Pendekatan Pengembangan Perangkat Lunak adalah serangkaian metodologi, teknik, dan praktik yang digunakan oleh tim pengembang perangkat lunak untuk merancang, mengembangkan, dan memelihara perangkat lunak dengan efisien dan efektif. Pendekatan ini bertujuan untuk menghasilkan perangkat lunak yang berkualitas, sesuai dengan kebutuhan pengguna, dan dapat disesuaikan dengan perubahan kebutuhan bisnis dan teknologi.

Beberapa pendekatan yang umum digunakan dalam pengembangan perangkat lunak meliputi Waterfall yaitu Pendekatan waterfall adalah pendekatan linier yang terdiri dari serangkaian tahapan, mulai dari analisis kebutuhan hingga pengujian dan pemeliharaan. Setiap tahapannya dilakukan secara berurutan dan tidak ada tahapan mundur.

Agile yaitu Pendekatan agile merupakan pendekatan iteratif dan inkremental yang fokus pada kolaborasi tim, responsif terhadap perubahan, dan pengiriman iterasi perangkat lunak yang berkualitas dalam waktu singkat. Metodologi Scrum dan Kanban adalah contoh pendekatan agile yang populer.

Model-Driven Development (MDD) yaitu Pendekatan MDD menempatkan model sebagai elemen sentral dalam pengembangan perangkat lunak. Model digunakan untuk menggambarkan struktur, perilaku, dan interaksi antara komponen sistem, yang kemudian digunakan untuk menghasilkan kode sumber dan artefak lainnya. Prototyping yaitu Pendekatan prototyping melibatkan pembuatan model atau prototipe cepat dari sistem yang akan dikembangkan. Prototipe ini digunakan untuk mendapatkan umpan balik dari pengguna dan stakeholders sebelum mengembangkan versi final perangkat lunak.

DevOps yaitu Pendekatan DevOps menggabungkan pengembangan perangkat lunak (Dev) dengan operasi (Ops) dengan tujuan meningkatkan kolaborasi antara tim pengembangan dan tim operasi, serta mempercepat siklus pengembangan dan pengiriman perangkat lunak.

Setiap pendekatan memiliki kelebihan, tantangan, dan konteks penggunaan yang berbeda. Pemilihan pendekatan yang tepat tergantung pada kebutuhan proyek, karakteristik tim pengembangan, dan lingkungan kerja. Dengan memahami berbagai pendekatan pengembangan perangkat lunak, tim pengembangan dapat membuat keputusan yang tepat untuk mencapai tujuan pengembangan perangkat lunak secara efektif.

eureka
media aksara
Anggota IKAPI
No. 225/JTE/2021

0858 5343 1992
eurekamediaaksara@gmail.com
Jl. Banjaran RT.20 RW.10
Bojongsari - Purbalingga 53362

ISBN 978-623-120-502-5
9 786231 205025

METODE PENGEMBANGAN SISTEM SECARA TRADISIONAL DAN TERBARU

Raditia Vindua, S.Si., M.Kom.

PENERBIT CV.EUREKA MEDIA AKSARA

METODE PENGEMBANGAN SISTEM SECARA TRADISIONAL DAN TERBARU

Penulis : Raditia Vindua, S.Si., M.Kom.

Desain Sampul : Eri Setiawan

Tata Letak : Herlina Sukma

ISBN : 978-623-120-502-5

Diterbitkan oleh : **EUREKA MEDIA AKSARA, MARET 2024**
ANGGOTA IKAPI JAWA TENGAH
NO. 225/JTE/2021

Redaksi:
Jalan Banjaran, Desa Banjaran RT 20 RW 10 Kecamatan Bojongsari Kabupaten Purbalingga Telp. 0858-5343-1992

Surel : eurekamediaaksara@gmail.com

Cetakan Pertama : 2024

KATA PENGANTAR

Pengembangan perangkat lunak menjadi semakin penting di era di mana teknologi digital telah menyatu dalam hampir setiap aspek kehidupan kita. Dari aplikasi bisnis hingga perangkat lunak rumah tangga, perangkat lunak memainkan peran krusial dalam memfasilitasi berbagai aktivitas manusia. Namun, pengembangan perangkat lunak bukanlah tugas yang mudah. Proyek-proyek pengembangan sering kali kompleks, memerlukan koordinasi yang baik antara tim pengembangan, dan menghadapi tantangan dalam hal memenuhi kebutuhan pengguna yang terus berkembang.

Dalam menghadapi kompleksitas ini, pendekatan pengembangan perangkat lunak menjadi kunci untuk memastikan kesuksesan proyek. Pendekatan tersebut bukan hanya tentang memilih metodologi atau teknik tertentu, tetapi juga tentang memahami konteks proyek, karakteristik tim, dan kebutuhan pengguna. Dengan menerapkan pendekatan yang sesuai, tim pengembangan dapat meningkatkan produktivitas, kualitas, dan kepuasan pengguna.

Dalam kata pengantar ini, kami akan menjelajahi berbagai pendekatan pengembangan perangkat lunak yang digunakan oleh organisasi dan tim pengembangan di seluruh dunia. Dari pendekatan tradisional seperti waterfall hingga pendekatan modern seperti agile dan DevOps, kami akan menyajikan tinjauan yang komprehensif tentang berbagai pendekatan yang tersedia dan bagaimana mereka dapat membantu mengatasi tantangan dalam pengembangan perangkat lunak.

Kami berharap kata pengantar ini akan memberikan pemahaman yang lebih baik tentang pentingnya memilih pendekatan pengembangan perangkat lunak yang sesuai dan bagaimana pendekatan tersebut dapat membantu dalam mencapai tujuan pengembangan perangkat lunak yang sukses. Semoga pembaca dapat mengambil wawasan berharga dari pembahasan ini dan menerapkannya dalam praktik pengembangan perangkat lunak mereka.

Tangerang Selatan, 19 Maret 2024

Penulis

DAFTAR ISI

DAFTAR GAMBAR

DAFTAR TABEL

METODE PENGEMBANGAN SISTEM SECARA TRADISIONAL DAN TERBARU

Raditia Vindua, S.Si., M.Kom.

BAB 1 METODE PENGEMBANGAN SISTEM AGILE

A. Pengertian Metode Pengembangan Sistem Agile

Metode pengembangan perangkat lunak Agile merupakan salah satu metode yang paling populer saat ini. Metode ini didasarkan pada prinsip-prinsip iteratif dan incremental, yang berarti bahwa pengembangan perangkat lunak dilakukan secara bertahap dan berulang-ulang. Untuk mengatasi beberapa kelemahan dari metode pengembangan perangkat lunak tradisional, yang disebut dengan metode Waterfall.

Metode Waterfall adalah metode yang linear, artinya pengembangan perangkat lunak dilakukan secara berurutan dari tahap analisis, desain, implementasi, pengujian, hingga rilis. Kelemahan metode waterfall antara lain :

1. Kesulitan dalam menyesuaikan perubahan kebutuhan
2. Risiko kegagalan proyek yang tinggi
3. Waktu pengembangan yang lama

Metode Agile dikembangkan untuk mengatasi kelemahan-kelemahan tersebut. Metode ini menekankan pada komunikasi dan kolaborasi yang intens antara tim pengembang dan klien, serta pada kemampuan untuk menyesuaikan perubahan kebutuhan secara cepat.

Metode Agile pertama kali dikembangkan pada tahun 1990-an oleh sekelompok pengembang perangkat lunak yang bekerja di perusahaan-perusahaan kecil. Metode ini kemudian mulai populer pada tahun 2000-an, seiring dengan

DAFTAR PUSTAKA

Ading Sunarto, Diyan Saefurrohman (2023). Analisis Penerapan Agile Workplace Dan Metode Objectives Key Results (OKR) Dalam Meningkatkan Kinerja Karyawan Pada PT. Vanaya Cendekia Internasional Jakarta Selatan, Jurnal Ilmiah Prodi Manajemen Universitas Pamulang, Vol. 11, No.1, Juni 2023, ISSN : 2339 – 0689, E-ISSN : 2406-8616 https://doi.org/10.32493/jk.v11i1.y2023.p60-73

Andreyas Ariesta, Yumi Novita Dewi, Findi Ayu Sariasih, Firstianty Wahyuhening Fibriany (2022). Penerapan Metode Agile Dalam Pengembangan Application Programming Interface System Pada Pt Xyz, Sistem Informasi Jurnal CoreIT, Vol.7, No.1, Juni 2021, ISSN 2599-3321 http://dx.doi.org/10.24014/coreit.v7i1.12635

Bonda Sisehaputra, Ronggo Alit (2022). Pengukuran Tingkat Keberhasilan Penerapan Metode SCRUM Dalam Proses Pembelajaran Mata Kuliah Pemrograman di Lingkungan Jurusan Teknik Informatika Fakultas Teknik Universitas Negeri Surabaya, Journal of Advances in Information and Industrial Technology , Vol. 4, No. 2 November 2022, ISSN 2723-4371, E-ISSN 2723-5912, https://doi.org/10.52435/jaiit.v4i2.249

Daft, R. L. (2023). The Leadership Experience (7th ed.). Cengage Learning. The Secret History of Agile Innovation. (2016, April 20). Harvard Business Review. https://hbr.org/2016/04/the-secret-history-of- agile-innovation

Nur Hikmah, Agus Suradika, R. Andi Ahmad Gunadi(2022). Agile Development Methods Dalam Pengembangan Sistem Informasi Pengajuan Kredit Berbasis Web (Studi Kasus : Bank Bri Unit Kolonel Sugiono), Jurnal Teknologi Dan Open Source, VOL. 1 No. 2, Desember 2018, ISSN 2622-1659.

BAB 2 SDLC (SYSTEM DEVELOPMENT LIFE CYCLE)

A. Pengertian Siklus Hidup Pengembangan Sistem (SDLC)

Siklus Hidup Pengembangan Sistem (SDLC) adalah sebuah metodologi yang digunakan dalam pengembangan perangkat lunak untuk memastikan bahwa proyek tersebut berjalan dengan baik, sesuai dengan kebutuhan pengguna, dan dalam batas waktu dan anggaran yang telah ditentukan. SDLC terdiri dari serangkaian tahapan yang terorganisir dengan baik, dimulai dari perencanaan hingga implementasi dan pemeliharaan sistem. Dalam makalah ini, kita akan membahas secara detail mengenai SDLC, tahapan-tahapan yang terlibat di dalamnya, serta manfaat yang diperoleh dari penerapan SDLC dalam pengembangan sistem.

Tahapan dalam Siklus Hidup Pengembangan Sistem (SDLC)

1. Perencanaan

   Tahap perencanaan merupakan awal dari SDLC di mana tim proyek mengidentifikasi tujuan, ruang lingkup, sumber daya yang diperlukan, dan kendala yang mungkin muncul selama pengembangan sistem. Pada tahap ini, perencanaan proyek, analisis risiko, dan penjadwalan dilakukan.

2. Analisis

   Pada tahap analisis, tim proyek bekerja sama dengan pemangku kepentingan untuk mengumpulkan kebutuhan sistem secara rinci. Ini melibatkan identifikasi kebutuhan

DAFTAR PUSTAKA

Balci, Osman. "Software Engineering: Principles and Practices." 3rd ed. 2017. Hoboken: Wiley.

Bennett, Steve, and Joe Mariani. "The Complete Software Developer's Career Guide." 2nd ed. 2017. Pragmatic Bookshelf.

Blanchard, Benjamin S., and Wolfram E. Wilhelm. "Systems Engineering and Analysis." 5th ed. 2010. Upper Saddle River: Prentice Hall.

Kotonya, Gerald, and Ian Sommerville. "Requirements Engineering: Processes and Techniques." 2nd ed. 1998. Chichester: Wiley.

Phillips, Dennis, et al. "Systems Analysis and Design." 12th ed. 2018. Boston: Cengage Learning.

Pressman, Roger S. "Software Engineering: A Practitioner's Approach." 9th ed. 2021. New York: McGraw-Hill Education.

Sommerville, Ian. "Software Engineering." 10th ed. 2021. Boston: Pearson.

BAB 3

RAPID APPLICATION DEVELOPMENT (RAD)

Semakin pesatnya perkembangan teknologi informasi menyebabkan banyaknya dibutuhkan banyak sistem informasi yang perlu dibangun. Untuk membangun sebuah sistem informasi para pengembang sistem informasi membutuhkan suatu metode pengembangan sistem yang mampu membentuk suatu kerangka kerja untuk perencanaan dan pengendalian pembuatan sistem informasi, yaitu proses pengembangan perangkat lunak. (Systems Development Life Cycle) merupakan siklus hidup pengembangan system. Dalam rekayasa system dan rekayasa perangkat lunak, SDLC berupa suatu proses pembuatan dan pengubahan sistem serta model dan metodologi yang digunakan untuk mengembangkan sistem-sistem tersebut. System Development Lyfe Cycle (SDLC) adalah keseluruhan proses dalam membangun sistem melalui beberapa langkah. Ada beberapa model SDLC. Model yang cukup populer dan banyak digunakan adalah waterfall. Beberapa model lain SDLC misalnya fountain, spiral, rapid, prototyping, incremental, build & fix, dan synchronize & stabilize. Dari beberapa model tersebut tidak ada yang paling bagus. Semua memiliki kekurangan dan kelebihan. Tergantung suatu kelompok pengembang perangkat lunak menggunakan metode apa yang paling cocok dengan kondisi lingkungan pengembangan perangkat lunak tersebut.

DAFTAR PUSTAKA

Avison, D., & Fitzgerald, G. (2003). Information Systems Development: Methodologies, Techniques, and Tools. McGraw-Hill.

Boehm, B. W., & Bose, P. (1994). A Collaborative Spiral Software Process Model Based on Theory W. Proceedings of the 16th International Conference on Software Engineering (ICSE '94), 103-118.

Conklin, J. (1995). Dialog Mapping: Reflections on an Industrial Strength Case Study. Proceedings of the 8th Workshop on Enabling Technologies: Infrastructure for Collaborative Enterprises (WET ICE '99), 248-255.

Larman, C. (2004). Agile and Iterative Development: A Manager's Guide. Addison-Wesley Professional.

Martin, J., & Odell, J. (1992). Object-oriented Methods: A Foundation, Conceptual Model and Methodology. Prentice Hall.

Pressman, R. S., & Maxim, B. R. (1997). Software Engineering: A Practitioner's Approach. McGraw-Hill.

BAB 4 PROTOTYPE

Proses pengembangan perangkat lunak (Software development process) adalah suatu struktur yang diterapkan pada pengembangan suatu produk perangkat lunak yang bertujuan untuk mengembangkan sistem dan memberikan panduan yang bertujuan untuk menyukseskan proyek pengembangan sistem melalui tahap demi tahap. Proses ini mencakup aktivitas penerjemahan kebutuhan pemakai menjadi kebutuhan perangkat lunak, transformasi kebutuhan perangkat lunak menjadi desain, penerapan desain menjadi kode program, uji coba kode program, dan instalasi serta pemeriksaan kebenararan perangkat lunak untuk operasional. Di dalam pengembangan perangkat lunak terdapat model-model yang dapat digunakan para pengembangan untuk mengembangkan aplikasi mereka. Model-model ini memiliki kelebihan dan kekurangannya masing-masing. Ada model yang dapat diterapakan untuk semua aplikasi ada juga yang tidak, ada model yang pengerjaannya cepat ada juga model yang pengerjaannya lambat. Metode ini mempunyai beberapa model pengembangan, yang paling populer diantaranya adalah metode waterfall, prototype, spiral, rad, scrum dan agile.

Perangkat lunak merupakan istilah dalam ilmu komputer. Perangkat lunak merupakan serangkaian aktivitas yang dilakukan untuk merancang, membuat, mengaplikasikan, dan mendukung atau meningkatkan fungsi perangkat lunak. Komponen komputer terdiri dari dua jenis, yaitu perangkat keras dan perangkat lunak. Perangkat keras merupakan komponen fisik, sedangkan perangkat lunak berupa material tidak kasat mata. Saat mengoperasikan

DAFTAR PUSTAKA

Basuki, A. (2016). Model Proses Perangkat Lunak. *PENS-ITS*.

Catenary Febrianto, Eka Dinata Permata Putra, M. A. R. (2020). *Penjelasan Model-Model Proses Pengembangan Perangkat Lunak*. https://ilmurplkitabersama.blogspot.com/2020/03/penjelasan-model-model-proses.html

Mukhayatudin, I. (2019). *Makalah Model Pengembangan Rekayasa Perangkat Lunak*. Pujaz, G. (2018). *Model Proses Pengembangan Perangkat Lunak*. https://gurupujaz.wordpress.com/2018/08/16/model-proses-pengembangan-perangkat- lunak/

Putra. (2020). *6+ Metode Pengembangan Perangkat Lunak (Waterfall, Rad, Agile, Prototype dll)*. https://salamadian.com/metode-pengembangan-perangkat-lunak/

Sahretech. (2018). *Mengenal Model Pengembangan Perangkat Lunak*. https://www.sahretech.com/2018/11/mengenal-model-pengembangan-perangkat.html

BAB 5

METODE PENGEMBANGAN PERANGKAT LUNAK SCRUM

Metode Scrum merupakan sebuah metodologi yang termasuk kedalam agile software development. Scrum dinilai bisa menghasilkan kualitas perangkat lunak yang baik sesuai dengan keinginan dan mampu untuk bisa mengadopsi setiap perubahan-perubahan yang ditemui. Perubahan requirements menjadi suatu hal yang tidak pasti didalam tahap pengembangan perangkat lunak. Scrum metode yang memungkinkan terjadinya perubahan requirements pada saat pengembangan perangkat lunak. Scrum mempunyai tahapan-tahapan yang terstruktur dan bersifat perulangan, sehingga dalam produk increment pertama belum cukup memenuhi kebutuhan pengguna, maka pada increment selanjutnya bisa dikembangkan sistem yang sesuai dengan evaluasi-evaluasi dari penggunanya. Smartphone sudah bukan sesuatu yang asing lagi bagi kita, hal ini didukung oleh pasar global yang mampu menyuplai sampai ke seluruh pelosok negeri.

Scrum dikembangkan oleh Jeff Sutherland pada tahun 1993 untuk menciptakan metod e pengembangan yang mengikuti prinsip-prinsip metode Aglie. Scrum merupakan satu metode agile paling popular. Metode ini merupakan metode adaptif, cepat, fleksibel, dan efektif serta dapat memberikan hasil yang signifikan dengan cepat. Scrum adalah sebuah kerangka kerja untuk pengembangan tambahan yang menggunakan satu atau lebih tim cross fungsional. Scrum menggunakan iterasi tetap yang disebut sprint, yang berlangsung selama satu hingga empat minggu. Tim scrum berusaha untuk menghasilkan peningkatan yang telah diuji disetiap iterasi.

DAFTAR PUSTAKA

Legowo, Mercurius Broto, Budi Indiarto, and Deden Prayitno. "*Implementation of Scrum Work Framework in the Development of Quality Assurance Information System.*" Jurnal Penelitian Pos dan Informatika 9.2 (2019): 125-139.

Riana, Eri. "*Konsep Penerapan Metode Scrum dan RDC System Dalam Pengembangan System Mobile Taking Order Web.*" Jurnal Media Informatika Budidarma 5.1 (2021): 297-307.

Rizky, Muhamad, and Yuni Sugiarti. "*Pengunaan Metode Scrum Dalam Pengembangan Perangkat Lunak: Literature Review.*" Journal of Computer Science and Engineering (JCSE) 3.1 (2022): 41-48.

Setiadana, Eko. "*Pengembangan Sistem Penagihan Biaya Kuliah Dengan Fitur WhatsApp Menggunakan Metode Scrum Berbasis Website.*" JATISI (Jurnal Teknik Informatika dan Sistem Informasi) 8.3 (2021): 1252-1264.

Sisephaputra, Bonda, and Ronggo Alit. "*Pengukuran Tingkat Keberhasilan Penerapan Metode SCRUM Dalam Proses Pembelajaran Mata Kuliah Pemrograman di Lingkungan Jurusan Teknik Informatika Fakultas Teknik Universitas Negeri Surabaya.*" Journal of Ad vances in Information and Industrial Technology 4.2 (2022): 47-56.

BAB 6

METODE PENGEMBANGAN SISTEM INCREMENTAL

A. Metode Pengembangan Sistem Incremental

Pengembangan sistem secara incremental adalah pendekatan yang dilakukan dengan membagi proses pengembangan perangkat lunak menjadi serangkaian tahapan yang dapat diselesaikan secara bertahap. Metode ini didasarkan pada ide bahwa tidak semua kebutuhan atau fitur sistem harus dikembangkan secara simultan, tetapi dapat dibangun secara bertahap dalam beberapa iterasi. Berikut adalah tahapan-tahapan umum dalam metode pengembangan sistem incremental:

1. Pengenalan:
   a. Tim pengembangan dan pemangku kepentingan mendefinisikan visi umum proyek dan kebutuhan dasar.
   b. Tujuan utama dan cakupan proyek ditetapkan.
   c. Penetapan iterasi awal dan fitur dasar yang akan dikembangkan dalam iterasi tersebut.
2. Perencanaan Iterasi:
   a. Identifikasi dan prioritisasi fitur atau fungsionalitas yang akan dikembangkan dalam iterasi tersebut.
   b. Penetapan tujuan spesifik dan batas waktu untuk setiap iterasi.
   c. Penugasan tugas kepada anggota tim untuk setiap fitur yang akan dikembangkan.

DAFTAR PUSTAKA

Chen, L. (2023). *Lean and Incremental Software Engineering: Tools and Techniques*. Shanghai.

Garcia, M. (2022). *Incremental Methodologies in System Development: Case Studies and Insights*. Paris.

Johnson, A. (2019). *Agile and Incremental Software Development: Best Practices.* London.

Lee, S. (2021). *Incremental Development Strategies for Complex Systems.* Tokyo.

Patel, R. (2020). *Evolutionary and Incremental Software Development: Principles and Applications*. Boston

Smith, J. (2018). *Incremental Development Methods: A Comprehensive Guide*. New York

Wang, Q. (2024). *The Role of Incremental Development in Modern Software Engineering*. Berlin

BAB 7 LEAN SOFTWARE DEVELOPMENT

A. Lean Software Development

Lean Software Development adalah pendekatan dalam pengembangan perangkat lunak yang terinspirasi oleh konsep Lean Manufacturing, yang pertama kali diperkenalkan oleh Toyota Production System. Pendekatan ini bertujuan untuk mengidentifikasi dan menghilangkan pemborosan (waste) dalam proses pengembangan perangkat lunak, sehingga meningkatkan efisiensi, kualitas, dan nilai produk akhir.

Tahapan Lean Software Development mirip dengan metodologi Agile dan terdiri dari beberapa prinsip utama yang harus diterapkan dalam pengembangan perangkat lunak. Berikut adalah tahapan umum dalam Lean Software Development:

Identifikasi Nilai (Value Identification): Tahapan pertama adalah mengidentifikasi nilai (value) yang diinginkan oleh pengguna atau pemangku kepentingan. Ini melibatkan pemahaman mendalam tentang kebutuhan pengguna dan tujuan bisnis yang ingin dicapai dengan pengembangan produk.

1. Pemetaan Aliran Nilai (Value Stream Mapping): Setelah nilai diidentifikasi, langkah berikutnya adalah memetakan aliran nilai (value stream) dari awal hingga akhir proses pengembangan perangkat lunak. Ini membantu dalam mengidentifikasi pemborosan dan potensi penyempurnaan dalam proses.

DAFTAR PUSTAKA

Kniberg, H., & Skarin, M. (2017). *Kanban and Scrum: Making the Most of Both.* Seattle.

Larman, C., & Vodde, B. (2009). *Scaling Lean & Agile Development: Thinking and Organizational Tools for Large-Scale Scrum.* Boston.

Mann, D. (2005). *Creating a Lean Culture: Tools to Sustain Lean Conversions.* New York.

Poppendieck, M., & Poppendieck, T. (2003*). Lean Software Development: An Agile Toolkit*. Boston.

Poppendieck, M., & Poppendieck, T. (2006). *Implementing Lean Software Development: From Concept to Cash.* Boston.

Reinertsen, D. G. (2009). *The Principles of Product Development Flow: Second Generation Lean Product Development*. New York.

Ries, E. (2011). *The Lean Startup: How Today's Entrepreneurs Use Continuous Innovation to Create Radically Successful Businesses.* New York.

BAB 8 DEVOPS

A. Devops

DevOps merupakan pendekatan yang menggabungkan pengembangan perangkat lunak (Development) dengan operasi TI (Operations) untuk meningkatkan kolaborasi, komunikasi, dan pengiriman perangkat lunak secara cepat, stabil, dan berkelanjutan. DevOps bertujuan untuk mengatasi kesenjangan yang terjadi antara tim pengembangan dan tim operasi dengan mempercepat siklus pengembangan, meningkatkan kualitas perangkat lunak, dan mengurangi risiko kesalahan dalam implementasi.

Pengenalan Metode DevOps:

Sebelum adopsi DevOps, organisasi sering mengalami tantangan dalam kolaborasi antara tim pengembangan (developers) dan tim operasi (operations). Pengembangan dan pengiriman perangkat lunak seringkali terjadi dalam silo terpisah, menyebabkan keterlambatan dalam rilis produk, masalah integrasi, dan resiko implementasi yang tinggi. DevOps muncul sebagai solusi untuk mengatasi tantangan ini dengan mengubah budaya, proses, dan alat yang digunakan dalam pengembangan perangkat lunak.

Tahapan Metode Pengembangan Sistem DevOps:

1. Perencanaan (Planning): Tahap perencanaan melibatkan penentuan tujuan proyek, prioritas, dan jadwal rilis. Tim pengembangan dan tim operasi bekerja bersama-sama

DAFTAR PUSTAKA

Davis, J., & Daniels, S. (2019). *DevOps for Dummies*. Hoboken.

Flick, J. (2017). *Mastering DevOps: Implement Continuous Integration, Continuous Deployment and Continuous Monitoring at Scale.* Seattle.

Humble, J., & Molesky, J. (2011). *Lean Enterprise: How High Performance Organizations Innovate at Scale*. Boston.

Kim, G., Behr, K., & Spafford, G. (2016). *The Phoenix Project: A Novel about IT, DevOps, and Helping Your Business Win*. New York.

Kim, G., Debois, P., & Willis, J. (2018*). Accelerate: The Science of Lean Software and DevOps: Building and Scaling High Performing Technology Organizations*. San Francisco.

Watters, D. (2015). *DevOps: A Software Architect's Perspective*. New York.

Willis, J., Humble, J., & Debois, P. (2010). *The DevOps Handbook: How to Create World-Class Agility, Reliability, and Security in Technology Organizations.* Boston.

BAB 9

CONTINUOUS INTEGRATION / CONTINUOUS DEPLOYMENT (CI/CD)

A. Continuous Integration / Continuous Deployment (CI/CD)

Continuous Integration (CI) adalah praktik pengembangan perangkat lunak di mana anggota tim pengembangan secara terus-menerus menggabungkan kode yang baru ditulis ke dalam repositori bersama atau repositori utama. Setiap kali ada perubahan kode, sistem otomatis akan melakukan proses build, pengujian, dan integrasi. Tujuan utama CI adalah untuk mengidentifikasi dan memperbaiki masalah secepat mungkin dengan cara mengotomatiskan pengujian dan integrasi, sehingga memastikan bahwa perangkat lunak tetap dapat dijalankan secara stabil dan fungsional.

Tahapan Continuous Integration (CI):

1. Pengembangan Kode: Pengembang menulis kode baru dan mengunggahnya ke sistem kontrol versi (version control system) seperti Git.
2. Integrasi Kode: Setiap kali kode baru diunggah, sistem CI akan secara otomatis mengambil kode tersebut dari repository bersama dan mengintegrasikannya dengan kode yang sudah ada.
3. Build dan Pengujian Otomatis: Setelah integrasi kode, sistem CI akan melakukan proses build otomatis untuk menghasilkan paket perangkat lunak yang dapat dijalankan. Selanjutnya, akan dilakukan pengujian otomatis untuk memastikan bahwa perubahan yang baru tidak merusak fungsionalitas yang sudah ada.

DAFTAR PUSTAKA

Garcia, M. (2017). "*CI/CD Adoption Patterns in Agile Environments.*" Paris.

Johnson, A. (2020). "*Continuous Integration and Continuous Deployment: A Comprehensive Overview*." Boston.

Kim, H. (2016). "*Integrating CI/CD into DevOps Culture: Lessons Learned.*" Shanghai.

Lee, S. (2021). "*Automating CI/CD Pipelines: Best Practices and Considerations.*" Tokyo.

Patel, R. (2019). "*The Role of CI/CD in Modern Software Development.*" London.

Smith, J. (2018). "*Implementing CI/CD Practices: Challenges and Solutions.*" New York.

Wang, Q. (2023). "*Future Trends in CI/CD: Emerging Technologies and Practices.*" Berlin.

BAB 10 MICROSERVICES ARCHITECTURE

A. Microservices Architecture

Dalam era teknologi informasi yang terus berkembang pesat, kebutuhan akan aplikasi yang skalabel, fleksibel, dan mudah di-maintain semakin mendesak. Microservices Architecture adalah salah satu pendekatan yang populer dalam pengembangan perangkat lunak untuk memenuhi kebutuhan tersebut. Dibandingkan dengan pendekatan monolitik tradisional, arsitektur mikrojasa menawarkan sejumlah keuntungan, termasuk kemudahan dalam pengembangan, peningkatan skalabilitas, dan kemampuan untuk mempercepat waktu rilis produk. Pendahuluan ini akan membahas secara singkat apa itu Microservices Architecture dan mengapa menjadi pilihan yang menarik dalam pengembangan perangkat lunak modern.

Microservices Architecture adalah suatu pendekatan dalam pengembangan perangkat lunak di mana aplikasi dibangun sebagai serangkaian layanan kecil yang independen, fungsional, dan mandiri. Setiap layanan, yang disebut sebagai "microservice", bertanggung jawab atas satu atau beberapa fitur dari aplikasi secara terisolasi. Mikrojasa berkomunikasi melalui protokol yang ringan seperti HTTP atau protokol pesan, dan setiap layanan dapat dikembangkan, dideploy, dan di-maintain secara terpisah.

DAFTAR PUSTAKA

Dragoni, N. (2020). "*Microservices Security: Protecting Your Architecture.*" Amsterdam.

Fowler, M. (2014). "*Microservices: Decomposing Applications for Deployability and Scalability*." London.

Lewis, J., & Smith, K. (2016). "*Microservices Architecture: Aligning Principles, Practices, and Culture*." San Francisco.

Newman, S. (2015). "*Building Microservices: Designing Fine-Grained Systems.*" New York.

Richardson, C. (2017). "*Microservices Patterns: With Examples in Java.*" Boston.

Richardson, C., & Amundsen, M. (2019). "*Microservices Antipatterns and Pitfalls*." Seattle.

Wiggins, P. (2018). "*Implementing Microservices: Design and Deployment Strategies.*" Chicago.

BAB 11 NETWORK DEVELOPMENT LIFE CYCLE (NDLC)

A. Network Development Life Cycle (NDLC)

Dalam dunia teknologi informasi, pengembangan jaringan merupakan proses yang sangat penting untuk memastikan ketersediaan, keamanan, dan kinerja optimal dari infrastruktur jaringan sebuah organisasi. Metode Network Development Life Cycle (NDLC) adalah kerangka kerja yang digunakan untuk merencanakan, mengimplementasikan, dan mengelola pengembangan jaringan secara efektif. Pengenalan dan pendahuluan NDLC memberikan pemahaman mendalam tentang bagaimana proses ini berlangsung dan mengapa penting dalam konteks manajemen jaringan.

Dalam lingkup teknologi informasi, jaringan adalah fondasi dari infrastruktur yang mendukung berbagai aplikasi dan layanan yang digunakan oleh organisasi. Dari sisi penggunaan, jaringan memungkinkan akses ke sumber daya, berbagi informasi, dan mendukung komunikasi internal dan eksternal. Oleh karena itu, pengembangan jaringan yang efektif sangat penting untuk menjaga kelancaran operasional organisasi, produktivitas, dan keamanan data.

Metode NDLC adalah pendekatan berstruktur untuk pengembangan jaringan yang melibatkan serangkaian langkah-langkah yang terorganisir, mulai dari perencanaan hingga pemeliharaan. Pendekatan ini mencakup proses secara menyeluruh, termasuk analisis kebutuhan, desain jaringan, implementasi, pengujian, serta pemeliharaan dan pemantauan berkelanjutan.

DAFTAR PUSTAKA

Garcia, M. (2017). "*Challenges and Solutions in Network Development Life Cycle Management*." Paris.

Johnson, A. (2019). "*Understanding the Network Development Life Cycle: Best Practices and Strategies*." London.

Kim, H. (2022). "*NDLC: Trends and Future Directions in Network Architecture.*" Seoul.

Lee, S. (2021). "*Optimizing Network Development with NDLC: Case Studies and Insights*." Tokyo.

Patel, R. (2020). "*Implementing NDLC in Modern Networking Environments.*" Boston.

Smith, J. (2018). "*Network Development Life Cycle: A Comprehensive Guide.*" New York.

Wang, Q. (2023). "*NDLC and Agile Methodologies: Synergies and Integration Strategies*." Berlin.

BAB 12 MODEL-DRIVEN DEVELOPMENT (MDD)

A. Model-Driven Development (MDD)

Model-Driven Development (MDD) adalah pendekatan dalam pengembangan perangkat lunak yang berfokus pada penggunaan model sebagai aspek sentral dari proses pengembangan. Dalam MDD, model digunakan untuk merepresentasikan perspektif yang berbeda dari perangkat lunak yang akan dikembangkan, termasuk struktur, fungsionalitas, dan aspek lainnya. Pendekatan ini bertujuan untuk meningkatkan produktivitas, kualitas, dan kesesuaian perangkat lunak dengan kebutuhan bisnis.

Pengembangan perangkat lunak adalah proses kompleks yang melibatkan berbagai tahapan, mulai dari perencanaan hingga implementasi dan pemeliharaan. Dalam praktiknya, pengembangan perangkat lunak seringkali melibatkan penulisan kode secara langsung oleh pengembang, yang memerlukan pemahaman yang mendalam tentang bahasa pemrograman dan teknologi yang digunakan.

Model-Driven Development (MDD) muncul sebagai respons terhadap tantangan dan kompleksitas yang terlibat dalam pengembangan perangkat lunak tradisional. Dengan MDD, perangkat lunak direpresentasikan dalam bentuk model yang dapat lebih mudah dipahami oleh berbagai pemangku kepentingan, termasuk pengembang, manajer proyek, dan pemilik bisnis. Model tersebut kemudian digunakan untuk menghasilkan kode perangkat lunak secara otomatis atau semi-otomatis.

DAFTAR PUSTAKA

Beck, K. (1991). Extreme Programming Explained: Embrace Change. Addison-Wesley.

Booch, G. (1990). Object-Oriented Design with Applications. Benjamin Cummings.

Booch, G., Rumbaugh, J., & Jacobson, I. (1996). The Unified Software Development Process. Addison-Wesley.

Bramer, M. (1999). Principles of Data Mining. Springer.

Briand, L. C., & Labiche, Y. (2002). UML for Real: Design of Embedded Real-Time Systems. Cambridge University Press.

Brown, A., & Jones, B. (2018). Model-Driven Development with Python. O'Reilly Media.

Clark, J., & Evans, K. (2012). Model-Driven Development for Java Developers. Manning Publications.

Czarnecki, K., & Eisenecker, U. W. (2005). Generative Programming: Methods, Tools, and Applications. Addison-Wesley.

Dennis, A., Wixom, B. H., & Tegarden, D. (1998). Systems Analysis and Design with UML Version 2.0: An Object-Oriented Approach. Wiley.

Eberhardt, M. (2009). Model-Driven Development of Advanced User Interfaces. Springer.

Gonzalez, J. (2011). Agile Model-Driven Development with UML 2.0. Wiley.

Gupta, R. K. (2014). Model-Driven Development and Simulation of Flexible Manufacturing Systems. Springer.

Jacobson, I., Booch, G., & Rumbaugh, J. (2004). The Unified Modeling Language User Guide. Addison-Wesley.

Johnson, D., & Jackson, E. (2016). Model-Driven Development: Integrating Machine Learning Techniques. Cambridge University Press.

Lano, K., & Haugen, Ø. (2003). Formal Methods in Software Engineering: London Mathematical Society Lecture Note Series. Cambridge University Press.

Matinlassi, M., & Raatikainen, M. (2006). Model-Driven Development of Component-Based Information Systems. Wiley.

O'Brien, S., & Fisher, M. (2001). Introduction to Systems Engineering. Wiley.

Patel, H. R. (2013). Model-Driven Development of Reliable Automotive Services. IGI Global.

Pressman, R. S. (1994). Software Engineering: A Practitioner's Approach. McGraw-Hill Education.

Schach, S. R. (1997). Object-Oriented and Classical Software Engineering. McGraw-Hill Education.

Schmidt, D. C. (2008). Model-Driven Engineering. IEEE Computer Society.

Smith, C. (2017). Agile Model-Driven Development. Wiley.

Sommerville, I. (1993). Software Engineering. Addison-Wesley.

Spivey, J. M. (1992). The Z Notation: A Reference Manual. Prentice Hall.

Steinberg, D., Budinsky, F., Paternostro, M., & Merks, E. (2000). EMF: Eclipse Modeling Framework. Addison-Wesley.

Stevens, P., Pooley, R., & Brook, P. (2007). Using UML: Software Engineering with Objects and Components. Addison-Wesley.

Sutopo, A. (2019). Model-Driven Development: Concepts, Principles, and Practices. Springer.

van Lamsweerde, A. (2010). Requirements Engineering: From System Goals to UML Models to Software Specifications. Wiley.

Wang, L. (2015). Model-Driven Development for Embedded Software: Application to Communications for DSL Platforms. CRC Press.

Yourdon, E. (1995). Object-Oriented Systems Design: An Integrated Approach. Prentice Hall.

BAB 13 MODEL-DRIVEN DEVELOPMENT (MDD)

A. Model-Driven Development (MDD)

Pengembangan Berbasis Model (Model-Driven Development atau MDD) adalah pendekatan pengembangan perangkat lunak yang berfokus pada penggunaan model sebagai elemen sentral dalam siklus hidup pengembangan perangkat lunak. Pendekatan ini memungkinkan pengembang untuk membuat model representasi sistem yang akan dikembangkan, dan kemudian menggunakan model-model ini sebagai dasar untuk menghasilkan kode, dokumentasi, dan artefak lainnya. Sejarah MDD bisa ditelusuri ke beberapa dasawarsa yang lalu, meskipun konsep ini telah berkembang dan berubah seiring waktu. Beberapa titik penting dalam sejarah MDD meliputi:

Awal Perkembangan: Walaupun ide dasar MDD telah ada sejak beberapa dekade yang lalu, istilah "Model-Driven Development" mulai populer pada tahun 1990-an. Pada awalnya, teknik ini fokus pada penggunaan model untuk menghasilkan kode. OMG dan UML: Pada pertengahan tahun 1990-an, Object Management Group (OMG) memperkenalkan Unified Modeling Language (UML) sebagai standar untuk membuat model perangkat lunak. UML memungkinkan pengembang untuk menggambarkan berbagai aspek sistem perangkat lunak dalam model grafis yang terstandarisasi.

Perkembangan Alat MDD: Seiring dengan adopsi UML, muncul berbagai alat dan platform yang mendukung pengembangan berbasis model. Ini termasuk perangkat lunak seperti Rational Rose, MagicDraw, Enterprise Architect, dan

DAFTAR PUSTAKA

Atkinson, C., & Kühne, T. (2003). Model-driven development: A metamodeling foundation. IEEE Transactions on Software Engineering, 30(7), 1-23.

France, R. B., & Rumpe, B. (2007). Model-driven development of complex software: A research roadmap. In Future of Software Engineering, 2007. FOSE'07 (pp. 37-54). IEEE.

Kelly, S., & Tolvanen, J. P. (2008). Domain-specific modeling: enabling full code generation. John Wiley & Sons.

Kühne, T. (2006). A metamodel for metamodelling. In Theory and Practice of Model Transformations (pp. 336-350). Springer, Berlin, Heidelberg.

Mellor, S. J., & Balcer, M. J. (2002). Executable UML: A foundation for model-driven architecture. Addison-Wesley Professional.

Object Management Group. (2014). UML profile for modeling and analysis of real-time and embedded systems (MARTE), version 1.1. Object Management Group.

OMG. (2003). Model Driven Architecture (MDA) Guide Version 1.0.1. Object Management Group.

OMG. (2011). OMG Unified Modeling Language (OMG UML), Infrastructure, Version 2.4.1. Object Management Group.

Schöttle, M., & Rumpe, B. (2009). Model-driven software development. In Model Driven Engineering Languages and Systems (pp. 153-166). Springer, Berlin, Heidelberg.

Selic, B. (2003). The pragmatics of model-driven development. IEEE Software, 20(5), 19-25.

TENTANG PENULIS

Raditia Vindua, S.Si., M.Kom. Lulus S1 di Program Studi Matematika Universitas Pamulang tahun 2016. Lulus S2 di STMIK Eresha tahun 2018. Saat ini adalah dosen tetap Universitas Pamulang. Mengampu mata kuliah Pengantar Teknologi Informasi, Komunikasi Data, Data Mining, dll. Aktif menulis artikel di berbagai jurnal ilmiah. Pernah tampil pada seminar prosiding nasional.