

PENERBIT BUKULOKA

DESAIN DAN PENGEMBANGAN SISTEM INFORMASI

Editor: Raden Ricky Atthariq Susetia, S.S.

Desain dan Pengembangan Sistem Informasi

Kiky Rizky Nova Wardani, M.Kom., dkk.

Editor: Raden Ricky Atthariq Susetia, S.S.

PT BUKULOKA LITERASI BANGSA

Anggota IKAPI: No. 645/DKI/2024

PENERBIT BUKULOKA

Desain dan Pengembangan Sistem Informasi

Penulis	: Kiky Rizky Nova Wardani, M.Kom., Ali Haidir, M.Kom., Fintri Indriyani, M.Kom., Euis Widanengsih, M.Kom., Agus Salim, M.Kom., Bilal Abdul Wahid, M.Kom., Henny Destiana,M.Kom., Hj. A.Irmayani Pawelloi, ST., M.T., Ghofar Taufik, M.Kom., Indra Riyana Rahadjeng, S.T., M.Kom., Aryo Tunjung Kusumo, M.Kom., dan Fajar Agustini, M.Kom.
ISBN	: 978-634-324-017-4 (PDF)
Penyunting Naskah	: Raden Ricky Atthariq Susetia, S.S.
Tata Letak	: Raden Ricky Atthariq Susetia, S.S.
Desain Sampul	: Novikean Keysah Sanisri

Penerbit

Penerbit PT Bukuloka Literasi Bangsa

Distributor: PT Yapindo

Kompleks Business Park Kebon Jeruk Blok I No. 21, Jl. Meruya Ilir Raya No. 88, Desa/Kelurahan Meruya Utara, Kecamatan Kembangan, Kota Adm. Jakarta Barat, Provinsi DKI Jakarta, Kode Pos: 11620

Email: penerbit.blb@gmail.com

Whatsapp: 0878-3483-2315

Website: bukuloka.com

© Hak cipta dilindungi oleh undang-undang

Berlaku selama 50 (lima puluh) tahun sejak ciptaan tersebut pertama kali dilakukan pengumuman.
Dilarang mengutip atau memperbanyak sebagian atau seluruh isi buku ini tanpa izin tertulis dari penerbit. Ketentuan Pidana Sanksi Pelanggaran Pasal 2 UU Nomor 19 Tahun 2002 Tentang Hak Cipta.
Barang siapa dengan sengaja dan tanpa hak melakukan perbuatan sebagaimana dimaksud dalam Pasal 2 ayat (1) atau Pasal 49 ayat (1) dan ayat (2) dipidana dengan pidana penjara masing-masing paling singkat 1 (satu) bulan dan/atau denda paling sedikit Rp1.000.000,00 (satu juta rupiah), atau pidana penjara paling lama 7 (Tujuh) tahun dan/atau denda paling banyak Rp5.000.000.000,00 (lima miliar rupiah).
Barang siapa dengan sengaja menyerahkan, menyiarkan, memamerkan, mengedarkan atau menjual kepada umum suatu ciptaan atau barang hasil pelanggaran Hak Cipta atau Hak Terkait sebagaimana dimaksud pada ayat (1) dipidana dengan pidana penjara paling lama 5 (lima) tahun dan/atau denda paling banyak Rp500.000.000,00 (lima ratus juta rupiah).

KATA PENGANTAR

Puji syukur ke hadirat Tuhan Yang Maha Esa atas rahmat dan karunia-Nya, sehingga buku berjudul *Desain dan Pengembangan Sistem Informasi* ini dapat tersusun dengan baik. Buku ini hadir sebagai bacaan yang membahas konsep, tahapan, dan penerapan sistem informasi secara runtut dengan bahasa yang mudah dipahami.

Isi buku ini menguraikan pentingnya perancangan sistem informasi dalam mendukung pengelolaan data, proses kerja, layanan, serta pengambilan keputusan di berbagai bidang. Pembahasan disusun agar pembaca dapat memahami bagaimana sebuah sistem dirancang, dikembangkan, dan digunakan untuk menjawab kebutuhan organisasi maupun kegiatan sehari-hari.

Buku ini dijual untuk masyarakat umum dan dapat dibaca oleh siapa saja yang ingin mengenal dunia sistem informasi secara praktis. Semoga buku ini memberikan manfaat serta menambah wawasan pembaca dalam memahami peran teknologi informasi pada era digital.

Jakarta, Juni 2026

Tim Penyusun

DAFTAR ISI

KATA PENGANTAR ..... **iii**

DAFTAR ISI ..... **iv**

BAB 1: Analisis Kebutuhan Sistem Informasi **1**

Kiky Rizky Nova Wardani, M.Kom. **1**

1.1 Konsep Kebutuhan Sistem 1

1.2 Identifikasi Pemangku Kepentingan 6

1.3 Teknik Pengumpulan Kebutuhan 11

1.4 Analisis Kebutuhan Fungsional dan Non-Fungsional 16

1.5 Dokumentasi Kebutuhan Sistem..... 21

BAB 2: Perencanaan Pengembangan Sistem..... **26**

Ali Haidir, M.Kom...... **26**

2.1 Siklus Hidup Pengembangan Sistem (SDLC)..... 27

2.2 Model Pengembangan Sistem 31

2.3 Perencanaan Sistem Proyek Informasi..... 35

2.4 Penjadwalan dan Estimasi Biaya 38

2.5 Manajemen Risiko Proyek 42

BAB 3: Pemodelan Proses Bisnis..... **47**

Fintri Indriyani, M.Kom. **47**

3.1 Konsep Proses Bisnis 47

3.2 Business Process Modeling 50

3.3 Diagram Alir (Flowchart)..... 53
3.4 Data Flow Diagram (DFD)..... 57
3.5 Evaluasi Proses Bisnis..... 61
BAB 4: Pemodelan Data Sistem..... 65
Euis Widanengsih, M.Kom. 65
4.1 Konsep Data dalam Sistem Informasi..... 65
4.2 Entity Relationship Diagram (ERD)..... 68
4.3 Normalisasi Database..... 72
4.4 Relasi Antar Tabel..... 75
4.5 Integritas Data..... 79
BAB 5: Perancangan Basis Data..... 84
Agus Salim, M.Kom..... 84
5.1 Konsep Data Dasar..... 84
5.2 Peranan Logis dan Fisik 87
5.3 DBMS dalam Sistem Informasi..... 90
5.4 Optimasi Struktur Basis Data 93
5.5 Data Dasar Keamanan..... 97
BAB 6: Perancangan Arsitektur Sistem 100
Bilal Abdul Wahid, M.Kom..... 100
6.1 Arsitektur Sistem..... 100
6.2 Arsitektur Software Design 108
6.3 Komponen System Architecture..... 112

6.4 Contoh System Architecture (Client-Server Architecture) 118
6.5 Diagram System Architecture 119
BAB 7: Desain Antarmuka Pengguna (UI/UX)..... 125
Henny Destiana,M.Kom. 125
7.1 Prinsip Desain Antarmuka..... 125
7.2 User Experience (UX)..... 129
7.3 Perancangan Prototipe..... 132
7.4 Evaluasi Usability 136
7.5 Standar Desain Antarmuka..... 139
BAB 8: Pelaksanaan Sistem Informasi 144
Hj. A.Irmayani Pawelloi, ST., M.T. 144
8.1 Konsep Sistem Implementasi 144
8.2 Pemrograman Sistem..... 148
8.3 Integrasi Modul Sistem 152
8.4 Pengujian Sistem..... 156
8.5 Sistem Penyebaran 160
BAB 9: Pengujian dan Evaluasi Sistem 165
Ghofar Taufik, M.Kom. 165
9.1 Konsep Pengujian Sistem..... 165
9.2 Jenis-Jenis Pengujian..... 169
9.3 Pengujian Fungsional dan Non-Fungsional 173
9.4 Evaluasi Kinerja Sistem 177

9.5 Dokumentasi Hasil Pengujian 181

BAB 10: Manajemen Keamanan Sistem Informasi 186

Indra Riyana Rahadjeng, S.T., M.Kom. 186

10.1 Konsep Keamanan Sistem 186

10.2 Ancaman dan Risiko Keamanan 190

10.3 Pengendalian Akses 194

10.4 Enkripsi dan Proteksi Data 198

10.5 Audit Keamanan Sistem 203

BAB 11: Sistem Pemeliharaan dan Pengembangan 208

Aryo Tunjung Kusumo, M.Kom. 208

11.1 Konsep Pemeliharaan Sistem 208

11.2 Jenis-Jenis Sistem Pemeliharaan 212

11.3 Sistem Pemantauan 215

11.4 Peningkatan dan Pengembangan Sistem 219

11.5 Dokumentasi Perubahan Sistem 222

BAB 12: Informasi Sistem Tata Kelola 226

Fajar Agustini, M.Kom. 226

12.1 Konsep Integrasi Sistem Informasi 226

12.2 Arsitektur Perusahaan 231

12.3 Tata Kelola TI (Tata Kelola TI) 235

12.4 Standar dan Kebijakan Sistem Informasi 241

12.5 Strategi Pengembangan Sistem Berkelanjutan .. 246

Profil Penulis **253**
Daftar Pustaka **265**

BAB 1: Analisis Kebutuhan Sistem Informasi

Kiky Rizky Nova Wardani, M.Kom.

Analisis kebutuhan sistem informasi merupakan tahap paling kritis dalam siklus pengembangan sistem informasi. Tahap ini menentukan kesuksesan atau kegagalan proyek pengembangan sistem secara keseluruhan. Proses analisis kebutuhan berfungsi untuk mengidentifikasi, mendokumentasikan, dan memvalidasi semua persyaratan yang diperlukan agar sistem informasi yang dikembangkan dapat memenuhi kebutuhan organisasi dan pengguna dengan sempurna. Tanpa analisis kebutuhan yang komprehensif dan terstruktur, proyek pengembangan sistem berisiko menghasilkan sistem yang tidak sesuai dengan ekspektasi *stakeholder*, memicu pemborosan sumber daya, dan pada akhirnya menyebabkan kegagalan proyek. Oleh karena itu, pemahaman yang mendalam tentang konsep, metodologi, dan teknik-teknik dalam analisis kebutuhan sistem informasi sangat penting bagi para profesional yang terlibat dalam pengembangan sistem.

1.1 Konsep Kebutuhan Sistem

1.1.1 Pengertian Dasar Kebutuhan Sistem Informasi

Kebutuhan sistem informasi secara fundamental dapat didefinisikan sebagai pernyataan lengkap mengenai fungsi, perilaku,

dan karakteristik yang harus dimiliki oleh sistem untuk memenuhi tujuan organisasi dan ekspektasi pengguna. Kebutuhan sistem merupakan terjemahan dari masalah bisnis atau peluang yang dihadapi organisasi ke dalam spesifikasi teknis yang dapat dipahami dan diimplementasikan oleh tim pengembang perangkat lunak. Proses untuk memahami dan mendokumentasikan kebutuhan ini disebut dengan analisis kebutuhan sistem (Tjahyanti & Sutama, 2025). Melalui analisis kebutuhan yang cermat, organisasi dapat memastikan bahwa setiap rupiah yang diinvestasikan dalam pengembangan sistem akan menghasilkan nilai maksimal bagi bisnis.

Kebutuhan sistem tidak hanya mencakup apa yang sistem harus lakukan, tetapi juga bagaimana sistem harus melakukannya, dalam kondisi apa, dan dengan batasan-batasan apa sistem harus beroperasi. Dengan demikian, kebutuhan sistem mencakup dimensi fungsional dan non-fungsional yang harus dipertimbangkan secara seimbang untuk menghasilkan sistem yang berkualitas dan sesuai dengan konteks organisasi.

1.1.2 Pentingnya Analisis Kebutuhan dalam Siklus Pengembangan

Analisis kebutuhan merupakan tahap krusial dalam siklus pengemabangan sistem informasi yang berperan sebagai fondasi bagi seluruh proses pengebanagn selanjutnya. Tahap ini tidak hanya berfungsi mengindentifikasi kebutuhan pengguna, tetapi juga untuk dapat memastikan bahwa sistem yang telah dikembangkan benar benar selaras dengan tujuan organisasi. Penelitian telah

menunjukkan bahwa ketika analisis kebutuhan dilakukan dengan sistematis dan komprehensif akan berkontribusi secara signifikant terhadap keberhasilan sebuah proyek SI. Proyek dengan analisis kebutuhan yang baik cenderung memiliki tingkat keberhasilan implementasi yang lebih tinggi, waktu pengembangan yang lebih efisien, serta pengendalian biaya yang lebih optimal. Sebaliknya, kelemahan dalam tahap ini sering kali menjadi penyebab utama kegagalan sistem, karena kesalahan yang terjadi akan terbawa ke tahap desain, pengembangan, hingga implementasi (Tjahyanti & Sutama, 2025).

Selain itu, kesalahan dalam fase analisis kebutuhan memiliki dampak yang sangat besar terhadap biaya proyek. Beberapa studi menunjukkan bahwa biaya perbaikan kesalahan pada tahap lanjutan dapat mencapai dua puluh hingga empat puluh kali lebih tinggi dibandingkan jika kesalahan tersebut diidentifikasi dan diperbaiki sejak tahap analisis. Hal ini menegaskan bahwa investasi waktu dan sumber daya pada tahap awal justru dapat mengurangi risiko dan biaya di masa depan.

Lebih lanjut, analisis kebutuhan juga berfungsi sebagai sarana komunikasi yang efektif antara berbagai pemangku kepentingan, termasuk manajemen, pengguna akhir, dan tim pengembang. Melalui proses ini, setiap pihak memperoleh pemahaman yang sama mengenai ruang lingkup, tujuan, serta spesifikasi sistem yang akan dibangun. Dengan demikian, potensi terjadinya kesalahpahaman (miscommunication) dapat diminimalkan, sehingga kolaborasi dalam proyek menjadi lebih terarah dan efisien.

1.1.3 Tahapan dan Proses Analisis Kebutuhan Sistem

Proses analisis kebutuhan sistem informasi dilakukan melalui tahapan yang terstruktur dan saling berkaitan.

Tahap awal adalah perencanaan dan persiapan yang mencakup identifikasi awal tujuan proyek, penentuan dan identifikasi *stakeholder* utama, dan penyiapan sumber daya yang diperlukan. Selanjutnya Tahap kedua adalah analisis dan klasifikasi pengumpulan kebutuhan, dilakukan pengumpulan kebutuhan (elicitation) dari pengguna dan pihak terkait menggunakan berbagai teknik yang relevan. melalui berbagai teknik elicitasi dari para *stakeholder* dan pengguna. Tahap ketiga adalah analisis dan klasifikasi kebutuhan yang telah dikumpulkan, di mana kebutuhan informasi yang diperoleh diorganisir secara sistematis ke dalam kategori-kategori yang logis dan saling berhubungan agar mudah dipahami (Widyastuti et al., 2024).

Tahap keempat adalah Hasil analisis akan dituangkan dan spesifikasi kebutuhan yang melibatkan pendokumentasian secara detail semua kebutuhan yang telah diidentifikasi dan dianalisis. Tahap kelima adalah validasi dan verifikasi kebutuhan untuk memastikan bahwa semua kebutuhan telah terdokumentasi dengan benar, lengkap, dan konsisten. Tahap keenam adalah presentasi dan persetujuan dokumentasi kebutuhan oleh semua *stakeholder* utama sehingga menjadi baseline yang dapat digunakan sebagai acuan untuk fase pengembangan selanjutnya. Setiap tahapan memiliki input, proses, dan output yang jelas, sehingga proses analisis kebutuhan menjadi lebih sistematis dan terukur.

1.1.4 Hubungan Analisis Kebutuhan dengan Metodologi Pengembangan Sistem

Metodologi pengembangan sistem yang digunakan dalam suatu proyek sangat mempengaruhi bagaimana proses analisis kebutuhan dilakukan dan didokumentasikan. Pada metodologi Waterfall, analisis kebutuhan umumnya dilakukan secara menyeluruh di tahap awal proyek. Hasilnya berupa dokumentasi yang lengkap, rinci, dan cenderung bersifat tetap (statis) sebagai acuan utama dalam tahap pengembangan berikutnya.

Sebaliknya, dalam metodologi Agile, analisis kebutuhan dilakukan secara iteratif dan inkremental. Kebutuhan tidak harus didefinisikan secara lengkap di awal, melainkan berkembang seiring berjalannya proses pengembangan. Dokumentasi kebutuhan biasanya disajikan dalam bentuk user stories yang lebih sederhana, fleksibel, dan mudah disesuaikan dengan perubahan (Widyastuti et al., 2024).

Perbedaan pendekatan ini menunjukkan bahwa pemilihan metodologi akan menentukan strategi dalam melakukan analisis kebutuhan. Oleh karena itu, pemahaman terhadap karakteristik masing-masing metodologi menjadi penting agar proses analisis dapat disesuaikan dengan konteks proyek yang dijalankan.

Meskipun demikian, prinsip dasar analisis kebutuhan tetap tidak berubah, yaitu mencakup pemahaman terhadap kebutuhan bisnis, identifikasi stakeholder secara menyeluruh, penggunaan teknik elicitasi yang tepat, serta penyusunan dokumentasi yang jelas dan terukur. Dengan menyesuaikan analisis kebutuhan terhadap

metodologi yang digunakan, tim pengembang dapat menjaga kualitas sistem tanpa mengabaikan fleksibilitas maupun ketepatan hasil.

1.2 Identifikasi Pemangku Kepentingan

1.2.1 Pemahaman tentang Pemangku Kepentingan dan Peran Mereka

Pemangku kepentingan (stakeholder) merupakan semua individu, kelompok, atau organisasi yang memiliki kepentingan, keterlibatan, maupun pengaruh terhadap sistem informasi yang akan dikembangkan. Stakeholder tidak hanya terbatas pada pengguna langsung sistem, tetapi juga mencakup manajemen organisasi, pemilik usaha, regulator pemerintah, mitra bisnis, serta pihak eksternal lain yang terdampak oleh implementasi sistem. Setiap stakeholder memiliki kebutuhan, harapan, serta sudut pandang yang berbeda, sehingga seluruh perspektif tersebut perlu diperhatikan dalam proses analisis kebutuhan (Tjahyanti & Sutama, 2025).

Pemahaman terhadap stakeholder tidak hanya sebatas pada identifikasi siapa saja yang terlibat, tetapi juga mencakup pemahaman mengenai peran, tanggung jawab, serta tingkat keterlibatan mereka dalam sistem. Sebagai contoh, pengguna operasional berfokus pada kemudahan penggunaan dan efisiensi kerja, sementara manajemen lebih menekankan pada aspek pengambilan keputusan dan ketersediaan informasi strategis. Di sisi

lain, pihak eksternal seperti regulator memiliki kepentingan terhadap kepatuhan sistem terhadap aturan yang berlaku.

Selain itu, penting untuk memahami bagaimana setiap stakeholder berinteraksi dengan sistem, baik secara langsung maupun tidak langsung. Interaksi ini akan mempengaruhi kebutuhan fungsional maupun non-fungsional yang harus dipenuhi oleh sistem. Oleh karena itu, analisis stakeholder yang mendalam akan membantu dalam mengidentifikasi kebutuhan secara lebih akurat serta meminimalkan risiko terjadinya kesalahan dalam pengembangan sistem.

Dengan demikian, pemahaman yang komprehensif terhadap stakeholder menjadi fondasi utama dalam analisis kebutuhan. Ketidaktepatan dalam mengidentifikasi atau memahami stakeholder dapat menyebabkan kebutuhan yang penting terlewatkan, sehingga sistem yang dikembangkan tidak mampu memenuhi harapan seluruh pihak yang berkepentingan. Oleh karena itu, keterlibatan stakeholder secara aktif sejak tahap awal pengembangan menjadi faktor penting dalam keberhasilan sistem informasi.

1.2.2 Klasifikasi dan Kategorisasi Stakeholder

Stakeholder dalam pengembangan sistem informasi dapat diklasifikasikan berdasarkan berbagai dimensi untuk mempermudah analisis dan pengelolaan keterlibatan mereka. Salah satu pendekatan yang umum digunakan adalah klasifikasi berdasarkan peran dalam organisasi. Dalam hal ini, stakeholder dapat dibedakan menjadi pengguna operasional yang berinteraksi langsung dengan sistem dalam aktivitas sehari-hari, pengguna manajerial yang

memanfaatkan sistem untuk pengambilan keputusan, pimpinan organisasi yang berperan dalam penentuan kebijakan dan alokasi sumber daya, serta pihak eksternal seperti regulator, mitra bisnis, atau pelanggan.

Selain berdasarkan peran, stakeholder juga dapat dikategorikan berdasarkan tingkat pengaruh (power) dan kepentingan (interest) mereka terhadap proyek. Pendekatan ini biasanya divisualisasikan dalam bentuk power-interest matrix yang membagi stakeholder ke dalam empat kategori utama (Widyastuti et al., 2024). Stakeholder dengan pengaruh dan kepentingan tinggi merupakan pihak yang harus dikelola secara aktif karena memiliki peran besar dalam keberhasilan proyek. Stakeholder dengan pengaruh tinggi namun kepentingan rendah tetap perlu diperhatikan untuk menjaga dukungan mereka terhadap proyek.

Di sisi lain, stakeholder dengan kepentingan tinggi namun pengaruh rendah perlu diberikan informasi yang cukup agar tetap terlibat dan mendukung sistem yang dikembangkan. Sementara itu, stakeholder dengan pengaruh dan kepentingan rendah cukup dipantau tanpa memerlukan keterlibatan intensif. Klasifikasi ini membantu tim pengembang dalam menentukan strategi komunikasi, tingkat keterlibatan, serta prioritas dalam memenuhi kebutuhan masing-masing stakeholder.

Dengan adanya kategorisasi yang jelas, proses analisis kebutuhan menjadi lebih terarah dan efisien. Tim pengembang dapat memfokuskan perhatian pada stakeholder yang paling berpengaruh

tanpa mengabaikan pihak lain yang tetap memiliki kontribusi terhadap keberhasilan sistem secara keseluruhan.

1.2.3 Metode dan Teknik Identifikasi Stakeholder

Proses identifikasi stakeholder merupakan langkah penting yang harus dilakukan secara sistematis agar seluruh pihak yang berkepentingan dapat terpetakan dengan baik. Identifikasi ini tidak hanya bertujuan untuk mengetahui siapa saja yang terlibat, tetapi juga untuk memahami hubungan, pengaruh, serta potensi kontribusi masing-masing stakeholder terhadap pengembangan sistem.

Salah satu teknik yang umum digunakan adalah brainstorming bersama tim proyek untuk mengidentifikasi pihak-pihak yang mungkin terlibat atau terdampak oleh sistem. Selain itu, analisis struktur organisasi juga dapat dilakukan untuk memahami hierarki, peran, serta unit kerja yang berkaitan dengan sistem yang akan dikembangkan. Studi terhadap dokumentasi proyek sebelumnya juga dapat menjadi referensi dalam menemukan pola stakeholder yang relevan (Subarkah et al., 2025).

Teknik lain yang efektif adalah melakukan wawancara awal dengan individu kunci, seperti manajer atau pengguna senior, guna memperoleh informasi lebih luas mengenai stakeholder yang mungkin belum teridentifikasi. Pendekatan ini membantu memperluas cakupan identifikasi serta mengurangi kemungkinan adanya pihak yang terlewat.

Hasil identifikasi kemudian dapat divisualisasikan menggunakan alat seperti stakeholder map atau power-interest matrix untuk mempermudah pemahaman posisi dan tingkat

kepentingan masing-masing stakeholder. Selain itu, dokumentasi yang sistematis dan komunikasi yang jelas juga diperlukan agar hasil identifikasi dapat dipahami dan digunakan sebagai dasar dalam tahap analisis kebutuhan selanjutnya.

Dengan menggunakan metode dan teknik yang tepat, proses identifikasi stakeholder dapat dilakukan secara lebih komprehensif, sehingga mendukung keberhasilan pengembangan sistem secara keseluruhan.

1.2.4 Analisis Kebutuhan dan Ekspektasi Stakeholder

Setelah stakeholder berhasil diidentifikasi, tahap selanjutnya adalah melakukan analisis terhadap kebutuhan dan ekspektasi masing-masing pihak. Analisis ini bertujuan untuk memahami secara mendalam apa yang diinginkan stakeholder, prioritas yang mereka miliki, serta bagaimana mereka menilai keberhasilan sistem yang akan dikembangkan (Syafitri et al., 2023). Proses ini penting untuk memastikan bahwa sistem yang dirancang benar-benar mampu menjawab kebutuhan nyata di lapangan.

Pengumpulan informasi dapat dilakukan melalui berbagai metode, seperti wawancara, diskusi kelompok (focus group discussion), maupun analisis dokumen yang relevan. Setiap metode memiliki kelebihan masing-masing dalam menggali kebutuhan, baik yang bersifat eksplisit maupun implisit. Oleh karena itu, penggunaan kombinasi teknik seringkali diperlukan agar hasil analisis menjadi lebih komprehensif.

Selain mengidentifikasi kebutuhan, analisis ini juga berperan dalam memetakan prioritas serta potensi konflik antar stakeholder.

Perbedaan kepentingan dan sudut pandang dapat menimbulkan ketidaksesuaian kebutuhan, sehingga diperlukan strategi untuk menyeimbangkan dan mengakomodasi berbagai kepentingan tersebut. Pendekatan yang tepat akan membantu meminimalkan risiko konflik serta meningkatkan penerimaan sistem oleh seluruh pihak.

Dengan melakukan analisis kebutuhan dan ekspektasi stakeholder secara sistematis, tim pengembang dapat menghasilkan spesifikasi kebutuhan yang lebih akurat, relevan, dan seimbang. Hal ini pada akhirnya akan meningkatkan kualitas sistem informasi yang dikembangkan serta mendukung keberhasilan implementasinya.

1.3 Teknik Pengumpulan Kebutuhan

1.3.1 Teknik Wawancara dalam Elicitasi Kebutuhan

Wawancara merupakan salah satu teknik utama dalam pengumpulan kebutuhan sistem informasi yang digunakan untuk menggali informasi secara langsung dari stakeholder. Teknik ini memungkinkan analis memperoleh pemahaman yang lebih mendalam mengenai kebutuhan, permasalahan, serta harapan pengguna terhadap sistem yang akan dikembangkan. Wawancara dapat dilakukan secara terstruktur dengan daftar pertanyaan yang telah disusun, maupun semi-terstruktur yang lebih fleksibel dan memungkinkan eksplorasi lebih lanjut terhadap topik tertentu (Tjahyanti & Sutama, 2025).

Keunggulan utama wawancara terletak pada kemampuannya menggali informasi secara detail, termasuk kebutuhan yang bersifat

implisit. Analis juga dapat mengajukan pertanyaan lanjutan untuk memperjelas informasi serta memahami konteks yang lebih luas. Selain itu, interaksi langsung dalam wawancara membantu membangun hubungan yang baik dengan stakeholder, sehingga meningkatkan keterbukaan dalam menyampaikan kebutuhan.

Namun demikian, teknik wawancara memiliki beberapa keterbatasan, seperti membutuhkan waktu yang relatif lama, potensi bias dari pewawancara, serta ketergantungan pada kemampuan komunikasi kedua belah pihak. Tidak semua stakeholder juga merasa nyaman dalam situasi wawancara, terutama jika berkaitan dengan topik yang sensitif.

Oleh karena itu, keberhasilan wawancara sangat ditentukan oleh kesiapan analis, baik dalam menyusun pertanyaan, kemampuan mendengarkan secara aktif, maupun menciptakan suasana yang kondusif. Dengan pelaksanaan yang tepat, wawancara menjadi teknik yang sangat efektif dalam menghasilkan informasi kebutuhan yang akurat dan mendalam.

1.3.2 Observasi dan Studi Dokumentasi

Observasi merupakan teknik pengumpulan kebutuhan yang dilakukan dengan cara mengamati secara langsung aktivitas dan proses bisnis yang berlangsung di organisasi. Melalui observasi, analis dapat memahami bagaimana pekerjaan dilakukan secara nyata, mengidentifikasi hambatan (bottleneck), serta menemukan kebutuhan yang mungkin tidak disadari atau tidak diungkapkan oleh pengguna (Widyastuti et al., 2024). Teknik ini sangat efektif terutama pada proses bisnis yang kompleks atau ketika terdapat

perbedaan antara prosedur yang dijelaskan dengan praktik di lapangan.

Keunggulan observasi terletak pada kemampuannya menangkap kondisi aktual dan perilaku pengguna secara objektif. Analis dapat melihat interaksi pengguna dengan sistem yang ada, alur kerja, serta potensi perbaikan yang diperlukan. Namun, observasi juga memiliki keterbatasan, seperti membutuhkan waktu yang cukup lama dan kemungkinan adanya perubahan perilaku pengguna karena merasa sedang diamati.

Selain observasi, studi dokumentasi juga menjadi teknik penting dalam pengumpulan kebutuhan. Dokumen seperti prosedur operasional standar (SOP), laporan, kebijakan organisasi, serta dokumentasi sistem yang sudah ada dapat memberikan gambaran yang jelas mengenai kebutuhan dan proses bisnis yang berjalan. Dokumen tersebut membantu analis memahami konteks organisasi serta mengidentifikasi kebutuhan yang telah terstruktur sebelumnya.

Kombinasi antara observasi dan studi dokumentasi menghasilkan pemahaman yang lebih menyeluruh karena informasi yang diperoleh berasal dari kondisi nyata di lapangan dan data yang terdokumentasi. Dengan demikian, kebutuhan sistem dapat diidentifikasi secara lebih akurat dan komprehensif (Subarkah et al., 2025).

1.3.3 Teknik Diskusi Kelompok dan Workshop

Diskusi kelompok atau focus group discussion (FGD) merupakan teknik pengumpulan kebutuhan yang melibatkan beberapa stakeholder dalam satu forum untuk membahas kebutuhan

sistem secara bersama-sama. Melalui diskusi ini, analis dapat mengidentifikasi berbagai sudut pandang, menemukan kesamaan maupun perbedaan kebutuhan, serta memahami prioritas dari masing-masing stakeholder (Tjahyanti & Sutama, 2025). Teknik ini juga efektif untuk membangun kesepahaman dan menyepakati kebutuhan secara kolektif.

Workshop merupakan bentuk lanjutan yang lebih terstruktur dari diskusi kelompok, di mana kegiatan dirancang secara sistematis dengan tujuan tertentu, seperti merumuskan kebutuhan, memvalidasi ide, atau menyusun prioritas. Dalam workshop, berbagai metode seperti brainstorming, simulasi skenario (role-playing), maupun prototype walkthrough dapat digunakan untuk membantu stakeholder memahami gambaran sistem secara lebih konkret.

Keunggulan teknik ini terletak pada kemampuannya menghasilkan ide yang lebih beragam dan inovatif melalui interaksi antar peserta. Selain itu, proses diskusi juga dapat mempercepat pengambilan keputusan karena melibatkan banyak pihak secara langsung. Namun, terdapat beberapa tantangan seperti dominasi peserta tertentu, potensi groupthink, serta kesulitan menggali kebutuhan yang bersifat sensitif dalam forum terbuka.

Oleh karena itu, peran fasilitator sangat penting dalam mengarahkan jalannya diskusi agar tetap fokus, seimbang, dan produktif. Dengan pengelolaan yang baik, teknik diskusi kelompok dan workshop dapat menjadi sarana efektif untuk menghasilkan kebutuhan yang lebih komprehensif dan disepakati bersama.

1.3.4 Teknik Prototyping dan Analisis Dokumen Existing

Prototyping merupakan teknik pengumpulan kebutuhan yang dilakukan dengan membuat representasi awal dari sistem yang akan dikembangkan. Bentuk prototipe dapat berupa sketsa sederhana, wireframe, mockup, hingga prototipe interaktif. Melalui prototipe ini, stakeholder dapat melihat gambaran sistem secara lebih nyata sehingga lebih mudah memberikan masukan dan umpan balik (Syafitri et al., 2023). Teknik ini sangat membantu dalam mengklarifikasi kebutuhan yang masih abstrak atau ambigu.

Keunggulan utama prototyping adalah kemampuannya meningkatkan pemahaman bersama antara analis dan stakeholder. Dengan adanya visualisasi, kebutuhan yang sebelumnya sulit dijelaskan menjadi lebih konkret, sehingga risiko kesalahan interpretasi dapat diminimalkan. Selain itu, prototyping juga memungkinkan proses perbaikan dilakukan secara lebih cepat melalui umpan balik berulang.

Namun, penggunaan prototipe juga memiliki keterbatasan, seperti potensi kesalahpahaman stakeholder yang menganggap prototipe sebagai sistem akhir, serta kebutuhan waktu dan sumber daya tambahan untuk pengembangannya. Oleh karena itu, penting untuk menjelaskan bahwa prototipe hanya merupakan alat bantu dalam proses analisis kebutuhan.

Selain prototyping, analisis terhadap dokumen existing juga menjadi sumber informasi yang sangat penting. Dokumen seperti sistem lama, laporan operasional, format input-output, serta catatan penggunaan sistem sebelumnya dapat memberikan gambaran

mengenai kebutuhan yang telah ada dan berkembang. Dari dokumen tersebut, analis dapat mengidentifikasi fungsi yang perlu dipertahankan, diperbaiki, atau dikembangkan lebih lanjut.

Dengan mengombinasikan teknik prototyping dan analisis dokumen existing, proses pengumpulan kebutuhan menjadi lebih akurat dan kontekstual. Pendekatan ini membantu memastikan bahwa sistem yang dikembangkan tidak hanya memenuhi kebutuhan baru, tetapi juga mampu mengakomodasi pengalaman dan kebutuhan yang telah terbentuk sebelumnya.

1.4 Analisis Kebutuhan Fungsional dan Non-Fungsional

1.4.1 Pengertian dan Karakteristik Kebutuhan Fungsional

Kebutuhan fungsional merupakan pernyataan yang menjelaskan apa saja yang harus dilakukan oleh sistem informasi, termasuk fitur dan fungsi yang diperlukan untuk mendukung proses bisnis organisasi. Kebutuhan ini biasanya dinyatakan dalam bentuk spesifikasi yang mencakup input yang diterima sistem, proses yang dilakukan, serta output yang dihasilkan (Subarkah et al., 2025). Sebagai contoh, pada sistem informasi akademik, kebutuhan fungsional dapat berupa kemampuan sistem untuk menerima input nilai, mengolah data, serta menghasilkan laporan hasil belajar siswa.

Karakteristik utama dari kebutuhan fungsional adalah sifatnya yang dapat diuji dan diverifikasi secara objektif. Artinya, setiap fungsi yang dirumuskan harus memiliki kriteria yang jelas sehingga dapat dipastikan apakah sistem telah memenuhi kebutuhan tersebut

atau belum. Oleh karena itu, kebutuhan fungsional harus ditulis secara spesifik, terukur, dan tidak ambigu agar mudah dipahami oleh analis, pengembang, maupun stakeholder.

Selain itu, kebutuhan fungsional harus memiliki keterlacakan (traceability) sepanjang siklus pengembangan sistem. Setiap kebutuhan harus dapat ditelusuri mulai dari tahap analisis hingga implementasi dan pengujian, sehingga tidak ada fungsi yang terlewat atau tidak terimplementasi. Dengan karakteristik tersebut, kebutuhan fungsional menjadi dasar utama dalam perancangan dan pembangunan sistem informasi yang sesuai dengan kebutuhan pengguna (Widyastuti et al., 2024)

1.4.2 Pengertian dan Karakteristik Kebutuhan Non-Fungsional

Kebutuhan non-fungsional merupakan kebutuhan yang tidak berkaitan langsung dengan fungsi spesifik sistem, tetapi lebih menekankan pada kualitas, karakteristik, dan batasan sistem secara keseluruhan. Kebutuhan ini mencakup aspek seperti keamanan, performa, keandalan (reliability), kemudahan penggunaan (usability), skalabilitas, serta kemampuan integrasi dengan sistem lain (Subarkah et al., 2025). Sebagai contoh, kebutuhan non-fungsional dapat berupa pernyataan bahwa sistem harus mampu merespons permintaan pengguna dalam waktu tertentu atau menjamin keamanan data melalui mekanisme autentikasi.

Berbeda dengan kebutuhan fungsional, kebutuhan non-fungsional berfokus pada bagaimana sistem bekerja, bukan apa yang dikerjakan. Oleh karena itu, kebutuhan ini sangat berpengaruh terhadap kualitas pengalaman pengguna dan keberlangsungan

sistem dalam jangka panjang. Sistem yang memiliki fungsi lengkap tetapi tidak memenuhi aspek non-fungsional seperti performa atau keamanan tetap berpotensi gagal dalam implementasi.

Karakteristik kebutuhan non-fungsional cenderung lebih sulit untuk diuji dan diverifikasi karena seringkali bersifat abstrak dan memerlukan parameter yang jelas agar dapat diukur. Oleh karena itu, kebutuhan ini sebaiknya dirumuskan dalam bentuk yang spesifik dan terukur, misalnya dengan menentukan batas waktu respon, tingkat ketersediaan sistem, atau standar keamanan tertentu (Syafitri et al., 2023).

Dengan perumusan yang tepat, kebutuhan non-fungsional dapat menjadi acuan penting dalam memastikan bahwa sistem tidak hanya berjalan sesuai fungsi, tetapi juga memiliki kualitas yang optimal dan mampu memenuhi tuntutan operasional organisasi secara menyeluruh.

1.4.3 Teknik Analisis Kebutuhan Fungsional

Analisis kebutuhan fungsional bertujuan untuk memahami secara mendalam proses bisnis yang akan didukung oleh sistem serta menentukan fungsi-fungsi yang harus disediakan. Proses ini menuntut analis untuk mengidentifikasi alur kerja, interaksi pengguna dengan sistem, serta bagaimana sistem dapat meningkatkan efektivitas dan efisiensi operasional (Subarkah et al., 2025).

Salah satu teknik yang umum digunakan adalah Use Case Analysis, yaitu metode yang menggambarkan interaksi antara pengguna (actor) dan sistem dalam berbagai skenario penggunaan.

Melalui use case, analis dapat mendeskripsikan langkah-langkah yang dilakukan pengguna dan respon sistem secara terstruktur. Hasilnya sering divisualisasikan dalam bentuk use case diagram yang memudahkan pemahaman terhadap fungsi-fungsi utama sistem.

Selain itu, dalam pendekatan Agile, teknik User Stories banyak digunakan untuk mendefinisikan kebutuhan fungsional secara sederhana dan berorientasi pada pengguna. Format umum yang digunakan adalah "Sebagai [peran], saya ingin [fungsi], sehingga [manfaat]". Pendekatan ini membantu menjaga fokus pada nilai yang diberikan sistem kepada pengguna.

Teknik lain yang mendukung analisis kebutuhan fungsional adalah penggunaan flowchart dan Business Process Model and Notation (BPMN) untuk memvisualisasikan alur proses bisnis. Dengan visualisasi ini, analis dapat mengidentifikasi titik-titik penting di mana sistem perlu memberikan dukungan atau melakukan otomatisasi (Widyastuti et al., 2024).

Dengan mengombinasikan berbagai teknik tersebut, analisis kebutuhan fungsional dapat dilakukan secara lebih sistematis dan terstruktur, sehingga menghasilkan spesifikasi yang jelas, mudah dipahami, dan siap untuk diimplementasikan dalam pengembangan sistem.

1.4.4 Teknik Analisis Kebutuhan Non-Fungsional

Analisis kebutuhan non-fungsional memerlukan pendekatan yang lebih sistematis karena berkaitan dengan kualitas dan karakteristik sistem secara keseluruhan. Salah satu kerangka kerja

yang sering digunakan adalah PIECES framework, yang membantu mengidentifikasi kebutuhan dari enam aspek utama, yaitu Performance (kinerja sistem), Information (kualitas informasi), Economics (biaya dan manfaat), Control (keamanan dan pengendalian), Efficiency (efisiensi penggunaan sumber daya), dan Service (kualitas layanan) (Syafitri et al., 2023). Dengan kerangka ini, analis dapat mengevaluasi kebutuhan non-fungsional secara lebih terstruktur dan menyeluruh.

Selain itu, teknik lain yang penting adalah mendefinisikan atribut kualitas dalam bentuk yang spesifik dan terukur. Misalnya, kebutuhan performa dapat dinyatakan dengan batas waktu respon tertentu, kebutuhan ketersediaan sistem dengan persentase uptime, serta kebutuhan keamanan dengan mekanisme autentikasi dan enkripsi data. Pendefinisian yang jelas ini membantu proses pengujian dan evaluasi di tahap implementasi.

Penggunaan skenario atau quality attribute scenarios juga dapat membantu dalam menganalisis kebutuhan non-fungsional. Teknik ini menggambarkan kondisi tertentu yang mungkin terjadi pada sistem, kemudian menentukan bagaimana sistem harus merespons kondisi tersebut. Dengan pendekatan ini, kebutuhan non-fungsional menjadi lebih konkret dan mudah dipahami.

Dengan menerapkan teknik analisis yang tepat, kebutuhan non-fungsional dapat diidentifikasi secara lebih akurat dan tidak bersifat ambigu. Hal ini penting untuk memastikan bahwa sistem yang dikembangkan tidak hanya berfungsi dengan baik, tetapi juga

memiliki kualitas yang sesuai dengan standar dan kebutuhan organisasi.

1.5 Dokumentasi Kebutuhan Sistem

1.5.1 Struktur dan Komponen Dokumen Spesifikasi Kebutuhan Perangkat Lunak

Dokumentasi kebutuhan sistem harus disusun secara terstruktur, jelas, dan komprehensif agar dapat dipahami oleh seluruh pihak yang terlibat dalam proyek, mulai dari stakeholder, manajemen, hingga tim pengembang. Salah satu bentuk dokumentasi yang umum digunakan adalah Software Requirements Specification (SRS), yang berfungsi sebagai acuan utama dalam proses pengembangan sistem.

Secara umum, dokumen SRS terdiri dari beberapa komponen utama. Bagian pendahuluan berisi tujuan, latar belakang, serta ruang lingkup sistem yang akan dikembangkan. Bagian gambaran umum menjelaskan deskripsi tingkat tinggi sistem, termasuk fungsi utama, karakteristik pengguna, serta hubungan sistem dengan lingkungan atau sistem lain (Syafitri et al., 2023).

Selanjutnya, dokumen memuat daftar kebutuhan fungsional yang menjelaskan secara rinci fungsi-fungsi sistem beserta input, proses, dan output yang dihasilkan. Bagian berikutnya adalah kebutuhan non-fungsional yang mencakup aspek kualitas seperti performa, keamanan, usability, dan maintainability yang dirumuskan secara spesifik dan terukur.

Selain itu, dokumen SRS juga dilengkapi dengan berbagai model dan diagram seperti use case diagram, entity relationship diagram (ERD), dan data flow diagram untuk membantu visualisasi struktur serta alur sistem. Terakhir, bagian kriteria penerimaan (acceptance criteria) menjelaskan standar atau kondisi yang harus dipenuhi agar setiap kebutuhan dapat dinyatakan terpenuhi (Subarkah et al., 2025).

Dengan struktur yang jelas dan lengkap, dokumen SRS menjadi alat komunikasi yang efektif sekaligus dasar yang kuat dalam memastikan bahwa sistem dikembangkan sesuai dengan kebutuhan yang telah ditetapkan.

1.5.2 Standar dan Format Dokumentasi Kebutuhan

Penerapan standar dalam dokumentasi kebutuhan sistem sangat penting untuk menjamin konsistensi, kualitas, dan kemudahan pemahaman dokumen oleh seluruh pihak yang terlibat. Salah satu standar yang banyak digunakan adalah ISO/IEC/IEEE 29148, yang memberikan pedoman mengenai struktur dokumen, cara penulisan kebutuhan, serta atribut yang harus disertakan dalam setiap spesifikasi kebutuhan (Syafitri et al., 2023). Dengan mengikuti standar ini, dokumentasi menjadi lebih sistematis dan sesuai dengan praktik terbaik dalam pengembangan sistem.

Selain standar, penggunaan format dan template yang konsisten juga sangat membantu dalam proses penyusunan dan pengelolaan dokumentasi. Template memastikan bahwa setiap kebutuhan ditulis dengan pola yang sama, sehingga memudahkan pembacaan, pencarian informasi, serta analisis perbandingan antar

kebutuhan. Hal ini juga mengurangi risiko adanya informasi penting yang terlewatkan dalam proses dokumentasi.

Dokumentasi yang terstandarisasi tidak hanya bermanfaat pada tahap pengembangan, tetapi juga pada tahap pemeliharaan sistem. Dengan format yang rapi dan konsisten, perubahan kebutuhan dapat dicatat dengan jelas, serta riwayat revisi dapat ditelusuri dengan mudah. Oleh karena itu, penerapan standar dan format dokumentasi menjadi salah satu faktor penting dalam menjaga kualitas dan keberlanjutan sistem informasi (Widyastuti et al., 2024).

1.5.3 Proses Verifikasi dan Validasi Dokumentasi Kebutuhan

Sebelum dokumentasi kebutuhan digunakan sebagai dasar pengembangan sistem, diperlukan proses verifikasi dan validasi untuk memastikan kualitas dan kesesuaiannya. Verifikasi bertujuan untuk memastikan bahwa dokumen telah disusun dengan benar, lengkap, konsisten, serta sesuai dengan standar yang digunakan. Proses ini biasanya dilakukan melalui review internal, penggunaan checklist, serta pemeriksaan kesesuaian antar bagian dokumen (Syafitri et al., 2023).

Sementara itu, validasi bertujuan untuk memastikan bahwa kebutuhan yang terdokumentasi benar-benar mencerminkan kebutuhan dan harapan stakeholder. Proses ini dilakukan dengan melibatkan stakeholder secara langsung, misalnya melalui walkthrough, presentasi dokumen, atau diskusi untuk memperoleh umpan balik dan persetujuan. Validasi juga dapat didukung dengan penggunaan prototipe atau proof of concept untuk memastikan

bahwa kebutuhan yang dirumuskan dapat diimplementasikan secara teknis.

Kedua proses ini saling melengkapi dalam menjamin kualitas dokumentasi kebutuhan. Verifikasi memastikan dokumen “dibuat dengan benar”, sedangkan validasi memastikan “kebutuhan yang ditulis adalah benar”. Dengan melakukan verifikasi dan validasi secara sistematis, risiko kesalahan, ketidaksesuaian, maupun kebutuhan yang terlewatkan dapat diminimalkan (Subarkah et al., 2025).

1.5.4 Manajemen Perubahan dan Pemeliharaan Dokumentasi Kebutuhan

Dokumentasi kebutuhan sistem bersifat dinamis dan dapat mengalami perubahan seiring perkembangan proyek, perubahan kebutuhan organisasi, maupun penyesuaian terhadap kondisi teknis. Oleh karena itu, diperlukan mekanisme change management yang terstruktur untuk mengelola setiap perubahan secara sistematis (Widyastuti et al., 2024). Proses ini mencakup pengajuan perubahan, analisis dampak, proses persetujuan oleh stakeholder, serta pencatatan perubahan secara lengkap.

Manajemen perubahan yang baik membantu memastikan bahwa setiap perubahan kebutuhan telah melalui pertimbangan yang matang dan tidak menimbulkan risiko terhadap konsistensi sistem. Selain itu, adanya sistem pelacakan (tracking) memungkinkan tim untuk mengetahui riwayat perubahan, alasan perubahan, serta pihak yang menyetujui, sehingga transparansi dan akuntabilitas tetap terjaga.

Di sisi lain, pemeliharaan dokumentasi juga menjadi aspek penting agar dokumen tetap relevan dan akurat sepanjang siklus hidup sistem. Hal ini dapat dilakukan melalui penerapan manajemen konfigurasi, pengelolaan versi dokumen, serta pembaruan berkala sesuai dengan perkembangan proyek. Dokumentasi yang terpelihara dengan baik akan memudahkan tim dalam melakukan pengembangan lanjutan, perbaikan sistem, maupun proses audit.

Dengan adanya manajemen perubahan dan pemeliharaan yang efektif, dokumentasi kebutuhan tidak hanya menjadi arsip, tetapi juga menjadi alat kerja yang aktif dan mendukung keberhasilan pengembangan sistem secara berkelanjutan (Syafitri et al., 2023).

BAB 2: Perencanaan Pengembangan Sistem

Ali Haidir, M.Kom.

Perencanaan pengembangan sistem merupakan fondasi penting dalam keberhasilan implementasi proyek sistem informasi. Fase ini mencakup serangkaian strategi aktivitas yang dirancang untuk memahami kebutuhan bisnis, menentukan pendekatan pengembangan yang tepat, mengalokasikan sumber daya dengan efisien, dan mengidentifikasi serta mengelola risiko yang mungkin timbul. Pentingnya perencanaan yang mencakup kemampuan untuk memberikan arah yang jelas kepada seluruh tim proyek, memastikan bahwa semua pemangku kepentingan memiliki pemahaman yang sama tentang tujuan dan strategi pengembangan, serta meminimalkan izin yang dapat menyebabkan keterlambatan atau pembengkakan biaya (Pargaonkar, 2023).

Dalam konteks industri modern yang dinamis, sistem perencanaan dan pengembangan harus fleksibel namun tetap terstruktur, mampu mengakomodasi perubahan tanpa mengorbankan integritas proyek secara keseluruhan. Bab ini akan menguraikan secara mendalam tentang komponen-komponen utama perencanaan sistem pengembangan, mulai dari pemahaman siklus hidup sistem pengembangan, pemilihan model pengembangan yang

sesuai, perencanaan terperinci tentang pelaksanaan proyek, penjadwalan dan estimasi biaya yang akurat, hingga manajemen risiko yang proaktif dan terencana.

2.1 Siklus Hidup Pengembangan Sistem (SDLC)

2.1.1 Pengertian dan Konsep Dasar SDLC

Siklus hidup sistem pengembangan, yang dikenal dengan istilah *Software Development* Life Cycle (SDLC), adalah kerangka metodologi yang komprehensif untuk mengelola setiap aspek informasi sistem pengembangan dari tahap konsepisasi awal hingga pemeliharaan dan peningkatan berkelanjutan. SDLC menyediakan struktur sistematis yang memandu tim pengembang, manajer proyek, dan pemangku kepentingan melalui serangkaian fase yang terdefinisi dengan jelas, masing-masing dengan tujuan spesifik, penyampaian yang diharapkan, dan kriteria keberhasilan yang terukur (Yas et al., 2023). Keadaan ini memastikan bahwa pengembangan sistem dilakukan dengan cara yang terkendali, terukur, dan dapat dipertanggungjawabkan.

Konsep dasar SDLC berpijak pada pemahaman bahwa sistem pengembangan adalah sebuah proses yang kompleks dan tidak dapat ditangani secara ad-hoc. Dengan menerapkan SDLC, organisasi dapat mengurangi pemanasan, meningkatkan kualitas produk akhir, dan menciptakan lingkungan kerja yang terorganisir dan efisien. SDLC juga berfungsi sebagai alat komunikasi yang efektif antar berbagai pemangku kepentingan, karena memberikan bahasa umum

dan Struktur yang dipahami oleh semua pihak yang terlibat dalam proyek.

2.1.2 Fase-Fase Utama SDLC

Meskipun detail implementasi dapat berbeda-beda tergantung pada model SDLC yang dipilih, secara umum lima fase utama yang menjadi fondasi dari setiap SDLC. Fase pertama adalah perencanaan (*planning*), di mana organisasi mengidentifikasi kebutuhan bisnis, menentukan tujuan proyek, mengalokasikan sumber daya awal, dan menetapkan ruang lingkup proyek. Dalam tahap ini, studi kelayakan dilakukan untuk menilai kelayakan teknis, ekonomi, dan operasional dari proyek yang diusulkan (Pargaonkar, 2023).

Fase kedua adalah analisis (*analysis*), di mana persyaratan bisnis diterjemahkan menjadi spesifikasi teknis yang detail. Tim analis bekerja sama dengan pengguna akhir dan pemangku kepentingan untuk mengumpulkan, mendokumentasikan, dan memvalidasi semua persyaratan fungsional dan non-fungsional. Fase ketiga adalah desain (*design*), di mana arsitektur sistem, struktur database, antarmuka pengguna, dan komponen-komponen teknis lainnya dirancang berdasarkan spesifikasi yang telah divalidasi. Desain mencakup *desain tingkat tinggi (high-level* design) yang menggambarkan struktur keseluruhan sistem dan desain tingkat rendah (low -*level* design) yang menggambarkan detail implementasi setiap komponen.

Fase keempat adalah implementasi (*implementation*) atau konstruksi, di mana kode program ditulis berdasarkan desain yang telah disetujui. Pengembang mengikuti standar pengkodean yang

ditetapkan, melakukan pengujian unit, dan memastikan kualitas kode melalui code review. Tahap kelima adalah pengujian *(testing)*, dimana sistem diuji secara menyeluruh untuk memastikan bahwa semua fungsi bekerja sesuai dengan spesifikasi dan persyaratan. Pengujian meliputi pengujian unit, pengujian integrasi, pengujian sistem, dan pengujian penerimaan pengguna. Fase terakhir adalah pemeliharaan dan dukungan (*maintenance and support*), di mana sistem memantau setelah implementasi, perbaikan bug, dan peningkatan dilakukan berdasarkan umpan balik pengguna (Chahar & Singh, 2024).

2.1.3 Manfaat dan Keuntungan Implementasi SDLC

Implementasi SDLC yang tepat memberikan sejumlah manfaat signifikan bagi organisasi dan sistem proyek informasi. Pertama, SDLC menyediakan kerangka kerja yang terstruktur yang mengurangi kenyamanan dan meningkatkan kontrol atas proses pengembangan. Dengan fase-fase yang jelas dan terdefinisi, manajer proyek dapat melacak kemajuan dengan lebih akurat dan mengidentifikasi masalah lebih awal. Kedua, SDLC memfasilitasi komunikasi yang lebih baik antar tim dan pemangku kepentingan dengan menyediakan dokumentasi yang komprehensif dan konsisten. Ketiga, penerapan SDLC meningkatkan kualitas sistem akhir karena setiap fase mencakup jaminan kualitas dan peninjauan yang memastikan produk memenuhi standar kualitas yang ditetapkan.

Keempat, SDLC membantu pengendalian biaya dan jadwal melalui perencanaan yang matang dan pemantauan yang konsisten.

Kelima, SDLC menciptakan basis pengetahuan yang dapat digunakan untuk proyek-proyek mendatang, memfasilitasi pembelajaran organisasi dan perbaikan berkelanjutan. Keenam, SDLC memastikan kepatuhan terhadap regulasi dan standar industri yang berlaku, karena proses yang terstruktur memudahkan dokumentasi dan jejak audit. Ketujuh, SDLC meningkatkan kepuasan pengguna karena keterlibatan pengguna dalam berbagai fase pengembangan memastikan bahwa sistem yang dihasilkan sesuai dengan kebutuhan dan harapan mereka (Yas et al., 2023).

2.1.4 Tantangan dan Batasan SDLC

Meskipun SDLC menawarkan banyak manfaat, penerapannya juga mampu mengatasi berbagai tantangan dan batasan yang perlu dipertimbangkan. Salah satu tantangan utama adalah struktur kaku dari beberapa model SDLC, khususnya model Waterfall, yang membuat sulit untuk mengakomodasi perubahan persyaratan yang tidak memerlukan antisipasi di awal proyek. Dalam lingkungan bisnis yang dinamis, kebutuhan pengguna dapat berubah dengan cepat, dan ketidakmampuan model SDLC tradisional untuk beradaptasi dapat menghasilkan sistem yang sudah usang sebelum implementasi selesai.

Tantangan lain adalah memperkirakan dampak positif dari struktur terhadap hasil akhir. Struktur yang ketat tidak menjamin keberhasilan jika tim tidak memiliki keahlian yang memadai atau jika komunikasi antar pemangku kepentingan tidak efektif. Selain itu, penerapan SDLC yang rigid dapat meningkatkan overhead administratif dan mengurangi kelincahan waktu dalam merespons

perubahan. Ada juga risiko dari kelumpuhan analisis, dimana terlalu banyak waktu yang dihabiskan untuk analisis dan perencanaan sehingga implementasinya tertunda. Organisasi perlu menemukan keseimbangan antara struktur yang diperlukan untuk kontrol dan sinkronisasi yang diperlukan untuk beradaptasi dengan perubahan (Pargaonkar, 2023).

2.2 Model Pengembangan Sistem

2.2.1 Model Air Terjun dan Karakteristiknya

Model Waterfall, juga dikenal sebagai model sekuensial atau linier, adalah salah satu model SDLC paling tradisional yang telah digunakan selama beberapa dekade dalam industri pengembangan sistem informasi. Dalam model ini, fase-fase pengembangan dilaksanakan secara berurutan, di mana setiap fase harus diselesaikan sepenuhnya sebelum fase berikutnya dimulai. Output dari satu fase menjadi input untuk fase berikutnya, menciptakan aliran kerja yang dapat divisualisasikan seperti air terjun yang bergerak ke bawah (Chahar & Singh, 2024).

Karakteristik utama model Waterfall mencakup persyaratan yang didefinisikan dengan jelas di awal proyek, perencanaan yang komprehensif sebelum implementasi dimulai, dan ringkasan yang jelas antara berbagai fase pengembangan. Model ini sangat cocok untuk proyek-proyek dengan persyaratan yang stabil, cakupan yang jelas, dan tingkat perubahan yang rendah. Keuntungan model Waterfall adalah kemudahan dalam perencanaan dan pelacakan

kemajuan, dokumentasi yang lengkap di setiap tahapan, dan kemudahan dalam mengelola proyek dengan tim yang terdistribusi atau besar. Namun, kelemahan model ini termasuk kekecewaan dalam menghadapi perubahan, identifikasi cacat terlambat yang menyebabkan biaya perbaikan tinggi, dan ketidakmampuan untuk melibatkan pengguna secara signifikan hingga fase pengujian atau bahkan implementasi (Pargaonkar, 2023).

2.2.2 Model Agile dan Metode Iteratif

Model Agile merepresentasikan paradigma yang berbeda dalam pengembangan sistem informasi, tekanan pada responsivitas terhadap perubahan, intensifikasi kolaborasi dengan pengguna, dan pengiriman nilai secara inkremental dalam periode waktu yang relatif singkat. Model Agile bekerja dalam iterasi pendek yang biasanya berlangsung dua hingga empat minggu, yang disebut *sprint* . Dalam setiap sprint, tim mengembangkan subset dari fungsionalitas yang dapat langsung diuji dan dievaluasi oleh pengguna, memungkinkan umpan balik real-time dan penyesuaian untuk sprint berikutnya (Yas et al., 2023).

Prinsip-prinsip fundamental model Agile mencakup individu dan interaksi dalam proses dan alat, perangkat lunak yang berfungsi dalam dokumentasi yang komprehensif, kolaborasi pelanggan dalam negosiasi kontrak, dan merespons perubahan dalam mengikuti rencana. Model Agile mengakui bahwa persyaratan tidak dapat didefinisikan dengan sempurna di awal proyek, dan bahwa perubahan adalah bagian alami dari pengembangan produk modern. Keuntungan model Agile mencakup tinggi dalam mengakomodasi

perubahan persyaratan, kualitas perangkat lunak yang lebih tinggi karena pengukuran berkelanjutan, tingkat kepuasan pelanggan yang lebih tinggi karena keterlibatan aktif, dan identifikasi awal masalah teknis. Namun, model Agile juga memiliki tantangan, termasuk kesulitan dalam merencanakan jadwal dan anggaran jangka panjang, ketergantungan pada ketersediaan dan komitmen pelanggan untuk berpartisipasi aktif, dan potensi untuk scope creep jika tidak dikelola dengan ketat (Chahar & Singh, 2024).

2.2.3 Model Hibrida dan Spiral

Model hybrid menggabungkan elemen-elemen terbaik dari model Waterfall dan Agile untuk menciptakan pendekatan yang disesuaikan dengan konteks proyek tertentu. Dalam model hybrid, fase-fase perencanaan awal dan desain tingkat tinggi dilakukan menggunakan pendekatan yang terstruktur mirip dengan Waterfall, memberikan kejelasan dan kontrol yang diperlukan untuk proyek besar atau kompleks. Namun, tahap implementasi dan pengujian menggunakan prinsip-prinsip Agile dengan iterasi dan feedback loop yang lebih ketat, memungkinkan permulaan dan responsivitas terhadap perubahan.

Model Spiral, yang diperkenalkan oleh Barry Boehm, adalah model berbasis risiko yang menggabungkan pembuatan prototipe dengan iterasi sistematis dan perhatian khusus terhadap manajemen risiko. Dalam model ini, pengembangan dilakukan dalam putaran spiral, di mana setiap putaran mencakup empat fase utama: perencanaan, analisis risiko, implementasi, dan evaluasi. Model Spiral sangat cocok untuk proyek-proyek besar dengan tingkat risiko

tinggi, karena memungkinkan identifikasi dan mitigasi risiko pada tahap awal. Model keuntungan Spiral adalah kemampuan menangani proyek yang kompleks dengan risiko tinggi, mengatur dalam mengakomodasi perubahan, dan pengurangan risiko melalui prototipe dan evaluasi iteratif. Namun, model ini memerlukan keahlian yang tinggi dalam manajemen risiko dan dapat lebih mahal dalam hal overhead proyek (Pargaonkar, 2023).

2.2.4 Pemilihan Model yang Tepat untuk Konteks Proyek

Pemilihan model pengembangan yang tepat adalah keputusan strategi yang sangat mempengaruhi keberhasilan proyek sistem informasi. Tidak ada satu model yang universal cocok untuk semua situasi; sebaliknya, pilihan model harus didasarkan pada karakteristik spesifik dari proyek dan organisasi. Faktor-faktor yang harus dipertimbangkan dalam pemilihan model meliputi kejelasan dan stabilitas dari persyaratan, ukuran dan kompleksitas proyek, tingkat risiko yang dihadapi, ketersediaan dan stabilitas waktu, budaya organisasi, dan garis waktu yang diberikan.

Untuk proyek-proyek kecil dengan persyaratan yang jelas dan stabil, model Waterfall mungkin adalah pilihan yang paling efisien. Untuk proyek-proyek dengan persyaratan yang tidak pasti atau sering berubah, dan di mana keterlibatan pelanggan yang tinggi dimungkinkan, model Agile akan lebih efektif. Untuk proyek-proyek berskala enterprise dengan risiko tinggi, model Hybrid atau Spiral akan memberikan keseimbangan terbaik antara pengendalian dan kelangsungan. Penelitian menunjukkan bahwa tidak ada model yang unggul secara universal, tetapi hasil yang lebih baik dicapai

ketika model yang dipilih sesuai dengan konteks proyek (Yas et al., 2023). Organisasi yang matang sering kali memiliki multiple model dalam toolkit mereka dan memilih atau mengadaptasi model berdasarkan kebutuhan spesifik setiap proyek.

2.3 Perencanaan Sistem Proyek Informasi

2.3.1 Analisis Kebutuhan dan Definisi Ruang Lingkup Proyek

Fase perencanaan analisis proyek dimulai dengan yang mendalam tentang kebutuhan bisnis dan pendefinisian ruang lingkup proyek yang jelas. Proses ini melibatkan wawancara intensif dengan berbagai pemangku kepentingan untuk memahami masalah-masalah yang dihadapi organisasi saat ini, peluang untuk perbaikan, dan bagaimana sistem informasi dapat berkontribusi terhadap pencapaian tujuan strategi. Kebutuhan dokumentasi harus mencakup baik kebutuhan fungsional yang menggambarkan apa yang harus dilakukan sistem, maupun kebutuhan non-fungsional yang berkaitan dengan kinerja, keamanan, skalabilitas, dan pemeliharaan.

Scope proyek harus didefinisikan dengan jelas untuk menghindari ambiguitas dan scope creep di kemudian hari. Pernyataan ruang lingkup harus mencakup apa yang akan dan tidak akan dimasukkan ke dalam proyek, memberikan kejelasan kepada semua pemangku kepentingan tentang batasan pengembangan. Praktik terbaik dalam analisis kebutuhan meliputi penggunaan berbagai teknik elisitasi seperti wawancara, diskusi kelompok terfokus, observasi, dan pembuatan prototipe untuk mengumpulkan

kebutuhan yang komprehensif dan akurat. Persyaratan yang dikumpulkan harus dijelaskan untuk kelengkapan, konsistensi, dan kelayakan sebelum disetujui sebagai baseline untuk pengembangan (Pargaonkar, 2023).

2.3.2 Identifikasi Stakeholder dan Manajemen Komunikasi

Identifikasi yang akurat terhadap semua pemangku kepentingan yang memiliki kepentingan dalam proyek adalah kritis untuk keberhasilan proyek. Analisis pemangku kepentingan harus mengklasifikasikan pemangku kepentingan berdasarkan tingkat kepentingan dan pengaruh mereka terhadap proyek. Stakeholder yang memiliki tingkat kepentingan dan pengaruh tinggi harus dikelola secara proaktif dan dilibatkan secara signifikan dalam pengambilan keputusan proyek. Strategi komunikasi yang efektif harus dikembangkan untuk memastikan bahwa informasi yang relevan dikomunikasikan kepada pemangku kepentingan yang tepat pada waktu yang tepat dalam format yang sesuai.

Saluran komunikasi harus ditetapkan dengan jelas, mencakup pertemuan rutin, laporan status, dan forum umpan balik. Tim proyek harus proaktif dalam mengidentifikasi dan mengatasi kekhawatiran pemangku kepentingan, membangun kepercayaan melalui transparansi dan responsivitas. Dokumentasi dari semua komunikasi penting harus disimpan untuk referensi dan jejak audit. Rencana komunikasi yang dikembangkan dengan baik akan mengurangi kesalahpahaman, meningkatkan dukungan dari pemangku kepentingan, dan memfasilitasi pengambilan keputusan yang lebih baik (Yas et al., 2023).

2.3.3 Alokasi Sumber Daya dan Pembentukan Tim Proyek

Alokasi sumber daya yang efisien merupakan kunci untuk memastikan bahwa proyek dapat dilaksanakan secara tepat waktu dan sesuai anggaran. Perencanaan sumber daya harus mempertimbangkan berbagai jenis sumber daya yang diperlukan, termasuk sumber daya manusia dengan berbagai keahlian, teknologi dan infrastruktur, fasilitas kerja, dan anggaran finansial. Untuk sumber daya manusia, harus diidentifikasi keahlian spesifik yang diperlukan untuk setiap fase proyek, tingkat ketersediaan sumber daya internal, dan kapan sumber daya eksternal atau outsourcing diperlukan.

Tim inti proyek biasanya terdiri dari manajer proyek yang bertanggung jawab untuk koordinasi keseluruhan, analis bisnis yang mengelola persyaratan, arsitek sistem yang merancang solusi teknis, pengembang yang mengimplementasikan sistem, dan spesialis jaminan kualitas yang melakukan pengujian. Peran dan tanggung jawab setiap anggota tim harus didefinisikan dengan jelas dalam piagam proyek dan matriks penugasan tanggung jawab. Kegiatan team building harus diadakan untuk memfasilitasi ikatan dan memastikan bahwa tim mencapai tujuan proyek dan berkomitmen untuk mencapainya. Investasi dalam pelatihan dan pengembangan akan meningkatkan kemampuan tim dan produktivitas secara keseluruhan (Chahar & Singh, 2024).

2.3.4 Dokumentasi Perencanaan dan Pembentukan Baseline

Dokumentasi perencanaan yang komprehensif adalah hasil nyata dari fase perencanaan yang memberikan referensi bagi semua

pihak yang terlibat dalam proyek. Piagam proyek adalah dokumen formal yang menetapkan otorisasi proyek, tujuan tingkat tinggi, persyaratan tingkat tinggi, dan kriteria keberhasilan. Rencana manajemen proyek menjabarkan bagaimana proyek akan direncanakan, dilaksanakan, dan dikontrol dalam berbagai bidang pengetahuan. Spesifikasi persyaratan rinci mendokumentasikan semua persyaratan dengan tingkat detail yang cukup untuk implementasi.

Baseline proyek, yang terdiri dari scope baseline, schedule baseline, dan cost baseline, harus ditetapkan sebagai patokan untuk mengukur kemajuan proyek. Scope baseline mengdefenisikan hasil dan pekerjaan yang akan dan tidak akan dilakukan dalam proyek. Jadwal baseline menentukan durasi dan urutan aktivitas serta tonggak pencapaian kunci. Cost baseline menunjukkan distribusi biaya proyek sepanjang timeline. Freeze formal dari baseline dilakukan setelah persetujuan dari steering committe, setelah itu setiap perubahan harus melalui proses kontrol perubahan formal. Dokumentasi yang terorganisir dengan baik dan mudah diakses akan memfasilitasi komunikasi yang lebih baik dan pengambilan keputusan yang lebih cepat (Pargaonkar, 2023).

2.4 Penjadwalan dan Estimasi Biaya

2.4.1 Teknik Estimasi Durasi dan Upaya

Estimasi durasi dan usaha merupakan salah satu aktivitas yang paling menantang dalam sistem informasi perencanaan proyek,

karena estimasi estimasi secara langsung mempengaruhi jadwal dan biaya proyek. Berbagai teknik estimasi dapat digunakan, tergantung pada tingkat detail yang tersedia dan keahlian yang dimiliki tim. Expert judgement memanfaatkan pengalaman dan intuisi dari para profesional berpengalaman yang telah menangani proyek serupa sebelumnya. Meskipun berguna, teknik ini dapat bias oleh optimisme atau pesimisme individu.

Estimasi analog membandingkan proyek saat ini dengan proyek-proyek historis yang serupa dan menggunakan data aktual dari proyek tersebut sebagai dasar estimasi. Teknik ini lebih akurat jika proyek historis memiliki karakteristik yang sangat mirip dengan proyek baru. Estimasi bottom-up melibatkan pemecahan pekerjaan menjadi unit-unit kecil yang dapat diestimasi dengan akurat, kemudian mengagregasikan estimasi untuk mendapatkan total. Teknik ini umumnya menghasilkan estimasi yang lebih akurat tetapi memerlukan waktu dan tenaga yang signifikan. Estimasi parametrik menggunakan model statistik yang berbasis pada data historis dan parameter proyek untuk menghasilkan estimasi. Model seperti COCOMO digunakan dalam perangkat lunak industri untuk mengestimasi upaya berdasarkan ukuran, kompleksitas, dan faktor tim (Maksymov, 2025).

2.4.2 Pengembangan Jadwal Proyek dan identifikasi Critical Path

Setelah durasi untuk setiap aktivitas diestimasi, langkah berikutnya adalah mengembangkan jadwal proyek yang menunjukkan urutan dan ketergantungan antar aktivitas. Metode

Jalur Kritis (CPM) adalah teknik yang digunakan untuk mengidentifikasi jalur kritis, yaitu rangkaian aktivitas yang memiliki durasi total paling lama dan menentukan durasi minimum keseluruhan proyek. Setiap tertunda pada aktivitas di jalur kritis akan menghasilkan tertunda pada tanggal penyelesaian proyek secara keseluruhan, sehingga aktivitas-aktivitas ini memerlukan pemantauan dan pengendalian yang ketat.

Aktivitas-aktivitas yang tidak berada pada jalur kritis memiliki slack atau float time, yang berarti mereka dapat melanjutkan dalam batasan tertentu tanpa mempengaruhi jadwal keseluruhan proyek. Pemahaman tentang jalur kritis sangat penting untuk memfokuskan upaya pengelolaan pada aktivitas-aktivitas yang benar-benar kritis. Resource leveling dapat dilakukan untuk menyelaraskan kebutuhan sumber daya dengan ketersediaannya, meskipun hal ini dapat memperpanjang durasi proyek. Perangkat lunak manajemen proyek seperti Microsoft Project, Primavera, atau Asana dapat digunakan untuk mengembangkan dan mengelola jadwal yang kompleks, menyediakan bagan Gantt, diagram jaringan, dan laporan untuk analisis dan komunikasi (Chahar & Singh, 2024).

2.4.3 Estimasi Biaya dan Pengendalian Anggaran

Estimasi biaya melibatkan prediksi tentang sumber daya finansial yang akan dibutuhkan untuk menyelesaikan proyek. Estimasi biaya harus mencakup semua kategori biaya termasuk tenaga kerja, teknologi dan peralatan, infrastruktur, pelatihan, dan kontinjensi. Keakuratan dalam memperkirakan biaya sangat penting, karena perkiraan yang terlalu rendah dapat mengakibatkan

pembengkakan anggaran yang membahayakan kelangsungan proyek, sementara perkiraan yang terlalu tinggi dapat mengakibatkan pemanfaatan sumber daya yang tidak efisien.

Pengendalian biaya meliputi pemantauan biaya aktual terhadap biaya yang dianggarkan, mengidentifikasi varians, menganalisis penyebab, dan mengambil tindakan perbaikan untuk mencegah pembengkakan biaya. Earned Value Management (EVM) adalah teknik yang kuat untuk mengintegrasikan ruang lingkup, jadwal, dan biaya dalam pengukuran kinerja proyek. EVM memungkinkan manajer proyek untuk melacak kinerja secara real-time dengan membandingkan nilai yang diperoleh (nilai pekerjaan yang diselesaikan) terhadap nilai yang direncanakan (nilai pekerjaan yang seharusnya selesai) dan biaya aktual (biaya yang benar-benar dikeluarkan). Metrik utama termasuk Indeks Kinerja Jadwal (SPI) dan Indeks Kinerja Biaya (CPI) yang menunjukkan efisiensi pelaksanaan proyek (Pargaonkar, 2023).

2.4.4 Manajemen Kontinjensi dan Cadangan

Karena pengembangan sistem informasi mengandung ketidakpastian yang melekat, cadangan kontinjensi harus dimasukkan ke dalam perkiraan jadwal dan biaya untuk mengakomodasi risiko-risiko yang tidak dapat diprediksi. Kontinjensi untuk biaya biasanya dihitung sebagai persentase dari total perkiraan biaya, dengan persentase ini berdasarkan penilaian risiko proyek. Demikian pula jadwal kontingensi dalam bentuk buffer time harus dialokasikan untuk aktivitas-aktivitas dengan

tingkat ketidakpastian tinggi, khususnya yang berada pada jalur kritis.

Cadangan darurat harus dikelola dengan ketat untuk memastikan bahwa cadangan ini digunakan hanya untuk kejadian yang benar-benar tidak terduga, bukan untuk perencanaan atau pelaksanaan yang buruk. Threshold atau trigger points harus ditentukan untuk mengizinkan penggunaan cadangan kontinjensi, memastikan konsistensi dalam pengelolaan. Cadangan manajemen adalah penyangga tambahan yang biasanya ditahan di tingkat komite pengarah untuk kejadian tak terduga atau perubahan strategis yang tidak dapat diantisipasi saat perencanaan awal. Transparansi dalam pelaporan tentang penggunaan kontinjensi dan cadangan manajemen akan membantu pemangku kepentingan memahami posisi keuangan dari proyek dan membuat keputusan yang tepat (Yas et al., 2023).

2.5 Manajemen Risiko Proyek

2.5.1 Identifikasi dan Kategorisasi Risiko

Manajemen risiko adalah proses sistematis untuk mengidentifikasi, menganalisis, merespons, dan memonitor risiko-risiko yang dapat mempengaruhi pencapaian tujuan proyek. Proses identifikasi risiko melibatkan pencarian proaktif terhadap potensi risiko yang dapat berdampak pada proyek. Berbagai teknik dapat digunakan untuk mengidentifikasi risiko, termasuk brainstorming dengan tim proyek, meninjau catatan sejarah dari proyek

sebelumnya, konsultasi dengan ahli di bidangnya, dan analisis asumsi dan batasan yang mendasari perencanaan proyek.

Risiko-risiko yang teridentifikasi harus dikategorisasi berdasarkan sumber-sumbernya untuk memfasilitasi analisis dan perencanaan respons yang lebih tepat sasaran. Kategori umum risiko dalam sistem informasi proyek mencakup risiko teknis yang berkaitan dengan pilihan teknologi dan keputusan arsitektur, risiko personel yang berkaitan dengan komposisi tim dan ketersediaan keterampilan, risiko organisasi yang berkaitan dengan struktur dan proses organisasi, risiko eksternal yang berasal dari faktor-faktor di luar organisasi seperti perubahan pasar atau perubahan peraturan, dan risiko manajemen proyek yang berkaitan dengan perencanaan, komunikasi, atau manajemen ruang lingkup (Maksymov, 2025).

2.5.2 Analisis Kualitatif dan Kuantitatif Risiko

Setelah risiko diidentifikasi dan dikategorisasi, langkah berikutnya adalah menganalisis risiko untuk memahami dampak dan probabilitas dari setiap risiko. Analisis risiko kualitatif melibatkan penilaian dampak dan probabilitas menggunakan skala ordinal seperti skala rendah, sedang, tinggi, atau numerik. Risiko-risiko ini kemudian diplot dalam matriks risiko yang menunjukkan kombinasi dari probabilitas dan dampak, memungkinkan penentuan prioritas risiko-risiko untuk perencanaan respons. Risiko dengan probabilitas tinggi dan dampak tinggi adalah prioritas tinggi dan memerlukan strategi respon yang aktif.

Analisis risiko kuantitatif melibatkan perhitungan numerik dari dampak dan probabilitas, sering kali menggunakan metode

probabilistik seperti analisis pohon keputusan, simulasi Monte Carlo, atau analisis sensitivitas. Teknik-teknik ini menghasilkan distribusi statistik dari kemungkinan hasil, memberikan manajer proyek dengan informasi yang lebih rinci tentang rentang hasil proyek yang mungkin dan probabilitas dari pencapaian atau melebihi target. Daftar risiko adalah dokumentasi formal dari semua risiko yang teridentifikasi, termasuk deskripsi, probabilitas, dampak, prioritas, strategi respons, pemilik, dan status. Daftar risiko harus dokumen hidup yang terus diperbarui sepanjang siklus hidup proyek (Maksymov, 2025).

2.5.3 Strategi Respon Risiko dan Implementasi Mitigasi

Setelah risiko dijelaskan dan diprioritaskan, strategi respons risiko harus dikembangkan untuk menangani risiko-risiko tersebut secara efektif. Terdapat empat strategi utama dalam respon terhadap risiko, yaitu menghindari, memitigasi, mentransfer, dan menerima. Penghindaran risiko melibatkan perubahan ruang lingkup, pendekatan, atau pilihan teknologi untuk mengeliminasi risiko sepenuhnya. Meskipun ini adalah strategi yang ideal, sering kali tidak praktis karena dapat mengubah sifat proyek atau tidak selaras dengan tujuan bisnis.

Mitigasi risiko adalah strategi proaktif untuk mengurangi probabilitas atau dampak risiko. Tindakan mitigasi harus dirancang untuk mengatasi akar penyebab risiko dan harus hemat biaya. Pengalihan risiko melibatkan pengalihan risiko kepada pihak ketiga melalui mekanisme seperti asuransi, outsourcing, atau perjanjian vendor. Penerimaan risiko adalah strategi reaktif di mana keputusan

dibuat untuk menerima konsekuensi potensial dari risiko dan tidak mengambil tindakan pencegahan, biasanya karena biaya mitigasi lebih tinggi daripada dampak yang diharapkan. Untuk setiap risiko berprioritas tinggi, rencana aksi harus dikembangkan yang secara jelas menentukan tindakan mitigasi, penanggung jawab, jangka waktu, dan sumber daya yang diperlukan (Chahar & Singh, 2024).

2.5.4 Pemantauan dan Pengendalian Risiko Sepanjang Proyek

Manajemen risiko bukanlah aktivitas satu kali pada perencanaan fase, melainkan proses berkelanjutan yang harus dilakukan sepanjang siklus hidup proyek. Pemantauan risiko meliputi pelacakan terus menerus terhadap status risiko yang teridentifikasi, pengamatan terhadap tanda peringatan dini, pencarian risiko baru yang mungkin muncul, dan evaluasi efektivitas respons risiko. Dalam setiap peninjauan status proyek, risiko harus dievaluasi ulang untuk menilai apakah probabilitas atau dampak telah berubah, apakah respons risiko telah efektif, dan apakah risiko baru telah teridentifikasi.

Pengendalian risiko melibatkan pengambilan tindakan korektif untuk mengatasi risiko yang telah terwujud atau tindakan mitigasi untuk mencegah realisasi risiko yang diantisipasi. Jika risiko terjadi, analisis akar permasalahan harus dilakukan untuk memahami mengapa tindakan pencegahan risiko tidak efektif, sehingga tindakan dapat diperbaiki untuk terjadinya risiko di masa depan. Pembelajaran dari pengalaman mengelola risiko pada proyek saat ini harus didokumentasikan dan dibagikan kepada organisasi untuk meningkatkan kemampuan manajemen risiko pada proyek-

proyek masa depan. Tim proyek harus menjaga pola pikir proaktif terhadap risiko, tidak menunggu sampai masalah terwujud sepenuhnya sebelum mengambil tindakan. Budaya risiko yang sehat dalam organisasi, di mana risiko dapat didiskusikan secara terbuka tanpa menyalahkan, sangat penting untuk manajemen risiko yang efektif (Pargaonkar, 2023).

BAB 3: Pemodelan Proses Bisnis

Fintri Indriyani, M.Kom.

3.1 Konsep Proses Bisnis

3.1.1 Pengertian Proses Bisnis dan Signifikansinya

Proses bisnis merupakan serangkaian aktivitas yang terkoordinasi dan terstruktur yang melibatkan berbagai sumber daya organisasi untuk mengubah input menjadi output yang memiliki nilai tambah bagi pelanggan dan stakeholder. Setiap organisasi, baik sektor publik maupun privat, memiliki proses bisnis yang dirancang untuk mencapai tujuan strategis dan meningkatkan kinerja operasional. Proses bisnis mencakup semua kegiatan dari tingkat strategis hingga operasional yang saling terhubung dalam suatu sistem yang terintegrasi. Signifikansi pemahaman proses bisnis semakin meningkat dalam era digital karena organisasi perlu mengoptimalkan operasi mereka untuk bersaing di pasar global yang dinamis dan kompleks. Dengan memahami proses bisnis secara menyeluruh, organisasi dapat mengidentifikasi peluang peningkatan efisiensi, mengurangi biaya operasional, meningkatkan kualitas produk atau layanan, dan pada akhirnya meningkatkan kepuasan pelanggan yang merupakan kunci kesuksesan jangka panjang.

3.1.2 Karakteristik Fundamental Proses Bisnis

Setiap proses bisnis memiliki karakteristik fundamental yang membedakannya dari aktivitas biasa dalam organisasi dan memberikan struktur pada bagaimana organisasi beroperasi. Pertama, proses bisnis memiliki tujuan yang jelas dan terukur yang selaras dengan visi dan misi organisasi secara keseluruhan. Kedua, proses bisnis melibatkan serangkaian aktivitas yang terikat secara logis dan kronologis dengan urutan tertentu yang harus diikuti untuk mencapai hasil yang optimal. Ketiga, setiap proses memiliki masukan atau *input* berupa sumber daya seperti bahan baku, data, atau tenaga kerja, dan menghasilkan keluaran atau *output* yang dapat diukur dan dievaluasi. Keempat, proses bisnis melibatkan pengambilan keputusan pada titik-titik tertentu yang mempengaruhi alur proses dan hasil akhir. Kelima, terdapat pemilik proses atau *process owner* yang bertanggung jawab atas keseluruhan implementasi, monitoring, dan peningkatan proses. Keenam, proses bisnis memiliki batas yang jelas dengan waktu siklus, biaya, dan sumber daya yang dapat diukur, sehingga dapat dievaluasi kinerjanya secara objektif dan berkelanjutan.

3.1.3 Klasifikasi dan Kategorisasi Proses Bisnis

Proses bisnis dalam organisasi dapat diklasifikasikan berdasarkan berbagai kriteria untuk memudahkan pemahaman dan manajemen. Berdasarkan fungsi dalam organisasi, proses bisnis dikategorikan menjadi proses strategis yang berkaitan dengan perencanaan jangka panjang dan pengembangan strategi, proses inti atau *core process* yang secara langsung menghasilkan produk atau

layanan untuk pelanggan eksternal, proses pendukung atau *supporting process* yang memberikan dukungan kepada proses inti seperti manajemen sumber daya manusia dan keuangan, serta proses manajemen yang mengawasi dan mengendalikan jalannya seluruh proses. Berdasarkan tingkat formalitas, proses bisnis dapat terstruktur dengan baik atau semi-terstruktur dengan dokumentasi lengkap, atau tidak terstruktur yang bersifat ad-hoc dan fleksibel. Berdasarkan dampak terhadap pelanggan, terdapat *front-office process* yang berinteraksi langsung dengan pelanggan dan *back-office process* yang bekerja di belakang layar. Klasifikasi ini membantu organisasi untuk mengalokasikan sumber daya dengan tepat dan mengimplementasikan strategi perbaikan yang sesuai dengan karakteristik unik setiap proses bisnis.

3.1.4 Peran Proses Bisnis dalam Transformasi Organisasi

Proses bisnis memainkan peran yang sangat penting dalam transformasi organisasi dan adaptasi terhadap perubahan lingkungan bisnis yang cepat dan kompleks. Proses bisnis yang dirancang dengan baik menjadi fondasi untuk implementasi teknologi informasi dan sistem digital yang meningkatkan efisiensi dan efektivitas operasi. Pemahaman mendalam tentang proses bisnis memungkinkan organisasi untuk melakukan restrukturisasi dan reengineering yang diperlukan untuk merespons peluang pasar baru atau ancaman yang muncul. Proses bisnis yang optimal menjadi katalis untuk transformasi digital dengan memungkinkan otomasi, integrasi sistem, dan penggunaan data analytics untuk pengambilan keputusan yang lebih baik. Proses bisnis juga menjadi sarana untuk

menyebarkan nilai-nilai dan budaya organisasi kepada semua karyawan sehingga menciptakan konsistensi dalam pelayanan dan kualitas output. Dengan demikian, investasi dalam pemahaman dan perbaikan proses bisnis merupakan investasi strategis yang memberikan dampak signifikan terhadap kesuksesan jangka panjang organisasi dalam mencapai keunggulan kompetitif di pasar yang terus berubah.

3.2 Business Process Modeling

3.2.1 Definisi dan Tujuan Business Process Modeling

Business Process Modeling atau BPM merupakan disiplin ilmu dan praktik manajemen yang berfokus pada pengidentifikasian, pendokumentasian, analisis, perbaikan, dan monitoring proses bisnis organisasi untuk meningkatkan kinerja dan efisiensi operasional. BPM mencakup serangkaian metodologi, tools, dan teknik yang memungkinkan organisasi untuk memahami proses bisnis mereka secara komprehensif dan membuat keputusan perbaikan yang didukung oleh data dan analisis yang objektif (Teixeira et al., 2024). Tujuan utama BPM adalah menciptakan transparansi penuh tentang bagaimana organisasi beroperasi, mengidentifikasi peluang peningkatan, mengoptimalkan alokasi sumber daya, meningkatkan kualitas output, dan pada akhirnya meningkatkan nilai yang dihasilkan bagi pelanggan dan stakeholder. BPM juga bertujuan untuk menciptakan agilitas organisasi sehingga dapat dengan cepat beradaptasi terhadap perubahan pasar dan kebutuhan pelanggan yang dinamis. Melalui implementasi BPM yang efektif, organisasi

dapat mencapai keunggulan operasional yang berkelanjutan dan menjadi pemimpin industri dalam hal efisiensi dan inovasi.

3.2.2 Komponen dan Elemen Utama Business Process Modeling

Business Process Modeling mencakup berbagai komponen dan elemen utama yang saling terintegrasi untuk menciptakan sistem manajemen proses yang komprehensif. Komponen pertama adalah *process discovery* atau penemuan proses yang melibatkan identifikasi dan dokumentasi semua proses bisnis yang ada dalam organisasi melalui observasi, wawancara, dan analisis data. Komponen kedua adalah *process modeling* yaitu representasi visual dari proses bisnis menggunakan notasi dan tools standar untuk memudahkan pemahaman dan analisis. Komponen ketiga adalah *process analysis* yang melibatkan evaluasi kritis terhadap proses untuk mengidentifikasi area-area yang tidak efisien, redundan, atau bermasalah. Komponen keempat adalah *process improvement* yang mencakup merancang dan mengimplementasikan perubahan untuk meningkatkan kinerja proses. Komponen kelima adalah *process monitoring* dan *control* yang memastikan proses terus berjalan sesuai rencana dan memberikan hasil yang diinginkan. Komponen keenam adalah *process optimization* yang merupakan aktivitas berkelanjutan untuk terus meningkatkan efisiensi dan efektivitas proses seiring waktu melalui pembelajaran dan inovasi yang konsisten.

3.2.3 Pendekatan dan Metodologi dalam Business Process Modeling

Terdapat berbagai pendekatan dan metodologi yang dapat digunakan dalam implementasi Business Process Modeling sesuai dengan konteks, budaya, dan kebutuhan organisasi yang spesifik. Pendekatan *Business Process Reengineering* atau BPR fokus pada redesign menyeluruh dan radikal dari proses bisnis untuk mencapai peningkatan dramatis dalam kinerja melalui pemanfaatan teknologi dan perubahan organisasi. Pendekatan *Lean* berfokus pada eliminasi pemborosan dalam proses dan memaksimalkan nilai yang dikirimkan kepada pelanggan dengan sumber daya minimum. Pendekatan *Six Sigma* menggunakan metodologi berbasis data dan analisis statistik untuk mengurangi variabilitas proses dan meningkatkan kualitas output. Pendekatan *Agile* memungkinkan organisasi untuk melakukan perbaikan proses secara iteratif dan responsif terhadap umpan balik pelanggan dengan siklus perbaikan yang singkat. Pendekatan *Total Quality Management* atau TQM menekankan keterlibatan semua anggota organisasi dalam perbaikan berkelanjutan dan fokus pada kepuasan pelanggan sebagai prioritas utama. Pemilihan pendekatan yang tepat sangat penting untuk memastikan implementasi BPM sesuai dengan karakteristik dan kemampuan organisasi.

3.2.4 Tools dan Implementasi Praktis Business Process Modeling

Implementasi praktis Business Process Modeling memerlukan penggunaan berbagai tools dan software yang memfasilitasi aktivitas modeling, analisis, simulasi, dan monitoring proses bisnis. Tools

seperti *Visual Paradigm*, *Signavio Process Manager*, dan *Microsoft Visio* memungkinkan tim untuk membuat diagram proses menggunakan notasi standar seperti BPMN dengan fitur kolaborasi yang mendukung kerja tim (M & Djamaluddin, 2022). Tools simulasi seperti *Arena* dan *Simul8* memungkinkan organisasi untuk melakukan analisis prediktif dan *what-if scenario* untuk mengevaluasi dampak perubahan proses sebelum implementasi aktual. Tools analitik seperti *Disco* dan *ProM* memungkinkan *proces mining* untuk mengekstrak pengetahuan tentang proses dari data event log sistem informasi. Implementasi praktis BPM memerlukan komitmen manajemen, alokasi sumber daya yang cukup, pelatihan karyawan, dan komunikasi yang jelas tentang tujuan dan manfaat BPM kepada semua stakeholder. Kesuksesan implementasi BPM bergantung pada kemampuan organisasi untuk mengubah mindset dari cara kerja tradisional menjadi pendekatan yang lebih terstruktur, data-driven, dan berorientasi pada perbaikan berkelanjutan.

3.3 Diagram Alir (Flowchart)

3.3.1 Pengertian dan Fungsi Fundamental Flowchart

Flowchart atau diagram alir merupakan representasi visual dari serangkaian langkah, aktivitas, keputusan, dan alur kontrol dalam sebuah proses menggunakan simbol-simbol standar yang saling terhubung dengan garis penunjuk arah. Flowchart berfungsi sebagai alat komunikasi yang sangat efektif dan efisien untuk menjelaskan proses yang kompleks kepada berbagai pihak termasuk karyawan,

manajemen, dan stakeholder eksternal tanpa memerlukan penjelasan verbal yang panjang dan berbelit-belit. Dalam konteks analisis proses bisnis, flowchart membantu mengidentifikasi langkah-langkah yang redundan atau tidak perlu, mengungkap bottleneck atau titik hambat dalam proses, dan memfasilitasi diskusi tentang peluang perbaikan. Flowchart juga berfungsi sebagai dokumentasi resmi dari prosedur operasional standar yang dapat direferensikan oleh karyawan baru atau sebagai pedoman dalam menjalankan tugas mereka dengan konsisten. Keunggulan utama flowchart adalah kesederhanaan dan kemudahan pembuatannya dibandingkan dengan notasi modeling yang lebih kompleks, sehingga dapat digunakan oleh berbagai kalangan tanpa memerlukan pelatihan khusus yang ekstensif.

3.3.2 Simbol-Simbol Standar dan Konvensi dalam Flowchart

Pembuatan flowchart yang efektif dan dapat dipahami oleh berbagai pihak memerlukan penggunaan simbol-simbol standar yang telah ditetapkan dan diakui secara internasional untuk memastikan konsistensi interpretasi. Simbol oval atau elips dengan label "Mulai" atau "Selesai" digunakan untuk menunjukkan titik awal dan akhir dari suatu proses. Simbol persegi panjang merepresentasikan aktivitas atau kegiatan spesifik yang dilakukan dalam proses bisnis. Simbol belah ketupat atau diamond shape merepresentasikan titik pengambilan keputusan di mana proses bercabang menjadi dua atau lebih jalur alternatif berdasarkan kondisi atau kriteria yang dievaluasi. Simbol jajargenjang atau parallelogram merepresentasikan masukan atau keluaran data dalam proses.

Simbol silinder merepresentasikan penyimpanan data atau file dalam sistem, baik berbentuk database digital maupun file fisik. Simbol lingkaran kecil merepresentasikan konektor atau titik sambung antar bagian flowchart. Garis dengan panah menunjukkan arah alur proses dari satu simbol ke simbol berikutnya. Penggunaan simbol-simbol ini secara konsisten memastikan bahwa flowchart dapat diinterpretasikan dengan benar dan mengurangi kemungkinan terjadinya miskomunikasi tentang proses yang digambarkan.

3.3.3 Prosedur Sistematis Pembuatan Flowchart

Pembuatan flowchart yang berkualitas tinggi dan akurat memerlukan mengikuti prosedur sistematis yang terstruktur dengan baik untuk memastikan semua aspek proses tercakup dengan benar. Tahap pertama adalah pengumpulan informasi komprehensif tentang proses melalui berbagai metode seperti observasi langsung di lapangan, wawancara mendalam dengan pelaksana proses, review dokumentasi yang ada, dan analisis sistem informasi yang digunakan. Tahap kedua adalah identifikasi semua langkah atau aktivitas dalam proses dan menentukan urutan logis atau sequence yang benar untuk mencapai hasil akhir. Tahap ketiga adalah identifikasi semua titik pengambilan keputusan dan menentukan kondisi atau kriteria yang akan mengarahkan proses ke jalur yang berbeda. Tahap keempat adalah pembuatan draft flowchart dengan menggunakan simbol-simbol standar dan menggambarkan alur logika yang benar sesuai dengan proses yang dianalisis. Tahap kelima adalah validasi flowchart dengan melibatkan pelaksana proses, manajer, dan stakeholder lainnya untuk memastikan akurasi

dan kelengkapan representasi proses. Tahap keenam adalah finalisasi dan dokumentasi flowchart dalam format yang dapat dipublikasikan dan digunakan sebagai referensi dalam organisasi untuk keperluan pelatihan, analisis, dan perbaikan proses.

3.3.4 Aplikasi Praktis Flowchart dalam Berbagai Konteks Organisasi

Flowchart memiliki aplikasi yang sangat luas dan praktis dalam berbagai konteks bisnis, fungsi organisasi, dan industri karena fleksibilitas, kesederhanaan, dan kemampuannya untuk mengkomunikasikan proses kompleks dengan cara yang mudah dipahami. Dalam fungsi operasional, flowchart digunakan untuk mendokumentasikan standar prosedur operasional yang harus diikuti oleh karyawan dalam melaksanakan tugas-tugas harian mereka dengan konsisten. Dalam fungsi akuntansi dan keuangan, flowchart digunakan untuk merepresentasikan proses-proses seperti siklus pengeluaran kas, proses penggajian, proses penerimaan pembayaran pelanggan, dan proses pelaporan keuangan (Tuasamu et al., 2023). Dalam fungsi sistem informasi, flowchart digunakan untuk merancang logika program komputer, proses pengolahan data dalam sistem informasi, dan alur transaksi dalam sistem informasi perusahaan. Dalam fungsi pemasaran dan penjualan, flowchart digunakan untuk merepresentasikan proses akuisisi pelanggan, proses penanganan pesanan, dan proses layanan purna jual. Dalam fungsi manajemen sumber daya manusia, flowchart digunakan untuk mengkomunikasikan proses rekrutmen, proses orientasi karyawan baru, dan proses penyelesaian perselisihan karyawan. Aplikasi

flowchart yang luas ini membuktikan nilai fundamental dari tool ini sebagai sarana komunikasi dan analisis yang efektif untuk mendukung berbagai aspek manajemen organisasi.

3.4 Data Flow Diagram (DFD)

3.4.1 Konsep Fundamental Data Flow Diagram

Data Flow Diagram atau DFD merupakan alat pemodelan sistem yang menggunakan notasi visual khusus untuk merepresentasikan bagaimana data mengalir melalui sistem, bagaimana data ditransformasi dalam setiap proses, tempat data disimpan atau diakses, dan pihak-pihak eksternal yang berinteraksi dengan sistem. DFD berbeda secara fundamental dengan flowchart karena fokus utama DFD adalah pada aliran dan transformasi data bukan pada aliran kendali atau pengambilan keputusan logis. DFD memberikan gambaran komprehensif tentang bagaimana sistem informasi beroperasi dengan menunjukkan sumber data eksternal, tujuan data, proses pengolahan data, penyimpanan data, dan hubungan antara berbagai elemen sistem. Konsep dasar DFD meliputi pemahaman tentang keempat elemen utama sistem yaitu entitas eksternal yang merupakan aktor atau sistem di luar batas sistem yang mengirim atau menerima data, proses yang mengubah dan memproses data input menjadi output yang berguna, penyimpanan data atau *datastore* yang menyimpan data untuk diakses oleh berbagai proses, dan aliran data yang menunjukkan pergerakan data dari satu elemen sistem ke elemen lain (Lumat,

2025). Pemahaman yang mendalam tentang konsep dasar DFD sangat penting untuk dapat merancang sistem informasi yang efektif dan menganalisis sistem yang sudah ada.

3.4.2 Level Hierarki dan Dekomposisi dalam Data Flow Diagram

DFD dirancang dalam bentuk hierarki dengan multiple level dekomposisi yang memungkinkan pengguna untuk memahami sistem pada berbagai tingkat detail dan abstraksi dari tinggi hingga rendah. *Context Diagram* atau DFD Level 0 merupakan diagram tingkat tertinggi yang memberikan gambaran keseluruhan sistem dalam bentuk yang sangat ringkas dengan menunjukkan batas sistem, entitas eksternal utama, dan aliran data utama antara sistem dan lingkungannya. DFD Level 1 melakukan dekomposisi terhadap proses utama dalam diagram konteks menjadi sub-proses yang lebih detail dengan menunjukkan aliran data antar proses utama dan penyimpanan data yang digunakan. DFD Level 2 melanjutkan penguraian dengan menjabarkan proses-proses Level 1 menjadi sub-proses yang lebih spesifik dan terperinci sesuai kebutuhan analisis. Penguraian dapat berlanjut ke level yang lebih dalam, umumnya hingga level 3 atau 4, sesuai dengan tingkat detail yang diperlukan untuk sepenuhnya memahami sistem informasi yang sedang dianalisis. Hierarki ini memungkinkan tim pengembang sistem untuk mulai dengan pemahaman umum tentang sistem dan secara bertahap menggali detail implementasi yang spesifik, sehingga memfasilitasi komunikasi yang lebih baik antara berbagai

stakeholder dengan tingkat pemahaman teknis yang bervariasi (Ridwan et al., 2022).

3.4.3 Elemen dan Notasi Standar dalam Data Flow Diagram

Pembuatan DFD yang akurat dan dapat dipahami dengan baik memerlukan pemahaman yang komprehensif tentang elemen-elemen utama dan notasi-notasi standar yang digunakan dalam representasi DFD. Lingkaran atau bubble merepresentasikan proses atau transformasi data yang terjadi dalam sistem, umumnya diberi nomor identifikasi unik dan label deskriptif yang menunjukkan jenis transformasi data yang dilakukan. Garis lurus dengan panah atau *data flow* menunjukkan pergerakan atau aliran data dari satu elemen ke elemen lain, dengan label yang jelas menunjukkan jenis atau nama data yang mengalir. Persegi panjang atau *external entity* merepresentasikan sistem, organisasi, departemen, atau individu di luar batas sistem yang mengirim atau menerima data dari sistem yang sedang dianalisis. Dua garis sejajar horizontal atau *data store* merepresentasikan lokasi penyimpanan data yang dapat berupa file, database, filing cabinet, atau arsip manual lainnya. Setiap elemen dalam DFD harus diberi label yang jelas, deskriptif, dan konsisten sehingga model dapat diinterpretasikan dengan benar oleh semua pihak yang membacanya termasuk pengembang sistem, pengguna, dan manajemen organisasi (Reawaruw et al., 2025).

3.4.4 Proses Pengembangan dan Analisis Data Flow Diagram

Proses pengembangan DFD yang efektif dan menghasilkan model sistem yang akurat melibatkan serangkaian langkah sistematis yang terstruktur dengan baik dari tahap perencanaan hingga

finalisasi. Langkah pertama adalah identifikasi dan dokumentasi semua entitas eksternal atau pengguna sistem yang berinteraksi dengan sistem yang akan dimodelkan. Langkah kedua adalah identifikasi semua proses utama atau fungsi sistem yang melakukan transformasi data dari input menjadi output yang berguna. Langkah ketiga adalah identifikasi dan dokumentasi semua penyimpanan data atau lokasi penyimpanan informasi yang digunakan oleh sistem. Langkah keempat adalah identifikasi semua aliran data antar elemen dan membuat *context diagram* atau DFD Level 0 sebagai gambaran tingkat tinggi sistem. Langkah kelima adalah merinci proses utama menjadi sub-proses yang lebih detail dan membuat DFD Level 1 dengan menunjukkan aliran data yang lebih spesifik antar sub-proses. Langkah keenam adalah melanjutkan dekomposisi hingga level yang diperlukan sesuai dengan kompleksitas sistem dan detail yang ingin dikomunikasikan. Dalam analisis DFD, sangat penting untuk memverifikasi bahwa semua data yang masuk ke sebuah proses juga keluar dari proses, tidak ada data yang hilang atau tidak terakses dalam sistem, dan aliran data konsisten di berbagai level DFD. Analisis ini membantu mengidentifikasi inkonsistensi, redundansi, atau masalah dalam desain sistem informasi sehingga dapat dikoreksi sebelum implementasi sistem dimulai.

3.5 Evaluasi Proses Bisnis

3.5.1 Tujuan Strategis dan Manfaat Evaluasi Proses Bisnis

Evaluasi proses bisnis merupakan aktivitas sistematis dan berkelanjutan untuk menilai kinerja proses bisnis yang sedang berjalan terhadap standar, target, dan benchmark yang telah ditetapkan organisasi untuk memastikan proses memberikan hasil yang optimal. Tujuan utama evaluasi adalah mengidentifikasi area-area dalam proses yang tidak berjalan sesuai ekspektasi atau tidak mencapai tingkat efisiensi dan efektivitas yang diinginkan oleh manajemen organisasi. Evaluasi yang komprehensif memungkinkan organisasi untuk mengidentifikasi akar penyebab masalah atau ketidaksempurnaan dalam proses dan mengembangkan strategi perbaikan yang tepat sasaran dan berbasis pada bukti atau data yang objektif. Pentingnya evaluasi proses bisnis meningkat secara signifikan dalam lingkungan bisnis yang dinamis dan kompleks di mana organisasi perlu terus beradaptasi dan meningkatkan keunggulan kompetitifnya untuk tetap relevan di pasar. Melalui evaluasi berkala dan sistematis, organisasi dapat memastikan bahwa sumber daya dialokasikan dan digunakan secara optimal, kualitas output tetap terjaga dan konsisten, kepuasan pelanggan terus meningkat, dan organisasi mempertahankan efisiensi operasional yang tinggi dalam mencapai target bisnis.

3.5.2 Metrik Kinerja dan Key Performance Indicators untuk Proses

Evaluasi proses bisnis yang efektif dan bermakna memerlukan penetapan metrik kinerja atau *Key Performance Indicators* yang jelas, terukur, dan secara akurat mencerminkan efektivitas dan efisiensi proses bisnis dalam menghasilkan value bagi organisasi. Metrik efisiensi seperti waktu siklus atau *cycle time* proses mengukur durasi total dari awal hingga akhir proses, sementara metrik produktivitas mengukur jumlah output yang dihasilkan per unit input atau per unit waktu kerja. Metrik kualitas mengukur tingkat akurasi output, tingkat kesalahan atau *defect rate*, tingkat kepatuhan terhadap standar kualitas yang telah ditetapkan, dan tingkat cacat atau *non-conformity* yang ditemukan pada output (Teixeira et al., 2024). Metrik kepuasan pelanggan mengukur seberapa puas pelanggan terhadap produk atau layanan yang dihasilkan oleh proses, yang dapat dikumpulkan melalui survei pelanggan, feedback pengguna, atau analisis keluhan pelanggan. Metrik biaya mengukur efisiensi penggunaan sumber daya finansial dalam proses termasuk biaya operasional per unit output, biaya per transaksi, dan total biaya penyelenggaraan proses. Metrik risiko mengukur potensi terjadinya masalah atau hambatan dalam proses yang dapat mempengaruhi pencapaian tujuan organisasi. Penetapan metrik yang tepat sangat penting karena akan mengarahkan fokus dan perhatian organisasi pada area-area yang benar-benar penting dan strategis untuk mencapai tujuan bisnis jangka pendek dan panjang.

3.5.3 Teknik Analisis Mendalam untuk Evaluasi Proses

Berbagai teknik dan metode analisis mendalam dapat digunakan untuk melakukan evaluasi proses bisnis secara menyeluruh dan mengungkap peluang perbaikan yang signifikan. *Root Cause Analysis* digunakan untuk mengidentifikasi akar penyebab masalah atau kinerja yang tidak optimal dalam proses melalui analisis sistematis dengan menggunakan tools seperti diagram *fishbone, why-why analysis*, atau *fault tree analysis*. *Value-Added Analysis* mengidentifikasi dan mengkategorikan setiap aktivitas dalam proses apakah menambah nilai bagi pelanggan atau tidak, sehingga aktivitas yang tidak menambah nilai dapat dieliminasi atau diminimalkan untuk meningkatkan efisiensi. *Benchmarking* membandingkan kinerja proses bisnis organisasi dengan proses serupa di organisasi lain yang dianggap sebagai best practice dalam industri untuk mengidentifikasi gap dan peluang peningkatan. *Process Simulation* menggunakan model komputer untuk mensimulasikan operasi proses dan menganalisis dampak berbagai perubahan atau skenario alternatif sebelum implementasi aktual untuk meminimalkan risiko. *Gap Analysis* secara sistematis mengidentifikasi perbedaan atau gap antara kondisi proses saat ini dan kondisi target yang diinginkan, menjadi dasar perencanaan dan implementasi perbaikan yang terukur dan terarah (Teixeira et al., 2024).

3.5.4 Implementasi Perbaikan dan Peningkatan Berkelanjutan

Hasil dari evaluasi proses bisnis harus diikuti dengan implementasi perbaikan yang terencana, terstruktur, dan

berkelanjutan untuk mencapai peningkatan kinerja yang signifikan dan berkelanjutan seiring waktu. Langkah pertama adalah prioritasi area-area perbaikan berdasarkan dampak potensial terhadap kinerja organisasi, sumber daya yang diperlukan untuk implementasi, dan kesiapan organisasi untuk melakukan perubahan tersebut. Langkah kedua adalah pengembangan rencana perbaikan detail yang mencakup tujuan spesifik yang ingin dicapai, aktivitas konkret yang akan dilakukan, sumber daya yang diperlukan, timeline implementasi yang jelas, dan identifikasi penanggung jawab untuk setiap aspek implementasi. Langkah ketiga adalah implementasi perbaikan dengan melibatkan semua stakeholder yang terkait dalam proses, memberikan pelatihan kepada karyawan tentang perubahan proses, dan memfasilitasi adaptasi organisasi terhadap proses baru. Langkah keempat adalah monitoring dan evaluasi berkelanjutan terhadap hasil implementasi untuk memastikan perbaikan mencapai target yang ditetapkan dan memberikan value yang diharapkan. Peningkatan berkelanjutan atau *continuous improvement* memerlukan budaya organisasi yang mendukung dan menghargai inovasi, pembelajaran, dan perbaikan berkesinambungan dalam operasi bisnis. Siklus evaluasi dan perbaikan ini harus berjalan terus-menerus dan konsisten untuk memastikan organisasi tetap kompetitif, responsif terhadap perubahan kebutuhan pasar, perkembangan teknologi, dan mampu memberikan value yang semakin baik kepada pelanggan dan stakeholder.

BAB 4: Pemodelan Data Sistem

Euis Widanengsih, M.Kom.

4.1 Konsep Data dalam Sistem Informasi

4.1.1 Definisi dan Peran Data dalam Sistem Informasi

Data merupakan representasi dari fakta, nilai, atau pengamatan yang belum diproses dan belum memiliki konteks atau makna khusus. Dalam konteks sistem informasi, data merupakan elemen paling fundamental yang menjadi basis dari semua operasi sistem. Data adalah bahan mentah yang akan diubah menjadi informasi melalui proses pengolahan. Sistem informasi dirancang untuk mengumpulkan, menyimpan, memproses, dan menyajikan data agar dapat menjadi informasi yang berguna bagi pengambilan keputusan. Peran data dalam sistem informasi sangat kritis karena kualitas dan akurasi data secara langsung mempengaruhi kualitas informasi yang dihasilkan, yang pada gilirannya mempengaruhi kualitas keputusan yang diambil oleh organisasi.

Dalam era digital dan transformasi bisnis modern, data telah menjadi aset paling berharga bagi organisasi. Organisasi yang dapat mengelola data dengan efektif dan efisien akan memiliki keunggulan kompetitif yang signifikan. Data dapat berasal dari berbagai sumber termasuk transaksi bisnis, sensor, media sosial, dan sistem-sistem lain. Untuk mengelola data yang kompleks dengan baik, diperlukan suatu struktur atau model yang jelas dan terorganisir. Pemodelan

data adalah proses untuk membuat representasi abstrak dari data yang ada dalam sistem, menggambarkan hubungan dan aturan-aturan yang berlaku terhadap data tersebut.

4.1.2 Jenis-Jenis Data dan Karakteristiknya

Data dapat diklasifikasikan menjadi berbagai jenis berdasarkan karakteristik dan sumbernya. Berdasarkan struktur, data dapat dibagi menjadi data terstruktur yang memiliki format dan organisasi yang jelas seperti data dalam bentuk tabel atau database, dan data tidak terstruktur seperti teks, gambar, video, dan dokumen yang tidak memiliki format standar. Data semi-terstruktur merupakan kategori yang berada di antara keduanya, seperti data *JSON* atau *XML* yang memiliki beberapa struktur namun lebih fleksibel dari data terstruktur tradisional.

Berdasarkan tipe nilainya, data dapat berupa data numerik yang terdiri dari angka-angka, data alfanumerik yang merupakan kombinasi huruf dan angka, data tanggal dan waktu, dan data logika (*boolean*) yang hanya memiliki dua nilai yaitu benar atau salah. Setiap jenis data memiliki karakteristik unik yang menentukan bagaimana data tersebut harus disimpan, diproses, dan dianalisis. Pemahaman mendalam tentang jenis-jenis data ini sangat penting dalam proses perancangan database dan sistem informasi untuk memastikan bahwa data disimpan dengan efisien dan dapat diakses dengan cepat sesuai kebutuhan organisasi.

4.1.3 Siklus Hidup Data dalam Sistem Informasi

Siklus hidup data dalam sistem informasi dimulai dari pengumpulan data yang merupakan tahap pertama di mana data

dikumpulkan dari berbagai sumber. Tahap kedua adalah penyimpanan data di mana data disimpan dalam sistem penyimpanan yang terorganisir seperti database. Tahap ketiga adalah pemrosesan data di mana data diubah, divalidasi, dan dianalisis untuk menghasilkan informasi yang bermakna. Tahap keempat adalah diseminasi atau penyebaran informasi di mana informasi yang telah diproses disajikan kepada pengguna yang membutuhkan.

Tahap kelima adalah penggunaan informasi di mana pengguna menggunakan informasi untuk membuat keputusan atau melakukan tindakan tertentu. Tahap terakhir adalah penyimpanan arsip atau *archiving* di mana data yang tidak lagi aktif digunakan disimpan dalam penyimpanan jangka panjang untuk keperluan audit, compliance, atau referensi historis. Memahami siklus hidup data ini penting untuk perancangan sistem informasi yang efektif, karena setiap tahap memiliki persyaratan dan tantangan tersendiri yang harus ditangani dengan baik untuk memastikan integritas dan keamanan data (Amran et al., 2019).

4.1.4 Atribut Data dan Identifikasi Entitas

Atribut data adalah karakteristik atau properti dari suatu entitas yang menggambarkan informasi spesifik tentang entitas tersebut. Setiap atribut memiliki nama, tipe data, dan kemungkinan nilai-nilai yang dapat diterima. Misalnya, pada entitas *Karyawan*, atribut yang mungkin ada termasuk *Nama Karyawan* bertipe teks, *Tanggal Lahir* bertipe tanggal, dan *Gaji* bertipe numerik. Atribut harus dipilih dengan hati-hati untuk memastikan bahwa semua informasi

penting tentang entitas telah ditangkap tanpa ada redundansi yang tidak perlu.

Identifikasi entitas merupakan langkah kritis dalam proses pemodelan data. Entitas adalah sesuatu yang mewakili suatu objek atau konsep dalam dunia nyata yang ingin dilacak dan disimpan datanya dalam sistem informasi. Setiap entitas harus memiliki identifikasi unik yang membedakannya dari entitas lainnya, biasanya melalui atribut yang disebut *primary key* atau kunci utama. Proses identifikasi entitas memerlukan pemahaman mendalam tentang domain bisnis atau masalah yang akan dimodelkan, serta interaksi dan hubungan antar entitas dalam sistem (Mo, 2024).

4.2 Entity Relationship Diagram (ERD)

4.2.1 Konsep Dasar dan Notasi ERD

Entity Relationship Diagram (ERD) adalah representasi grafis dari model konseptual yang menggambarkan entitas dalam sistem, atribut entitas tersebut, dan hubungan (*relationship*) antara entitas-entitas tersebut. ERD merupakan alat yang sangat penting dalam fase desain konseptual database karena memungkinkan komunikasi yang jelas antara desainer database dengan pengguna atau pemangku kepentingan tentang struktur data yang akan dibangun. ERD menggunakan notasi tertentu untuk merepresentasikan berbagai elemen dan konsepnya dengan konsisten.

Dalam notasi Chen yang paling umum digunakan, entitas digambarkan sebagai persegi panjang, atribut digambarkan sebagai

oval, dan hubungan digambarkan sebagai belah ketupat. Garis-garis menghubungkan entitas, atribut, dan hubungan untuk menunjukkan hubungan-hubungan tersebut. Notasi tambahan digunakan untuk menunjukkan kardinalitas dari hubungan, yaitu jumlah maksimum dan minimum dari kejadian suatu entitas yang dapat berhubungan dengan kejadian entitas lain. Kardinalitas dinotasikan dengan berbagai cara tergantung pada jenis notasi yang digunakan, misalnya notasi *crow's foot* menggunakan simbol khusus untuk menunjukkan hubungan satu-ke-satu, satu-ke-banyak, banyak-ke-banyak, dan sebagainya.

4.2.2 Tahapan Pengembangan ERD

Pengembangan ERD melibatkan beberapa tahapan sistematis yang harus diikuti dengan cermat. Tahap pertama adalah identifikasi dan analisis kebutuhan bisnis, di mana desainer database harus memahami domain bisnis, proses bisnis, dan persyaratan pengguna secara mendalam. Tahap ini melibatkan wawancara dengan pengguna, analisis dokumen, dan observasi sistem yang ada untuk mengumpulkan informasi yang lengkap dan akurat tentang data apa yang perlu disimpan dan bagaimana data tersebut digunakan.

Tahap kedua adalah identifikasi entitas utama di mana desainer mengidentifikasi objek-objek utama atau konsep-konsep penting dalam domain bisnis yang perlu disimpan datanya. Tahap ketiga adalah identifikasi atribut untuk setiap entitas, di mana desainer menentukan properti atau karakteristik apa yang perlu disimpan untuk setiap entitas. Tahap keempat adalah identifikasi hubungan antara entitas, di mana desainer menentukan bagaimana

entitas-entitas saling berkaitan satu sama lain. Tahap kelima adalah penentuan kardinalitas hubungan untuk memastikan bahwa hubungan-hubungan tersebut benar-benar mewakili aturan-aturan bisnis yang berlaku. Tahap keenam adalah validasi dan penyempurnaan ERD dengan melibatkan pengguna untuk memastikan bahwa model yang dibuat sudah akurat dan lengkap (Amran et al., 2019).

4.2.3 Entitas, Atribut, dan Kunci dalam ERD

Dalam ERD, entitas merupakan elemen dasar yang merepresentasikan sesuatu yang nyata atau abstrak yang ingin dilacak. Setiap entitas harus memiliki nama yang jelas dan deskriptif yang menggunakan kata benda tunggal. Contoh entitas termasuk *Siswa*, *Mata Kuliah*, *Dosen*, dan sebagainya. Atribut dari suatu entitas menggambarkan informasi spesifik tentang entitas tersebut. Beberapa atribut mungkin bersifat wajib atau harus ada, sementara atribut lain bersifat opsional. Atribut juga dapat diklasifikasikan berdasarkan cara nilai-nilainya ditentukan.

Atribut sederhana adalah atribut yang memiliki nilai tunggal yang tidak dapat dipecah lagi, sementara atribut komposit adalah atribut yang dapat dipecah menjadi atribut-atribut yang lebih kecil. Atribut single-valued adalah atribut yang hanya memiliki satu nilai untuk setiap kejadian entitas, sementara atribut multi-valued adalah atribut yang dapat memiliki lebih dari satu nilai untuk setiap kejadian entitas. Kunci atau *key* dalam ERD digunakan untuk mengidentifikasi secara unik setiap kejadian entitas. *Primary key* atau kunci utama adalah atribut atau kombinasi atribut yang

secara unik mengidentifikasi setiap baris dalam tabel entitas. *Candidate key* atau kunci calon adalah atribut atau kombinasi atribut lain yang juga dapat mengidentifikasi secara unik setiap baris, namun tidak dipilih sebagai kunci utama.

4.2.4 Hubungan dan Kardinalitas dalam ERD

Hubungan dalam ERD merepresentasikan asosiasi atau koneksi antara dua entitas atau lebih. Hubungan biasanya digambarkan dengan belah ketupat dan garis yang menghubungkan entitas-entitas yang terlibat. Setiap hubungan harus memiliki nama yang jelas yang menggambarkan sifat hubungan tersebut dengan menggunakan kata kerja atau frasa yang deskriptif. Kardinalitas hubungan menunjukkan jumlah maksimum dan minimum dari kejadian suatu entitas yang dapat berhubungan dengan kejadian entitas lain.

Ada empat jenis kardinalitas utama yang umum dalam model relasional. Pertama adalah hubungan satu-ke-satu (*one-to-one*) di mana satu kejadian entitas A dapat berhubungan dengan paling banyak satu kejadian entitas B dan sebaliknya. Kedua adalah hubungan satu-ke-banyak (*one-to-many*) di mana satu kejadian entitas A dapat berhubungan dengan banyak kejadian entitas B, tetapi setiap kejadian entitas B hanya dapat berhubungan dengan satu kejadian entitas A. Ketiga adalah hubungan banyak-ke-banyak (*many-to-many*) di mana satu kejadian entitas A dapat berhubungan dengan banyak kejadian entitas B dan sebaliknya. Keempat adalah hubungan yang sifatnya opsional atau wajib ada yang menunjukkan

apakah suatu hubungan harus ada atau boleh tidak ada untuk setiap kejadian entitas (Hayat & Kuswandi, 2025).

4.3 Normalisasi Database

4.3.1 Konsep Normalisasi dan Tujuannya

Normalisasi database adalah proses terstruktur untuk mengorganisir data dalam basis data relasional dengan tujuan mengurangi atau menghilangkan redundansi data dan anomali data. Redundansi data terjadi ketika data yang sama disimpan di beberapa tempat atau ditampilkan berkali-kali dalam tabel yang sama, yang menyebabkan pemborosan ruang penyimpanan dan meningkatkan risiko inkonsistensi data. Anomali data adalah situasi di mana modifikasi data dalam basis data mengarah pada ketidakkonsistenan atau kehilangan data yang tidak diinginkan. Tujuan utama normalisasi adalah untuk menghasilkan skema database yang efisien, konsisten, dan mudah untuk dirawat dan dimodifikasi.

Normalisasi juga membantu meningkatkan performa query karena data diorganisir dengan cara yang meminimalkan jumlah tabel yang perlu dijoin untuk menjawab query tertentu. Selain itu, normalisasi memfasilitasi integritas data dengan memastikan bahwa setiap data hanya disimpan di satu tempat, sehingga jika ada perubahan pada data tersebut, perubahan tersebut secara otomatis tercermin di semua tempat di mana data tersebut digunakan. Normalisasi juga membantu dalam merancang database yang lebih

fleksibel dan mudah beradaptasi terhadap perubahan kebutuhan bisnis di masa depan.

4.3.2 Bentuk Normal: 1NF, 2NF, dan 3NF

Bentuk normal adalah tingkat standarisasi yang diterapkan pada tabel database untuk mengurangi redundansi data dan anomali. Ada beberapa bentuk normal yang diterima secara luas, namun tiga bentuk pertama adalah yang paling sering digunakan dalam praktik. Bentuk Normal Pertama (*First Normal Form*, 1NF) memerlukan bahwa setiap atribut dalam tabel hanya mengandung nilai-nilai atomik atau tidak dapat dipecah lagi, dan tidak ada kelompok atribut yang berulang. Dengan kata lain, setiap sel dalam tabel hanya boleh mengandung satu nilai tunggal, bukan serangkaian nilai atau array.

Bentuk Normal Kedua (*Second Normal Form*, 2NF) memerlukan bahwa tabel pertama kali memenuhi kriteria 1NF, dan semua atribut non-kunci harus bergantung sepenuhnya pada kunci utama, bukan hanya sebagian dari kunci utama. Ini berarti bahwa jika kunci utama terdiri dari beberapa atribut (*composite key*), maka setiap atribut non-kunci harus bergantung pada seluruh kunci utama, bukan hanya pada beberapa atribut dari kunci utama tersebut. Bentuk Normal Ketiga (*Third Normal Form*, 3NF) memerlukan bahwa tabel pertama kali memenuhi kriteria 2NF, dan tidak ada atribut non-kunci yang bergantung secara transitif pada atribut non-kunci lainnya. Dengan kata lain, semua atribut non-kunci harus bergantung secara langsung pada kunci utama, dan tidak ada ketergantungan tidak langsung melalui atribut non-kunci lainnya.

4.3.3 Proses Normalisasi dan Decomposisi Tabel

Proses normalisasi melibatkan penganalisan tabel yang ada dan mengidentifikasi masalah-masalah redundansi dan anomali, kemudian menguraikan tabel-tabel tersebut menjadi tabel-tabel yang lebih kecil yang memenuhi bentuk normal yang diinginkan. Proses ini disebut decomposisi atau pemisahan tabel. Untuk mengubah tabel menjadi 1NF, setiap atribut yang mengandung nilai-nilai berulang atau multivalued harus dipisahkan ke dalam tabel terpisah, atau atribut tersebut harus diubah sehingga setiap baris hanya mengandung nilai tunggal.

Untuk mengubah tabel dari 1NF menjadi 2NF, atribut-atribut yang hanya bergantung pada sebagian dari kunci utama (*composite key*) harus dipisahkan ke dalam tabel terpisah dengan kunci utama baru yang terdiri dari atribut-atribut yang benar-benar bergantung. Untuk mengubah tabel dari 2NF menjadi 3NF, atribut-atribut yang memiliki ketergantungan transitif dengan atribut non-kunci lainnya harus dipisahkan ke dalam tabel terpisah. Proses normalisasi ini harus dilakukan dengan hati-hati untuk memastikan bahwa tidak ada informasi yang hilang, dan bahwa hubungan antara tabel-tabel yang dihasilkan tetap dapat dinyatakan dengan jelas melalui kunci-kunci asing (*foreign keys*) (Mo, 2024).

4.3.4 Denormalisasi dan Trade-off dalam Normalisasi

Meskipun normalisasi sangat penting untuk menghasilkan database yang terstruktur dengan baik, dalam beberapa situasi tertentu, denormalisasi atau proses sebaliknya dari normalisasi mungkin perlu dilakukan untuk meningkatkan performa sistem.

Denormalisasi adalah proses yang sengaja menambahkan redundansi terkontrol ke dalam database dengan tujuan untuk mengurangi jumlah join operasi yang diperlukan dalam query, sehingga meningkatkan kecepatan eksekusi query. Namun, denormalisasi harus dilakukan dengan bijak karena dapat meningkatkan kompleksitas dalam memelihara konsistensi data.

Trade-off yang terjadi dalam normalisasi adalah antara kompleksitas desain database di satu sisi, dan performa query serta kemudahan pemeliharaan database di sisi lain. Database yang terlalu ternormalisasi mungkin memerlukan banyak join operasi untuk menjawab query yang kompleks, yang dapat memperlambat performa sistem. Di sisi lain, database yang kurang ternormalisasi mungkin memiliki performa query yang lebih cepat tetapi memiliki redundansi data yang tinggi, yang dapat menyebabkan anomali data dan mempersulit pemeliharaan database. Keputusan tentang seberapa tinggi tingkat normalisasi yang harus diterapkan harus didasarkan pada analisis mendalam terhadap kebutuhan bisnis, pola akses data, dan persyaratan performa sistem.

4.4 Relasi Antar Tabel

4.4.1 Konsep Kunci Asing dan Integritas Referensial

Relasi antar tabel dalam model relasional dibangun melalui penggunaan kunci asing (*foreign key*) yang merupakan atribut atau kombinasi atribut dalam satu tabel yang mereferensikan kunci utama dalam tabel lain. Kunci asing memungkinkan untuk menjaga

hubungan antar tabel dan memastikan bahwa data dalam tabel-tabel tersebut tetap konsisten dan terintegrasi dengan baik. Integritas referensial adalah prinsip yang menyatakan bahwa setiap nilai dalam kolom kunci asing harus sesuai dengan nilai yang ada dalam kolom kunci utama tabel yang direferensikan, atau boleh bernilai null jika diizinkan.

Dengan menegakkan integritas referensial, sistem dapat mencegah situasi di mana terdapat data yatim piatu (*orphan data*) yaitu baris dalam tabel child yang mereferensikan baris dalam tabel parent yang sudah tidak ada. Integritas referensial juga membantu dalam memastikan konsistensi data, karena jika ada upaya untuk menghapus baris dalam tabel parent yang masih direferensikan oleh baris-baris dalam tabel child, sistem dapat menolak operasi tersebut atau secara otomatis menghapus baris-baris yang direferensikan tersebut sesuai dengan aturan yang telah didefinisikan.

4.4.2 Jenis-Jenis Relasi: Satu-ke-Satu, Satu-ke-Banyak, dan Banyak-ke-Banyak

Jenis-jenis relasi dalam model relasional mencerminkan berbagai cara entitas dapat berhubungan satu sama lain dalam dunia nyata. Relasi satu-ke-satu (*one-to-one*) terjadi ketika satu baris dalam tabel A dapat berhubungan dengan paling banyak satu baris dalam tabel B dan sebaliknya. Contohnya adalah relasi antara tabel *Karyawan* dan tabel *Akun* di mana setiap karyawan memiliki tepat satu akun pengguna komputer, dan setiap akun pengguna hanya dimiliki oleh satu karyawan. Dalam implementasi, relasi satu-ke-

satu dapat direpresentasikan dengan menyimpan kunci utama tabel B sebagai kunci asing dalam tabel A, atau sebaliknya.

Relasi satu-ke-banyak (*one-to-many*) terjadi ketika satu baris dalam tabel A dapat berhubungan dengan banyak baris dalam tabel B, namun setiap baris dalam tabel B hanya dapat berhubungan dengan satu baris dalam tabel A. Contohnya adalah relasi antara tabel *Departemen* dan tabel *Karyawan* di mana satu departemen memiliki banyak karyawan, tetapi setiap karyawan hanya termasuk dalam satu departemen. Dalam implementasi, relasi satu-ke-banyak direpresentasikan dengan menyimpan kunci utama tabel A sebagai kunci asing dalam tabel B.

Relasi banyak-ke-banyak (*many-to-many*) terjadi ketika banyak baris dalam tabel A dapat berhubungan dengan banyak baris dalam tabel B dan sebaliknya. Contohnya adalah relasi antara tabel *Siswa* dan tabel *Mata Kuliah* di mana satu siswa dapat mengambil banyak mata kuliah, dan satu mata kuliah dapat diambil oleh banyak siswa. Dalam implementasi relasi banyak-ke-banyak, diperlukan tabel perantara atau tabel hubungan yang disebut tabel *junction* atau tabel asosiasi, yang menyimpan kunci utama dari kedua tabel yang berhubungan sebagai kunci asing (Hayat & Kuswandi, 2025).

4.4.3 Implementasi Relasi dalam Model Relasional

Implementasi relasi dalam model relasional memerlukan perencanaan yang cermat untuk memastikan bahwa relasi-relasi tersebut dapat direpresentasikan dengan efektif dan efisien. Untuk relasi satu-ke-satu, ada dua strategi implementasi yang umum

digunakan. Strategi pertama adalah menyimpan referensi satu arah, yaitu menyimpan kunci utama dari salah satu tabel sebagai kunci asing dalam tabel lainnya. Strategi kedua adalah mengintegrasikan kedua tabel menjadi satu tabel jika relasi tersebut bersifat wajib di kedua sisi dan tidak ada alasan untuk memisahkan data.

Untuk relasi satu-ke-banyak, implementasi relatif sederhana, yaitu dengan menyimpan kunci utama dari tabel parent sebagai kunci asing dalam tabel child. Hal ini memungkinkan untuk melacak hubungan antara data di kedua tabel dengan mudah. Untuk relasi banyak-ke-banyak, diperlukan tabel junction yang menyimpan kunci utama dari kedua tabel sebagai kunci asing gabungan. Tabel junction ini dapat juga mengandung atribut tambahan yang mendeskripsikan sifat dari relasi tersebut, seperti tanggal hubungan dimulai atau peran dari entitas dalam hubungan tersebut.

4.4.4 Query dan Join untuk Mengakses Data Relasi

Query adalah perintah untuk mengakses dan memanipulasi data dalam database. Dalam model relasional, salah satu operasi paling penting adalah join, yang memungkinkan untuk mengombinasikan baris dari dua atau lebih tabel berdasarkan kolom-kolom yang terkait. Ada beberapa jenis join yang umum digunakan dalam SQL. *Inner join* mengembalikan hanya baris-baris yang memiliki nilai yang cocok di kedua tabel. *Left outer join* mengembalikan semua baris dari tabel sebelah kiri, dan baris-baris yang cocok dari tabel sebelah kanan. *Right outer join* adalah kebalikan dari left outer join. *Full outer join* mengembalikan semua baris dari kedua tabel.

Penggunaan join memungkinkan untuk memanfaatkan relasi antara tabel-tabel untuk menjawab query yang kompleks dengan menggabungkan informasi dari beberapa tabel. Namun, penggunaan join yang berlebihan dapat mempengaruhi performa query karena database harus melakukan operasi yang kompleks untuk mencocokkan baris-baris dari berbagai tabel. Oleh karena itu, pemahaman yang mendalam tentang bagaimana relasi-relasi dibangun dan cara mengakses data melalui relasi-relasi tersebut sangat penting untuk merancang sistem database yang efisien (Amran et al., 2019).

4.5 Integritas Data

4.5.1 Konsep Integritas Data dan Jenis-Jenisnya

Integritas data adalah keadaan di mana data yang tersimpan dalam database akurat, lengkap, konsisten, dan dapat dipercaya. Integritas data adalah salah satu aspek terpenting dalam manajemen database karena data yang tidak integral dapat menyebabkan pengambilan keputusan yang salah dan merusak reputasi organisasi. Ada beberapa jenis integritas data yang perlu dipertahankan dalam sistem database. Integritas domain adalah jenis integritas yang memastikan bahwa setiap atribut hanya menerima nilai-nilai yang valid dan sesuai dengan tipe data dan aturan-aturan yang telah ditentukan.

Integritas entitas memastikan bahwa setiap baris dalam tabel dapat diidentifikasi secara unik melalui kunci utama dan tidak ada

duplikat yang tidak perlu. Integritas referensial memastikan bahwa nilai-nilai dalam kolom kunci asing sesuai dengan nilai-nilai yang ada dalam kolom kunci utama tabel yang direferensikan, atau bernilai null jika diizinkan. Integritas semantik memastikan bahwa makna dan konteks dari data tetap terpelihara ketika data diakses atau dimodifikasi. Integritas temporal memastikan bahwa perubahan-perubahan pada data dicatat dengan waktu yang akurat sehingga dapat dilacak kapan suatu data diubah dan oleh siapa.

4.5.2 Aturan-Aturan Integritas dan Constraints

Aturan-aturan integritas dapat diimplementasikan dalam database melalui penggunaan constraints atau batasan-batasan yang dikenakan pada tabel dan kolom. *Primary key constraint* memastikan bahwa setiap baris dalam tabel dapat diidentifikasi secara unik dan tidak boleh ada nilai null. *Unique constraint* memastikan bahwa semua nilai dalam kolom atau kombinasi kolom adalah unik dan tidak boleh ada duplikat. *Foreign key constraint* memastikan integritas referensial dengan memverifikasi bahwa nilai-nilai dalam kolom kunci asing sesuai dengan nilai-nilai yang ada dalam tabel yang direferensikan.

Not null constraint memastikan bahwa kolom tertentu harus selalu memiliki nilai dan tidak boleh kosong. *Check constraint* memastikan bahwa nilai-nilai dalam kolom memenuhi kondisi logika yang telah ditentukan, misalnya bahwa nilai harus berada dalam rentang tertentu atau memenuhi pola tertentu. *Default constraint* menyediakan nilai default yang akan digunakan jika nilai tidak secara eksplisit disediakan ketika memasukkan data baru.

Dengan menerapkan constraints-constraints ini dengan benar, database dapat secara otomatis menegakkan aturan-aturan integritas dan mencegah pemasukan data yang tidak valid atau inkonsisten (Mo, 2024).

4.5.3 Validasi Data dan Pemeriksaan Integritas

Validasi data adalah proses untuk memverifikasi bahwa data yang dimasukkan ke dalam sistem memenuhi semua aturan integritas yang telah didefinisikan. Validasi dapat dilakukan pada berbagai tingkat, mulai dari tingkat aplikasi di mana validasi dilakukan sebelum data dikirim ke database, hingga tingkat database di mana validasi dilakukan oleh sistem manajemen database itu sendiri. Validasi pada tingkat aplikasi dapat memberikan feedback yang lebih baik kepada pengguna dan meningkatkan user experience, sementara validasi pada tingkat database memberikan perlindungan tambahan terhadap data yang tidak valid yang mungkin memasuki database melalui cara lain.

Pemeriksaan integritas melibatkan proses verifikasi secara berkala bahwa data dalam database masih memenuhi semua aturan integritas yang telah didefinisikan. Ini dapat dilakukan melalui query yang dirancang khusus untuk mendeteksi anomali atau inkonsistensi dalam data. Jika ditemukan pelanggaran integritas, tindakan korektif harus diambil untuk memperbaiki data dan memastikan bahwa penyebab dari pelanggaran tersebut diidentifikasi dan ditangani untuk mencegah terjadinya kembali di masa depan. Database administrator harus secara rutin melakukan pemeriksaan integritas dan memelihara prosedur backup dan recovery untuk memastikan

bahwa data dapat dipulihkan jika terjadi kerusakan atau kehilangan data.

4.5.4 Mekanisme Proteksi dan Pemulihan Data

Mekanisme proteksi data dirancang untuk mencegah kehilangan atau kerusakan data akibat berbagai faktor seperti kegagalan hardware, kesalahan pengguna, atau serangan keamanan. Salah satu mekanisme proteksi paling penting adalah backup data yang merupakan duplikasi lengkap dari data dalam database yang disimpan secara terpisah. Backup harus dilakukan secara berkala dengan frekuensi yang disesuaikan dengan seberapa sering data berubah dan seberapa penting data tersebut bagi organisasi. Ada berbagai strategi backup seperti full backup yang menyalin semua data, incremental backup yang hanya menyalin data yang telah berubah sejak backup terakhir, dan differential backup yang menyalin data yang telah berubah sejak full backup terakhir.

Recovery atau pemulihan data adalah proses mengembalikan data yang hilang atau rusak dari backup. Sistem database harus memiliki strategi recovery yang jelas dan prosedur yang telah diuji secara berkala untuk memastikan bahwa data dapat dipulihkan dengan cepat jika terjadi bencana. Transaction logging adalah mekanisme lain yang penting di mana setiap perubahan pada data dicatat dalam log file yang terpisah. Jika terjadi kegagalan, data dapat dipulihkan ke titik tertentu dengan menggunakan log file. Point-in-time recovery memungkinkan untuk memulihkan data ke kondisi pada waktu tertentu di masa lalu. Database administrator harus memiliki pemahaman yang mendalam tentang mekanisme-

mekanisme ini dan harus mengimplementasikannya dengan benar untuk memastikan bahwa data organisasi terlindungi dengan baik (Hayat & Kuswandi, 2025).

BAB 5: Perancangan Basis Data

Agus Salim, M.Kom.

5.1 Konsep Data Dasar

5.1.1 Definisi dan Tujuan Dasar Data

Basis data merupakan kumpulan data terorganisir yang disimpan dan diakses melalui suatu sistem manajemen basis data (Database Management System , DBMS) (Delplanque et al., 2018) . Basis data dirancang untuk menyimpan, mengelola, dan mengambil kembali data secara efisien dan aman. Tujuan utama basis data adalah menyediakan suatu mekanisme yang memungkinkan berbagai aplikasi dan pengguna untuk mengakses data yang sama dengan cara yang konsisten dan terkontrol, sekaligus menjaga integritas dan keamanan data.

Konsep basis data telah berkembang seiring dengan kemajuan teknologi informasi. Sebelum era basis data modern, organisasi menyimpan data dalam sistem file tradisional yang tersebar di berbagai aplikasi, menyebabkan redundansi data, inkonsistensi, dan kesulitan dalam pengelolaan data. Dengan mengadopsi basis data modern, organisasi dapat mengelola volume data yang besar dengan cara yang lebih terstruktur, efisien, dan aman. Basis data memungkinkan integrasi data dari berbagai sumber dan

memfasilitasi pengambilan keputusan yang lebih baik berdasarkan analisis data yang komprehensif.

5.1.2 Model Data dan Jenis-Jenis Basis Data

Model data mendefinisikan struktur logistik bagaimana data disimpan, diorganisir, dan diakses dalam basis data. Model data relasional merupakan model yang paling luas digunakan dalam praktik industri, di mana data diorganisir dalam bentuk tabel dengan baris dan kolom (Mandapuram & Hosen, 2018) . Setiap tabel mewakili entitas dalam domain bisnis, dan kolom mewakili atribut atau properti dari entitas tersebut. Hubungan antar tabel dibangun melalui kunci asing (foreign key *)* yang memungkinkan integrasi data lintas tabel.

Model data lain yang penting termasuk model data berorientasi objek yang memungkinkan penyimpanan data yang lebih kompleks dalam bentuk objek dengan atribut dan metode, serta model data NoSQL yang dioptimalkan untuk data tidak terstruktur dan semi-terstruktur (Mandapuram & Hosen, 2018) . Pemilihan model data bergantung pada karakteristik aplikasi, jenis data yang akan disimpan, dan kebutuhan kinerja yang spesifik. Basis data relasional tetap menjadi pilihan utama untuk aplikasi transaksional yang memerlukan konsistensi data yang ketat, sementara NoSQL semakin populer untuk aplikasi *big data* dan *analisis real-time* .

5.1.3 Karakteristik dan Keuntungan Basis Data

Basis data modern memiliki beberapa karakteristik fundamental yang membedakannya dari sistem penyimpanan data tradisional. Karakteristik pertama adalah *kemandirian data* atau

kemandirian data, yang memungkinkan perubahan pada struktur fisik dasar data tanpa mempengaruhi aplikasi yang menggunakannya. Kedua adalah *integritas data* atau integritas data yang memastikan data selalu konsisten dan akurat melalui mekanisme batasan dan validasi. Ketiga adalah *keamanan data* atau keamanan data yang melindungi data dari akses tidak sah dan manipulasi yang tidak diinginkan.

Keuntungan utama menggunakan basis data mencakup penghapusan redundansi data, peningkatan konsistensi data, peningkatan keamanan data, dan peningkatan kinerja dalam pengambilan dan manipulasi data (Delplanque et al., 2018) . Dengan basis data yang dirancang dengan baik, organisasi dapat mengurangi duplikasi data yang tidak perlu, memastikan bahwa semua aplikasi bekerja dengan informasi yang sama dan terbaru, dan mempermudah audit dan kontrol akses data. Selain itu, basis data memungkinkan skalabilitas yang lebih baik ketika organisasi berkembang dan volume data meningkat.

5.1.4 Evolusi dan Tantangan Kontemporer dalam Basis Data

Evolusi basis data dari waktu ke waktu mencerminkan perubahan kebutuhan bisnis dan kemajuan teknologi. Fase awal basis data didominasi oleh sistem hierarki dan jaringan, kemudian beralih ke model relasional yang evolusioner karena kemudahannya dan ketidakmampuan dalam query. Saat ini, tren menunjukkan penggunaan hybrid *polyglot persistence* di mana organisasi menggunakan kombinasi basis data relasional dan NoSQL untuk memenuhi kebutuhan yang beragam (Pokorný, 2018) .

Tantangan kontemporer dalam basis data mencakup pengelolaan volume data yang sangat besar (*big data*), kebutuhan akan *pemrosesan real-time* , integrasi data dari sumber heterogen, dan keamanan data di era ancaman siber yang semakin canggih. Perkembangan teknologi seperti cloud computing, blockchain, dan kecerdasan buatan membuka peluang dan tantangan baru dalam desain dan manajemen basis data. Organisasi harus memilih teknologi basis data yang paling sesuai dengan kebutuhan spesifik mereka sambil mempertimbangkan faktor-faktor seperti Skalabilitas, ketersediaan, dan biaya operasional.

5.2 Peranan Logis dan Fisik

5.2.1 Data Dasar Tahapan Perancangan

Perancangan basis data yang efektif memerlukan suatu pendekatan sistematis yang mencakup beberapa tahapan penting. Tahap pertama adalah *analisis kebutuhan* atau analisis kebutuhan, di mana analis bisnis dan desainer bekerja sama mengidentifikasi kebutuhan data organisasi, entitas-entitas utama, atribut-atribut yang perlu disimpan, dan hubungan antar entitas. Tahap kedua adalah perancangan konseptual (*conceptual design*) yang menghasilkan diagram *entitas-relationship* (ER) yang menggambarkan struktur data pada tingkat abstrak tanpa mempertimbangkan detail implementasi teknis.

Tahap ketiga adalah perancangan logistik (logical *design*) yang mengubah model konseptualisasi menjadi skema basis data

relasional dengan tabel-tabel, kolom-kolom, dan batasan-batasan yang spesifik. Tahap keempat adalah perancangan fisik (*desain fisik*) yang menentukan bagaimana data akan disimpan secara fisik di disk, termasuk penentuan indeks, partisi data, dan organisasi penyimpanan untuk mengoptimalkan kinerja. Tahap kelima adalah implementasi (*implementasi*) dan pengujian untuk memastikan data dasar berfungsi sesuai dengan spesifikasi yang telah ditetapkan.

5.2.2 Normalisasi dan Desain Skema Relasional

Normalisasi adalah proses merancang tabel-tabel dalam basis data relasional untuk menghilangkan redundansi data dan memastikan integritas data (Delplanque et al., 2018) . Proses normalisasi didasarkan pada teori *ketergantungan fungsional (fungsional* dependency) dan dilakukan melalui serangkaian bentuk normal (*bentuk normal*). Bentuk normal pertama (1NF) memastikan bahwa setiap atribut hanya mengandung nilai atomik yang tidak dapat dipecahkan lebih lanjut. Bentuk normal kedua (2NF) memastikan bahwa setiap atribut non-kunci bergantung penuh pada kunci utama, bukan hanya sebagian dari kunci komposit.

Bentuk normal ketiga (3NF) memastikan bahwa tidak ada ketergantungan transitif antara atribut-atribut non-kunci. Bentuk normal Boyce-Codd (BCNF) adalah bentuk normal paling ketat yang memastikan setiap penentu adalah kunci kandidat. Meskipun normalisasi bermanfaat dalam mengeliminasi anomali pembaruan, penghapusan, dan penyisipan, normalisasi yang berlebihan dapat menyebabkan banyak *join* query yang kompleks dan lambat. Oleh karena itu, beberapa desainer memilih untuk melakukan

denormalisasi yang terkontrol untuk meningkatkan kinerja query dengan mengorbankan beberapa aspek mendasar.

5.2.3 Penentuan Kunci, Kendala, dan Relasi

Kunci merupakan elemen penting dalam desain skema relasional yang berfungsi identifikasi secara unik setiap baris dalam tabel. Kunci utama (*primary key*) adalah atribut atau kombinasi atribut yang secara unik mengidentifikasi setiap baris, sedangkan kunci kandidat adalah atribut lain yang juga dapat berfungsi sebagai kunci unik. Kunci asing (*foreign key*) adalah atribut yang mereferensikan kunci utama dari tabel lain, membangun hubungan antar tabel dan memastikan integritas referensial.

Batasan adalah aturan-aturan yang membatasi nilai-nilai yang dapat disimpan dalam basis data untuk memastikan integritas data. Jenis-jenis kendala meliputi *NOT NULL* yang atribut selalu memiliki nilai, *UNIQUE* yang memastikan tidak ada duplikat nilai, *PRIMARY KEY* , *FOREIGN KEY* , dan *CHECK* yang membatasi nilai dalam rentang tertentu. Relationship atau relasi menggambarkan hubungan antara entitas, seperti one-to-one, one-to-many, dan many-to-many, yang perlu direpresentasikan dengan benar dalam skema basis data untuk memastikan integritas data dan kemudahan dalam query.

5.2.4 Dokumentasi dan Review Desain

Dokumentasi desain data dasar yang komprehensif sangat penting untuk memudahkan pemeliharaan, pengembangan, dan pelatihan pengguna. Dokumentasi harus mencakup diagram ER yang menunjukkan semua entitas, atribut, dan relasi, deskripsi untuk

setiap tabel dan kolom yang menjelaskan tujuan dan batasan nilai, serta daftar semua kunci dan batasan. Dokumentasi juga harus mencakup informasi tentang integrasi data dasar dengan aplikasi, prosedur untuk pembaruan skema, dan rencana backup serta pemulihan bencana.

Tinjauan desain melibatkan diskusi mendalam dengan pemangku kepentingan, pengembang, dan administrator basis data untuk memastikan desain memenuhi semua kebutuhan bisnis dan teknis. Tinjauan harus mencakup verifikasi bahwa semua entitas dan atribut yang diperlukan telah diidentifikasi, bahwa normalisasi telah diterapkan dengan tepat, bahwa integritas data dijamin melalui batasan yang sesuai, dan bahwa kinerja yang diharapkan dapat dicapai dengan desain yang diusulkan. Perbaikan desain berdasarkan feedback dari review dapat dilakukan sebelum implementasi untuk menghindari perubahan skema yang mahal di kemudian hari.

5.3 DBMS dalam Sistem Informasi

5.3.1 Fungsi dan Komponen Utama DBMS

Database Management System (DBMS) adalah perangkat lunak yang berfungsi mengelola data dasar dan menyediakan antarmuka bagi pengguna dan aplikasi untuk mengakses data (Mandapuram & Hosen, 2018) . DBMS menyediakan fungsi-fungsi penting seperti pembuatan data basis, penambahan dan perubahan struktur tabel, penyisipan, pembaruan, dan penghapusan data, serta pengambilan data melalui query yang kompleks. DBMS

juga melakukan kontrol akses, memastikan bahwa hanya pengguna yang berwenang yang dapat mengakses data tertentu, dan mengelola transaksi untuk memastikan konsistensi data ketika banyak pengguna mengakses data dasar secara bersamaan.

Komponen utama DBMS mencakup prosesor kueri yang memproses perintah SQL atau bahasa kueri lain yang dikirimkan oleh pengguna, mesin penyimpanan yang mengelola bagaimana data disimpan dan diambil dari disk, manajer transaksi yang mengelola transaksi untuk memastikan properti ACID (*Atomicity*, *Consistency*, *Isolation*, *Durability*), manajer keamanan yang mengelola kontrol akses dan data terenkripsi, dan manajer pemulihan yang memastikan data dasar dapat terjadi dari kegagalan sistem atau bencana.

5.3.2 DBMS Relasional vs NoSQL vs Hibrid

DBMS relasional seperti MySQL, PostgreSQL, dan Oracle telah menjadi standar industri selama puluhan tahun karena kehandalannya dalam mengelola data terstruktur dan kemampuan untuk menjamin konsistensi data melalui properti ACID (Pokorný, 2018) . Namun, meningkatnya volume data yang tidak terstruktur dan kebutuhan untuk skalabilitas horizontal membawa perubahan dalam lanskap DBMS. DBMS NoSQL seperti MongoDB, Cassandra, dan DynamoDB dirancang untuk menangani volume data besar dengan skema fleksibel dan kemampuan skalabilitas yang lebih baik, meskipun dengan tradeoff dalam hal jaminan konsistensi data.

Tren terkini menunjukkan penerapan pendekatan hybrid atau persistensi poliglot, di mana organisasi menggunakan kombinasi

basis data relasional untuk data terstruktur yang memerlukan konsistensi ketat, dan basis data NoSQL untuk data tidak terstruktur atau semi-terstruktur yang memerlukan skalabilitas tinggi (Pokorný, 2018) . Integrasi antara DBMS relasional dan NoSQL menciptakan tantangan teknis dalam memastikan konsistensi data lintas sistem, namun memberikan hasil maksimal dalam memilih teknologi terbaik untuk use case yang spesifik.

5.3.3 Integrasi DBMS dengan Aplikasi dan Cloud Computing

Integrasi DBMS dengan aplikasi enterprise memerlukan pemahaman mendalam tentang arsitektur aplikasi, pola akses data, dan kebutuhan kinerja. Aplikasi modern sering menggunakan kerangka Object-Relational Mapping (ORM) seperti Hibernate dan Entity Framework untuk mengabstraksi kompleksitas akses database dan memudahkan konstruksi query. Komputasi awan telah mengubah cara basis data dikelola, dengan penawaran seperti AWS RDS, Azure SQL Database, dan Google Cloud SQL yang menyediakan DBMS terkelola tanpa perlu mengelola infrastruktur yang mendasarinya (Jena et al., 2020) .

Manfaat penggunaan DBMS berbasis cloud mencakup skalabilitas otomatis, backup dan pemulihan bencana yang terkelola, dan biaya yang lebih dapat diprediksi dengan model pay-as-you-go. Namun, organisasi harus mempertimbangkan faktor-faktor seperti latensi jaringan, keamanan data di cloud, kepatuhan terhadap regulasi residensi data, dan optimalisasi biaya. Pendekatan cloud hybrid yang menggabungkan data dasar lokal dengan sumber daya

cloud juga semakin populer untuk organisasi yang memerlukan fleksibilitas maksimal dalam pengelolaan data.

5.3.4 DBMS dan Teknologi Baru

Perkembangan teknologi seperti blockchain, kecerdasan buatan, dan internet of things membawa implikasi baru dalam desain dan manajemen DBMS. Sistem basis data berbasis blockchain menawarkan kekekalan dan transparansi yang berguna untuk aplikasi yang memerlukan jejak audit yang tidak dapat diubah, seperti dalam manajemen rantai pasokan dan layanan kesehatan. AI dan pembelajaran mesin semakin terintegrasi ke dalam DBMS untuk optimasi kueri otomatis, pemeliharaan prediktif, dan deteksi anomali dalam pola akses data.

Internet of Things membawa tantangan baru dalam hal volume data yang sangat besar dan variasi tinggi dalam struktur data. Database time-series seperti InfluxDB dan Prometheus menjadi semakin penting untuk menangani sensor data yang dikirim secara kontinyu dari jutaan perangkat. DBMS masa depan akan memerlukan lebih adaptif terhadap perubahan kebutuhan aplikasi, lebih cerdas dalam optimasi kinerja, dan lebih tangguh dalam menjaga keamanan dan privasi data.

5.4 Optimasi Struktur Basis Data

5.4.1 Pengindeksan dan Optimasi Kueri

Indexing merupakan salah satu teknik optimasi database paling efektif yang bekerja dengan membuat struktur data khusus

untuk mempercepat pencarian data. Index utama yang paling sering digunakan adalah B-tree index yang mengorganisir data dalam struktur pohon yang seimbang, memungkinkan pencarian biner pencarian dengan kompleksitas O(log n). Index hash berguna untuk pencarian pencocokan tepat dengan kompleksitas O(1) rata-rata, sedangkan indeks bitmap efisien untuk kolom dengan kardinalitas rendah seperti status atau jenis gender.

Optimasi kueri adalah proses mengubah query SQL menjadi rencana eksekusi yang paling efisien. Pengoptimal kueri dalam DBMS menganalisis struktur kueri, data statistik, dan sumber daya yang tersedia untuk menentukan rencana eksekusi terbaik. Teknik-teknik optimasi kueri mencakup penulisan ulang kueri untuk menggabungkan subquery, join reordering untuk meminimalkan hasil perantara yang besar, dan predicate pushdown untuk memfilter data lebih awal dalam proses query. Administrator basis data perlu memahami cara membaca rencana eksekusi kueri dan mengidentifikasi kemacetan yang mungkin terjadi, seperti pemindaian tabel lengkap yang dapat dihindari dengan pengindeksan yang tepat.

5.4.2 Partisi dan Sharding

Partisi adalah teknik membagi data dalam satu tabel besar menjadi beberapa partisi yang lebih kecil berdasarkan kriteria tertentu seperti rentang nilai atau fungsi hash. Partisi horizontal membagi baris data menjadi beberapa tabel atau database yang berbeda, berguna untuk meningkatkan kinerja query dan memudahkan pemeliharaan seperti backup dan pengarsipan data

lama. Partisi vertikal membagi kolom data menjadi beberapa tabel yang berbeda, berguna ketika hanya subset kolom tertentu yang sering diakses bersama-sama.

Sharding adalah bentuk khusus dari partisi horizontal yang didistribusikan ke beberapa database server yang berbeda untuk mencapai skalabilitas horizontal. Kunci sharding harus dipilih dengan hati-hati untuk memastikan distribusi data yang merata dan meminimalkan kueri lintas shard yang mahal. Tantangan dalam sharding mencakup kesulitan dalam melakukan join lintas query shard, kompleksitas dalam resharding ketika menambah server baru, dan tantangan konsistensi dalam mengelola data terdistribusi.

5.4.3 Caching dan Basis Data Dalam Memori

Caching adalah teknik menyimpan data yang sering diakses di memori untuk mengurangi beban query ke database utama. Redis dan Memcached adalah cache in-memory yang populer yang dapat mempercepat akses data secara signifikan dibandingkan dengan akses disk. Strategi caching yang baik memerlukan pemahaman tentang pola akses data dan perlunya menyeimbangkan antara ukuran cache yang terbatas dengan kehangatan data yang diperlukan untuk aplikasi.

Database dalam memori seperti MemSQL dan Aerospike menyimpan seluruh dataset di RAM untuk mencapai latensi paling rendah dan throughput tertinggi. Basis data dalam memori berguna untuk aplikasi yang memerlukan analisis waktu nyata dan respons kueri latensi sangat rendah, namun memerlukan investasi signifikan dalam sumber daya memori. Kombinasi dari database on-disk

dengan cache layer dan in-memory database dapat memberikan arsitektur optimal untuk aplikasi dengan kebutuhan performa yang beragam.

5.4.4 Pemantauan dan Penyesuaian Kinerja

Pemantauan database secara kontinyu sangat penting untuk mengidentifikasi kemacetan dan masalah kinerja sebelum menjadi masalah kritis. Metrik-metrik penting yang perlu dimonitor mencakup pemanfaatan CPU, penggunaan memori, kecepatan I/O disk, waktu eksekusi kueri, dan status kumpulan koneksi. Tools seperti Enterprise Manager dari Oracle, Performance Insights dari AWS, dan Grafana dapat memvisualisasikan metrik-metrik ini dan memberikan peringatan ketika ambang batas tertentu terlampaui.

Penyetelan kinerja meliputi analisis query yang lambat, identifikasi indeks yang hilang, optimasi operasi gabungan, dan penyesuaian parameter database seperti ukuran buffer pool dan sort memory. Alat penasihat basis data dalam DBMS modern dapat memberikan rekomendasi otomatis tentang indeks yang perlu dibuat atau kueri yang perlu dioptimalkan. Perencanaan kapasitas juga penting untuk memastikan data dasar memiliki sumber daya yang cukup untuk pertumbuhan data dan beban query yang meningkat di masa depan.

5.5 Data Dasar Keamanan

5.5.1 Kontrol Akses dan Otentikasi

Keamanan dasar data dimulai dengan kontrol akses yang ketat untuk memastikan hanya pengguna yang berwenang yang dapat mengakses data. Otentikasi adalah proses verifikasi identitas pengguna melalui kredensial seperti nama pengguna dan kata sandi, atau melalui metode yang lebih aman seperti otentikasi multi-faktor. DBMS modern mendukung integrasi dengan layanan direktori seperti LDAP atau Active Directory untuk manajemen pengguna terpusat dan sistem masuk tunggal.

Otorisasi adalah proses menentukan apa yang dapat dilakukan pengguna setelah diautentikasi. Model kontrol akses seperti Discretionary Access Control (DAC) memungkinkan pemilik data untuk menentukan siapa yang dapat mengakses data mereka, sedangkan Role-Based Access Control (RBAC) mengorganisir izin berdasarkan peran yang dimainkan oleh pengguna. DBMS menyediakan kontrol izin granular yang dapat mencakup tingkat database, tabel, kolom, bahkan tingkat baris untuk mengimplementasikan kontrol akses yang terperinci.

5.5.2 Enkripsi dan Perlindungan Data

Enkripsi adalah teknik mengubah data menjadi bentuk terenkripsi yang tidak dapat dibaca tanpa kunci dekripsi yang tepat. Enkripsi at-rest melindungi data yang disimpan di disk dengan mengenkripsi file database, sedangkan enkripsi in-transit melindungi data yang dikirim melalui jaringan dari server database

ke aplikasi client. Enkripsi Data Transparan (TDE) menyediakan enkripsi saat tidak digunakan tanpa memerlukan perubahan aplikasi, secara otomatis mengenkripsi dan mendekripsi data saat diakses (Natarajan & Shaik, 2020) .

Manajemen kunci enkripsi sangat penting karena jika kunci hilang atau dikompromikan, data tidak dapat bocor atau dapat diakses oleh pihak yang tidak berwenang. Modul Keamanan Perangkat Keras (HSM) dan layanan manajemen kunci berbasis cloud seperti AWS KMS dapat menyediakan penyimpanan kunci yang aman dan manajemen rotasi kunci. Organisasi harus memiliki kebijakan yang jelas tentang bagaimana kunci dihasilkan, disimpan, diputar, dan dihancurkan.

5.5.3 Pencatatan Audit dan Deteksi Ancaman

Audit logging mencatat semua aktivitas yang terjadi pada database untuk tujuan akuntabilitas, kepatuhan, dan penyelidikan forensik. Entri log harus mencakup informasi tentang siapa yang melakukan aksi, aksi apa yang dilakukan, waktu aksi dilakukan, dan hasil dari aksi. Deteksi ancaman tingkat lanjut menggunakan pembelajaran mesin untuk mengidentifikasi pola akses yang tidak normal atau mencurigakan, seperti akses ke data sensitif di luar jam kerja normal atau dari lokasi geografis yang tidak biasa.

Sebagian besar regulasi kepatuhan besar seperti GDPR, HIPAA, dan SOC 2 memerlukan organisasi untuk mempertahankan log audit yang komprehensif dan mampu menyelidiki dan merespons insiden keamanan. Sistem logging terpusat dan Security Information and Event Management (SIEM) dapat mengagregasi log dari

berbagai sumber analisis untuk yang lebih baik dan peringatan dini terhadap ancaman keamanan.

5.5.4 Manajemen Kerentanan dan Praktik Terbaik Keamanan

Manajemen kerentanan melibatkan identifikasi, penilaian, dan remediasi kerentanan keamanan dalam DBMS dan aplikasi yang mengakses basis data. Patch dan pembaruan keamanan harus diterapkan secara tepat waktu untuk menutup kerentanan yang diketahui sebelum dapat dieksploitasi oleh penyerang. Praktik terbaik keamanan basis data mencakup prinsip hak istimewa paling rendah yang hanya memberikan izin minimum yang diperlukan untuk setiap pengguna, penilaian keamanan berkala dan pengujian penetrasi, dan klasifikasi data untuk menentukan tingkat proteksi yang tepat untuk berbagai jenis data.

Pemulihan bencana dan perencanaan kesinambungan bisnis merupakan aspek penting dari keseluruhan data dasar strategi keamanan. Pencadangan rutin dan kemampuan memulihkan dari pencadangan di lokasi alternatif memastikan bahwa data dasar dapat disimpan dengan cepat dari berbagai jenis bencana seperti kegagalan perangkat keras, kerusakan data, atau serangan ransomware. Pengujian prosedur pemulihan bencana secara berkala sangat penting untuk memastikan bahwa rencana benar-benar dapat dijalankan pada saat diperlukan.

BAB 6: Perancangan Arsitektur Sistem

Bilal Abdul Wahid, M.Kom.

6.1 Arsitektur Sistem

6.1.1 Pengertian Arsitektur Sistem

Arsitektur Sistem merupakan suatu kerangka kerja komprehensif yang mendeskripsikan bentuk dan struktur komponenkomponennya dan bagaimana mereka saling sesuai satu dengan lainnya. Arsitektur sistem menerjemahkan desain logis dari suatu sistem informasi ke dalam struktur fisik yang meliputi hardware, software, dukungan jaringan, metode pengolahan, dan keamanan. Produk akhir dari tahap desain sistem adalah desain sistem spesifikasi. Sama seperti seorang arsitek dimulai proyek dengan daftar persyaratan pemilik, seorang analis sistem harus mendekati arsitektur sistem dengan daftar keseluruhan. Sebelum mengambil keputusan, analis harus mempertimbangkan tujuh isu khusus yang akan mempengaruhi pilihan arsitektur diantaranya:

- Enterprise resource planning (ERP)
- Inisialisasi dan total cost of ownership (TCO)
- Skalabilitas
- Integrasi web
- Sistem legacy

- Pemilihan proses
- Isu keamanan

Dalam ilmu sistem informasi, “sistem arsitektur” merujuk pada rencana atau kerangka kerja yang menggambarkan struktur, komponen, interaksi, dan tata letak sistem informasi. Secara lebih rinci, sistem arsitektur adalah kerangka kerja konseptual yang membantu dalam merancang, mengembangkan, dan mengelola sistem informasi secara efektif.

Beberapa fungsi pada sistem arsitektur dalam sistem informasi dapat mencakup berbagai hal, yaitu:

Sistem arsitektur dapat mengorganisir dan menentukan struktur keseluruhan dari sistem informasi, bagaimana komponen sistem diatur, bagaimana data disimpan, dan bagaimana aliran informasi antara komponen-komponen sistem diatur.

Sistem arsitektur membantu pengelolaan manajemen sumber daya yang digunakan dalam sistem informasi, seperti perangkat keras, perangkat lunak, dan jaringan. Hal ini termasuk dalam pemilihan, konfigurasi, dan alokasi sumber daya yang diperlukan untuk menjalankan sistem secara efisien dan efektif.

Memastikan integrasi yang baik antara komponen-komponen sistem. Ini termasuk integrasi perangkat keras, perangkat lunak, basis data, dan aplikasi agar semuanya dapat bekerja dengan baik.

Sistem arsitektur memiliki keamanan informasi yang tinggi. Ini melibatkan implementasi langkah-langkah keamanan seperti autentikasi, otorisasi, enkripsi data, dan pemantauan kegiatan yang mencurigakan.

Arsitektur harus mendukung kinerja yang baik dan skalabilitas sistem informasi. Sistem harus mampu mengatasi pertumbuhan volume data dan permintaan pengguna tanpa mengalami degradasi kinerja yang signifikan.

Sistem arsitektur memiliki fleksibilitas yang tinggi agar dapat beradaptasi dengan perubahan kebutuhan bisnis dan teknologi. Ini memungkinkan perusahaan untuk menambahkan atau mengubah komponen sistem dengan mudah.

Sistem arsitektur bertanggung jawab atas manajemen data, termasuk pemodelan data, pengolahan data, dan penyimpanan data. Ini termasuk pemilihan basis data yang sesuai dan desain struktur data yang efisien.

Memiliki desain antarmuka pengguna agar tampilan, navigasi, dan interaksi antara pengguna dan sistem terlihat teratur dan sistematis.

Contoh sistem arsitektur dalam sistem informasi adalah mikroservis. Arsitektur berbasis mikroservis merupakan pendekatan yang populer dalam pengembangan sistem informasi modern. Dalam arsitektur ini, aplikasi dibangun sebagai sekumpulan mikroservis independen yang masing-masing memiliki fungsionalitas spesifik dan dapat berkomunikasi satu sama lain melalui antarmuka yang telah ditentukan. Keuntungan dari arsitektur berbasis mikroservis yaitu mampu mengelola traffic dengan mudah, fleksibilitas dalam pemilihan teknologi, manajemen pemeliharaan yang lebih mudah, dan tingkat keamanan yang tinggi.

Arsitektur sistem adalah cetak biru (*blueprint*) tingkat tinggi yang menggambarkan struktur dan operasi dari sistem teknologi informasi (IT) yang kompleks. Pada dasarnya, arsitektur sistem menetapkan kerangka kerja yang menjelaskan bagaimana berbagai elemen teknis—termasuk perangkat keras (*hardware*), perangkat lunak (*software*), jaringan, database, keamanan, dan integrasi—dikoordinasikan untuk bekerja bersama guna mencapai tujuan organisasi. Konsep arsitektur ini tidak hanya fokus pada komponen teknis, tetapi juga pada bagaimana komponen-komponen tersebut dirancang untuk memenuhi kebutuhan bisnis, operasional, dan pengguna akhir.

Arsitektur sistem sering diuraikan dalam beberapa lapisan atau tingkat abstraksi untuk memudahkan pemahaman dan pengelolaan. Lapisan-lapisan ini bisa mencakup:

Arsitektur Bisnis: Lapisan ini menggambarkan proses bisnis inti dan alur kerja yang didukung oleh sistem IT. Arsitektur bisnis menunjukkan bagaimana teknologi membantu organisasi mencapai tujuan strategisnya, seperti meningkatkan efisiensi, memaksimalkan keuntungan, atau memperkuat hubungan dengan pelanggan.

Arsitektur Aplikasi: Berfokus pada desain dan organisasi berbagai aplikasi yang digunakan oleh organisasi. Ini mencakup pemetaan aplikasi ke proses bisnis, alur data antar-aplikasi, dan bagaimana aplikasi tersebut diintegrasikan satu sama lain. Arsitektur aplikasi bertujuan untuk memastikan bahwa berbagai aplikasi yang digunakan dapat berkomunikasi dengan lancar dan mendukung fungsi bisnis yang diperlukan.

Arsitektur Data: Berhubungan dengan struktur, penyimpanan, pengelolaan, dan akses data dalam sistem. Ini mencakup perencanaan basis data, model data, dan arus data antar-komponen. Arsitektur data memastikan bahwa data yang relevan tersedia, akurat, dan dapat diakses dengan mudah oleh aplikasi atau pengguna yang membutuhkan serta dilindungi dari akses yang tidak sah.

Arsitektur Infrastruktur Teknologi: Mencakup perangkat keras, jaringan, dan elemen infrastruktur lainnya yang mendukung pengoperasian aplikasi dan penyimpanan data. Arsitektur ini memetakan kebutuhan sumber daya seperti *server*, perangkat penyimpanan, perangkat jaringan, dan teknologi *cloud* untuk memastikan performa, keandalan, dan skalabilitas sistem.

Arsitektur Keamanan: Berfokus pada perlindungan terhadap ancaman keamanan internal dan eksternal. Ini mencakup kebijakan keamanan, prosedur, alat dan teknologi seperti *firewall*, enkripsi, sistem deteksi intrusi, serta manajemen akses dan identitas. Arsitektur keamanan harus dirancang sedemikian rupa sehingga melindungi data sensitif dan menjaga kepatuhan terhadap peraturan yang berlaku.

Arsitektur Integrasi: Menggambarkan bagaimana sistem dan aplikasi yang berbeda dalam organisasi terhubung dan berkomunikasi satu sama lain. Ini termasuk penggunaan *middleware*, API (*Application Programming Interface*), ESB (*Enterprise Service Bus*), atau mekanisme lainnya

untuk mengintegrasikan aplikasi yang mungkin berjalan di berbagai platform dan lingkungan.

Arsitektur sistem yang efektif adalah kunci untuk mencapai efisiensi operasional, fleksibilitas, skalabilitas, dan keamanan yang optimal dalam lingkungan IT. Berikut adalah beberapa alasan mengapa arsitektur sistem yang baik sangat penting:

Efisiensi Operasional: Arsitektur yang dirancang dengan baik dapat mengurangi redundansi dan meningkatkan efisiensi dengan meminimalkan kompleksitas sistem. Ini memastikan bahwa sumber daya teknologi digunakan secara optimal dan tidak ada komponen yang tumpang tindih atau tidak diperlukan.

Skalabilitas dan Fleksibilitas: Organisasi perlu merespons perubahan bisnis dengan cepat, seperti ekspansi pasar, perubahan kebijakan, atau peningkatan permintaan pengguna. Arsitektur sistem yang efektif memungkinkan penambahan atau pengurangan kapasitas dengan mudah tanpa harus melakukan perombakan besar-besaran pada sistem yang ada.

Keamanan dan Kepatuhan: Dalam dunia yang semakin terhubung, keamanan menjadi salah satu aspek paling kritis. Arsitektur sistem yang dirancang dengan prinsip keamanan yang baik dapat melindungi data sensitif, mencegah serangan siber, dan memastikan kepatuhan terhadap regulasi yang berlaku, seperti GDPR atau ISO 27001.

Integrasi yang Mulus: Organisasi sering kali menggunakan berbagai aplikasi dan layanan dari vendor yang berbeda. Arsitektur sistem yang kuat memfasilitasi integrasi yang mulus antara berbagai

sistem sehingga data dan informasi dapat mengalir secara efisien di seluruh organisasi tanpa hambatan.

6.1.2 Langkah-Langkah Mengembangkan Arsitektur Sistem yang Efektif

Mengembangkan arsitektur sistem yang efektif memerlukan pendekatan yang sistematis dan terstruktur. Berikut adalah langkah-langkah kunci yang perlu dipertimbangkan:

Identifikasi Kebutuhan Bisnis dan Teknologi: Langkah pertama dalam mengembangkan arsitektur sistem adalah memahami kebutuhan bisnis organisasi. Ini mencakup pemahaman tentang tujuan strategis, proses bisnis inti, kebutuhan pengguna, dan tantangan yang dihadapi. Selain itu, identifikasi teknologi yang ada dan evaluasi untuk memenuhi kebutuhan tersebut. Analisis ini akan membantu merancang arsitektur sistem yang mendukung tujuan bisnis secara efektif.

Pilih Pendekatan Arsitektur yang Tepat: Ada beberapa pendekatan dalam merancang arsitektur sistem, seperti arsitektur monolitik, mikrolayanan (*microservices*), dan arsitektur berbasis *cloud*. Pilih pendekatan yang paling sesuai dengan kebutuhan bisnis dan teknologi organisasi. Misalnya, arsitektur mikrolayanan cocok untuk organisasi yang membutuhkan skalabilitas tinggi dan pengembangan yang cepat, sedangkan arsitektur monolitik mungkin lebih sesuai untuk aplikasi dengan kebutuhan interaksi yang ketat antara komponen-komponennya.

Rancang Komponen Utama dan Hubungan Antar-Komponen: Setelah pendekatan arsitektur dipilih, langkah

selanjutnya adalah merancang komponen utama dari sistem dan bagaimana mereka saling terhubung. Ini mencakup desain modul perangkat lunak, antarmuka aplikasi (API), basis data, serta infrastruktur jaringan dan keamanan. Pertimbangkan bagaimana setiap komponen berinteraksi serta bagaimana data akan diakses dan dikelola.

Pertimbangkan Aspek Keamanan dari Awal: Keamanan harus menjadi prioritas utama dalam merancang arsitektur sistem. Integrasikan keamanan di setiap lapisan arsitektur, mulai dari autentikasi dan otorisasi pengguna, enkripsi data, *firewall*, hingga monitoring dan deteksi ancaman. Dengan memasukkan prinsip keamanan dari awal, risiko keamanan dapat diminimalkan dan kepatuhan terhadap regulasi lebih mudah dicapai.

Uji dan Validasi Arsitektur: Sebelum diimplementasikan secara penuh, arsitektur sistem harus diuji dan divalidasi untuk memastikan bahwa desain yang diusulkan dapat memenuhi kebutuhan bisnis dan teknis. Lakukan uji coba untuk mengidentifikasi potensi kelemahan, seperti *bottleneck* performa, masalah integrasi, atau celah keamanan. Umpan balik dari proses ini harus digunakan untuk menyempurnakan desain arsitektur.

Dokumentasi dan Pemeliharaan: Dokumentasi yang baik sangat penting untuk keberhasilan jangka panjang arsitektur sistem. Dokumentasikan setiap keputusan desain, diagram arsitektur, serta panduan implementasi dan pemeliharaan. Dokumentasi ini akan sangat membantu dalam proses pemeliharaan, *troubleshooting*, dan pengembangan lebih lanjut di masa depan.

Review **dan Iterasi Berkala:** Teknologi dan kebutuhan bisnis terus berkembang. Oleh karena itu, penting untuk melakukan *review* dan iterasi berkala terhadap arsitektur sistem yang ada. Dengan melakukan penilaian rutin, organisasi dapat menyesuaikan arsitektur mereka dengan kebutuhan dan tantangan baru sehingga sistem tetap relevan dan efektif.

6.2 Arsitektur Software Design

Arsitektur software design merupakan fondasi penting dalam pengembangan perangkat lunak. Arsitektur ini menggambarkan struktur sistem secara keseluruhan, termasuk komponen-komponen utama, hubungan antar komponen, serta prinsip-prinsip desain yang digunakan untuk mencapai tujuan fungsional dan non-fungsional. Desain arsitektur yang baik dapat meningkatkan kualitas software, memudahkan pemeliharaan, dan mempercepat proses pengembangan.

6.3.1 Pengertian Arsitektur Software

Arsitektur software adalah kerangka kerja yang menentukan bagaimana bagian-bagian dari sebuah sistem perangkat lunak diorganisasi dan berinteraksi satu sama lain. Arsitektur ini mengatur keputusan strategis seperti pemilihan teknologi, pola desain, serta standar yang digunakan dalam pembangunan software. Menurut Bass, Clements, dan Kazman (2012), arsitektur perangkat lunak menjelaskan struktur sistem perangkat lunak dan menyediakan peta jalan bagi pengembang.

6.3.2 Komponen Utama Arsitektur Software Design

Beberapa komponen utama dalam arsitektur software design meliputi:

- Komponen: Modul atau unit fungsional dari software yang menjalankan tugas tertentu.
- Konektor: Mekanisme komunikasi antar komponen, bisa berupa API, protokol komunikasi, atau middleware.
- Konfigurasi: Cara pengorganisasian komponen dan konektor dalam sistem.

6.3.3 Pola Arsitektur Software

Berbagai pola arsitektur telah dikembangkan untuk memenuhi kebutuhan sistem yang berbeda, antara lain:

- Layered Architecture: Memisahkan sistem menjadi beberapa lapisan dengan tanggung jawab berbeda, misalnya presentasi, bisnis, dan data.
- Microservices Architecture: Mengembangkan aplikasi sebagai kumpulan layanan kecil yang independen dan dapat di-deploy secara terpisah.
- Event-Driven Architecture: Sistem yang beroperasi berdasarkan event atau kejadian yang terjadi secara asinkron.
- Client-Server Architecture: Pemisahan antara penyedia layanan (server) dan pengguna layanan (client).

6.3.4 Faktor yang Mempengaruhi Desain Arsitektur

Beberapa faktor yang perlu diperhatikan dalam desain arsitektur antara lain:

- Kebutuhan Fungsional dan Non-Fungsional: Seperti performa, skalabilitas, keamanan, dan maintainabilitas.
- Teknologi yang Digunakan: Bahasa pemrograman, platform, dan framework.
- Tim Pengembang: Keahlian dan pengalaman pengembang.
- Batasan Proyek: Anggaran, waktu, dan sumber daya.

6.3.5 Manfaat Arsitektur Software Design

- Memudahkan komunikasi antar tim pengembang.
- Meningkatkan kualitas dan konsistensi software.
- Mempermudah perawatan dan pengembangan lanjutan.
- Mendukung skalabilitas dan fleksibilitas sistem.

6.3.6 Tantangan dalam Arsitektur Software Design

- Menyesuaikan desain dengan kebutuhan bisnis yang dinamis.
- Menghadapi kompleksitas sistem yang meningkat.
- Mengelola risiko teknologi baru yang belum teruji.
- Menjaga keseimbangan antara performa dan maintainabilitas.

Arsitektur software design adalah elemen krusial yang menentukan keberhasilan pengembangan perangkat lunak. Dengan memahami pola-pola arsitektur, faktor pendukung, serta tantangan yang ada, pengembang dapat membuat sistem yang handal, efisien, dan mudah dikembangkan di masa depan.

Definisi, Komponen, Contoh, dan Diagram System Architecture dalam Konteks Sistem Informasi

System Architecture adalah model konseptual yang mendefinisikan struktur, perilaku, dan lebih banyak pandangan dari

sebuah sistem. System Architecture sendiri dapat terdiri dari komponen-komponen sistem dan subsistem yang dikembangkan, yang akan bekerja sama untuk mengimplementasikan sistem secara keseluruhan. Penerapan system architecture yang baik dalam suatu organisasi sangat diperlukan supaya sistem informasi yang berjalan dapat sesuai dengan kebutuhan dan sistem yang digunakan dapat berjalan dengan baik.

System Architecture juga dilengkapi dengan sistem-sistem modern yang terdapat untuk membantu proses berjalannya sistem tersebut. Pembangunan system architecture yang baik dapat dibantu dengan memanfaatkan computing modern yang merupakan bagian dari kemajuan teknologi yang memungkinkan membantu dalam membangun system architecture yang lebih baik dan efisien. Hal ini umumnya dijalankan dengan computing devices yang memadai sesuai kebutuhannya masing-masing.

Computing devices, merupakan sebuah perangkat komputasi yang terdiri dari unit mandiri / beberapa unit yang saling berhubungan. Perangkat-perangkat yang menyediakan serangkaian fungsi tertentu, seperti server, telepon, pengatur pribadi, atau fungsi yang lebih umum seperti pada PC/ laptop. Dalam sistem modern, menggunakan server dimana server tersebut digunakan untuk mengelola data sumber dan mengaktifkan/ membuka akses kepada user dan juga komputer-komputer untuk dapat diakses data sumbernya, biasanya tempat untuk penyimpanan data yang penggunaannya tergolong tinggi akan dilakukan pada server farm, dimana terdapat banyak dan database besar. Namun, dalam artikel

ini, akan lebih berfokus pada pembahasan mengenai komponen yang diperlukan, contoh, dan diagram dari system architecture.

6.3 Komponen System Architecture

6.3.1 Perangkat Keras (Hardware)

Merupakan komponen dalam system architecture yang memungkinkan untuk menyediakan perangkat keras untuk mengaplikasikan system architecture ke dalam suatu sistem yang dirancang. Hardware yang dikaitkan dengan system architecture umumnya terdiri atas storage sebagai tempat penyimpanan dalam wujud fisik, CPU, ataupun perangkat I/O seperti keyboard dan mouse. Dalam membangun system architecture, sangat perlu diperhatikan spesifikasi minum yang diperlukan supaya hardware yang digunakan mampu untuk menjalankan sistem tersebut, sehingga proses yang dijalankan oleh sistem dapat berjalan dengan baik dan mendukung aktivitas yang berjalan. Beberapa hal yang perlu dipertimbangkan dalam memilih hardware untuk mendukung suatu system architecture adalah kinerja, kebutuhan pengguna, dan biaya, supaya nantinya hardware yang dipilih mampu membantu sistem berjalan dengan baik.

6.3.2 Jaringan (Network)

Komponen selanjutnya adalah jaringan (Network). Untuk network contoh yang dapat ditemukan dan kita pakai dalam kehidupan sehari-hari adalah Local Area Network (LAN), Wide Area Network (WAN), World Wide Web (WWW), Uniform Resource Locator (URL), dan Hyperlink. Untuk LAN dari kata local

itu sudah kita ketahui kalau jaringan komputernya itu hanya mencakup suatu wilayah local tertentu saja dan hanya dapat digunakan oleh orang yang berada di area tersebut. Kelebihan dari LAN adalah tidak terlalu mengeluarkan biaya yang banyak karena mencakup area yang kecil dan untuk kekurangannya adalah area jaringannya terbatas karena hanya area lokal saja. Untuk WAN dari kata wide itu sendiri sudah berarti jaringan komputer itu luas cakupannya bisa satu negara hingga antar benua. Kelebihan WAN adalah dapat mengantar suatu informasi atau file untuk antar benua dan untuk kekurangannya adalah karena WAN ini mencakup area yang besar seperti antar benua, maka perlu mengeluarkan biaya yang lebih mahal daripada LAN. Untuk World Wide Web adalah sebuah kumpulan situs web yang terhubung dengan komputer melalui internet, di dalam web tersebut dapat menyajikan berupa server informasi seperti www.google.com.

Selanjutnya untuk URL adalah sebuah rangkaian karakter yang memiliki format tertentu dimana menjadi sebuah alamat dari internet. URL ini disebut sebuah alamat karena dapat menunjukan ke alamat World Wide Web atau WWW, selain itu URL juga dapat digunakan untuk mengunduh video, gambar, dan lain-lain. Contoh URL adalah https://www.google.com. Jika dilihat dari struktur tersebut https:// adalah protokolnya dimana menunjukan jenis server, www adalah subdomain nya, google adalah domainnya, dan yang terakhir .com adalah top level domain. Yang terakhir adalah hyperlink, hyperlink itu sendiri digunakan untuk menghubungkan suatu halaman dengan halaman lainnya yang terdapat pada suatu

artikel, berita, dan lain-lain yang ditemukan pada internet. Hyperlink ini dibuat dalam bentuk teks dan kemudian disambungkan dengan halaman yang dituju.

6.3.3 Perangkat Lunak (Software)

Software juga merupakan hal penting, dimana software sendiri dibagi menjadi application software, app, dan system software. Application software memiliki arti bahwa program yang menjalankan pekerjaan untuk pengguna. App menjelaskan bahwa sebuah custom program biasanya untuk laptop atau smartphone, dan system software yang dimana bekerja seperti penghubung yang memegang segalanya bersatu. Di dalam sebuah server, software sendiri dapat berupa terdapat operating system, web server, application programs, dan database management system. Dan di dalam personal computing devices, software juga dapat berupa operating system, web browser, plug-ins, dan apps. Untuk web-based applications, karakteristiknya adalah menggunakan sebuah web browser, diakses melalui URL, terdapat pada web server, dan menggunakan IP protocol standard. Untuk Embedded Software, karakteristiknya adalah seperti aplikasi software atau metode yang tertanam pada aplikasi lain seperti browser, dan juga berupa toolbars, plug-ins, dann widgets.

6.3.4 Database

Database merupakan komponen opsi bagi pengguna sistem untuk memiliki system architecture yang menggunakan penyimpanan, pengelolaan, dan memanipulasi data yang ada. Tentunya komponen ini cukup penting untuk diperhatikan

keamanannya, karena menyangkut dengan data yang termuat dari sistem yang ada. Beberapa keamanan yang bisa diterapkan terhadap database dari suatu sistem bisa dilakukan dengan melakukan verifikasi dan autentikasi terhadap pengakses dari data dan melakukan enkripsi terhadap data pada database. Selain itu, perlu juga diperhatikan tentang kemampuan database untuk saling terintegrasi antar komputer dalam organisasi, hal ini bertujuan untuk memungkinkan terjadinya aktivitas informasi yang baik dan efisien. Hal lainnya yang perlu diperhatikan adalah desain database yang perlu disesuaikan dengan kebutuhan dari sistem yang dibuat, supaya database tersebut dapat sesuai dengan kebutuhan pengguna dan mendukung proses yang dijalankan sistem dengan mampu meningkatkan kinerja sistem.

6.3.5 Protocols and Web Protocols

Protocol merupakan sebuah bahasa atau aturan yang mengatur terjadinya hubungan, komunikasi, dan perpindahan data antara 2 atau lebih komputer. Tujuan penggunaan protocols adalah untuk menghubungkan pengirim dan pengguna agar komunikasi dapat berjalan lancar. Fungsi lain dari protocols adalah untuk sebagai pelengkap informasi yang akan dikirimkan bersama dengan alamat, kode-kode koreksi, dan lainnya. Paket data ini dinamakan dengan Frame. Data kemudian dikirimkan dalam blok-blok dan dikendalikan oleh Protocol Data Unit (PDU) dimana masing-masing PDU berisi kontrol informasi dan data.

Protocols dapat memberikan kontrol penuh terhadap flow, komunikasi dan error yang mungkin terjadi di dalam pertukaran

data. Kontrol terhadap komunikasi adalah dengan menggunakan protocols maka user dapat membuat hubungan, mengirim data dan juga mengakhiri hubungan dengan mudah. Selain itu kontrol terhadap flow berarti menggunakan protocols user dapat mengatur seluruh kelancaran maupun mengatur pemberhentian proses pengiriman dan penerimaan data. Dan protocols juga dapat mengawasi apakah di saat transfer data terdapat error atau tidak. Tentu saja bukan hanya ini saja fungsi dari protocols tetapi sangat banyak lagi seperti protocols bisa digunakan sebagai alat pengatur keamanan, maupun membagi data yg dikirim ke beberapa data kecil, dll.

Web Protocols atau yang lebih dikenal sebagai IP merupakan sebuah protocols lapisan yang digunakan oleh TCP/IP yang berada di internet dan digunakan untuk melakukan pengalamatan dan routing data di antara host di dalam jaringan komputer. Protokol IP adalah salah satu protokol terpenting dari keluarga protokol TCP/IP. Paket IP membawa data aktual yang dikirim dari satu titik ke titik lain di seluruh jaringan. Metode yang digunakan adalah Connectionless, artinya tidak ada sesi koneksi yang perlu dibuat dan dipelihara. Selain itu, protokol ini juga tidak menjamin pengiriman data, melainkan diserahkan kepada protokol lapisan atas (lapisan transport pada model referensi OSI atau lapisan antar-host pada model referensi DARPA), yakni protocols TCP.

Contoh – Contoh Protocols

Seperti yang sudah di jelaskan di atas bahwa protocols adalah sebuah aturan yang mengatur hubungan maupun perpindahan data

antar komputer. Tentu saja protocols memiliki banyak jenisnya seperti :

1. TCP/IP

merupakan standar transfer data yang digunakan oleh komunitas Internet untuk bertukar data dari satu komputer ke komputer lain di Internet. Protokol ini tidak dapat berdiri sendiri tanpa bantuan komponen lain, karena TCP/IP sendiri hanya berfungsi sebagai protokol suite.

protocols ini merupakan protokol yang paling banyak digunakan saat ini dan biasanya berupa perangkat lunak dalam sistem operasi. Perangkat lunak yang menangani masalah TCP/IP biasanya disebut aplikasi stack TCP/IP. Protokol ini pertama kali dikembangkan pada tahun 1970-an dan 1980-an dan dianggap sebagai protokol standar untuk menghubungkan komputer ke jaringan area luas, khususnya WAN.

2. HTTP

protocols ini merupakan protocols yang sering di jumpai di dalam web dikarenakan protocols ini digunakan oleh para webmaster untuk melakukan transfer halaman web. Selain itu tugas dari HTTP adalah untuk menerjemahkan suatu pesan agar bisa di format dan di kirim dari server ke user

3. TelNet

TelNet merupakan sebuah protocols dalam LAN (Local Area Network) dahulu TelNet sangat sering digunakan tetapi seiring berjalannya waktu TelNet sudah jarang dikarenakan memiliki kekurangan dan resiko terhadap keamanan.

6.4 Contoh System Architecture (Client-Server Architecture)

System Architecture design memiliki tujuan untuk menentukan komponen dari sebuah software sistem informasi yang akan dijalankan pada hardware yang terdapat di dalam sistem. Salah satu system architecture yang sering dijumpai adalah Client-Server Architecture, yang merupakan model arsitektur yang memiliki pembagian interaksi antara komputer klien dengan server. Dalam hal ini, klien bertindak sebagai peminta layanan atau data, dan server sebagai penyedia layanan atau data. Terdapat 3 macam architecture yang biasanya dipakai, yakni 2-Tiered client-server architecture, 3-Tiered client server architecture, dan N-tiered client server architecture.

1. Two-tiered client server architecture

Sistem ini menggunakan 2 set komputer yang disebut sebagai client dan server. Server yang ada menangani data dan client yang ada menangani aplikasi dan logika presentasi yang dilihat oleh pengguna. Two tiered client server biasanya digunakan pada organisasi yang biasanya kecil, karena jenis ini mudah untuk dipelajari, memiliki skalabilitas yang mudah dan kinerja yang cepat.

2. Three-tiered client server architecture

Pada three-tiered, konsep akan menggunakan 3 set komputer yang terdiri dari client devices, application server, dan database server. Dimana client hanya menampilkan logika presentasi, application server yang menangani logika bisnis dan database server

menangani logika data access dan data storage. Konsep ini biasa digunakan pada perusahaan yang cukup besar, dimana memiliki tingkat keamanan yang lebih tinggi.

3. N-tiered client server architecture

Konsep n tier membagi aplikasi menjadi lapisan logis dan tingkatan fisik, dimana lapisan berdiri sebagai cara untuk memisahkan tanggung jawab dan mengelola ketergantungan. Setiap lapisan memiliki tanggung jawab tertentu. Lapisan yang lebih tinggi dapat menggunakan layanan lapisan yang lebih rendah, tetapi tidak untuk sebaliknya.

6.5 Diagram System Architecture

System Architecture juga dilengkapi dengan beberapa diagram yang dapat membantu menjelaskan sebuah konsep tertentu. Diagram yang terdapat dapat mengabstraksi hubungan, batasan dan batas antara komponen sistem perangkat lunak. Ini merupakan sebuah alat yang penting yang dapat memberikan ikhtisar komprehensif tentang peta jalan penerapan dan pengembangan fisik sistem di perangkat lunak. Diagram – diagram ini harus melakukan berbagai tugas, mulai dari menyampaikan informasi tentang desain sistem sehingga pengguna dapat memahami dan menggunakannya, bisa juga sebagai pembantu membuat keputusan. Jenis-jenis diagram akan membantu banyak tim berbeda di seluruh perusahaan, termasuk tim marketing, tim IT, tim Teknik, dan pihak lainnya yang memiliki kepentingan untuk berurusan dengan proses yang melibatkan tahapan. Diagram

akan membantu membagi bisnis menjadi beberapa lapisan yang mengilustrasikan interaksi antara berbagai sistem dengan konsumen. Diagram – diagram tersebut adalah location diagrams, network diagrams, dan deployment diagrams.

1. Location Diagram

Dimulai dari yang pertama yaitu location diagrams, dimana berfungsi untuk mengidentifikasi lokasi penempatan geografis dari sebuah hardware, software, dan users. Menggambarkan di lokasi mana yang menghosting di aplikasi mana, juga mengidentifikasi teknologi ataupun suatu aplikasi yang digunakan di lokasi tertentu, juga mengidentifikasi di lokasi mana penggunaan bisnis biasanya berinteraksi dengan aplikasi tersebut.

2. Network Diagram

Selanjutnya, terdapat network diagrams, yang berguna untuk menjelaskan bagaimana application software itu diluncurkan melewati hardware dan sistem software. Network Diagram berdiri sebagai representasi visual dari arsitektur jaringan. Ini memetakan struktur jaringan dengan berbagai simbol dan koneksi garis yang berbeda – beda. Network diagram menggunakan sekumpulan node dan garis penghubung untuk menampilkan bagaimana elemen-elemen dalam suatu jaringan terhubung.

3. Deployment Diagram

Terakhir, adalah deployment diagrams, dimana berguna untuk menjelaskan bagaimana komponen dari sebuah network saling berhubungan. Deployment diagram adalah sebuah jenis diagram UML yang menunjukkan arsitektur eksekusi sistem, termasuk node

seperti lingkungan eksekusi hardware atau software, dan juga middleware yang menghubungkannya. Deployment diagram biasanya digunakan untuk memvisualisasikan fisik hardware dan software dari suatu sistem.



Gambar: *Architectur Development Cycle*

Tahapan-tahapan tersebut dapat dijelaskan sebagai berikut[14]:

1. Phase A:

Architecture Vision (Visi Arsitektur) Tahap ini menggambarkan batasan-batasan dari rancangan arsitektur. Pada tahapan ini dilakukan pendefinisian ruang lingkup, batasan-batasan dan ekspektasi dari rancangan arsitektur, untuk kemudian menetapkan visi arsitektur yang diusulkan. Konteks bisnis divalidasi untuk menyusun statement of architecture work.

2. Phase B:

Business Architecture (Arsitektur Bisnis) Mendefinisikan kondisi awal arsitektur bisnis, menentukan arsitektur bisnisyang diinginkan, dan penentuan tools yang akan digunakan.

3. Phase C:

Information System Architecture (Arsitektur Sistem Informasi) Membangun arsitektur sistem informasi yang diinginkan, arsitektur ini meliputi 2 (dua) arsitektur yaitu data dan aplikasi.

- Arsitektur Data (Data Architecture) Arsitektur data melakukan identifikasi entitas data, serta menggambarkan asosiasi data dengan proses dan skema data. Identifikasi entitas data dilakukan berdasarkan arsitektur bisnis yang ada. Aliran informasi antara sistem didekomposisiskan sebagai entitas data.
- b. Arsitektur Aplikasi (Applications Architecture) Sebagai bagian dari tahap Arsitektur Sistem Informasi, pada tahap ini arsitektur dari apliakasi-aplikasi yang tersedia dan relevan dalam Enterprise Continum diidentifikasi dan dipertimbangkan. Pada tahap ini, arsitektur aplikasi ddiusul sesuai dengan kebutuhan.

4. Phase D :

Technology Architcture (Arsitektur Teknologi)

Sasaran dari tahapan ini adalah untuk membangun arsitektur teknoogi yang akan dijadikan dasar pada saat implementasi.

5. Phase E :

Opportunities and Solutions (Peluan dan Solusi) Pada tahap ini peluang-peluang bisnis baru dari arsitektur pada tahap-tahap sebelumnya yang mungkin muncul diidentifikasi.hasil dari fase ini merupakan dasar dari rencana implementasi yang diperlukan untuk mencapai sasaran rancangan arsitektur.

6. Phase F :

Migration Planning (Perencanaan Migrasi) Tahap ini untuk membuat suatu rencana migrasi, termasuk prioritas pekerjaan. Sasaran dari tahap ini adalah memilih beberapa proyek-proyek implementasi berdasarkan prioritas utama. Pada tahap ini roodmap dari keseluruhan implementasi disusun.

7. Phase G :

Implementation Governance (Tata Kelola Implementasi) Tahapan ini bertujuan untuk menyusun suatu tata laksana implementasi, termasuk menyusun dan memformalisasi tim, menyusun manajemen proyek, membuat suatu manajeman komunikasi dari proyek tersebut, dll.

8. Phase H :

Architecture Change Management (Arsitektur Manajemen Perubahan) Tahapan ini merupakan tahapan penting karena infrastruktur TI akan terus berkembang menyesuaikan dengan kebutuhan bisnis yang ada. Sasaran dari tahapan ini adalah membangun suatu arsitektur proses manajemen perubahan bagi dasar arsitektur yang baru yang mana dilakukan setelah tahapan tata laksana implementasi dilaksanakan.

Kedelapan tahapan utama tersebut didukung oleh tahapan persiapan (preliminary) dan tahapan manajemen prasyarat (requirement management) di akhir proses.

- Preliminary Phase:Framework and Principles Fase Preliminary merupakan fase awal yang merupakan persiapan sebelum merencanakan sebuah arsitektur enterprise, fase ini bertujuan untuk menjelaskan setiap tahapan-tahapan dari kerangka kerja serta metodologi dari setiap perencanaan, melaksanakan tools arsitektur dan prinsip-prinsip Arsitektur Enterprise.
- Requirement Management Tahapan ini bertujuan untuk menyediakan proses pengelolaan kebutuhan arsitektur, mengidentifikasi kebutuhan enterprise, menyimpan lalu memberikannya kepada fase yang relevan.

BAB 7: Desain Antarmuka Pengguna (UI/UX)

Henny Destiana,M.Kom.

7.1 Prinsip Desain Antarmuka

7.1.1 Definisi dan Elemen Dasar User Interface

User Interface (UI) merupakan bagian dari sistem yang memungkinkan interaksi antara pengguna dan mesin, mencakup semua elemen visual dan interaktif yang pengguna lihat dan gunakan untuk berkomunikasi dengan sistem. Elemen dasar UI meliputi teks, tombol, ikon, menu, jendela dialog, formulir input, dan berbagai komponen interaktif lainnya yang dirancang untuk memfasilitasi komunikasi dua arah antara pengguna dan aplikasi. Setiap elemen UI harus dirancang dengan hati-hati untuk memastikan bahwa fungsinya jelas, dapat diakses dengan mudah, dan konsisten dengan desain keseluruhan sistem.

Prinsip dasar dalam desain UI adalah kesederhanaan, konsistensi, dan kejelasan. Kesederhanaan berarti menghindari elemen-elemen yang tidak perlu dan memastikan bahwa antarmuka tidak membingungkan pengguna dengan terlalu banyak informasi atau pilihan. Konsistensi berarti menggunakan elemen, warna, tipografi, dan pola interaksi yang sama di seluruh sistem sehingga pengguna dapat dengan mudah mempelajari bagaimana sistem

bekerja. Kejelasan berarti membuat tujuan setiap elemen UI jelas sehingga pengguna tidak perlu menebak-tebak apa yang akan terjadi ketika mereka mengklik tombol atau mengisi formulir. Desain UI yang baik harus transparan, artinya pengguna fokus pada tugas mereka bukan pada antarmuka itu sendiri (Riski et al., 2025).

7.1.2 Prinsip Desain Visual dan Estetika

Desain visual yang efektif adalah fondasi dari UI yang berkualitas tinggi. Prinsip-prinsip desain visual mencakup penggunaan warna, tipografi, ruang putih, dan tata letak yang harmonis untuk menciptakan antarmuka yang menarik secara visual namun tetap fungsional. Warna memiliki dampak psikologis yang signifikan pada persepsi pengguna, dengan warna-warna hangat seperti merah dan orange cenderung menarik perhatian, sementara warna-warna sejuk seperti biru dan hijau memiliki efek menenangkan. Pemilihan palet warna harus dilakukan dengan pertimbangan yang cermat untuk memastikan kontras yang cukup untuk aksesibilitas dan kesatuan visual yang kohesif.

Tipografi juga memainkan peran penting dalam desain UI. Pemilihan font yang tepat, ukuran teks, dan jarak antara baris mempengaruhi keterbacaan dan estetika antarmuka. Ruang putih atau *negative space* adalah area kosong dalam desain yang membantu mengurangi kebisingan visual dan membuat konten menjadi lebih mudah dibaca. Tata letak atau *layout* harus mengikuti prinsip-prinsip seperti keseimbangan, penekanan hierarki visual yang jelas untuk membimbing pengguna melalui informasi dalam urutan yang tepat. Dengan menggabungkan prinsip-prinsip desain

visual ini dengan hati-hati, desainer dapat menciptakan antarmuka yang tidak hanya indah secara visual tetapi juga fungsional dan mudah digunakan (Masveta & Manyangara, 2025).

7.1.3 Prinsip Interaksi dan Feedback

Interaksi yang baik dalam UI melibatkan memberikan umpan balik yang jelas kepada pengguna tentang hasil dari tindakan mereka. Umpan balik visual seperti perubahan warna tombol ketika diklik, animasi transisi, atau pesan konfirmasi sangat penting untuk membantu pengguna memahami bahwa sistem telah menerima dan memproses input mereka. Tanpa umpan balik yang jelas, pengguna mungkin merasa bingung tentang apakah tindakan mereka berhasil atau tidak, yang dapat menyebabkan pengalaman pengguna yang frustratif.

Waktu respons sistem juga penting dalam prinsip interaksi. Sistem harus memberikan umpan balik dalam waktu yang cepat sehingga pengguna tidak merasa tertunda atau bingung. Jika sistem memerlukan waktu untuk memproses permintaan, indikator kemajuan atau pesan penunggu harus ditampilkan untuk membuat pengguna tetap terinformasi. Konsistensi dalam pola interaksi juga sangat penting. Jika pengguna telah belajar bahwa double-click membuka item, pola ini harus konsisten di seluruh sistem. Prinsip interaksi yang baik juga mencakup memberikan kontrol kepada pengguna dan memungkinkan mereka untuk membatalkan tindakan jika diperlukan, yang mengurangi rasa takut melakukan kesalahan.

7.1.4 Desain Responsif dan Aksesibilitas

Desain responsif adalah prinsip penting dalam UI modern yang memastikan bahwa antarmuka dapat beradaptasi dengan baik pada berbagai ukuran layar dan perangkat, dari smartphone hingga monitor desktop. Pendekatan *mobile-first* semakin populer di mana desainer mulai dengan merancang untuk perangkat mobile yang lebih kecil, kemudian memperluas desain untuk layar yang lebih besar. Ini memastikan bahwa antarmuka tetap fungsional dan dapat digunakan pada semua ukuran layar.

Aksesibilitas adalah prinsip desain yang memastikan bahwa antarmuka dapat digunakan oleh semua orang, termasuk mereka yang memiliki keterbatasan fisik atau kognitif. Ini mencakup memastikan kontras warna yang cukup untuk pengguna dengan penglihatan yang lemah, menyediakan alternatif teks untuk gambar bagi pengguna yang menggunakan pembaca layar, memastikan bahwa semua fungsi dapat diakses melalui keyboard untuk pengguna yang tidak dapat menggunakan mouse, dan memastikan bahwa teks dapat dibaca tanpa kesulitan dengan menggunakan ukuran font yang cukup besar. Standar aksesibilitas web seperti *Web Content Accessibility Guidelines* (WCAG) telah dikembangkan untuk membantu desainer menciptakan UI yang inklusif dan dapat diakses oleh semua orang (Mustafa et al., 2025).

7.2 User Experience (UX)

7.2.1 Definisi dan Dimensi User Experience

User Experience (UX) adalah pengalaman keseluruhan yang dirasakan pengguna saat berinteraksi dengan sistem, mencakup tidak hanya antarmuka visual tetapi juga semua aspek dari perjalanan pengguna, mulai dari menemukan sistem, mempelajari cara menggunakannya, hingga mencapai tujuan mereka. UX adalah konsep yang jauh lebih luas daripada UI, yang hanya berfokus pada elemen visual dan interaktif. UX mempertimbangkan emosi, kepuasan, dan keseluruhan pengalaman pengguna.

Dimensi utama dari UX mencakup kegunaan (*usefulness*) yang mengacu pada seberapa baik sistem memenuhi kebutuhan pengguna, kemudahan penggunaan (*ease of use*) yang mengacu pada seberapa mudah pengguna dapat mempelajari dan menggunakan sistem, efisiensi yang mengacu pada seberapa cepat pengguna dapat menyelesaikan tugas mereka, kepuasan yang mengacu pada seberapa puas pengguna dengan pengalaman mereka, dan aksesibilitas yang mengacu pada seberapa mudah sistem dapat diakses oleh semua pengguna termasuk mereka yang memiliki keterbatasan. Desain UX yang baik mempertimbangkan semua dimensi ini secara holistik untuk menciptakan pengalaman yang memuaskan bagi pengguna (Masveta & Manyangara, 2025).

7.2.2 Penelitian Pengguna dan Pemahaman Kebutuhan

Penelitian pengguna adalah langkah kritis dalam proses desain UX yang melibatkan pemahaman siapa pengguna, apa kebutuhan

mereka, bagaimana mereka berperilaku, dan apa yang memotivasi mereka. Metode penelitian pengguna mencakup wawancara mendalam, survei kuesioner, fokus grup, observasi pengguna dalam konteks nyata mereka, dan analisis data penggunaan sistem yang ada. Teknik seperti pembuatan *user persona* yang merepresentasikan segmen pengguna tipikal dan pembuatan *user journey map* yang menggambarkan perjalanan pengguna melalui sistem sangat membantu dalam memahami kebutuhan dan pain points pengguna.

Empati adalah kunci dalam memahami kebutuhan pengguna. Desainer harus menempatkan diri mereka dalam posisi pengguna dan berpikir tentang apa yang akan membuat pengalaman mereka lebih baik. Ini melibatkan mendengarkan feedback pengguna dengan serius, mengamati bagaimana pengguna sebenarnya menggunakan sistem daripada hanya mengandalkan asumsi, dan secara terus-menerus menguji asumsi desain dengan pengguna nyata. Pemahaman yang mendalam tentang pengguna adalah fondasi dari desain UX yang efektif dan membantu memastikan bahwa sistem yang dirancang benar-benar memecahkan masalah pengguna.

7.2.3 Perjalanan Pengguna dan Sentuhan Titik

Perjalanan pengguna adalah narasi tentang apa yang dilakukan pengguna, pikirkan, dan rasakan saat mereka mencoba mencapai tujuan dengan sistem. Perjalanan pengguna dimulai dari sebelum pengguna bahkan menyadari mereka membutuhkan sistem, melewati tahap kesadaran di mana mereka mengetahui sistem ada, tahap pertimbangan di mana mereka mengevaluasi apakah sistem memenuhi kebutuhan mereka, tahap keputusan di mana mereka

memutuskan untuk menggunakan sistem, tahap penggunaan di mana mereka benar-benar menggunakan sistem, dan tahap advokasi di mana mereka berbagi pengalaman mereka dengan orang lain.

Titik sentuh pengguna adalah momen-momen spesifik dalam perjalanan pengguna di mana mereka berinteraksi dengan sistem atau merek. Setiap titik sentuh adalah kesempatan untuk memberikan pengalaman yang positif dan memperkuat persepsi pengguna tentang sistem. Dengan memetakan perjalanan pengguna dan mengidentifikasi semua titik sentuh, desainer dapat memahami di mana pengalaman pengguna mungkin gagal dan di mana ada peluang untuk improvement. Fokus pada setiap titik sentuh memastikan bahwa keseluruhan pengalaman pengguna konsisten dan memuaskan dari awal hingga akhir (Riski et al., 2025).

7.2.4 Emosi Pengguna dan Kepuasan

Emosi pengguna memainkan peran penting dalam menentukan kepuasan mereka dengan sistem. Desain yang baik tidak hanya harus fungsional tetapi juga harus membangkitkan emosi positif pada pengguna. Emosi seperti kepercayaan, kegembiraan, dan rasa pencapaian berkontribusi pada kepuasan pengguna yang tinggi, sementara emosi negatif seperti frustrasi, kebingungan, dan kekecewaan mengurangi kepuasan. Strategi desain yang berfokus pada penciptaan emosi positif mencakup penggunaan micro-interactions yang menyenangkan, animasi yang smooth, umpan balik yang positif dan mendorong, dan desain yang intuitif yang membuat pengguna merasa percaya diri dalam menggunakan sistem.

Kepuasan pengguna dapat diukur melalui berbagai metrik seperti *System Usability Scale* (SUS), *Net Promoter Score* (NPS), dan survey kepuasan pengguna yang lebih komprehensif. Pengukuran kepuasan pengguna secara teratur membantu desainer memahami apakah desain mereka efektif dan di mana masih ada ruang untuk perbaikan. Dengan fokus pada kepuasan emosional pengguna sebagai tujuan desain, sistem yang dirancang akan lebih likely diadopsi dan digunakan dengan antusiasme oleh pengguna (Khairunnisa & Emriadi, 2026).

7.3 Perancangan Prototipe

7.3.1 Konsep Prototipe dan Tingkat Fidelitas

Prototipe adalah representasi awal dari sistem yang dirancang yang digunakan untuk menguji konsep desain, mengeksplorasi ide-ide alternatif, dan mendapatkan umpan balik dari pengguna sebelum investasi besar dalam pengembangan penuh dilakukan. Prototipe dapat berkisar dari sketsa sederhana di atas kertas hingga simulasi interaktif yang hampir identik dengan produk akhir. Fidelitas prototipe mengacu pada tingkat detail dan kecanggihan dari prototipe. Prototipe fidelitas rendah seperti sketsa atau wireframe sederhana sangat berguna untuk fase awal desain ketika banyak ide masih dieksplorasi, sementara prototipe fidelitas tinggi lebih berguna dalam fase lanjut ketika desain telah lebih difinalisasi.

Prototipe fidelitas rendah dapat berupa sketsa tangan, storyboard yang menggambarkan perjalanan pengguna, atau

wireframe digital yang menunjukkan tata letak dan alur navigasi dasar. Prototipe ini cepat dibuat dan mudah diubah, membuatnya ideal untuk eksperimen dan iterasi cepat. Prototipe fidelitas menengah mungkin termasuk mockup visual yang menampilkan warna dan tipografi tetapi belum memiliki interaktivitas penuh. Prototipe fidelitas tinggi adalah representasi interaktif yang sangat mirip dengan produk akhir, memungkinkan pengguna untuk benar-benar berinteraksi dengan sistem dan memberikan umpan balik yang sangat berharga tentang pengalaman pengguna.

7.3.2 Alat dan Teknologi Prototyping

Ada berbagai alat dan teknologi yang tersedia untuk membuat prototipe desain. Alat prototyping tradisional mencakup kertas dan pensil untuk sketsa, papan putih untuk brainstorming dan ideation, dan kartu indeks untuk membuat prototipe aliran tugas. Dalam era digital, ada banyak alat khusus yang dapat digunakan untuk membuat prototipe digital. *Wireframing tools* seperti Balsamiq dan OmniGraffle memungkinkan desainer untuk membuat tata letak struktural dari sistem dengan cepat. *Design tools* seperti Figma, Adobe XD, dan Sketch memungkinkan desainer untuk membuat mockup visual dengan warna, tipografi, dan elemen desain lainnya.

Prototyping tools seperti InVision, Principle, dan Framer memungkinkan desainer untuk menambahkan interaktivitas ke mockup mereka, menciptakan simulasi dari aliran pengguna dan transisi. Beberapa alat seperti Figma juga memungkinkan kolaborasi real-time antara desainer, yang memfasilitasi proses desain yang lebih efisien. Pemilihan alat yang tepat tergantung pada kebutuhan

spesifik proyek, tingkat fidelitas yang diinginkan, dan preferensi desainer. Banyak desainer menggunakan kombinasi berbagai alat untuk mencapai hasil terbaik, mulai dari sketsa manual untuk ideation awal hingga alat digital canggih untuk prototyping final.

7.3.3 Proses Iteratif dan Design Thinking

Proses perancangan prototipe adalah iteratif, yang berarti desain diuji, diumpan balik dikumpulkan, dan kemudian desain diperbaiki berdasarkan umpan balik tersebut. Siklus ini diulangi berkali-kali hingga desain mencapai tingkat kepuasan yang diinginkan. Metodologi *Design Thinking* adalah pendekatan populer yang mengorganisir proses desain iteratif menjadi lima tahap: Empathize, Define, Ideate, Prototype, dan Test. Tahap Empathize melibatkan pemahaman mendalam tentang pengguna dan kebutuhan mereka. Tahap Define melibatkan artikulasi masalah desain yang jelas. Tahap Ideate melibatkan brainstorming ide-ide alternatif untuk memecahkan masalah.

Tahap Prototype melibatkan pembuatan representasi awal dari solusi desain. Tahap Test melibatkan menguji prototipe dengan pengguna nyata dan mengumpulkan umpan balik. Setelah testing, proses berulang kembali ke tahap Ideate atau Prototype untuk membuat perbaikan berdasarkan umpan balik yang diterima. Pendekatan iteratif ini memastikan bahwa desain terus berkembang menuju solusi yang lebih baik dan lebih memenuhi kebutuhan pengguna. Kunci kesuksesan dengan Design Thinking adalah kemauan untuk melakukan banyak iterasi dan tidak takut untuk

membuang ide-ide yang tidak bekerja demi ide-ide yang lebih baik (Riski et al., 2025).

7.3.4 Validasi Konsep dan Pembelajaran Awal

Validasi konsep adalah proses menentukan apakah konsep desain yang dikembangkan benar-benar memecahkan masalah pengguna dengan cara yang efektif dan efisien. Prototipe digunakan sebagai alat untuk validasi ini. Dengan menunjukkan prototipe kepada pengguna dan mengamati bagaimana mereka berinteraksi dengannya, desainer dapat memperoleh pembelajaran berharga tentang apakah konsep desain fundamental berhasil. Pembelajaran awal ini jauh lebih berharga dan ekonomis daripada menemukan masalah besar dalam produk yang sudah sepenuhnya dikembangkan.

Teknik seperti *think aloud testing* di mana pengguna mengomentari pemikiran mereka saat menggunakan prototipe, *task-based testing* di mana pengguna diminta untuk menyelesaikan tugas spesifik dengan prototipe, dan *observation* di mana desainer hanya mengamati bagaimana pengguna menggunakan prototipe tanpa intervensi, semua memberikan wawasan berharga. Data kualitatif dari testing ini dapat dikombinasikan dengan data kuantitatif seperti waktu penyelesaian tugas dan tingkat kesalahan untuk mendapatkan pemahaman komprehensif tentang performa desain. Dengan validasi awal ini, desainer dapat membuat keputusan desain yang lebih informed dan menghindari menghabiskan sumber daya pada ide-ide yang tidak akan berhasil (Khairunnisa & Emriadi, 2026).

7.4 Evaluasi Usability

7.4.1 Definisi dan Metrik Usability

Usability adalah sejauh mana sistem dapat digunakan oleh pengguna untuk mencapai tujuan yang spesifik dengan efektivitas, efisiensi, dan kepuasan dalam konteks penggunaan yang spesifik. Efektivitas mengacu pada akurasi dan kelengkapan dengan mana pengguna dapat mencapai tujuan tertentu. Efisiensi mengacu pada jumlah usaha dan sumber daya yang diperlukan untuk mencapai tujuan tersebut. Kepuasan mengacu pada persepsi pengguna tentang pengalaman mereka menggunakan sistem, termasuk apakah mereka merasa nyaman, aman, dan puas dengan cara mereka dapat menggunakan sistem.

Metrik usability yang umum digunakan mencakup *task success rate* yang mengukur persentase pengguna yang berhasil menyelesaikan tugas yang ditugaskan, *task completion time* yang mengukur berapa lama waktu yang diperlukan pengguna untuk menyelesaikan tugas, *error rate* yang mengukur berapa banyak kesalahan yang dibuat pengguna saat menggunakan sistem, *System Usability Scale* (SUS) yang merupakan kuesioner 10-item yang memberikan skor kepuasan pengguna, dan *Net Promoter Score* (NPS) yang mengukur seberapa mungkin pengguna akan merekomendasikan sistem kepada orang lain. Kombinasi dari metrik-metrik ini memberikan gambaran komprehensif tentang usability sistem (Khairunnisa & Emriadi, 2026).

7.4.2 Metode Evaluasi: Usability Testing dan Heuristic Evaluation

Usability testing adalah metode evaluasi di mana pengguna nyata atau representatif dari pengguna target diminta untuk menggunakan sistem dan menyelesaikan tugas spesifik sementara peneliti mengamati dan mencatat masalah yang mereka alami. Usability testing dapat dilakukan dengan berbagai cara, mulai dari testing one-on-one di mana satu pengguna dievaluasi sekaligus, hingga testing kelompok di mana beberapa pengguna dievaluasi bersama-sama. Testing dapat dilakukan secara sinkron di mana peneliti hadir saat testing berlangsung, atau asinkron di mana pengguna melakukan testing sendiri dalam waktu mereka sendiri dengan sistem tracking otomatis.

Heuristic Evaluation adalah metode alternatif di mana usability experts mengevaluasi sistem berdasarkan himpunan prinsip desain yang dikenal sebagai heuristics. Heuristics yang paling terkenal adalah 10 Usability Heuristics yang dikembangkan oleh Jacob Nielsen, yang mencakup visibility dari system status, match antara sistem dan real world, user control dan freedom, consistency dan standards, error prevention, recognition rather than recall, flexibility dan efficiency of use, aesthetic dan minimalist design, help users recognize diagnose dan recover dari errors, dan help dan documentation. Keuntungan heuristic evaluation adalah relatif cepat dan murah dibanding usability testing, namun kekurangannya adalah tidak mendapatkan feedback dari pengguna nyata yang mungkin memiliki perspektif berbeda.

7.4.3 Analisis Data Evaluasi dan Identifikasi Masalah

Data yang dikumpulkan dari usability testing dan heuristic evaluation perlu dianalisis dengan cermat untuk mengidentifikasi pola masalah dan peluang perbaikan. Analisis kualitatif melibatkan mengkategorikan masalah yang diamati ke dalam tema-tema umum, misalnya masalah navigasi, masalah pemahaman konten, masalah aksesibilitas, dan lain-lain. Severity rating adalah proses mengklasifikasikan masalah berdasarkan seberapa parah dampak masalah terhadap user experience, dengan rating mulai dari minor hingga critical. Prioritas perbaikan dapat ditentukan berdasarkan kombinasi dari severity masalah dan jumlah pengguna yang dipengaruhi oleh masalah tersebut.

Analisis kuantitatif melibatkan pengumpulan dan analisis statistik dari metrik seperti task success rate, completion time, dan error rate. Visualisasi data seperti grafik dan tabel membantu dalam memahami pola dalam data dan membuat temuan lebih mudah dikomunikasikan kepada stakeholder. Data analysis tools seperti spreadsheet, statistical software, dan specialized usability analysis tools dapat membantu dalam proses analisis ini. Dokumentasi temuan yang jelas dan actionable recommendations adalah hasil akhir dari analisis evaluasi yang baik (Khairunnisa & Emriadi, 2026).

7.4.4 Iterasi Desain Berdasarkan Feedback

Setelah masalah diidentifikasi melalui evaluasi, langkah selanjutnya adalah melakukan iterasi desain untuk memperbaiki masalah-masalah tersebut. Proses ini melibatkan brainstorming

solusi alternatif, memilih solusi terbaik berdasarkan pertimbangan cost-benefit, mengimplementasikan solusi dalam prototipe yang diperbarui, dan kemudian menguji ulang untuk memverifikasi bahwa perbaikan yang dilakukan benar-benar memecahkan masalah tanpa menciptakan masalah baru.

Iterasi yang sistematis dan data-driven ini memastikan bahwa desain terus berkembang menuju solusi yang lebih baik. Penting untuk tidak membuat terlalu banyak perubahan sekaligus, karena ini dapat membuat sulit untuk menentukan perubahan mana yang benar-benar meningkatkan usability. Pendekatan yang lebih baik adalah membuat perubahan yang terukur dan menguji efeknya. Dengan mengulangi siklus evaluasi-perbaikan ini berkali-kali, desain akan konvergen menuju solusi yang memiliki usability yang tinggi dan memenuhi kebutuhan pengguna (Masveta & Manyangara, 2025).

7.5 Standar Desain Antarmuka

7.5.1 Pedoman dan Standar Industri

Standar dan pedoman desain telah dikembangkan untuk membantu desainer dalam menciptakan antarmuka yang konsisten, usable, dan accessible. Standar-standar ini dikembangkan berdasarkan penelitian ekstensif tentang bagaimana pengguna berinteraksi dengan sistem dan apa yang membuat pengalaman pengguna efektif. Pedoman desain sistem operasi besar seperti iOS Human Interface Guidelines dari Apple, Material Design dari Google, dan Windows Design Guidelines dari Microsoft

memberikan rekomendasi detail tentang bagaimana merancang untuk platform tertentu.

Standar aksesibilitas seperti *Web Content Accessibility Guidelines* (WCAG) menyediakan pedoman untuk membuat konten web dapat diakses oleh orang-orang dengan berbagai jenis disabilitas. Standar ergonomi seperti ISO 9241 yang berfokus pada *Ergonomics of Human-System Interaction* memberikan rekomendasi tentang bagaimana merancang sistem yang nyaman dan aman untuk digunakan. Standar usability seperti ISO 9126 mendefinisikan karakteristik kualitas perangkat lunak termasuk usability. Dengan mematuhi standar-standar ini, desainer dapat memastikan bahwa sistem mereka memenuhi ekspektasi industri dan memberikan pengalaman pengguna yang konsisten dengan produk lain yang sudah familiar bagi pengguna.

7.5.2 Desain untuk Aksesibilitas dan Inklusivitas

Aksesibilitas adalah prinsip fundamental dalam desain antarmuka modern yang memastikan bahwa sistem dapat digunakan oleh semua orang termasuk mereka yang memiliki keterbatasan fisik, sensori, atau kognitif. Aksesibilitas bukan hanya tentang mematuhi standar regulasi tetapi tentang menciptakan pengalaman yang inklusif dan memuaskan untuk semua pengguna. Desain yang accessible mencakup memastikan kontras warna yang cukup untuk membaca teks, menyediakan alternatif teks untuk gambar, memastikan navigasi keyboard yang lengkap, menggunakan ukuran font yang readable, dan menghindari flash atau animasi yang dapat memicu epilepsi.

Prinsip desain universal atau *universal design* berpendapat bahwa sistem harus dirancang dari awal untuk dapat digunakan oleh semua orang dengan sedikit atau tanpa modifikasi. Ini berbeda dengan accessibility yang sering ditambahkan setelah sistem sudah dirancang. Dengan mengadopsi universal design principles dari awal, desainer dapat menciptakan sistem yang lebih inklusif dan tidak perlu dimodifikasi kemudian. Desain inklusif juga mempertimbangkan konteks penggunaan yang beragam, seperti penggunaan sistem di bawah cahaya matahari terang, penggunaan dengan tangan basah, penggunaan dalam lingkungan yang bising, dan penggunaan oleh pengguna dalam kondisi stress atau terburu-buru (Mustafa et al., 2025).

7.5.3 Konsistensi dan Pola Desain

Konsistensi dalam desain antarmuka adalah elemen kritis untuk usability yang baik. Konsistensi visual berarti menggunakan elemen desain yang sama, seperti warna, font, ikon, dan gaya tombol di seluruh sistem. Konsistensi interaksi berarti menggunakan pola interaksi yang sama untuk tindakan serupa di seluruh sistem, misalnya jika double-click membuka item di satu tempat, pola ini harus konsisten di seluruh sistem. Konsistensi terminologi berarti menggunakan istilah yang sama untuk konsep yang sama di seluruh antarmuka dan dokumentasi.

Design patterns adalah solusi yang terbukti efektif untuk masalah desain umum. Pattern-pattern ini dokumentasikan dan dibagikan sehingga desainer dapat menggunakannya kembali dalam proyek mereka sendiri, menghemat waktu dan memastikan

konsistensi. Contoh-contoh design patterns termasuk *navigation patterns*, *form patterns*, *search patterns*, dan *error handling patterns*. Dengan menggunakan established design patterns, desainer menciptakan antarmuka yang familiar bagi pengguna, yang mengurangi kurva pembelajaran dan meningkatkan usability. Design systems adalah dokumentasi lengkap dari design patterns, components, dan guidelines yang digunakan dalam organisasi, memastikan konsistensi di semua produk yang dikembangkan organisasi (Masveta & Manyangara, 2025).

7.5.4 Evolusi Tren Desain dan Best Practices

Tren desain terus berkembang seiring dengan perubahan teknologi, preferensi pengguna, dan kesadaran akan pentingnya user experience. Tren desain terbaru termasuk minimalism yang menghilangkan elemen visual yang tidak perlu untuk fokus pada yang esensial, dark mode yang mengurangi kelelahan mata dalam cahaya rendah, microinteractions yang memberikan umpan balik halus untuk tindakan pengguna, dan neumorphism yang menggabungkan elemen skeuomorphic dan flat design. Kesadaran akan pentingnya accessibility telah menjadi trend penting di mana aksesibilitas bukan lagi afterthought tetapi pertimbangan utama dalam desain.

Personalisasi adalah trend lain yang berkembang dengan cepat di mana antarmuka beradaptasi dengan preferensi individu pengguna berdasarkan data penggunaan mereka. Dengan pembelajaran mesin dan AI, sistem dapat belajar preferensi pengguna dan mempersonalisasi pengalaman mereka secara real-time. Voice

interfaces dan conversational design juga semakin populer dengan munculnya smart speakers dan voice assistants. Best practices dalam desain antarmuka modern mencakup fokus yang kuat pada user research, iterasi design yang cepat berdasarkan feedback pengguna, prioritas pada accessibility dan inklusivitas, dan adopsi dari established design patterns dan standards. Dengan tetap update dengan tren terbaru sambil tetap fokus pada fundamental principles dari good design, desainer dapat menciptakan antarmuka yang tidak hanya modern tetapi juga efektif dan memuaskan bagi pengguna (Riski et al., 2025).

BAB 8: Pelaksanaan Sistem Informasi

Hj. A.Irmayani Pawelloi, ST., M.T.

8.1 Konsep Sistem Implementasi

8.1.1 Definisi dan Implementasi Ruang Lingkup

Implementasi sistem informasi merupakan tahap krusial dalam siklus pengembangan sistem di mana desain sistem yang telah diubah menjadi sistem operasional yang siap digunakan oleh pengguna akhir. Implementasi tidak sekedar aktivitas teknis seperti instalasi perangkat lunak dan konfigurasi database, tetapi merupakan proses transformasi organisasi yang melibatkan aspek manusia, proses bisnis, dan teknologi secara terintegrasi (Mahomedov, 2025). Ruang lingkup implementasi mencakup sistem pemrograman, modul integrasi, pengujian menyeluruh, persiapan data, migrasi dari sistem lama, pelatihan pengguna, dan pemantauan pasca-implementasi untuk memastikan sistem berfungsi optimal dalam mendukung operasi bisnis organisasi.

Kesuksesan sistem implementasi diukur tidak hanya dari perspektif teknis seperti uptime dan kinerja, tetapi juga dari perspektif bisnis seperti tingkat adopsi pengguna, laba atas investasi, dan sistem kontribusi terhadap pencapaian tujuan strategis organisasi. Fase implementasi memerlukan koordinasi yang intensif

antara berbagai pemangku kepentingan termasuk tim pengembang, pengguna bisnis, proyek manajemen, administrator sistem, dan departemen operasi untuk memastikan setiap aspek dipertimbangkan dan dijalankan dengan sempurna.

8.1.2 Tahapan dan Metodologi Implementasi

Sistem implementasi proses mengikuti tahapan terstruktur yang memastikan setiap komponen sistem dipersiapkan dengan matang sebelum diluncurkan ke lingkungan produksi (Panguraj, 2021). Tahap persiapan implementasi meliputi finalisasi rencana implementasi yang detail, alokasi sumber daya yang tepat baik dalam hal personel maupun infrastruktur teknologi, persiapan lingkungan produksi yang mencakup server, database, networking, dan sistem keamanan, serta penyiapan dokumentasi teknis dan user manual yang komprehensif.

Tahap selanjutnya adalah pemrograman dan pengembangan komponen sistem yang masih dalam tahap finalisasi, diikuti dengan pengujian sistem secara menyeluruh untuk memverifikasi bahwa setiap komponen berfungsi sesuai spesifikasi yang telah ditetapkan dan diintegrasikan dengan baik dengan komponen lainnya. Tahap persiapan cutover atau transisi yang melibatkan migrasi data dari sistem lama ke sistem baru, persiapan strategi cutover yang disesuaikan dengan karakteristik organisasi dan risiko yang dapat ditoleransi, serta pelatihan intensif kepada pengguna tentang cara mengoperasikan sistem baru. Tahap pelaksanaan cutover merupakan momen kritis di mana sistem baru mulai dioperasikan secara riil, sementara tahap tinjauan pasca implementasi dilakukan untuk

mencapai kesuksesan implementasi dan mengidentifikasi pembelajaran yang dapat diterapkan pada implementasi sistem berikutnya.

8.1.3 Implementasi Strategi Cutover dan Risiko

Peralihan strategi yang dipilih sangat mempengaruhi tingkat risiko dan dampak implementasi terhadap operasi bisnis yang sedang berjalan (Mahomedov, 2025). Strategi *big bang* menggantikan sistem lama sepenuhnya dengan sistem baru pada waktu tertentu yang telah ditentukan, memberikan keuntungan berupa kecepatan transisi dan penghindaran kompleksitas menjalankan dua sistem secara bersamaan, namun membawa risiko tinggi karena jika terjadi masalah tidak ada fallback yang tersedia. Strategi *peluncuran bertahap* mengimplementasikan sistem secara bertahap per modul, departemen, atau lokasi geografis, sehingga memungkinkan identifikasi dan penyelesaian masalah lebih awal sebelum sistem sepenuhnya berganti, tetapi memerlukan periode transisi yang lebih panjang dan kompleksitas pengelolaan dalam dua sistem yang berbeda beroperasi secara bersamaan.

Strategi *parallel running* menjalankan sistem lama dan sistem baru secara bersamaan untuk periode tertentu, memberikan jaminan tertinggi terhadap kontinuitas operasi karena tim dapat dengan cepat kembali ke sistem lama jika masalah serius terjadi, namun memerlukan investasi sumber daya yang sangat besar karena kedua sistem harus dijalankan dan dirawat secara bersamaan. Identifikasi dan mitigasi risiko harus dilakukan secara proaktif melalui penilaian risiko menyeluruh, perencanaan kontingensi untuk berbagai

skenario kegagalan, dan pembentukan mekanisme eskalasi yang jelas untuk menangani masalah kritis selama implementasi.

8.1.4 Faktor Kesuksesan dan Manajemen Perubahan

Sistem implementasi kesuksesan bergantung pada berbagai faktor yang saling terkait dari sisi teknis, organisasi, dan manusia (Panguraj, 2021). Dukungan yang kuat dari manajemen puncak sangat penting untuk memberikan prioritas pada implementasi proyek, mengalokasikan sumber daya yang diperlukan, dan memfasilitasi penyelesaian masalah yang memerlukan keputusan tingkat strategi. Keterlibatan aktif pengguna sejak tahap perancangan sistem memastikan bahwa sistem yang dibangun benar-benar memenuhi kebutuhan bisnis mereka dan mengurangi resistensi terhadap perubahan.

Manajemen perubahan yang efektif melibatkan komunikasi yang transparan dan berkelanjutan tentang alasan perubahan, manfaat yang akan diperoleh, dampak terhadap pekerjaan sehari-hari pengguna, dan dukungan yang akan diberikan selama transisi. Program pelatihan yang komprehensif dan berkelanjutan tidak hanya mempersiapkan pengguna secara teknis untuk mengoperasikan sistem baru, tetapi juga membangun kepercayaan diri mereka dalam menggunakan sistem dan mengurangi kecemasan terhadap teknologi baru. Tim implementasi yang memiliki kombinasi optimal antara keahlian teknis dan keterampilan soft skill seperti komunikasi dan pemecahan masalah, serta pengalaman dalam informasi sistem implementasi sebelumnya, akan meningkatkan kemungkinan keberhasilan implementasi secara signifikan.

8.2 Pemrograman Sistem

8.2.1 Standar dan Prinsip Pemrograman

Pemrograman sistem adalah tahap di mana desain sistem konversi menjadi kode sumber yang dapat dieksekusi dan berfungsi sesuai dengan spesifikasi teknis yang telah didefinisikan (Mahomedov, 2025). Penerapan standar pemrograman yang ketat dan konsisten memastikan bahwa kode yang dihasilkan tidak hanya berfungsi dengan benar tetapi juga mudah dipahami, mudah dirawat, dan mudah dimodifikasi di masa depan oleh pengembang lain. Standar pemrograman mencakup konvensi penamaan yang konsisten untuk variabel, fungsi, kelas, dan modul yang mencerminkan tujuan dari elemen tersebut, dokumentasi kode yang lengkap melalui komentar dan javadoc yang menjelaskan logika kompleks, penanganan error dan pengecualian yang tepat untuk menangani situasi abnormal, dan penggunaan pola desain yang telah terbukti efektif dalam menyelesaikan masalah pemrograman tertentu.

Prinsip-prinsip solid engineering practice seperti *Single Responsibility Prinsip* yang menyatakan setiap fungsi atau kelas harus memiliki satu tanggung jawab tunggal, *DRY* (Don't Repeat Yourself) yang menghindari duplikasi kode yang tidak perlu, dan *KISS* (Keep It Simple Stupid) yang menekankan kompresi dalam desain dan implementasi kode, harus diterapkan secara konsisten dalam seluruh proses pengembangan menghasilkan untuk

codebase yang berkualitas tinggi, mudah dipahami, dan mudah di-maintain.

8.2.2 Pemilihan Bahasa dan Platform Pengembangan

Pemilihan bahasa pemrograman dan pengembangan platform adalah keputusan strategi yang mempengaruhi produktivitas waktu pengembang, kinerja sistem, dan kemampuan untuk melakukan pemeliharaan dan evolusi sistem di masa depan (Panguraj, 2021). Pertimbangan dalam pemilihan bahasa mencakup kesesuaian bahasa dengan karakteristik aplikasi yang akan dibangun, misalnya Java atau C# untuk aplikasi enterprise yang kompleks, Python untuk aplikasi data science dan machine learning, atau JavaScript untuk pengembangan web frontend, ketersediaan perpustakaan dan framework yang matang yang dapat mempercepat waktu pengembangan, keahlian dan pengalaman tim pengembang dengan bahasa tersebut, serta pertimbangan non-fungsional seperti kinerja, skalabilitas, dan keamanan.

Pemilihan lingkungan pengembangan terintegrasi (IDE) yang tepat meningkatkan produktivitas pengembang secara signifikan melalui fitur seperti penyelesaian kode, penyorotan sintaksis, alat debugging bawaan, dan integrasi dengan sistem kontrol versi. Framework dan perpustakaan yang matang seperti Spring untuk Java, Django atau FastAPI untuk Python, atau React untuk JavaScript frontend dapat mempercepat waktu pengembangan secara dramatis dengan menyediakan komponen-komponen siap pakai yang telah teruji dan mengikuti praktik terbaik yang telah disepakati dalam komunitas pengembang. Namun penggunaan

teknologi baru yang belum matang harus dipertimbangkan dengan cermat karena risiko menghadapi bug yang tidak terduga, dokumentasi yang tidak lengkap, atau dukungan komunitas yang belum terbentuk.

8.2.3 Kontrol Versi dan Kolaborasi Tim

Sistem kontrol versi seperti Git adalah fondasi penting untuk kolaborasi efektif antara anggota tim pengembang yang bekerja pada basis kode yang sama (Panguraj, 2021). Kontrol versi memungkinkan waktu untuk melacak setiap perubahan yang dilakukan terhadap kode, siapa yang melakukan perubahan, kapan perubahan dilakukan, dan apa alasan di balik perubahan tersebut melalui pesan komit yang informatif. Strategi percabangan yang jelas seperti Git Flow atau Trunk-Based Development memastikan bahwa pengembang dapat bekerja pada fitur atau perbaikan bug yang berbeda secara paralel tanpa konflik, dengan integrasi yang dikontrol melalui pull request atau merge request yang memerlukan persetujuan dari rekan tim sebelum kode digabungkan ke dalam cabang utama.

Code review adalah praktik penting di mana kode yang ditulis oleh satu pengembang ditinjau secara menyeluruh oleh rekan tim sebelum diintegrasikan ke dalam codebase utama. Melalui tinjauan kode, tim dapat mengidentifikasi bug, masalah desain, dan peluang untuk perbaikan lebih awal sebelum kode masuk ke produksi, memastikan kepatuhan terhadap standar pengkodean dan praktik terbaik yang telah ditetapkan, dan memfasilitasi transfer pengetahuan antar anggota tim sehingga pengetahuan tidak

tersentralisasi pada satu orang saja. Alat seperti GitHub, GitLab, dan Bitbucket menyediakan fitur pull request/merge request terintegrasi dengan kemampuan peninjauan kode bawaan yang memfasilitasi proses ini secara terstruktur dan transparan.

8.2.4 Pengujian dan Jaminan Kualitas dalam Pengembangan

Pengembangan kode yang berkualitas tinggi memerlukan pengujian yang komprehensif dan dilakukan sejak tahap awal pengembangan, bukan hanya dilakukan menjelang akhir siklus hidup proyek (Mahomedov, 2025). Pengujian unit menguji fungsi individu atau metode dalam isolasi untuk memastikan bahwa setiap unit berfungsi sesuai dengan spesifikasinya melalui kasus uji otomatis yang ditulis oleh pengembang itu sendiri atau oleh tim QA khusus. Test-Driven Development (TDD) adalah pendekatan di mana test ditulis terlebih dahulu sebelum mengimplementasikan kode, memastikan bahwa setiap fitur yang dikembangkan benar-benar teruji dan mencegah pengembang dari menulis kode yang tidak dapat diuji.

Continuous Integration (CI) mengotomatisasi proses pembuatan aplikasi, menjalankan pengujian otomatis, dan melakukan analisis kode statis setiap kali ada perubahan kode yang di-commit ke repositori. Ini memberikan umpan balik yang cepat kepada pengembang tentang kualitas kode mereka dan memastikan bahwa kode yang di-commit tidak memecahkan fungsionalitas yang sudah ada atau memperkenalkan bug. Metrik cakupan yang dihasilkan dari pengujian otomatis memberikan visibilitas tentang persentase kode yang telah ditemukan oleh pengujian, meskipun

cakupan yang tinggi tidak selalu menjamin bahwa kualitas kode juga tinggi, tetapi cakupan yang rendah menunjukkan bahwa ada risiko dari jalur kode yang belum diuji.

8.3 Integrasi Modul Sistem

8.3.1 Perencanaan dan Desain Integrasi

Integrasi modul adalah proses menggabungkan berbagai komponen atau modul yang dikembangkan secara terpisah oleh tim yang berbeda menjadi satu sistem yang terpadu dan berfungsi secara harmonis sesuai dengan kebutuhan sistem secara keseluruhan. Perencanaan komunikasi harus dilakukan sejak tahap desain sistem untuk mendeteksi dan mendokumentasikan antarmuka yang jelas antara modul-modul, protokol komunikasi yang akan digunakan untuk pertukaran data antar modul, format data yang distandarisasi untuk memastikan kompatibilitas, dan mekanisme penanganan kesalahan untuk menangani situasi ketika komunikasi antar modul mengalami kegagalan. Desain antarmuka yang komprehensif memfasilitasi pengembangan modul secara paralel karena tim dapat bekerja berdasarkan kontrak antarmuka yang telah disepakati tanpa perlu menunggu modul lain selesai dikembangkan.

Strategi integrasi dapat dilakukan secara inkremental (bottom-up atau top-down) di mana modul terintegrasi satu per satu secara bertahap, atau secara big-bang di mana semua modul terintegrasi sekaligus pada saat yang sama. Integrasi inkremental umumnya lebih disukai karena memungkinkan identifikasi masalah integrasi

lebih awal dan memfasilitasi debugging yang lebih efektif karena masalah dapat dengan lebih mudah diisolasi ke modul atau antarmuka tertentu. Mock Objects dan Test Stubs dapat digunakan untuk menggantikan modul yang belum siap atau layanan eksternal sehingga pengujian integrasi dapat dilakukan sebelum semua modul siap, mempercepat siklus pengujian dan memungkinkan pengujian dan pengembangan paralel.

8.3.2 Pengujian Integrasi dan Penyelesaian Masalah

Pengujian integrasi memastikan bahwa modul-modul yang telah berinteraksi bekerja bersama-sama dengan baik dan sesuai dengan spesifikasi sistem secara keseluruhan, bukan hanya bekerja dengan benar secara individu (Panguraj, 2021). Pengujian integrasi mencakup pengujian terhadap berbagai aspek seperti antarmuka antar modul untuk memastikan bahwa data disimpan dengan benar dalam format yang diharapkan, aliran data antar modul untuk memastikan bahwa data mengalir melalui sistem dengan urutan dan logika yang benar, interaksi dengan sistem eksternal seperti database atau layanan pihak ketiga untuk memastikan integrasi berfungsi dengan benar, dan perilaku sistem end-to-end dalam berbagai skenario penggunaan termasuk skenario normal maupun kasus tepi.

Masalah atau cacat yang ditemukan selama pengujian integrasi harus dicatat secara terstruktur dalam sistem pelacakan cacat dengan informasi lengkap seperti deskripsi masalah yang jelas dan spesifik, langkah-langkah untuk mereproduksi masalah sehingga dapat diproduksi secara konsisten, tingkat keparahan dari masalah yang mengindikasikan dampaknya terhadap fungsionalitas sistem (kritis,

besar, kecil), dan prioritas yang menunjukkan urgensi penyelesaian. Proses triage dilakukan untuk menentukan tim mana yang bertanggung jawab atas resolusi masalah tersebut berdasarkan modul mana yang bermasalah, sementara root cause analysis dilakukan untuk memahami akar penyebab masalah agar dapat dicegah di masa depan. Komunikasi yang efektif antar tim menjadi sangat penting ketika masalah melibatkan banyak modul atau tim untuk memastikan bahwa penyelesaian dapat dilakukan dengan cepat dan efisien.

8.3.3 API dan Middleware dalam Integrasi

Application Programming Interface (API) mendefinisikan kontrak komunikasi yang jelas antara komponen-komponen sistem, menentukan endpoint yang tersedia, parameter yang diterima, format respon yang dikembalikan, dan kode kesalahan yang mungkin terjadi. Dokumentasi API yang komprehensif sangat penting untuk memudahkan pengembang lain mengintegrasikan dengan API tersebut. Standar seperti OpenAPI/Swagger dan GraphQL memfasilitasi dokumentasi API yang terstruktur dan dapat dibaca di mesin, memungkinkan pembuatan otomatis dari kode klien dan portal pengembang yang interaktif untuk menguji API.

Middleware seperti perantara pesan, gateway API, atau bus layanan berfungsi sebagai perantara yang memfasilitasi komunikasi antar komponen sistem yang mungkin heterogen dalam teknologi yang digunakan (Mahomedov, 2025). Penggunaan middleware mengurangi kopling antar aplikasi sehingga perubahan pada salah satu aplikasi tidak perlu mempengaruhi aplikasi lain, memfasilitasi

pola komunikasi yang kompleks seperti publikasi-langganan atau permintaan-balas, dan menyediakan kemampuan tambahan seperti perutean, transformasi, dan pengayaan data. Teknologi seperti Apache Kafka, RabbitMQ, atau AWS SQS menyediakan infrastruktur perpesanan andal yang dapat menangani pesan dalam jumlah besar dengan jaminan pengiriman dan pemesanan.

8.3.4 Integrasi Pemantauan dan Pemecahan Masalah

Setelah integrasi selesai dan sistem sudah aktif, pemantauan berkelanjutan diperlukan untuk memastikan bahwa modul-modul terus berfungsi dengan baik dan dapat saling berkomunikasi tanpa masalah. Pemantauan mencakup metrik pelacakan seperti waktu respons yang mengukur berapa lama waktu yang diperlukan untuk proses permintaan end-to-end, throughput yang mengukur jumlah permintaan yang dapat diproses per satuan waktu, tingkat kesalahan yang mengukur persentase permintaan yang mengalami kegagalan, dan pemanfaatan sumber daya untuk memastikan bahwa sistem memiliki sumber daya yang cukup untuk menangani beban yang ada. Aturan peringatan harus diterapkan untuk memberitahu operasi tim secara proaktif jika terjadi anomali atau degradasi kinerja sebelum berdampak signifikan terhadap pengguna akhir.

Pemecahan masalah integrasi yang mengalami masalah memerlukan alat logging dan tracing yang canggih untuk memahami aliran data melalui berbagai modul dan mengidentifikasi di mana masalah yang terjadi (Panguraj, 2021). Platform logging terpusat seperti ELK Stack, Splunk, atau Datadog mengumpulkan log dari berbagai modul dan sistem ke dalam satu tempat sehingga tim dapat

menganalisisnya secara terpadu. Alat penelusuran terdistribusi seperti Jaeger atau Zipkin membantu melacak permintaan individu ketika permintaan tersebut melalui berbagai modul, memberikan visibilitas penuh tentang jalur yang telah dilalui dan latensi di setiap langkah, memudahkan mengidentifikasi titik kemacetan dan kegagalan dalam sistem.

8.4 Pengujian Sistem

8.4.1 Strategi dan Tingkat Pengujian

Pengujian sistem adalah aktivitas sistematis yang bertujuan untuk memverifikasi bahwa sistem memenuhi semua persyaratan yang telah ditetapkan dan tidak terdapat cacat kritis yang dapat mengganggu sistem operasi (Rajapakse et al., 2021). Strategi pengujian yang komprehensif harus mencakup berbagai tingkat pengujian yang masing-masing memiliki fokus dan ruang lingkup yang berbeda tetapi secara bersama-sama berkontribusi pada memastikan kualitas akhir sistem. Pengujian unit menguji fungsi individu dalam isolasi untuk memastikan bahwa setiap unit berfungsi sesuai dengan spesifikasinya, biasanya dilakukan oleh developer sendiri menggunakan framework seperti JUnit, NUnit, atau pytest.

Pengujian integrasi menguji bagaimana beberapa unit atau modul bekerja bersama-sama untuk memastikan bahwa antarmuka dan aliran data antar modul berfungsi dengan benar. Pengujian sistem menguji sistem secara keseluruhan terhadap kelengkapan

spesifikasi kebutuhan untuk memastikan bahwa sistem memenuhi persyaratan end-to-end baik persyaratan fungsional maupun non-fungsional. User Acceptance Testing (UAT) adalah pengujian oleh pengguna akhir atau sponsor bisnis untuk memastikan bahwa sistem memenuhi kebutuhan bisnis mereka dan dapat diterima untuk diterapkan ke lingkungan produksi. Setiap tingkat pengujian memerlukan kasus uji yang dirancang dengan cermat berdasarkan kebutuhan dan skenario penggunaan yang diharapkan.

8.4.2 Pengujian Otomatis dan Alat Otomatisasi Pengujian

Otomatisasi pengujian menggunakan alat dan skrip untuk menjalankan kasus uji secara otomatis, meningkatkan efisiensi dan konsistensi pengujian khususnya untuk pengujian regresi yang memerlukan eksekusi kasus uji yang sama berulang kali. Alat-alat seperti Selenium untuk otomatisasi aplikasi web, JUnit atau TestNG untuk pengujian unit Java, Cypress atau Playwright untuk pengujian end-to-end, dan LoadRunner untuk pengujian kinerja memungkinkan tim untuk membuat rangkaian pengujian komprehensif yang dapat dijalankan secara otomatis. Pengujian otomatis memberikan umpan balik yang cepat tentang kualitas kode, memfasilitasi integrasi berkelanjutan di mana pengujian dijalankan otomatis setiap kali ada perubahan kode, dan mengurangi beban pengujian manual sehingga tim QA dapat fokus pada pengujian yang lebih kompleks dan pengujian eksplorasi.

Praktik terbaik dalam otomatisasi pengujian mencakup memilih alat yang kompatibel dengan tumpukan teknologi yang digunakan, menulis kasus pengujian yang kuat dan dapat dipelihara

dengan menghindari nilai hardcoding dan menggunakan lapisan abstraksi seperti model objek halaman untuk pengujian web, menjaga rangkaian pengujian tetap relevan dengan memperbarui pengujian secara berkala ketika persyaratan berubah, dan mengelola data pengujian dengan baik untuk memastikan pengujian dapat berjalan secara konsisten dan dapat direproduksi. Konsep piramida pengujian merekomendasikan bahwa sebagian besar pengujian harus berupa pengujian unit yang cepat dijalankan, diikuti oleh proporsi yang lebih kecil dari pengujian integrasi, dan paling kecil adalah pengujian end-to-end yang lambat dijalankan.

8.4.3 Pengujian Kinerja dan Pengujian Keamanan

Pengujian kinerja memastikan bahwa sistem dapat memproses volume transaksi yang diharapkan dalam waktu respons yang dapat diterima dan tidak mengalami penurunan kinerja yang signifikan di bawah beban normal (Rajapakse et al., 2021). Pengujian beban di sekitar beban pengguna yang realistis terhadap sistem untuk mengidentifikasi kemacetan, memastikan bahwa sistem dapat berskala untuk menangani beban puncak yang diharapkan, dan mengidentifikasi titik di mana sistem mulai degradasi atau kegagalan. Stress testing mendorong sistem melampaui kapasitas normalnya untuk menentukan titik puncaknya, mengobservasi bagaimana sistem berperilaku dalam kondisi ekstrem, dan memastikan mekanisme degradasi atau pemulihan yang baik dan kuat.

Pengujian keamanan memastikan bahwa sistem tahan terhadap berbagai vektor serangan dan tidak rentan terhadap eksploitasi

keamanan yang diketahui (Mahomedov, 2025). Pengujian keamanan mencakup pengujian penetrasi di mana peretas etis mencoba melanggar keamanan sistem, pemindaian kerentanan untuk memastikan kerentanan yang diketahui dalam aplikasi dan infrastruktur, tinjauan kode untuk mengidentifikasi kelemahan keamanan dalam kode sumber, dan pengujian kepatuhan untuk memastikan sistem memenuhi persyaratan peraturan seperti GDPR, HIPAA, atau standar khusus industri. Hasil dari pengujian keamanan harus diprioritaskan dan ditangani sebelum penerapan produksi untuk meminimalkan risiko pelanggaran keamanan.

8.4.4 Manajemen Cacat dan Kriteria Keluar

Manajemen cacat adalah proses terstruktur untuk memindai, menganalisis, dan menyelesaikan bug atau masalah yang ditemukan selama pengujian (Panguraj, 2021). Setiap cacat harus dicatat dalam sistem pelacakan cacat dengan informasi lengkap termasuk deskripsi yang jelas tentang masalah, langkah-langkah untuk mereproduksi yang memungkinkan QA atau pengembang lain Menghasilkan masalah secara konsisten, tingkat keparahan yang menunjukkan dampak terhadap fungsionalitas (kritis yang menyebabkan sistem crash atau hilangnya fungsionalitas sepenuhnya, mayor yang menyebabkan hilangnya fungsionalitas secara signifikan, minor yang menyebabkan ketidaknyamanan kecil), dan prioritas yang mengindikasikan perbaikan urgensi (tinggi yang harus diselesaikan sebelum rilis produksi, medium yang dapat diselesaikan dalam rilis mendatang, rendah yang dapat).

Kriteria keluar untuk setiap fase pengujian harus didefinisikan dengan jelas sejak awal sehingga tim memiliki kejelasan tentang kapan fase pengujian dapat dinyatakan selesai dan dapat beralih ke fase berikutnya (Rajapakse et al., 2021). Kriteria keluar yang biasanya mencakup semua kasus pengujian yang direncanakan harus dijalankan dengan tingkat keberhasilan minimum tertentu misalnya 95%, semua cacat kritis dan besar harus telah diselesaikan, cakupan pengujian minimum harus tercapai misalnya 80%, dan tidak ada masalah penghenti pertunjukan yang tersisa yang dapat mengganggu penerapan produksi. Metrik dan laporan kualitas harus dihasilkan secara teratur untuk memberikan visibilitas kepada pemangku kepentingan tentang kemajuan pengujian dan status kualitas dari sistem yang sedang diuji.

8.5 Sistem Penyebaran

8.5.1 Persiapan dan Perencanaan Deployment

Deployment adalah fase kritis di mana sistem dipindahkan dari staging environment ke lingkungan produksi di mana sistem akan digunakan oleh pengguna akhir sebenarnya dalam volume dan intensitas sesungguhnya. Persiapan penerapan harus dimulai jauh hari sebelum tanggal penerapan sebenarnya dan mencakup berbagai aktivitas seperti finalisasi rencana penerapan yang detail dengan prosedur langkah demi langkah, rencana darurat untuk berbagai skenario kegagalan, dan prosedur eskalasi yang jelas, persiapan lingkungan produksi termasuk penyediaan server, konfigurasi aturan

jaringan dan firewall, pengaturan database dan infrastruktur cadangan, dan memastikan mekanisme pemantauan dan peringatan yang tepat sudah siap.

Pemeriksaan pra-penerapan harus dilakukan untuk memverifikasi bahwa semua komponen dari sistem siap untuk penerapan, ketergantungan pada sistem eksternal atau layanan pihak ketiga telah divalidasi, skrip migrasi data telah diuji dan siap dijalankan, prosedur rollback telah diuji dan didokumentasikan, dan komunikasi dengan pemangku kepentingan tentang jendela penerapan dan potensi dampak telah dilakukan. Prosedur pencadangan dan pemulihan bencana harus dilakukan untuk memastikan bahwa organisasi memiliki mekanisme untuk memulihkan dari kegagalan penerapan atau kegagalan bencana selama operasi produksi. Alokasi sumber daya dan penjadwalan harus direncanakan dengan baik untuk memastikan tersedianya personel yang cukup dari tim pengembangan, operasi, dan bisnis untuk mendukung penerapan dan periode pasca-penempatan segera.

8.5.2 Penerapan Strategi dan Manajemen Perubahan

Strategi deployment yang dipilih harus disesuaikan dengan toleransi risiko organisasi, kepentingan sistem yang akan dideploy, dan toleransi terhadap downtime. Penerapan Big Bang menggantikan sistem lama sepenuhnya dengan sistem baru dalam satu waktu pada tanggal tertentu, memberikan keuntungan dari kecepatan transisi dan menghindari kompleksitas dari menjalankan dua sistem secara paralel, namun memiliki risiko tinggi karena jika masalah ditemukan selama penerapan, tidak ada fallback langsung.

Penerapan bertahap mengimplementasikan sistem secara bertahap per modul atau per unit bisnis, memungkinkan tim untuk mengidentifikasi dan menyelesaikan masalah pada subset pengguna sebelum peluncuran penuh, tetapi memerlukan periode transisi yang lebih lama dan manajemen perubahan yang kompleks.

Penerapan biru-hijau mempertahankan dua lingkungan produksi yang identik dengan satu lingkungan (biru) melayani lalu lintas pengguna aktual sementara lingkungan lain (hijau) menerima versi terbaru dari sistem. Setelah sistem diperbarui diuji di lingkungan hijau, lalu lintas dialihkan ke lingkungan hijau, memungkinkan rollback instan ke lingkungan biru jika ditemukan masalah. Penerapan Canary meluncurkan sistem yang diperbarui ke persentase kecil dari pengguna terlebih dahulu untuk memantau perilaku dan mengumpulkan umpan balik, secara bertahap meningkatkan persentase pengguna yang menerima sistem yang diperbarui jika metrik kinerja dan tingkat kesalahan dapat diterima. Tanda fitur memungkinkan tim untuk menerapkan fitur baru ke produksi tanpa segera membuat fitur terlihat oleh semua pengguna, memberikan fleksibilitas untuk mengaktifkan/menonaktifkan fitur secara dinamis berdasarkan pemantauan data dan keputusan bisnis.

8.5.3 Penerapan Berkelanjutan dan Praktik DevOps

Continuous Deployment (CD) adalah latihan di mana setiap perubahan yang lulus tes otomatis secara otomatis di-deploy ke lingkungan produksi tanpa intervensi manual (Panguraj, 2021). Ini merupakan tingkat kematangan tertinggi dari praktik pengiriman berkelanjutan, memberikan kemampuan untuk merilis fitur kepada

pengguna akhir dengan frekuensi yang sangat tinggi, berpotensi beberapa kali per hari. Penerapan Berkelanjutan memerlukan infrastruktur yang sangat andal dan otomatis, pengujian otomatis komprehensif yang memberikan keyakinan bahwa perubahan tidak menimbulkan regresi, dan pemantauan dan peringatan yang kuat untuk mendeteksi dan merespons masalah di produksi dengan cepat.

DevOps adalah gerakan budaya dan serangkaian praktik yang mengintegrasikan tim pengembangan dan operasi, mengotomatisasi proses sebanyak mungkin dalam siklus hidup pengiriman perangkat lunak, dan meningkatkan visibilitas dan pengukuran di setiap tahap dari pengembangan hingga operasi produksi (Panguraj, 2021). Pipeline CI/CD adalah alur kerja otomatis yang mencakup setiap tahap dari penerapan kode, pembuatan artefak, menjalankan pengujian otomatis, dan penerapan ke produksi. Infrastructure as Code (IaC) adalah praktik di mana konfigurasi infrastruktur di-define dalam kode dan dapat di-versi terkontrol, dapat direproduksi, dan otomatis, memungkinkan penyediaan dan modifikasi infrastruktur dengan cepat. Kontainerisasi menggunakan alat seperti Docker untuk mengemas aplikasi beserta dependensi-nya ke dalam container yang konsisten di seluruh lingkungan pengembangan, pengujian, dan produksi, sehingga mengurangi masalah "berfungsi pada mesin saya".

8.5.4 Pemantauan Pasca-Penerapan dan Keunggulan Operasional

Setelah penerapan selesai, pemantauan yang berkelanjutan dan komprehensif sangat penting untuk memastikan bahwa sistem

berfungsi optimal dan memenuhi ekspektasi kinerja dalam lingkungan produksi. Pelacakan metrik pemantauan kinerja aplikasi seperti waktu respons, throughput, tingkat kesalahan, dan metrik bisnis yang menunjukkan kesehatan sistem dari perspektif pengguna akhir. Pemantauan infrastruktur mengukur pemanfaatan sumber daya dari server, database, dan jaringan untuk memastikan bahwa infrastruktur memiliki kapasitas yang cukup untuk menangani beban dan mengidentifikasi kemacetan yang mungkin memerlukan optimasi atau penskalaan.

Keunggulan operasional dalam lingkungan produksi memerlukan runbook dan pedoman yang jelas untuk tugas operasional umum dan prosedur pemecahan masalah sehingga tim operasi dapat merespons masalah dengan cepat. Logging dan observability yang komprehensif memberikan visibilitas penuh tentang apa yang terjadi dalam sistem untuk memfasilitasi debugging dan analisis akar penyebab dari masalah produksi. Prosedur rotasi panggilan dan manajemen insiden harus ditetapkan untuk memastikan bahwa permasalahan kritis dapat diselesaikan dengan segera tanpa penundaan. Tinjauan pasca-insiden harus dilakukan untuk menganalisis akar penyebab masalah produksi dan menerapkan tindakan pencegahan untuk mencegah masalah yang sama terulang di masa depan, menciptakan budaya pembelajaran dan perbaikan berkelanjutan.

BAB 9: Pengujian dan Evaluasi Sistem

Ghofar Taufik, M.Kom.

9.1 Konsep Pengujian Sistem

9.1.1 Definisi dan Pengertian Pengujian Sistem

Pengujian sistem merupakan tahapan krusial dalam pengembangan perangkat lunak yang bertujuan memastikan aplikasi atau sistem informasi telah berfungsi sesuai dengan spesifikasi yang telah ditetapkan sebelumnya. Pengujian sistem dapat didefinisikan sebagai serangkaian aktivitas terstruktur yang dilakukan untuk menemukan kesalahan, cacat, atau *bug* dalam suatu aplikasi sebelum sistem tersebut diimplementasikan kepada pengguna akhir. Dalam konteks pengembangan perangkat lunak modern, pengujian sistem bukan lagi merupakan kegiatan sampingan tetapi menjadi bagian integral dari proses pengembangan yang mempengaruhi kualitas akhir produk (Salshabila et al., 2026).

Pentingnya pengujian sistem tidak dapat diabaikan mengingat semakin kompleksnya sistem informasi yang dikembangkan oleh organisasi. Setiap sistem informasi yang dikembangkan harus melalui berbagai fase pengujian untuk memastikan bahwa tidak ada cacat yang terlewatkan sebelum sistem beroperasi secara penuh. Tingginya ketergantungan organisasi terhadap teknologi informasi

untuk mendukung proses bisnis mereka menjadikan pengujian sistem sebagai investasi penting dalam menjaga kontinuitas dan keandalan operasional.

9.1.2 Tujuan dan Manfaat Pengujian Sistem

Tujuan utama dari pengujian sistem adalah untuk mengidentifikasi dan mendokumentasikan semua cacat atau kesalahan yang mungkin terdapat dalam aplikasi sebelum digunakan secara operasional. Dengan melakukan pengujian secara sistematis dan menyeluruh, tim pengembang dapat memastikan bahwa setiap fitur yang telah dikembangkan berfungsi dengan baik dan sistem mampu menangani berbagai skenario penggunaan yang mungkin dihadapi oleh pengguna akhir (Salshabila et al., 2026). Selain itu, pengujian sistem juga bertujuan untuk memvalidasi bahwa sistem telah memenuhi semua kebutuhan bisnis yang telah didefinisikan sejak awal proyek.

Manfaat yang diperoleh dari pengujian sistem sangat beragam dan berdampak signifikan terhadap kesuksesan implementasi sistem. Manfaat pertama adalah peningkatan kualitas produk perangkat lunak secara keseluruhan, yang akan meningkatkan kepercayaan pengguna terhadap sistem. Manfaat kedua adalah pengurangan risiko kegagalan sistem setelah implementasi, karena cacat-cacat telah diidentifikasi dan diperbaiki pada tahap pengujian. Manfaat ketiga adalah pemenuhan standar keamanan yang diperlukan untuk melindungi data dan informasi sensitif. Manfaat keempat adalah peningkatan kepuasan pengguna karena sistem yang digunakan telah

diuji secara menyeluruh dan berfungsi sesuai dengan harapan mereka.

9.1.3 Tahapan dan Fase Pengujian Sistem

Pengujian sistem biasanya dilakukan dalam beberapa tahapan yang terstruktur mengikuti kerangka kerja yang dikenal sebagai *Software Testing Life Cycle* (STLC). Tahapan pertama adalah fase analisis kebutuhan pengujian, di mana tim pengujian mempelajari dokumentasi spesifikasi sistem, analisis persyaratan, dan dokumen desain untuk mengidentifikasi semua kebutuhan pengujian yang harus dipenuhi dan aspek-aspek yang harus diuji. Fase ini sangat penting karena kesuksesan pengujian keseluruhan bergantung pada pemahaman yang mendalam tentang apa yang harus diuji (Salshabila et al., 2026).

Tahapan kedua adalah fase perencanaan pengujian, yang melibatkan penyusunan strategi pengujian komprehensif, identifikasi sumber daya yang diperlukan termasuk alat pengujian dan personel, penentuan jadwal pelaksanaan pengujian, serta penetapan kriteria penerimaan. Tahapan ketiga adalah fase desain kasus pengujian, di mana tim membuat skenario pengujian yang komprehensif dengan mempertimbangkan berbagai kondisi dan kemungkinan situasi yang dapat terjadi. Tahapan keempat adalah fase persiapan lingkungan pengujian, yang mencakup penyiapan data uji yang representatif, konfigurasi sistem operasi dan perangkat lunak pendukung, serta pengaturan perangkat keras yang diperlukan agar lingkungan pengujian sedekat mungkin dengan lingkungan produksi.

Tahapan kelima adalah fase eksekusi pengujian, di mana semua kasus pengujian dijalankan sesuai dengan rencana yang telah disusun dengan cermat, dan hasil pengujian dicatat secara detail. Tahapan terakhir adalah fase analisis hasil pengujian dan pelaporan, di mana semua temuan pengujian didokumentasikan dengan jelas, diserahkan kepada tim pengembang untuk perbaikan, dan dilakukan evaluasi untuk memastikan semua cacat telah diperbaiki dengan memuaskan sebelum sistem dinyatakan siap untuk produksi.

9.1.4 Stakeholder dan Tanggung Jawab dalam Pengujian

Dalam proses pengujian sistem yang sukses, terdapat berbagai pihak atau pemangku kepentingan yang terlibat dengan tanggung jawab dan peran yang berbeda-beda. Tim *Quality Assurance* atau pengujian bertanggung jawab untuk merencanakan, merancang, melaksanakan pengujian sesuai dengan standar kualitas yang telah ditetapkan, dan melaporkan semua temuan yang diperoleh (Hidayat et al., 2025). Tim pengembang atau *developer* bertanggung jawab untuk membangun sistem sesuai dengan spesifikasi, memperbaiki cacat yang ditemukan selama pengujian, dan melakukan pengujian ulang untuk memverifikasi bahwa cacat telah diperbaiki dengan baik.

Perwakilan dari pengguna akhir atau pihak bisnis memberikan masukan dan validasi tentang kebutuhan fungsional, proses bisnis yang harus diuji, serta perspektif pengguna terhadap sistem. Manajer proyek bertugas untuk memastikan bahwa pengujian berjalan sesuai dengan jadwal dan anggaran yang telah ditetapkan, serta memfasilitasi komunikasi antar tim. *Database Administrator*

memastikan bahwa data pengujian tersedia, terintegrasi dengan baik, dan mencerminkan skenario dunia nyata. Dengan koordinasi yang baik di antara semua pihak ini, proses pengujian dapat berjalan efektif dan menghasilkan sistem yang berkualitas tinggi.

9.2 Jenis-Jenis Pengujian

9.2.1 Pengujian Unit (Unit Testing)

Unit testing adalah pengujian tingkat terendah yang fokus pada pengujian komponen atau modul individual dari sistem secara terisolasi. Dalam pengujian ini, setiap fungsi, metode, atau komponen kecil diuji secara terpisah untuk memastikan bahwa logika yang dikembangkan berfungsi dengan benar sesuai dengan spesifikasi yang telah ditetapkan. *Unit testing* biasanya dilakukan oleh para pengembang sendiri menggunakan alat pengujian otomatis yang telah disediakan untuk bahasa pemrograman yang digunakan, seperti *JUnit* untuk Java, *PHPUnit* untuk PHP, atau alat sejenis lainnya untuk bahasa pemrograman lain (Hidayat et al., 2025).

Keuntungan utama dari *unit testing* adalah dapat mendeteksi kesalahan pada tahap awal pengembangan, sehingga mempermudah perbaikan dan secara signifikan mengurangi biaya perbaikan yang mungkin terjadi jika cacat baru diketahui pada tahap-tahap pengujian yang lebih lanjut. Selain itu, *unit testing* yang baik mendokumentasikan perilaku yang diharapkan dari setiap komponen, sehingga mempermudah pemahaman kode oleh pengembang lain yang mungkin harus melakukan pemeliharaan di

masa depan. Dalam praktiknya, *unit testing* harus mencakup pengujian kasus positif di mana input yang diberikan adalah input yang valid, serta pengujian kasus negatif di mana input yang tidak valid diberikan untuk memastikan setiap komponen dapat menangani berbagai situasi dengan baik.

9.2.2 Pengujian Integrasi (Integration Testing)

Integration testing adalah pengujian yang fokus pada interaksi dan komunikasi antara berbagai modul atau komponen dalam sistem ketika digabungkan menjadi satu kesatuan. Tujuan dari pengujian ini adalah untuk memastikan bahwa ketika beberapa modul yang telah diuji secara individual digabungkan, mereka dapat bekerja sama dengan baik tanpa terjadi kesalahan dalam pertukaran data, komunikasi, atau koordinasi antar modul (Hidayat et al., 2025). Ada beberapa pendekatan strategis dalam melakukan *integration testing* yang dapat dipilih sesuai dengan karakteristik proyek.

Pendekatan *bottom-up integration testing* dimulai dari modul tingkat terendah dan secara bertahap naik ke modul tingkat lebih tinggi, menguji integrasi antar modul secara progresif. Pendekatan *top-down integration testing* melakukan hal sebaliknya, di mana pengujian dimulai dari modul tingkat tinggi dan secara bertahap turun ke modul tingkat lebih rendah. Pendekatan *sandwich integration testing* menggabungkan kedua pendekatan tersebut untuk mencapai efisiensi yang lebih baik dengan menguji bagian-bagian tertentu secara paralel. Pengujian integrasi membantu mengidentifikasi masalah-masalah yang tidak dapat terlihat pada pengujian unit, seperti kesalahan dalam antarmuka antar modul,

ketidaksesuaian format data yang dipertukarkan, atau ketidakselarasan dalam asumsi yang dibuat oleh setiap modul.

9.2.3 Pengujian Sistem (System Testing)

System testing adalah pengujian holistik yang dilakukan terhadap seluruh sistem yang telah terintegrasi sebagai satu kesatuan untuk memastikan bahwa sistem memenuhi semua kebutuhan dan spesifikasi yang telah ditetapkan dalam dokumen kebutuhan. Dalam pengujian ini, sistem diuji sebagai satu entitas utuh dengan menjalankan berbagai skenario penggunaan yang realistis dan sesuai dengan proses bisnis yang sesungguhnya (Hidayat et al., 2025). *System testing* mencakup pengujian fungsional untuk memverifikasi bahwa semua fitur berfungsi dengan benar, serta pengujian non-fungsional untuk memverifikasi aspek-aspek seperti performa, keamanan, skalabilitas, dan keandalan sistem.

Pengujian sistem melibatkan pengujian alur proses bisnis end-to-end yang menggambarkan skenario penggunaan aktual, pengujian integrasi dengan sistem eksternal yang mungkin terhubung, pengujian dalam lingkungan yang mendekati lingkungan produksi sedekat mungkin, serta pengujian terhadap konfigurasi perangkat keras dan perangkat lunak yang akan digunakan di produksi. Temuan dari pengujian sistem seringkali lebih signifikan dan berdampak dibandingkan dengan pengujian pada level yang lebih rendah, karena pengujian sistem dapat mengungkap masalah-masalah yang kompleks dan sulit dideteksi pada tahap-tahap pengujian sebelumnya yang fokus pada komponen individual.

9.2.4 Pengujian Penerimaan Pengguna (User Acceptance Testing)

User Acceptance Testing atau *UAT* adalah pengujian yang melibatkan pengguna akhir secara langsung untuk memvalidasi dan memverifikasi bahwa sistem memenuhi kebutuhan bisnis mereka dan dapat diterima untuk digunakan dalam operasional sehari-hari. Dalam *UAT*, pengguna akhir atau perwakilan dari pengguna menjalankan skenario penggunaan yang sesuai dengan aktivitas bisnis mereka sehari-hari dan memberikan umpan balik yang berharga tentang apakah sistem berfungsi sesuai dengan harapan dan kebutuhan mereka (Salshabila et al., 2026). *UAT* sangat penting karena melibatkan perspektif pengguna akhir yang seringkali berbeda dengan perspektif tim teknis yang telah mengembangkan sistem.

Melalui *UAT*, dapat diidentifikasi masalah-masalah yang berkaitan dengan kegunaan dan kemudahan penggunaan sistem, alur kerja bisnis yang mungkin tidak sesuai dengan proses bisnis yang sesungguhnya, atau kebutuhan khusus yang mungkin terlewatkan pada tahap-tahap pengujian sebelumnya. Hasil dari *UAT* merupakan faktor penentu yang sangat penting untuk mengambil keputusan apakah sistem siap untuk diimplementasikan secara penuh ke lingkungan produksi atau masih memerlukan perbaikan dan pengujian tambahan sebelum diluncurkan ke pengguna nyata.

9.3 Pengujian Fungsional dan Non-Fungsional

9.3.1 Pengujian Fungsional (Functional Testing)

Pengujian fungsional adalah pengujian yang memverifikasi dan memastikan bahwa setiap fitur dan fungsi dari sistem berfungsi sesuai dengan spesifikasi fungsional yang telah disepakati antara tim pengembang dan pihak bisnis atau pengguna. Pengujian ini fokus pada "apa yang dilakukan sistem" atau functionality dari sistem, bukan pada "bagaimana cara kerjanya" atau implementasi internal dari sistem. Dalam pengujian fungsional, seorang *tester* memberikan input tertentu kepada sistem dan membandingkan output atau hasil yang dihasilkan oleh sistem dengan output yang diharapkan menurut spesifikasi yang telah ditetapkan (Hidayat et al., 2025).

Pengujian fungsional dapat dilakukan menggunakan dua pendekatan utama yang berbeda dalam hal pengetahuan tentang struktur internal kode. *Black box testing* adalah pengujian yang dilakukan tanpa mempertimbangkan atau mengetahui struktur internal kode, hanya melihat input dan output sistem sebagai *black box*, sehingga fokus pengujian adalah pada fungsionalitas eksternal. Sebaliknya, *white box testing* melibatkan analisis mendalam terhadap struktur internal kode dan logika program untuk merancang kasus pengujian yang lebih komprehensif dan efektif berdasarkan pemahaman tentang implementasi internal. Pengujian fungsional meliputi berbagai aspek penting seperti pengujian validasi input data, pengujian alur proses bisnis secara end-to-end, pengujian

integrasi data antar modul, serta pengujian terhadap kebenaran laporan dan output yang dihasilkan oleh sistem.

9.3.2 Teknik-Teknik Pengujian Fungsional

Ada berbagai teknik yang telah terbukti efektif dan dapat digunakan untuk melakukan pengujian fungsional secara sistematis dan efisien. Teknik *Boundary Value Analysis* (BVA) fokus pada pengujian nilai-nilai yang berada di batas rentang input yang valid dan tidak valid, karena pengalaman menunjukkan bahwa banyak kesalahan terjadi tepatnya pada nilai-nilai batas antara domain input yang valid dan tidak valid. Teknik *Equivalence Partitioning* membagi seluruh domain input data menjadi kelompok-kelompok atau partisi yang ekuivalen dan mewakili serta menguji perwakilan dari setiap kelompok, sehingga dapat mengurangi jumlah kasus pengujian yang diperlukan tanpa mengurangi efektivitas pengujian (Hidayat et al., 2025).

Teknik *Decision Table Testing* berguna untuk menguji kombinasi kondisi yang kompleks dan interaksi antar kondisi dalam sistem, terutama ketika terdapat banyak kondisi yang saling mempengaruhi. Teknik *Cause-Effect Graphing* membantu mengidentifikasi hubungan sebab-akibat antar input untuk menghasilkan kasus pengujian yang lebih efektif. Penggunaan kombinasi dari berbagai teknik-teknik ini dalam pengujian fungsional memastikan bahwa cakupan pengujian atau *test coverage* menjadi lebih komprehensif dan efisien. Dengan menerapkan teknik-teknik ini secara strategis, seorang *tester* dapat mengurangi jumlah kasus pengujian yang diperlukan tanpa

mengorbankan efektivitas pengujian dalam menemukan cacat-cacat yang mungkin terdapat dalam sistem.

9.3.3 Pengujian Non-Fungsional (Non-Functional Testing)

Pengujian non-fungsional adalah pengujian yang memverifikasi aspek-aspek sistem yang tidak langsung terkait dengan fungsionalitas atau fitur bisnis yang spesifik, tetapi sangat penting dan krusial untuk kualitas, keandalan, dan kegunaan sistem secara keseluruhan. Pengujian ini mencakup berbagai dimensi kualitas penting seperti performa atau responsiveness sistem, keamanan sistem dari ancaman, skalabilitas kemampuan sistem menangani pertumbuhan, ketersediaan atau *uptime* sistem, pemulihan dari kegagalan atau disaster recovery, dan kemudahan penggunaan atau *usability* (Hidayat et al., 2025).

Pengujian performa mengevaluasi bagaimana sistem merespons dan memberikan layanan ketika diakses oleh beban kerja yang berbeda-beda, mulai dari beban normal hingga beban ekstrem. Pengujian keamanan memverifikasi bahwa sistem memiliki kontrol keamanan yang memadai dan efektif untuk melindungi dari akses tidak sah, pencurian data, dan serangan keamanan lainnya. Pengujian skalabilitas menentukan seberapa baik dan sampai mana sistem dapat menangani pertumbuhan dalam jumlah pengguna simultan atau volume data yang diproses. Pengujian ketersediaan memastikan bahwa sistem dapat beroperasi dengan *uptime* yang tinggi dan dapat dipulihkan dengan cepat apabila terjadi kegagalan. Setiap aspek pengujian non-fungsional memiliki standar dan metrik

kualitas yang spesifik untuk mengukur keberhasilan dan keefektifan pengujian.

9.3.4 Standar dan Kriteria Penerimaan Pengujian

Setiap pengujian, baik yang bersifat fungsional maupun non-fungsional, harus memiliki kriteria penerimaan yang jelas dan terukur yang mendefinisikan kapan pengujian dapat dinyatakan berhasil dan sistem dapat diterima atau ditolak. Kriteria penerimaan untuk pengujian fungsional biasanya berdasarkan pada spesifikasi kebutuhan yang telah didokumentasikan, di mana semua skenario penggunaan positif dan negatif harus berfungsi sesuai dengan yang ditentukan tanpa ada pengecualian atau penyimpangan (Hidayat et al., 2025). Kriteria penerimaan untuk pengujian non-fungsional biasanya berdasarkan pada standar industri yang berlaku atau persyaratan khusus yang ditetapkan oleh organisasi, seperti waktu respons maksimal yang dapat ditoleransi, tingkat keamanan minimum yang harus dipenuhi, atau tingkat ketersediaan sistem yang diharapkan.

Penetapan kriteria penerimaan yang jelas, terukur, dan realistis sebelum pengujian dimulai sangat membantu untuk memastikan bahwa pengujian fokus pada hal-hal yang benar-benar penting bagi organisasi dan pengguna, memudahkan pengambilan keputusan tentang apakah sistem siap untuk tahap implementasi berikutnya, dan mengurangi subjektivitas dalam menilai hasil pengujian. Kriteria penerimaan juga berfungsi sebagai definisi dari apa yang dimaksud dengan "kualitas yang dapat diterima" dalam konteks proyek atau organisasi spesifik.

9.4 Evaluasi Kinerja Sistem

9.4.1 Konsep dan Pentingnya Evaluasi Kinerja

Evaluasi kinerja sistem adalah proses sistematis mengukur, menganalisis, dan mengevaluasi bagaimana sistem beroperasi dalam menjalankan fungsinya di bawah berbagai kondisi dan beban kerja yang berbeda-beda. Evaluasi kinerja sistem berbeda dengan pengujian fungsional dalam hal fokus dan tujuannya. Jika pengujian fungsional fokus pada "apa yang dilakukan sistem" atau kebenaran fungsionalitas, evaluasi kinerja fokus pada "seberapa baik sistem melakukannya" atau kualitas dan efisiensi dalam melaksanakan fungsinya (Hamidah et al., 2025). Evaluasi kinerja melibatkan pengukuran metrik-metrik kuantitatif seperti waktu respons, throughput atau jumlah transaksi per detik, tingkat penggunaan CPU, penggunaan memori, dan tingkat kesalahan atau kegagalan transaksi.

Evaluasi kinerja sistem sangat penting dan kritis dalam memastikan bahwa sistem dapat memberikan pengalaman pengguna yang memuaskan dan dapat menangani beban kerja yang diharapkan tanpa mengalami degradasi kinerja yang signifikan. Dalam era digital modern di mana pengguna sangat sensitif dan demanding terhadap kecepatan, responsivitas, dan keandalan sistem, evaluasi kinerja menjadi aspek yang tidak dapat diabaikan dalam pengembangan sistem informasi. Sistem yang lambat atau tidak responsif dapat mengakibatkan frustrasi pengguna, penurunan

produktivitas, dan bahkan kehilangan bisnis apabila sistem digunakan untuk aplikasi bisnis kritis.

9.4.2 Jenis-Jenis Pengujian Kinerja

Ada beberapa jenis pengujian kinerja yang dapat dilakukan tergantung pada tujuan spesifik dari evaluasi kinerja yang ingin dicapai. *Load testing* mengevaluasi dan menganalisis bagaimana sistem berperforma ketika diakses oleh sejumlah pengguna yang sesuai dengan beban kerja normal yang diharapkan dalam operasional sehari-hari, untuk memastikan sistem dapat menangani beban normal dengan performa yang dapat diterima. *Stress testing* menentukan batasan atau limit sistem dengan terus meningkatkan beban secara progresif hingga sistem mengalami penurunan performa atau kegagalan, sehingga dapat diidentifikasi kapasitas maksimal sistem dan titik-titik di mana sistem mulai mengalami masalah (Hamidah et al., 2025).

Endurance testing atau *soak testing* mengevaluasi kestabilan dan keandalan sistem ketika beroperasi terus-menerus dalam jangka waktu yang panjang untuk mendeteksi kebocoran memori, resource leak, atau degradasi performa yang terjadi secara bertahap seiring waktu berjalan. *Spike testing* mengevaluasi bagaimana sistem merespons dan beradaptasi ketika ada lonjakan tiba-tiba dalam jumlah pengguna atau volume transaksi yang melampaui beban normal, untuk memahami kemampuan sistem mengatasi situasi peak. Setiap jenis pengujian kinerja memberikan wawasan dan informasi yang berbeda tentang perilaku dan karakteristik sistem di berbagai kondisi operasional yang berbeda-beda.

9.4.3 Alat, Metrik, dan Parameter Evaluasi Kinerja

Untuk melakukan evaluasi kinerja sistem secara efektif dan akurat, diperlukan alat-alat pengujian khusus yang dapat mensimulasikan beban pengguna yang realistis dan mengukur metrik-metrik kinerja dengan presisi tinggi. Salah satu alat yang paling populer dan banyak digunakan adalah *Apache JMeter*, yang dapat digunakan untuk melakukan pengujian beban, stress testing, dan jenis pengujian performa lainnya pada aplikasi web, REST *API*, dan berbagai protokol jaringan lainnya dengan interface yang user-friendly dan fitur-fitur yang komprehensif (Hamidah et al., 2025). Alat lain yang juga populer dan banyak digunakan termasuk *LoadRunner* dari Micro Focus, *Gatling* yang berbasis bahasa Scala, dan *HP UFT* atau *Unified Functional Testing*.

Metrik-metrik yang biasanya diukur dalam evaluasi kinerja meliputi waktu respons atau *response time* yang merupakan waktu yang diperlukan sistem untuk merespons permintaan dari pengguna, *throughput* yang menunjukkan jumlah transaksi atau request yang dapat diproses sistem dalam satuan waktu, tingkat penggunaan sumber daya sistem seperti persentase penggunaan CPU, penggunaan memori, dan utilisasi disk, serta tingkat kesalahan atau kegagalan transaksi yang mencerminkan reliabilitas sistem. Interpretasi yang cermat terhadap metrik-metrik ini membantu dalam mengidentifikasi *bottleneck* atau hambatan dalam sistem dan area-area spesifik yang memerlukan optimasi untuk meningkatkan kinerja sistem secara keseluruhan.

9.4.4 Rekomendasi dan Strategi Perbaikan Kinerja

Setelah melakukan evaluasi kinerja sistem, hasil analisis dan temuan harus diterjemahkan menjadi rekomendasi perbaikan yang konkrit, terukur, dan dapat diimplementasikan oleh tim pengembang. Jika ditemukan bahwa waktu respons sistem melebihi standar yang telah ditetapkan, rekomendasi perbaikan dapat mencakup optimasi *query* basis data untuk mengurangi waktu pemrosesan data, implementasi mekanisme *cache* untuk mengurangi akses ke basis data, atau peningkatan spesifikasi hardware server untuk meningkatkan kapasitas pemrosesan (Hamidah et al., 2025). Jika penggunaan memori menunjukkan tren peningkatan yang stabil dan berkelanjutan yang mengindikasikan *memory leak*, rekomendasi perbaikan dapat fokus pada identifikasi dan perbaikan kebocoran memori dalam kode atau library yang digunakan.

Rekomendasi perbaikan harus diprioritaskan berdasarkan dampaknya terhadap pengalaman pengguna dan sumber daya yang diperlukan untuk mengimplementasikan perbaikan tersebut. Beberapa perbaikan mungkin memerlukan perubahan arsitektur sistem yang signifikan dan memakan waktu, sementara perbaikan lainnya dapat diimplementasikan dengan modifikasi yang relatif sederhana pada konfigurasi atau parameter sistem. Strategi perbaikan kinerja harus mempertimbangkan trade-off antara biaya implementasi, waktu implementasi, dan manfaat yang akan diperoleh dari setiap perbaikan yang diusulkan.

9.5 Dokumentasi Hasil Pengujian

9.5.1 Pentingnya dan Fungsi Dokumentasi Pengujian

Dokumentasi hasil pengujian merupakan aspek kritis dan fundamental dalam proses pengujian sistem yang seringkali kurang mendapat perhatian dan alokasi resources yang memadai, padahal dokumentasi yang baik dan lengkap sangat penting untuk berbagai keperluan dan kebutuhan organisasi. Dokumentasi pengujian berfungsi sebagai catatan resmi dan komprehensif dari semua aktivitas pengujian yang telah dilakukan, temuan dan hasil pengujian yang diperoleh, cacat yang ditemukan, dan keputusan-keputusan yang dibuat terkait kualitas dan status siap pakai sistem (Qystiar & Wijayanto, 2025). Dokumentasi yang baik memfasilitasi komunikasi yang lebih baik antar anggota tim, membantu dalam penelusuran akar penyebab atau *root cause analysis* dari masalah yang ditemukan, dan menyediakan referensi historis untuk pengujian di masa depan atau untuk versi sistem yang akan dikembangkan kemudian.

Dokumentasi pengujian juga sangat penting untuk keperluan *compliance*, audit, dan keperluan legal atau regulasi, terutama dalam industri atau sektor yang memiliki regulasi ketat seperti perbankan, asuransi, kesehatan, atau pemerintahan. Dengan dokumentasi pengujian yang baik dan terorganisir dengan rapi, organisasi dapat menunjukkan kepada auditor eksternal bahwa mereka telah melakukan pengujian yang memadai dan sesuai dengan standar industri yang berlaku. Dokumentasi juga membantu dalam

estimasi dan perencanaan pengujian untuk proyek-proyek serupa di masa depan berdasarkan pengalaman dan data historis.

9.5.2 Komponen-Komponen Utama Dokumentasi Pengujian

Dokumentasi pengujian yang komprehensif dan lengkap harus mencakup beberapa komponen penting yang saling melengkapi. Rencana pengujian atau *Test Plan* merupakan dokumen induk yang mendokumentasikan secara detail strategi pengujian yang akan dilakukan, ruang lingkup atau scope dari pengujian, sumber daya yang diperlukan termasuk alat dan personel, jadwal rencana pengujian, risiko-risiko pengujian, dan kriteria penerimaan atau exit criteria yang harus dipenuhi agar pengujian dinyatakan berhasil (Qystiar & Wijayanto, 2025). Spesifikasi kasus pengujian atau *Test Case Specification* mendeskripsikan setiap kasus pengujian secara detail dan terstruktur, termasuk kondisi awal atau *precondition*, langkah-langkah pengujian yang harus dijalankan, input data yang akan diberikan, output yang diharapkan, dan kriteria lulus atau gagal untuk setiap kasus.

Laporan eksekusi pengujian atau *Test Execution Report* mencatat hasil pengujian untuk setiap kasus pengujian yang telah dijalankan, termasuk tanggal dan waktu eksekusi, hasil pengujian (*pass* atau *fail*), dan catatan atau observasi tentang masalah yang ditemukan selama eksekusi. Laporan cacat atau *Bug Report* mendokumentasikan secara detail setiap cacat atau masalah yang ditemukan selama pengujian, termasuk deskripsi singkat dari cacat, langkah-langkah untuk mereproduksi cacat, tingkat keparahan atau severity, dan status perbaikan dari cacat. Ringkasan hasil

pengujian atau *Test Summary Report* memberikan gambaran keseluruhan tentang hasil pengujian, tingkat keberhasilan pengujian dalam persentase atau metrik lainnya, dan rekomendasi tentang apakah sistem siap untuk tahap implementasi berikutnya atau masih memerlukan perbaikan tambahan.

9.5.3 Standar dan Kerangka Kerja Dokumentasi

Untuk memastikan konsistensi, profesionalisme, dan kualitas dari dokumentasi pengujian di dalam organisasi, banyak organisasi mengadopsi dan mengikuti standar internasional yang telah ditetapkan untuk dokumentasi pengujian. Standar *IEEE 829* atau *IEEE Standard for Software and System Test Documentation* merupakan standar internasional yang komprehensif dan memberikan panduan detail tentang format, struktur, dan konten dari berbagai jenis dokumen pengujian yang harus dibuat (Qystiar & Wijayanto, 2025). Standar ini membantu memastikan bahwa dokumentasi pengujian konsisten, terstandar, dan mudah dipahami oleh semua pihak yang terlibat dalam proses pengujian maupun pihak eksternal yang memerlukan dokumentasi ini.

Standar lain yang juga relevan dan banyak digunakan adalah *ISO/IEC 29119* yang berjudul *Information technology – Software and system engineering – Software testing*, yang menyediakan panduan komprehensif tentang proses pengujian perangkat lunak secara keseluruhan, termasuk panduan detail tentang dokumentasi pengujian. Dengan mengadopsi standar-standar internasional ini, organisasi dapat memastikan bahwa proses pengujian berjalan secara profesional, sistematis, dan terstruktur

dengan baik, serta menghasilkan dokumentasi pengujian yang berkualitas tinggi, konsisten, dan dapat dipercaya.

9.5.4 Manajemen dan Pengelolaan Dokumentasi Pengujian

Dokumentasi pengujian yang berjumlah besar dan kompleks memerlukan sistem manajemen yang baik dan terstruktur untuk memastikan bahwa dokumen dapat diakses dengan mudah, dapat dicari dengan cepat, dan dapat diperbarui atau dimodifikasi dengan aman tanpa kehilangan integritas data. Banyak organisasi modern menggunakan sistem manajemen alat pengujian atau *test management tools* seperti *TestRail*, *Zephyr*, atau *HP Application Lifecycle Management* (*HP ALM*) untuk mengorganisir, menyimpan, mengelola, dan mengendalikan dokumentasi pengujian secara terpusat dan terintegrasi (Qystiar & Wijayanto, 2025). Sistem-sistem ini menyediakan interface yang user-friendly, fitur pencarian yang powerful, dan kemampuan untuk melacak versi dokumen.

Sistem pelacakan masalah atau *bug tracking system* seperti *Jira*, *Bugzilla*, atau *Azure DevOps* membantu dalam mencatat, melacak, dan mengelola status perbaikan dari setiap cacat atau masalah yang ditemukan selama pengujian. Integrasi yang baik antara sistem manajemen pengujian dan sistem pelacakan masalah memungkinkan alur kerja yang lebih efisien dan transparan tentang status pengujian dan perbaikan masalah yang ditemukan. Dengan sistem manajemen dokumentasi yang baik dan terorganisir, dokumentasi pengujian bukan hanya memenuhi keperluan operasional saat ini, tetapi juga menjadi aset berharga yang dapat

dimanfaatkan untuk peningkatan proses pengujian secara berkelanjutan di masa depan dan pembelajaran organisasi.

BAB 10: Manajemen Keamanan Sistem Informasi

Indra Riyana Rahadjeng, S.T., M.Kom.

10.1 Konsep Keamanan Sistem

10.1.1 Definisi dan Pentingnya Keamanan Informasi

Keamanan sistem informasi telah menjadi aspek kritis dalam operasional organisasi modern yang semakin bergantung pada teknologi digital. Dalam era transformasi digital saat ini, data telah menjadi salah satu aset paling berharga bagi setiap organisasi, baik itu institusi pemerintah, perusahaan swasta, maupun lembaga pendidikan. Keamanan informasi dapat didefinisikan sebagai upaya perlindungan komprehensif terhadap informasi dan sistem yang memprosesnya dari ancaman yang dapat menyebabkan kerusakan, kehilangan, atau pengungkapan yang tidak sah (Fayayola et al., 2024). Pentingnya keamanan informasi tidak dapat dipisahkan dari kebutuhan organisasi untuk menjaga kepercayaan pemangku kepentingan, mempertahankan keunggulan kompetitif, dan memenuhi persyaratan regulasi yang semakin ketat.

Sistem manajemen keamanan informasi merupakan pendekatan terstruktur untuk mengelola keamanan data dan sistem informasi dalam organisasi secara berkelanjutan. Dengan menerapkan keamanan informasi yang efektif, organisasi dapat

meminimalkan risiko terhadap ancaman siber yang terus berkembang, melindungi data sensitif pelanggan dan karyawan, serta memastikan kontinuitas bisnis dalam menghadapi insiden keamanan yang tidak terduga. Investasi dalam keamanan informasi bukan sekadar pengeluaran operasional, tetapi merupakan investasi strategis yang berkontribusi pada pertumbuhan dan keberlanjutan jangka panjang organisasi.

10.1.2 Pilar Utama Keamanan Informasi

Keamanan informasi dibangun atas fondasi tiga pilar utama yang sering dikenal dengan istilah *CIA* (Confidentiality, Integrity, Availability). Pertama, kerahasiaan atau *Confidentiality* memastikan bahwa informasi hanya dapat diakses oleh pihak-pihak yang telah diberi otorisasi dan wewenang untuk mengaksesnya. Aspek ini sangat penting dalam melindungi informasi rahasia perusahaan, data pribadi pelanggan, dan dokumen-dokumen sensitif lainnya dari akses yang tidak sah baik dari pihak luar maupun dari dalam organisasi itu sendiri (Fayayola et al., 2024). Kedua, integritas atau *Integrity* menjamin bahwa informasi tidak mengalami perubahan, penghapusan, atau pemalsuan tanpa persetujuan yang resmi dari pihak yang berwenang.

Aspek integritas memastikan bahwa data tetap akurat, lengkap, dan dapat dipercaya sepanjang siklus hidupnya, dari saat pengumpulan hingga penyimpanan dan pemrosesan. Ketiga, ketersediaan atau *Availability* menjamin bahwa informasi dan sistem dapat diakses dengan mudah oleh pengguna yang berwenang kapan pun dibutuhkan untuk mendukung operasi bisnis yang

berkelanjutan. Ketiga pilar ini saling terkait dan sama pentingnya dalam membangun sistem keamanan informasi yang komprehensif dan efektif bagi organisasi.

10.1.3 Aset Informasi dan Identifikasi Aset

Aset informasi mencakup semua bentuk data, sistem, perangkat keras, perangkat lunak, serta sumber daya manusia yang terlibat dalam pengelolaan informasi di dalam organisasi. Identifikasi aset merupakan langkah fundamental dalam manajemen keamanan informasi karena tanpa mengetahui secara pasti apa yang perlu dilindungi, organisasi tidak dapat merancang strategi perlindungan yang efektif dan efisien. Aset informasi dapat diklasifikasikan menjadi berbagai kategori, termasuk basis data yang berisi data pelanggan dan karyawan, aplikasi bisnis yang kritis untuk operasi, infrastruktur jaringan yang menghubungkan sistem-sistem tersebut, perangkat penyimpanan data baik yang berada di lokasi fisik maupun di cloud, dokumen-dokumen penting baik dalam bentuk digital maupun cetak, serta kekayaan intelektual dan informasi strategis organisasi.

Proses identifikasi aset harus dilakukan secara menyeluruh, sistematis, dan melibatkan semua departemen serta unit kerja dalam organisasi. Setiap aset yang teridentifikasi harus didokumentasikan dengan rinci mencakup deskripsi lengkap aset, lokasi penyimpanan baik fisik maupun digital, pemilik atau *owner* aset, tingkat sensitivitas atau nilai aset bagi organisasi, serta risiko yang mungkin terjadi jika aset tersebut terkena dampak dari ancaman keamanan. Dokumentasi yang akurat dan terpelihara dengan baik akan sangat

memudahkan proses manajemen keamanan selanjutnya dan memastikan bahwa tidak ada aset penting yang terlewatkan dalam implementasi pengendalian keamanan.

10.1.4 Standar dan Kerangka Kerja Internasional

Dalam menerapkan keamanan sistem informasi, organisasi memiliki berbagai pilihan standar internasional dan kerangka kerja yang telah terbukti efektivitasnya dalam memandu implementasi keamanan yang komprehensif. ISO/IEC 27001 merupakan standar internasional yang paling banyak diadopsi dan menjadi acuan utama untuk membangun, mengimplementasikan, memelihara, dan terus meningkatkan Sistem Manajemen Keamanan Informasi dalam organisasi (Fayayola et al., 2024). Standar ini menyediakan panduan komprehensif mengenai berbagai aspek yang harus dipertimbangkan dalam keamanan informasi, mulai dari kebijakan keamanan tingkat organisasi, manajemen risiko, pengendalian akses, enkripsi, hingga penanganan insiden keamanan.

Selain ISO/IEC 27001, terdapat kerangka kerja lain yang relevan dan banyak digunakan, seperti COBIT (Control Objectives for Information and Related Technology) yang fokus pada tata kelola teknologi informasi yang lebih luas, serta NIST Cybersecurity Framework yang memberikan panduan praktis untuk mengidentifikasi, melindungi, mendeteksi, merespons, dan memulihkan dari serangan siber. Dengan menggunakan standar dan kerangka kerja yang sesuai dengan konteks organisasi, perusahaan dapat memastikan bahwa implementasi keamanan informasi mereka

mengikuti praktik terbaik yang telah diakui secara global dan terus berkembang seiring dengan dinamika ancaman keamanan modern.

10.2 Ancaman dan Risiko Keamanan

10.2.1 Jenis-Jenis Ancaman Keamanan Informasi

Ancaman keamanan informasi dapat berasal dari berbagai sumber dan memiliki karakteristik yang sangat beragam dalam hal teknik, dampak, dan tingkat kesophistikasiannya. Ancaman dapat diklasifikasikan secara umum menjadi dua kategori utama, yaitu ancaman eksternal yang berasal dari luar organisasi dan ancaman internal yang berasal dari dalam organisasi itu sendiri. Ancaman eksternal mencakup serangan siber yang dilakukan oleh pihak ketiga yang tidak berwenang seperti peretas, kelompok kejahatan siber terorganisir, atau bahkan negara lain yang melakukan serangan bertarget terhadap infrastruktur kritis organisasi (Fayayola et al., 2024). Jenis-jenis serangan eksternal termasuk distribusi *malware* melalui internet, percobaan pencurian data oleh kompetitor bisnis, penipuan berbasis internet atau *phishing*, *denial of service* attacks, serta serangan terkoordinasi dari kelompok peretas profesional yang berniat mencuri informasi berharga.

Ancaman internal meliputi tindakan yang dilakukan oleh karyawan, kontraktor, atau pengguna internal yang memiliki akses sah ke sistem informasi organisasi. Ancaman internal dapat berupa pencurian data yang disengaja oleh karyawan yang tidak puas, penggunaan akses sistem untuk kepentingan pribadi yang tidak

sesuai dengan kebijakan, atau pengungkapan informasi rahasia kepada pihak yang tidak berwenang. Selain itu, faktor manusia seperti kelalaian dalam menangani data sensitif, ketidakpatuhan terhadap kebijakan keamanan yang telah ditetapkan, atau kurangnya kesadaran keamanan juga dapat menciptakan celah keamanan yang dapat dieksploitasi oleh penyerang baik yang berasal dari dalam maupun dari luar organisasi.

10.2.2 Kerentanan Sistem dan Mekanisme Eksploitasi

Kerentanan merupakan kelemahan atau celah dalam desain, implementasi, operasi, atau manajemen sistem informasi yang dapat dieksploitasi oleh ancaman untuk mencapai tujuan mereka yang biasanya merugikan organisasi. Kerentanan dapat timbul dari berbagai sumber yang kompleks, termasuk perangkat lunak yang memiliki *bug* atau cacat desain (*flaw*), konfigurasi sistem yang tidak optimal atau tidak sesuai dengan standar keamanan, kurangnya pembaruan keamanan atau *patch* yang diperlukan, protokol keamanan yang lemah, atau proses manajemen akses yang tidak memadai (Ajiga et al., 2024). Kerentanan yang paling umum ditemukan dalam sistem informasi modern antara lain mekanisme otentikasi yang lemah, enkripsi yang tidak memadai atau tidak digunakan sama sekali, tidak adanya validasi input pada aplikasi web yang dapat menyebabkan *injection attacks*, konfigurasi default yang tidak diubah pada perangkat jaringan, dan manajemen patch yang tidak teratur sehingga sistem terus menjalankan versi perangkat lunak yang sudah diketahui memiliki kerentanan.

Untuk mengurangi kerentanan sistem informasi, organisasi harus secara proaktif melakukan identifikasi kerentanan melalui berbagai metode seperti *vulnerability scanning*, analisis kode, dan *penetration testing*, kemudian mendokumentasikan setiap kerentanan yang ditemukan, menilai tingkat keparahannya, dan membuat rencana perbaikan yang jelas dengan prioritas berdasarkan tingkat risiko. Proses ini sering disebut sebagai manajemen kerentanan atau *vulnerability management* yang merupakan bagian integral dari strategi keamanan informasi yang efektif dan proaktif.

10.2.3 Manajemen Risiko Keamanan Informasi Komprehensif

Manajemen risiko keamanan informasi adalah proses sistematis dan berkelanjutan untuk mengidentifikasi, menganalisis, mengevaluasi, dan menangani risiko keamanan informasi dalam organisasi dengan cara yang terukur dan terdokumentasi. Risiko merupakan perpaduan antara kemungkinan terjadinya ancaman keselamatan dan dampak atau kerugian yang dapat ditimbulkan jika ancaman tersebut berhasil terealisasi dan mengakibatkan terganggunya sistem atau hilangnya data penting (Ajiga et al., 2024). Proses manajemen risiko dimulai dengan fase komunikasi dan konsultasi dengan semua pemangku kepentingan untuk memastikan pemahaman yang sama tentang konteks dan tujuan manajemen risiko.

Tahap berikutnya adalah penetapan konteks risiko yang jelas dan terukur, dilanjutkan dengan fase penilaian risiko yang meliputi identifikasi risiko potensial yang mungkin terjadi, analisis kemungkinan dan dampak dari setiap risiko yang teridentifikasi,

serta evaluasi tingkat risiko dengan membandingkan hasil analisis terhadap kriteria risiko yang telah ditetapkan sebelumnya. Setelah risiko teridentifikasi dan dievaluasi dengan baik, organisasi harus menentukan strategi perlakuan risiko yang sesuai dengan konteks bisnis organisasi. Strategi ini dapat berupa mitigasi risiko dengan menerapkan pengendalian keamanan yang efektif, penerimaan risiko jika tingkat risiko masih berada dalam batas toleransi yang dapat diterima organisasi, transfer risiko kepada pihak ketiga melalui mekanisme asuransi siber, atau penghindaran risiko dengan menghentikan aktivitas yang membawa risiko tersebut.

10.2.4 Analisis Dampak Bisnis dan Prioritas Penanganan

Setiap risiko keamanan memiliki tingkat dampak yang berbeda-beda terhadap operasi bisnis dan tujuan strategis organisasi secara keseluruhan. Analisis dampak bisnis atau *Business Impact Analysis* membantu organisasi memahami konsekuensi yang dapat terjadi jika sistem informasi tertentu mengalami gangguan, kerusakan, atau tidak tersedia dalam jangka waktu tertentu. Dampak dapat diukur dari berbagai dimensi yang kompleks, termasuk dampak finansial berupa kerugian langsung atau kehilangan pendapatan yang potensial, dampak operasional berupa gangguan pada proses bisnis kritis yang dapat menghambat layanan kepada pelanggan, dampak reputasi berupa hilangnya kepercayaan dan loyalitas pelanggan terhadap organisasi, serta dampak hukum dan kepatuhan terhadap regulasi yang berlaku.

Dengan memahami dampak bisnis dari setiap risiko keamanan secara mendalam, organisasi dapat memprioritaskan alokasi sumber

daya keamanan dengan lebih efektif dan efisien untuk mencapai perlindungan maksimal. Risiko yang memiliki kombinasi kemungkinan tinggi dan dampak besar harus diberikan prioritas lebih tinggi dalam penanganannya dibandingkan dengan risiko yang memiliki kemungkinan rendah atau dampak yang kecil terhadap bisnis (Fayayola et al., 2024). Proses prioritisasi yang terstruktur ini memastikan bahwa organisasi dapat memaksimalkan efektivitas investasi keamanan informasi dengan fokus pada risiko yang paling kritis bagi keberlangsungan dan kesuksesan bisnis dalam jangka panjang.

10.3 Pengendalian Akses

10.3.1 Prinsip-Prinsip Pengendalian Akses Berbasis Peran

Pengendalian akses merupakan mekanisme keamanan yang dirancang untuk memastikan bahwa hanya pengguna yang berwenang yang dapat mengakses aset informasi dan sistem tertentu sesuai dengan peran dan tanggung jawab yang diberikan kepada mereka dalam organisasi. Pengendalian akses berbasis peran atau *Role-Based Access Control* (RBAC) adalah model pengendalian akses yang paling banyak diterapkan dalam organisasi modern karena kesederhanaan dan fleksibilitasnya. Dalam model RBAC, setiap pengguna diassignkan ke satu atau lebih peran yang telah ditentukan sebelumnya, dan setiap peran memiliki hak akses yang spesifik dan terbatas terhadap sumber daya sistem tertentu (Ajiga et al., 2024). Misalnya, peran *administrator* sistem

mungkin memiliki hak akses penuh terhadap semua konfigurasi sistem, sementara peran *user* biasa hanya memiliki hak akses terbatas untuk menggunakan aplikasi sesuai kebutuhan pekerjaan mereka.

Prinsip dasar dalam pengendalian akses adalah prinsip *least privilege*, yang menyatakan bahwa setiap pengguna hanya diberikan hak akses minimum yang benar-benar diperlukan untuk menjalankan tugas-tugas spesifik mereka dengan baik dan tidak lebih dari itu. Pendekatan ini membantu mengurangi risiko keamanan secara signifikan dengan membatasi potensi kerusakan yang dapat ditimbulkan oleh pengguna yang berkompromi, sistem yang terinfeksi, atau oleh pengguna yang bertindak nakal dengan tujuan jahat. RBAC juga memudahkan pengelolaan akses dalam organisasi yang besar karena administrator keamanan hanya perlu mengelola peran-peran yang ada, bukan setiap akun pengguna individual secara satu per satu.

10.3.2 Autentikasi Pengguna dan Verifikasi Identitas

Autentikasi pengguna merupakan proses fundamental dalam keamanan sistem informasi untuk memverifikasi dan memastikan bahwa pengguna yang mengklaim diri mereka adalah benar-benar orang atau entitas yang mereka klaim. Autentikasi dapat dilakukan melalui berbagai mekanisme dan faktor yang dirancang untuk memberikan tingkat keyakinan yang berbeda-beda. Faktor autentikasi pertama adalah sesuatu yang diketahui (*something you know*) seperti kata sandi, PIN, atau jawaban pertanyaan keamanan. Faktor kedua adalah sesuatu yang dimiliki (*something you have*)

seperti kartu identitas, token fisik, atau perangkat mobile yang menerima kode verifikasi. Faktor ketiga adalah sesuatu yang menjadi ciri diri (*something you are*) seperti sidik jari, pengenalan wajah, atau pola iris mata (Ajiga et al., 2024).

Autentikasi multi-faktor atau *Multi-Factor Authentication* (MFA) merupakan pendekatan keamanan yang jauh lebih kuat dan direkomendasikan untuk melindungi akun dan sistem yang sangat sensitif. MFA memerlukan bukti identitas dari lebih dari satu faktor, sehingga bahkan jika salah satu faktor berhasil dikompromikan atau dicuri, penyerang masih memerlukan faktor lain untuk dapat berhasil mengakses sistem. Misalnya, pengguna harus memasukkan kata sandi mereka dan juga memberikan kode verifikasi enam digit yang diterima melalui aplikasi otentikator atau pesan singkat. Dengan menerapkan MFA secara konsisten dan luas, organisasi dapat secara signifikan mengurangi risiko akses tidak sah bahkan dalam skenario di mana kata sandi pengguna berhasil dicuri, ditebak, atau dikompromikan oleh penyerang yang canggih.

10.3.3 Manajemen Hak Akses dan Audit Berkala

Manajemen hak akses melibatkan proses berkelanjutan untuk pemberian, pembaruan, dan pencabutan hak akses pengguna terhadap sistem dan sumber daya informasi sesuai dengan kebutuhan pekerjaan dan peran mereka yang berubah dari waktu ke waktu. Ketika karyawan baru bergabung dengan organisasi, sistem akses yang sesuai dengan peran dan departemen mereka harus disiapkan terlebih dahulu berdasarkan prinsip *least privilege*. Sebaliknya, ketika karyawan keluar dari organisasi, berhenti dari posisi tertentu,

atau berpindah ke posisi lain dalam organisasi, hak akses mereka harus segera diperbarui atau dicabut untuk mencegah penggunaan akses yang tidak sah atau terlalu luas (Swanzy et al., 2024). Proses ini harus didokumentasikan dengan baik dan dipantau secara ketat untuk memastikan kepatuhan.

Organisasi harus melakukan audit atau peninjauan berkala dan sistematis terhadap hak akses yang telah diberikan kepada semua pengguna untuk memastikan bahwa hak akses masih sesuai dengan peran dan tanggung jawab terkini mereka dalam organisasi. Proses audit ini sangat penting untuk mengidentifikasi akun-akun yang sudah tidak digunakan atau *orphan accounts*, hak akses yang berlebihan dan melebihi kebutuhan nyata, atau anomali lain yang dapat mengindikasikan kompromi keamanan atau penyalahgunaan hak akses. Dengan melakukan pembaruan, audit, dan monitoring hak akses secara berkala dan konsisten, organisasi dapat memastikan bahwa pengendalian akses tetap efektif dalam mengontrol siapa yang dapat mengakses sistem informasi kritis dan melindungi dari penggunaan akses yang tidak sah.

10.3.4 Segregasi Tugas dan Pencegahan Fraud

Segregasi tugas atau *Separation of Duties* (SoD) adalah prinsip kontrol internal yang sangat penting dan fundamental untuk mencegah penyalahgunaan wewenang, kecurangan, serta kesalahan baik yang disengaja maupun tidak disengaja dalam proses bisnis kritis. Prinsip ini mengharuskan bahwa tugas-tugas yang berpotensi menghasilkan dampak signifikan dalam sebuah proses bisnis tidak dapat semuanya dijalankan atau dikendalikan oleh satu orang saja.

Misalnya, dalam proses persetujuan pengeluaran dana organisasi, tugas pembuatan permintaan, verifikasi kebutuhan, persetujuan, dan pembayaran harus dilakukan oleh orang-orang yang berbeda dan independen untuk mencegah satu individu dapat mengotomatisasi seluruh proses sesuai dengan kepentingan pribadi mereka tanpa pengawasan (Naiho et al., 2024).

Dengan menerapkan segregasi tugas dalam sistem informasi dan proses bisnis organisasi, perusahaan dapat mencegah individu berkompromi atau karyawan nakal untuk melakukan tindakan yang merugikan organisasi, mencuri aset, atau memanipulasi data tanpa diketahui atau disetujui oleh pihak lain yang independen. Dalam konteks teknologi informasi, segregasi tugas dapat diimplementasikan melalui konfigurasi sistem yang mengharuskan persetujuan dari beberapa pengguna dengan peran berbeda dan independen sebelum tindakan kritis dapat dieksekusi, serta dengan membatasi hak akses administratif kepada personel yang benar-benar berwenang dan dapat dipercaya.

10.4 Enkripsi dan Proteksi Data

10.4.1 Konsep Dasar Enkripsi dan Algoritma Kriptografi

Enkripsi adalah proses konversi data dari bentuk aslinya yang dapat dibaca (*plaintext*) menjadi bentuk yang tidak dapat dibaca atau *ciphertext* menggunakan algoritma matematis yang kompleks dan kunci keamanan tertentu yang hanya diketahui oleh pihak yang berwenang. Tujuan utama enkripsi adalah melindungi kerahasiaan

data dengan memastikan bahwa hanya pihak yang memiliki kunci dekripsi yang tepat dan sesuai yang dapat mengakses kembali data asli dalam bentuk yang dapat dibaca dan dipahami (Fayayola et al., 2024). Enkripsi merupakan salah satu teknik perlindungan data yang paling fundamental dan sangat banyak digunakan dalam sistem keamanan informasi modern untuk berbagai keperluan.

Enkripsi dapat diklasifikasikan menjadi dua jenis utama yang memiliki karakteristik dan kegunaan berbeda, yaitu enkripsi simetris dan enkripsi asimetris. Enkripsi simetris menggunakan kunci yang sama untuk proses enkripsi data dan juga untuk dekripsi data, sehingga lebih cepat dan efisien dalam hal performa untuk data bervolume besar. Namun, tantangan utama dari enkripsi simetris adalah distribusi kunci yang aman dan terjaga, karena kedua belah pihak yang melakukan komunikasi harus memiliki kunci yang sama tanpa membiarkan pihak ketiga mengetahuinya. Sebaliknya, enkripsi asimetris menggunakan sepasang kunci yang berbeda secara matematis, yaitu kunci publik untuk enkripsi dan kunci privat untuk dekripsi, yang memudahkan distribusi kunci namun lebih lambat dibandingkan enkripsi simetris dalam hal kecepatan pemrosesan data (Swanzy et al., 2024).

10.4.2 Enkripsi Data dalam Transit dan Data dalam Keadaan Istirahat

Data dalam perjalanan atau *data in transit* mengacu pada data yang sedang ditransmisikan melalui jaringan komunikasi dari satu lokasi ke lokasi tujuan lain, baik melalui jaringan publik internet maupun jaringan privat organisasi. Data jenis ini sangat rentan

terhadap penyadapan atau intersepsi oleh pihak yang tidak berwenang ketika melewati infrastruktur jaringan publik yang tidak aman atau jaringan pihak ketiga yang tidak dapat dikendalikan sepenuhnya oleh organisasi. Untuk melindungi data dalam perjalanan secara efektif, organisasi harus menggunakan protokol komunikasi yang aman dan telah dienkripsi secara kuat seperti HTTPS untuk komunikasi web, TLS (*Transport Layer Security*) untuk berbagai protokol komunikasi, atau VPN (*Virtual Private Network*) untuk menghubungkan jaringan-jaringan yang tersebar di berbagai lokasi (Swanzy et al., 2024).

Data dalam keadaan istirahat atau *data at rest* mengacu pada data yang disimpan dalam media penyimpanan berbagai macam seperti basis data, server file, perangkat penyimpanan eksternal, atau cloud storage yang tidak sedang diakses atau ditransmisikan. Meskipun data ini tidak sedang ditransmisikan atau bergerak, namun tetap sangat rentan terhadap akses yang tidak sah jika media penyimpanan dapat diakses secara fisik oleh pihak yang tidak berwenang atau dapat diikat melalui serangan cyber. Untuk melindungi data dalam keadaan istirahat dengan memadai, organisasi harus menerapkan enkripsi pada berbagai tingkatan termasuk tingkat basis data dengan enkripsi kolom sensitif, tingkat sistem file dengan enkripsi folder atau partisi, atau tingkat perangkat penyimpanan dengan enkripsi seluruh volume penyimpanan untuk memastikan bahwa bahkan jika media penyimpanan dicuri atau diakses secara tidak sah, data tetap terlindungi dan tidak dapat dibaca tanpa kunci enkripsi yang sesuai.

10.4.3 Manajemen Kunci Kriptografi yang Aman

Manajemen kunci kriptografi adalah aspek yang sangat kritis dari implementasi enkripsi dan sering kali menjadi kelemahan yang paling serius dalam sistem keamanan organisasi yang seharusnya dapat diandalkan. Kunci kriptografi harus dihasilkan dengan benar dan menggunakan proses yang cukup acak, disimpan dengan aman dalam lingkungan yang terlindungi dari akses yang tidak sah, didistribusikan hanya kepada pihak yang berwenang dan membutuhkannya, dan diperbarui atau dirotasi secara berkala untuk mengurangi risiko kompromi kunci (Fayayola et al., 2024). Organisasi harus memiliki kebijakan dan prosedur yang jelas, terdokumentasi, dan dipatuhi mengenai siklus hidup lengkap kunci, termasuk proses generasi kunci, penyimpanan yang aman, penggunaan yang terkontrol, rotasi berkala, dan penghancuran kunci yang sudah tidak diperlukan lagi.

Sistem manajemen kunci atau *Key Management System* (KMS) dapat digunakan untuk mengotomatisasi dan memusatkan pengelolaan kunci kriptografi dalam organisasi sehingga dapat dipantau secara terpusat dan konsisten. KMS menyediakan fitur-fitur penting seperti generasi kunci yang aman dengan tingkat entropi tinggi, penyimpanan kunci yang dienkripsi berlapis dengan proteksi berlapis, distribusi kunci otomatis kepada sistem yang membutuhkan, audit trail komprehensif untuk memantau dan mencatat setiap penggunaan kunci, serta rotasi kunci yang terjadwal secara otomatis dan dapat disesuaikan. Dengan menggunakan KMS yang baik, organisasi dapat mengurangi risiko

manusia dalam pengelolaan kunci, meningkatkan kepatuhan terhadap kebijakan keamanan, dan memastikan bahwa kunci-kunci kritis tetap aman dari akses yang tidak sah atau kehilangan yang tidak disengaja.

10.4.4 Proteksi Data Pribadi dan Kepatuhan Regulasi Global

Dalam era digital saat ini di mana privasi data telah menjadi isu utama bagi konsumen dan regulator, perlindungan data pribadi telah menjadi prioritas yang sangat penting baik dari perspektif bisnis organisasi maupun dari perspektif hukum dan regulasi. Organisasi yang menangani data pribadi pelanggan, karyawan, atau pihak ketiga lainnya harus memastikan dengan sungguh-sungguh bahwa data tersebut dilindungi dengan baik dari akses, penggunaan, atau pengungkapan yang tidak sah kepada pihak yang tidak berwenang. Selain itu, organisasi juga harus mematuhi berbagai regulasi dan undang-undang perlindungan data yang berlaku di berbagai yurisdiksi, seperti Regulasi Perlindungan Data Pribadi di Indonesia, GDPR (*General Data Protection Regulation*) di Eropa, atau CCPA (*California Consumer Privacy Act*) di Amerika Serikat (Fayayola et al., 2024).

Untuk memenuhi persyaratan perlindungan data pribadi dan kepatuhan regulasi secara menyeluruh, organisasi harus menerapkan berbagai mekanisme teknis dan administratif yang saling mendukung dan terintegrasi. Mekanisme teknis mencakup enkripsi data sensitif menggunakan algoritma yang kuat, pengendalian akses yang ketat terhadap data pribadi, audit trail atau log audit yang komprehensif, dan *data masking* atau penyembunyian informasi

sensitif dalam laporan atau database pengembangan. Mekanisme administratif mencakup kebijakan privasi yang jelas dan mudah dipahami, persetujuan pengguna yang eksplisit untuk pengumpulan dan penggunaan data, proses penanganan permintaan akses data dari individu, dan prosedur respons yang cepat dan efektif dalam hal terjadi pelanggaran data atau *data breach* yang serius (Swanzy et al., 2024).

10.5 Audit Keamanan Sistem

10.5.1 Tujuan dan Ruang Lingkup Audit Keamanan Komprehensif

Audit keamanan sistem informasi adalah proses evaluasi sistematis, terstruktur, dan independen terhadap implementasi keamanan informasi yang sudah ada dalam organisasi untuk menilai efektivitas pengendalian keamanan, kesesuaian dengan standar, kebijakan, dan persyaratan regulasi yang berlaku di industri. Tujuan utama audit keamanan adalah memberikan keyakinan yang memadai kepada manajemen puncak dan pemangku kepentingan lainnya bahwa sistem informasi dan data organisasi telah dilindungi dengan memadai terhadap ancaman keamanan yang terus berkembang dan berubah (Ajiga et al., 2024). Audit keamanan juga bertujuan untuk mengidentifikasi kelemahan dan celah keamanan yang masih ada, memberikan rekomendasi untuk perbaikan, serta mendokumentasikan status keamanan organisasi sebagai bukti kepatuhan terhadap regulasi.

Ruang lingkup audit keamanan dapat mencakup berbagai aspek yang sangat luas, mulai dari aspek teknis seperti keamanan infrastruktur jaringan dan perimeter, keamanan aplikasi dan basis data, keamanan sistem operasi dan perangkat, hingga aspek administratif dan manajerial seperti kebijakan keamanan informasi, prosedur manajemen akses, pelatihan kesadaran keamanan untuk karyawan, perencanaan keamanan, dan manajemen risiko keamanan. Audit keamanan dapat difokuskan pada area tertentu yang dianggap memiliki risiko tinggi bagi organisasi, atau mencakup seluruh sistem informasi organisasi secara menyeluruh dan komprehensif, tergantung pada kebutuhan bisnis, prioritas manajemen, dan sumber daya yang tersedia.

10.5.2 Metodologi Audit dan Teknik Pengujian Keamanan

Metodologi audit keamanan harus mengikuti standar dan best practice yang telah diakui secara internasional untuk memastikan bahwa audit dilakukan secara objektif, konsisten, dapat dipercaya, dan komprehensif dalam mengevaluasi keamanan sistem informasi. Salah satu kerangka kerja yang sangat sering digunakan adalah COBIT 5, yang menyediakan panduan komprehensif untuk mengevaluasi tata kelola teknologi informasi termasuk keamanan informasi dalam konteks bisnis organisasi (Naiho et al., 2024). Kerangka kerja lain yang sangat relevan adalah ISO/IEC 27001 yang menguraikan persyaratan lengkap untuk Sistem Manajemen Keamanan Informasi yang dapat diaudit untuk kesesuaian dan efektivitasnya.

Teknik audit keamanan yang umum digunakan antara lain pemeriksaan kebijakan keamanan dan dokumentasi lengkap, wawancara mendalam dengan personel keamanan dan pengguna sistem untuk memahami implementasi praktis, observasi langsung terhadap implementasi pengendalian keamanan untuk memverifikasi kesesuaian dengan prosedur, pengujian pengendalian keamanan melalui teknik teknis seperti *vulnerability assessment* dan *penetration testing* yang disimulasikan, serta analisis mendalam terhadap sistem log dan audit trail untuk mendeteksi anomali atau indikasi aktivitas yang mencurigakan dan tidak normal.

10.5.3 Penilaian Kematangan Keamanan Informasi Organisasi

Penilaian kematangan keamanan informasi atau *Security Maturity Assessment* adalah proses evaluasi yang komprehensif terhadap tingkat kedewasaan atau kematangan implementasi keamanan informasi dalam organisasi secara keseluruhan dan terintegrasi. Penilaian ini biasanya menggunakan model tingkat kematangan atau *maturity model* yang terdiri dari beberapa level yang terstruktur, mulai dari level paling dasar hingga level paling matang dalam implementasi keamanan informasi yang ideal. Model kematangan yang umum digunakan adalah model yang didefinisikan dalam kerangka kerja COBIT 5, NIST Cybersecurity Framework, atau model khusus lainnya yang dikembangkan sesuai dengan kebutuhan spesifik industri dan organisasi (Naiho et al., 2024).

Setiap level kematangan menggambarkan karakteristik spesifik dari implementasi keamanan informasi pada level tersebut.

Level 1 biasanya menggambarkan implementasi yang belum terstruktur atau *ad-hoc* dan reaktif terhadap insiden. Level 2 menggambarkan implementasi yang sudah terstruktur berdasarkan prosedur namun belum optimal. Level 3 menggambarkan implementasi yang sudah terstandar dan terdefinisi dengan jelas dalam organisasi. Level 4 menggambarkan implementasi yang sudah diukur dan dikelola dengan baik berdasarkan metrik kinerja. Level 5 menggambarkan implementasi yang optimal, terus-menerus ditingkatkan, dan inovatif. Dengan melakukan penilaian kematangan secara berkala, organisasi dapat mengidentifikasi area-area yang perlu ditingkatkan dan merencanakan inisiatif peningkatan keamanan yang realistis dan sesuai dengan tujuan bisnis jangka panjang organisasi.

10.5.4 Laporan Audit dan Tindak Lanjut Rekomendasi

Hasil dari audit keamanan sistem informasi harus didokumentasikan dalam bentuk laporan audit yang komprehensif, terstruktur, dan dapat dipahami oleh berbagai tingkat manajemen dalam organisasi mulai dari level operasional hingga level pimpinan tertinggi. Laporan audit harus berisi ringkasan eksekutif yang menjelaskan temuan audit secara umum dan rekomendasi utama dalam bahasa yang mudah dipahami oleh pemimpin bisnis, temuan detail yang menguraikan setiap kelemahan keamanan yang ditemukan beserta tingkat keparahannya dan dampak potensialnya, serta rekomendasi perbaikan yang spesifik, terukur, dan dapat ditindaklanjuti dengan jelas (Ajiga et al., 2024).

Rekomendasi perbaikan harus diprioritaskan berdasarkan tingkat risiko dengan fokus pada kelemahan yang memiliki risiko tertinggi dan dampak bisnis paling besar terlebih dahulu. Setiap rekomendasi harus disertai dengan estimasi biaya implementasi, sumber daya manusia yang diperlukan, jangka waktu implementasi yang realistis, dan manfaat yang diharapkan dari implementasi rekomendasi tersebut untuk bisnis. Manajemen organisasi kemudian dapat menggunakan informasi ini untuk membuat keputusan strategis mengenai alokasi sumber daya dan prioritas inisiatif keamanan informasi yang harus dijalankan. Tindak lanjut terhadap rekomendasi audit harus dipantau secara berkala dan terencana untuk memastikan bahwa rencana perbaikan diimplementasikan sesuai dengan jadwal yang telah ditetapkan dan mencapai hasil yang diharapkan dalam meningkatkan postur keamanan organisasi.

BAB 11: Sistem Pemeliharaan dan Pengembangan

Aryo Tunjung Kusumo, M.Kom.

11.1 Konsep Pemeliharaan Sistem

11.1.1 Definisi dan Tujuan Pemeliharaan Sistem

Pemeliharaan sistem merupakan serangkaian kegiatan yang dirancang untuk menjaga sistem informasi agar tetap berfungsi optimal dan dapat memberikan nilai tambah bagi organisasi. Dalam konteks manajemen siklus hidup perangkat lunak, pemeliharaan bukan sekadar perbaikan reaktif terhadap masalah yang timbul, melainkan sebuah pendekatan strategi yang mencakup perencanaan, implementasi, dan evaluasi berkelanjutan (Goyal, 2022) . Tujuan utama menjaga sistem mencakup memastikan ketersediaan layanan tanpa gangguan, meningkatkan kinerja sistem, memperpanjang umur sistem, dan mendukung adaptasi terhadap perubahan kebutuhan bisnis.

Di era transformasi digital, sistem pemeliharaan telah berkembang dari pendekatan tradisional yang bersifat perbaikan darurat menjadi strategi proaktif yang mengintegrasikan analitik prediktif dan manajemen aset yang terukur. Konsep pemeliharaan modern yang mengarahkan organisasi tidak hanya untuk menanggapi permasalahan yang ada, namun juga mengantisipasi dan

mencegah kemungkinan kegagalan sebelum hal tersebut berdampak pada operasional bisnis. Dengan mengimplementasikan sistem pemeliharaan yang terencana dengan baik, organisasi dapat mengurangi biaya operasional hingga tiga puluh persen dan meningkatkan efisiensi produksi secara signifikan.

11.1.2 Prinsip-Prinsip Dasar Pemeliharaan

Prinsip-prinsip dasar pemeliharaan sistem dibangun atas kekurangan, efisiensi, dan transparansi dalam pengelolaan aset informasi teknologi. Pertama, prinsip keseimbangan antara biaya dan manfaat mengutamakan keputusan pemeliharaan yang didasarkan pada analisis mendalam terhadap dampak finansial dan operasional. Kedua, prinsip pencegahan menempatkan upaya proaktif di atas respons reaktif, dengan fokus pada mengidentifikasi dini potensi masalah melalui pemantauan berkelanjutan dan diagnostik kondisi sistem. Ketiga, prinsip-prinsip yang komprehensif memastikan bahwa setiap keputusan pemeliharaan, perubahan sistem, dan hasil evaluasi dicatat dengan jelas untuk memfasilitasi pembelajaran organisasi dan pengambilan keputusan yang lebih baik di masa mendatang.

Prinsip lainnya yang sama pentingnya adalah integrasi antara berbagai fungsi operasional, di mana pemeliharaan sistem harus berkoordinasi dengan departemen pengembangan, keamanan siber, dan manajemen perubahan untuk memastikan kohesi dalam strategi sistem pengelolaan secara keseluruhan. Prinsip implementasi-prinsip ini memerlukan komitmen organisasi terhadap investasi dalam pemantauan infrastruktur, pengembangan sumber daya

manusia, dan penerapan teknologi pendukung yang memungkinkan pengumpulan dan analisis data secara real-time untuk mendukung pengambilan keputusan yang lebih terinformasi.

11.1.3 Pentingnya Pemeliharaan Sistem dalam Bisnis

Sistem pemeliharaan memiliki peran penting dalam keberlangsungan operasional bisnis modern, terutama dalam menghadapi kompleksitas infrastruktur teknologi yang semakin tinggi. Sistem yang tidak dipelihara dengan baik cenderung mengalami degradasi kinerja, meningkatkan risiko keamanan, dan pada akhirnya dapat menyebabkan hilangnya data serta gangguan layanan yang merugikan reputasi dan keuangan organisasi. Studi empiris menunjukkan bahwa organisasi yang menerapkan strategi pemeliharaan yang matang mengalami peningkatan sistem ketersediaan, pengurangan waktu henti produksi, dan peningkatan kepuasan pelanggan yang signifikan.

Dalam konteks manajemen risiko, pemeliharaan sistem merupakan investasi preventif yang mengurangi kemungkinan terjadinya kejadian kritis yang dapat mengganggu kelangsungan bisnis. Selain itu, pemeliharaan yang efektif memungkinkan organisasi untuk tetap mematuhi regulasi industri yang semakin ketat, mendukung transformasi digital yang lebih efektif, dan menciptakan lingkungan yang mendukung untuk inovasi berkelanjutan. Oleh karena itu, sistem pemeliharaan bukan sekedar fungsi teknis operasional, melainkan strategi bisnis yang fundamental untuk mencapai keunggulan kompetitif dan pertumbuhan jangka panjang.

11.1.4 Siklus Hidup Pemeliharaan Sistem

Siklus hidup pemeliharaan sistem mengikuti pola yang terstruktur, dimulai dari fase perencanaan, implementasi, pemantauan, evaluasi, dan perbaikan berkelanjutan. Tahap perencanaan meliputi pemeliharaan kebutuhan pemeliharaan berdasarkan analisis kondisi sistem, penilaian risiko, dan proyeksi kebutuhan bisnis ke depan. Fase implementasi mencakup pelaksanaan aktivitas pemeliharaan sesuai rencana yang telah ditetapkan, baik dalam bentuk pembaruan rutin, perbaikan darurat, maupun upgrade sistem yang terencana. Fase monitoring dilakukan secara berkelanjutan untuk mendeteksi setiap perubahan status sistem yang dapat mengindikasikan potensi masalah atau penurunan kinerja.

Fase evaluasi dilakukan secara berkala untuk mengukur efektivitas strategi pemeliharaan yang telah diimplementasikan, mengidentifikasi kesenjangan antara target dan realisasi, serta mengumpulkan pembelajaran dari kejadian atau masalah yang terjadi selama periode pemeliharaan. Terakhir, tahap perbaikan berkelanjutan memastikan bahwa proses pemeliharaan secara terus-menerus disempurnakan berdasarkan data empiris, umpan balik pemangku kepentingan, dan perkembangan praktik terbaik industri. Siklus ini bersifat berulang dan berkelanjutan, menciptakan loop pembelajaran yang memungkinkan organisasi untuk terus meningkatkan kematangan melakukan praktik pemeliharaan sistem mereka sesuai dengan dinamika bisnis dan teknologi yang terus berkembang.

11.2 Jenis-Jenis Sistem Pemeliharaan

11.2.1 Pemeliharaan Korektif

Pemeliharaan korektif merupakan jenis pemeliharaan yang dilakukan untuk memperbaiki sistem yang mengalami kerusakan atau kegagalan fungsional. Jenis pemeliharaan ini bersifat reaktif, artinya dilakukan setelah masalah terjadi dan berdampak pada operasional sistem. Meskipun bersifat darurat, pemeliharaan korektif tetap memerlukan pendekatan yang terstruktur dan didukung oleh prosedur yang jelas agar perbaikan dapat dilakukan dengan cepat dan efektif, meminimalkan durasi downtime sistem, dan mencegah terjadinya kerusakan lebih lanjut.

Dalam melakukan pemeliharaan korektif, diperlukan tim yang mempelajari dan memiliki pemahaman mendalam tentang arsitektur sistem untuk dapat melakukan diagnosis dengan cepat terhadap masalah. Dokumentasi yang baik mengenai kejadian kejadian, langkah-langkah perbaikan yang diambil, dan hasil akhir dari perbaikan sangat penting untuk mendukung proses pembelajaran organisasi dan mencegah terjadinya masalah serupa di masa depan. Meskipun pemeliharaan korektif seringkali menimbulkan biaya tinggi karena bersifat darurat dan dapat mengganggu operasional, jenis pemeliharaan ini tetap diperlukan sebagai bagian dari strategi pengelolaan risiko yang komprehensif.

11.2.2 Pemeliharaan Preventif

Pemeliharaan preventif adalah jenis pemeliharaan yang dilakukan secara terencana dan berkala dengan tujuan mencegah

terjadinya kerusakan atau kegagalan sistem sebelum masalah tersebut benar-benar muncul. Jenis pemeliharaan ini bersifat proaktif dan merupakan pendekatan yang lebih efisien dibandingkan dengan pemeliharaan korektif dalam jangka panjang, karena dapat mengurangi kemungkinan terjadinya downtime yang tidak terduga. Pemeliharaan preventif biasanya diadakan pada waktu-waktu ketika sistem memiliki beban kerja yang minimal atau pada periode yang telah disepakati bersama dengan pemangku kepentingan terkait untuk meminimalkan dampak terhadap operasional bisnis.

Kegiatan pemeliharaan preventif meliputi pemeriksaan berkala terhadap kondisi komponen sistem, pembersihan data yang tidak terpakai, optimalisasi kinerja, pembaruan perangkat lunak untuk keperluan keamanan, dan penggantian komponen yang sudah mencapai umur operasionalnya. Strategi pemeliharaan preventif yang efektif memerlukan pemahaman yang mendalam tentang karakteristik sistem, pola kegagalan historis, dan rekomendasi dari vendor atau produsen sistem. Dengan menerapkan pemeliharaan preventif secara konsisten, organisasi dapat mengurangi risiko kerusakan sistem yang tidak terduga, meningkatkan keandalan dan ketersediaan sistem, serta mengoptimalkan total biaya kepemilikan dalam jangka panjang.

11.2.3 Pemeliharaan Prediktif

Pemeliharaan prediktif adalah pendekatan inovatif yang memanfaatkan teknologi pemantauan canggih dan data analitik untuk memprediksi kapan komponen sistem kemungkinan akan mengalami kegagalan, sehingga memungkinkan waktu

pemeliharaan untuk melakukan intervensi pada waktu yang optimal (Goyal, 2022) . Jenis pemeliharaan ini memanfaatkan sensor-sensor yang terpasang pada sistem untuk mengumpulkan data secara real-time mengenai kondisi operasional, kinematik, suhu, dan parameter teknis lainnya, yang kemudian diterapkan menggunakan algoritma pembelajaran mesin untuk mengidentifikasi pola yang mengindikasikan potensi kegagalan.

Keuntungan utama dari pemeliharaan prediktif adalah mampu mendeteksi masalah pada tahap awal sebelum berdampak pada kinerja sistem, memungkinkan perencanaan pemeliharaan yang lebih presisi, mengurangi downtime yang tidak terduga, dan mengoptimalkan alokasi sumber daya pemeliharaan. Implementasi pemeliharaan prediktif memerlukan investasi yang signifikan dalam pemantauan infrastruktur, keahlian pengembangan dalam analitik data, dan integrasi dengan sistem manajemen aset yang terintegrasi. Namun, laba atas investasi dari pendekatan ini umumnya sangat positif dalam jangka panjang, terutama untuk sistem-sistem kritis yang memiliki dampak operasional tinggi terhadap bisnis.

11.2.4 Pemeliharaan Adaptif

Pemeliharaan adaptif Merujuk pada jenis pemeliharaan yang menyesuaikan strategi dan frekuensinya berdasarkan perubahan kondisi operasional, karakteristik beban kerja, dan perkembangan teknologi yang relevan. Jenis pemeliharaan ini mengakui bahwa setiap sistem memiliki karakteristik unik dan kebutuhan pemeliharaan yang dapat berubah seiring dengan perkembangan bisnis dan lingkungan operasionalnya. Pemeliharaan adaptif

memerlukan perencanaan dan eksekusi, serta kemampuan untuk melakukan fine-tuning terhadap strategi pemeliharaan berdasarkan umpan balik real-time dan pembelajaran dari pengalaman operasional.

Implementasi pemeliharaan adaptif melibatkan mekanisme pengembangan untuk terus-menerus menyebarkan efektivitas strategi pemeliharaan yang sedang berjalan, mengidentifikasi area-area yang memerlukan penyesuaian, dan melakukan iterasi perbaikan secara sistematis. Pendekatan ini memberikan fleksibilitas yang diperlukan organisasi untuk merespons perubahan kebutuhan bisnis dengan cepat, mengadopsi teknologi baru yang lebih efisien, dan mengoptimalkan keseimbangan antara biaya, risiko, dan kinerja dalam sistem manajemen. Oleh karena itu, pemeliharaan adaptif merupakan pendekatan yang sangat relevan untuk organisasi yang beroperasi dalam lingkungan bisnis yang dinamis dan cepat berubah.

11.3 Sistem Pemantauan

11.3.1 Konsep dan Tujuan Pemantauan

Sistem pemantauan adalah proses berkelanjutan untuk mengamati, mengukur, dan menganalisis kinerja sistem secara real-time atau berkala untuk memastikan bahwa sistem beroperasi sesuai dengan target yang telah ditetapkan dan untuk mendeteksi setiap anomali yang memerlukan intervensi. Tujuan utama dari sistem pemantauan adalah untuk mempercepat pengambilan keputusan yang terinformasi dan tepat waktu mengenai kondisi sistem,

mengidentifikasi kemacetan atau area-area yang memerlukan optimasi, serta memberikan peringatan dini terhadap potensi masalah sebelum berdampak pada operasional bisnis (Valencia & Pinzón, 2021) . Pemantauan yang efektif memberikan visibilitas yang komprehensif terhadap semua aspek kinerja sistem, dari metrik teknis seperti penggunaan CPU, pemanfaatan memori, latensi jaringan, hingga metrik bisnis seperti throughput transaksi dan kepuasan pelanggan.

Implementasi sistem pemantauan yang matang memerlukan pemilihan metrik yang tepat, penetapan ambang batas atau pemicu peringatan yang realistis, dan dasbor pengembangan yang mudah digunakan untuk memfasilitasi interpretasi data oleh berbagai pemangku kepentingan. Selain itu, pemantauan harus dilengkapi dengan sistem peringatan yang dapat memberitahukan waktu operasional secara otomatis ketika ada kondisi yang memerlukan perhatian, memungkinkan respons yang cepat dan efektif terhadap potensi masalah.

11.3.2 Pemantauan Infrastruktur dan Peralatan

Pemantauan infrastruktur yang kuat memerlukan kombinasi dari berbagai komponen, termasuk sensor dan probe yang terpasang pada sistem untuk mengumpulkan data, pengumpulan platform yang mengagregasi data dari berbagai sumber, lapisan penyimpanan yang mampu menangani data volume yang besar, dan mesin analitik yang memproses data untuk menghasilkan wawasan yang dapat ditindaklanjuti. Alat pemantauan modern mengintegrasikan kemampuan untuk pengumpulan data real-time, analisis deret waktu,

deteksi anomali, dan visualisasi dalam satu platform yang koheren (Valencia & Pinzón, 2021) . Pemilihan alat monitoring yang tepat harus mempertimbangkan skalabilitas, keandalan, kemudahan integrasi dengan sistem yang ada, serta total biaya kepemilikan dalam jangka panjang.

Pemantauan infrastruktur yang baik juga harus memiliki mekanisme redundansi dan failover untuk memastikan kontinuitas proses pemantauan bahkan ketika terjadi kegagalan pada komponen pemantauan itu sendiri. Selain itu, pertimbangan keamanan menjadi penting karena infrastruktur pemantauan mengumpulkan dan menyimpan data sensitif mengenai sistem operasional, sehingga diperlukan kontrol akses, enkripsi, dan audit logging yang tepat untuk menjaga integritas dan kerahasiaan pemantauan data.

11.3.3 Pemantauan Metrik dan KPI

Pemilihan metrik yang tepat merupakan aspek kritis dalam penerapan sistem pemantauan yang efektif. Metrik dapat dibagi menjadi beberapa kategori, termasuk metrik infrastruktur yang mengukur kesehatan komponen fisik dan sistem virtual, metrik aplikasi yang mengukur kinerja aplikasi, dan metrik bisnis yang mengukur dampak sistem terhadap tujuan bisnis organisasi. Key Performance Indicators (KPI) adalah subset dari metrik yang dipilih secara strategis untuk memberikan visibilitas terhadap aspek-aspek yang paling penting bagi organisasi dalam mencapai tujuan bisnisnya.

Dalam konteks Maintenance 4.0, pemantauan KPI harus mencakup ketersediaan, keandalan, waktu respons, dan efisiensi

biaya. Penetapan target untuk setiap KPI harus realistis, dapat dicapai, namun tetap menantang untuk mendorong perbaikan berkelanjutan (Naik & Kirkire, 2025) . Tinjauan berkala terhadap kinerja KPI dan benchmarking terhadap standar industri membantu organisasi untuk memastikan bahwa target yang ditetapkan tetap relevan dan selaras dengan strategi bisnis yang berkembang.

11.3.4 Deteksi dan Peringatan Insiden

Sistem deteksi dan peringatan yang efektif memerlukan kombinasi logika berbasis aturan, analisis statistik, dan pembelajaran mesin untuk dapat mengidentifikasi berbagai jenis anomali dengan akurasi tinggi dan tingkat positif palsu yang minimal. Deteksi berbasis aturan menggunakan aturan ambang batas dan korelasi yang telah didefinisikan sebelumnya untuk mengidentifikasi kondisi-kondisi yang diketahui berisiko tinggi, sementara pendekatan statistik dan pembelajaran mesin dapat mendeteksi anomali yang lebih halus dan baru berdasarkan deviasi dari pola operasi normal.

Strategi peringatan harus dirancang untuk memastikan bahwa peringatan hanya dikirimkan ketika ada kondisi yang benar-benar memerlukan tindakan dari waktu operasional, menghindari kelelahan peringatan yang dapat menurunkan efektivitas pemantauan. Tingkat keparahan harus ditetapkan dengan jelas, dan mekanisme perutean peringatan harus memastikan bahwa peringatan sampai ke orang atau tim yang tepat berdasarkan sifat dan tingkat keparahan masalah yang terdeteksi.

11.4 Peningkatan dan Pengembangan Sistem

11.4.1 Sistem Peningkatan Strategi

Upgrade sistem merupakan kegiatan untuk meningkatkan versi atau rilis dari perangkat lunak, perangkat keras, atau komponen sistem secara keseluruhan dengan tujuan untuk memanfaatkan fitur-fitur baru, meningkatkan postur keamanan, meningkatkan kinerja, atau untuk menjaga kompatibilitas dengan teknologi atau standar yang berkembang. Peningkatan strategi yang sukses memerlukan perencanaan yang matang, penilaian mendalam terhadap risiko dan manfaat, serta pelaksanaan yang hati-hati untuk meminimalkan gangguan terhadap operasional bisnis. Dalam pendekatan Maintenance 4.0, upgrade bukan hanya tentang instalasi versi terbaru, tetapi tentang evolusi strategis dari sistem yang sejalan dengan roadmap bisnis dan teknologi organisasi (Naik & Kirkire, 2025) .

Strategi peningkatan harus mempertimbangkan berbagai faktor seperti kompatibilitas ke belakang, interoperabilitas dengan sistem-sistem terkait, persyaratan pelatihan untuk pengguna dan tim TI, dan rencana darurat untuk menangani potensi masalah yang mungkin timbul. Pendekatan peningkatan bertahap, dimana peningkatan dilakukan secara bertahap terhadap subset dari populasi pengguna sebelum peluncuran penuh, dapat mengurangi risiko dan memungkinkan pembelajaran dari fase awal untuk diterapkan pada fase berikutnya.

11.4.2 Perencanaan dan Penilaian Risiko

Peningkatan perencanaan memerlukan pendekatan yang komprehensif dan melibatkan pemangku kepentingan dari berbagai fungsi organisasi untuk memastikan bahwa semua perspektif dipertimbangkan. Penilaian risiko harus mengidentifikasi potensi risiko dari peningkatan, baik risiko teknis seperti masalah kompatibilitas, kehilangan data, penurunan kinerja, maupun risiko bisnis seperti gangguan terhadap operasional, dampak terhadap kepuasan pelanggan, dan implikasi finansial. Strategi mitigasi harus dikembangkan untuk setiap risiko yang teridentifikasi, dengan kepemilikan yang jelas dan rencana kontinjensi (Vilsmeier & Salehi, 2025) .

Strategi pengujian merupakan komponen penting dari perencanaan, di mana upgrade harus diuji secara menyeluruh dalam lingkungan yang representatif terhadap lingkungan produksi untuk memvalidasi bahwa upgrade akan memberikan manfaat yang diharapkan tanpa menimbulkan masalah yang tidak terduga. Pengujian beban, pengujian kompatibilitas, pengujian keamanan, dan pengujian penerimaan pengguna merupakan jenis-jenis pengujian yang biasanya diperlukan sebelum penerapan produksi.

11.4.3 Implementasi Peningkatan

Implementasi upgrade memerlukan koordinasi yang ketat antara berbagai tim, komunikasi yang jelas kepada pengguna mengenai jadwal dan dampak yang diharapkan, serta pemantauan yang intensif selama proses upgrade untuk dapat mendeteksi dan merespons masalah dengan cepat. Prosedur rollback harus

didefinisikan dengan jelas sebelum upgrade dilakukan, memastikan bahwa jika ada masalah yang tidak dapat diatasi, sistem dapat dikembalikan ke keadaan sebelumnya dengan gangguan minimal.

Implementasi upgrade modern sering menggunakan teknik seperti blue-green deployment atau canary deployment untuk mengurangi risiko dan memungkinkan quick rollback jika diperlukan. Komunikasi kepada semua pemangku kepentingan harus dilakukan secara proaktif sebelum, selama, dan setelah peningkatan untuk memastikan bahwa semua orang memahami apa yang terjadi dan bagaimana hal tersebut dapat mempengaruhi pekerjaan mereka.

11.4.4 Validasi dan Optimasi Pasca-Peningkatan

Setelah upgrade selesai dilakukan, validasi harus dilakukan secara komprehensif untuk memastikan bahwa semua komponen berfungsi sebagaimana diharapkan dan bahwa manfaat yang direncanakan dari upgrade telah tercapai. Baseline kinerja harus diukur dan dibandingkan dengan ekspektasi untuk mengidentifikasi apakah ada masalah yang memerlukan optimalisasi lebih lanjut. Umpan balik pengguna harus dikumpulkan dan dianalisis untuk mengidentifikasi apakah ada kesenjangan antara harapan pengguna dan kenyataan, dan untuk mengidentifikasi peluang perbaikan yang dapat dilakukan dalam rilis berikutnya.

Optimasi dapat mencakup penyesuaian konfigurasi, penyesuaian terhadap alokasi sumber daya, atau implementasi terhadap solusi yang telah diidentifikasi selama tahap pengujian. Dokumentasi harus diperbarui untuk merefleksikan perubahan yang

telah dilakukan, memastikan bahwa tim di masa depan memiliki informasi yang akurat mengenai keadaan sistem saat ini.

11.5 Dokumentasi Perubahan Sistem

11.5.1 Pentingnya Sistem Dokumentasi

Dokumentasi sistem merupakan komponen fundamental dalam sistem manajemen yang efektif, menyediakan catatan yang komprehensif mengenai bagaimana sistem dirancang, diimplementasikan, dioperasikan, dipelihara, dan setiap perubahan yang dilakukan terhadapnya. Dokumentasi yang baik memfasilitasi transfer pengetahuan antar anggota tim, memudahkan orientasi pengguna dan pengembang baru, mendukung pemecahan masalah dan penyelesaian masalah yang lebih cepat, serta memberikan jejak audit yang diperlukan untuk tujuan kepatuhan dan tata kelola (Vilsmeier & Salehi, 2025) . Dalam era di mana pergantian karyawan adalah normal, dokumentasi yang baik adalah aset berharga yang memastikan bahwa pengetahuan tidak hilang ketika anggota tim meninggalkan organisasi.

Dokumentasi juga memiliki nilai yang signifikan dalam konteks pemulihan bencana dan kelangsungan bisnis, di mana dokumentasi yang akurat dapat mempercepat waktu pemulihan ketika terjadi kejadian yang serius. Lebih dari itu, dokumentasi yang baik mendukung perbaikan proses dan inovasi, karena memberikan landasan yang kuat untuk analisis dan pemahaman terhadap sistem

keadaan saat ini yang dapat memfasilitasi identifikasi terhadap peluang perbaikan.

11.5.2 Jenis-Jenis Dokumentasi Perubahan

Dokumentasi perubahan sistem dapat diklasifikasikan menjadi beberapa jenis berdasarkan tujuan dan audiensnya. Dokumentasi teknis meliputi diagram arsitektur, spesifikasi desain, dokumentasi konfigurasi, dan prosedur operasi standar yang membantu tim teknis dalam memahami dan mengelola sistem. Dokumentasi operasional mencakup runbook, panduan pemecahan masalah, dan prosedur respons insiden yang membantu tim operasi dalam menjalankan operasi sehari-hari. Dokumentasi pengguna mencakup panduan pengguna, tutorial, dan FAQ yang membantu pengguna akhir dalam menggunakan sistem secara efektif. Dokumentasi perubahan mencakup permintaan perubahan, log perubahan, dan penilaian dampak yang memberikan catatan lengkap mengenai setiap perubahan yang dilakukan terhadap sistem.

Dokumentasi peraturan dan kepatuhan mencakup kebijakan, prosedur, dan bukti mengenai kepatuhan terhadap persyaratan peraturan yang berlaku. Setiap jenis dokumentasi memiliki format, tingkat detail, dan frekuensi pembaruan yang berbeda-beda, namun semua harus dimasukkan ke dalam sistem manajemen pengetahuan yang koheren untuk memastikan aksesibilitas dan keutuhan dokumentasi.

11.5.3 Manajemen Perubahan dan Kontrol Versi

Manajemen perubahan merupakan proses formal untuk mengelola perubahan-perubahan terhadap sistem dengan cara yang

terstruktur, terkendali, dan terdokumentasi. Tujuan dari manajemen perubahan adalah untuk memastikan bahwa perubahan dilakukan dengan kesadaran penuh terhadap dampak-nya, bahwa perubahan dievaluasi dan disetujui oleh pemangku kepentingan yang tepat sebelum implementasi, dan bahwa semua pihak yang terkena dampak oleh perubahan diinformasikan secara tepat waktu (Vilsmeier & Salehi, 2025) . Kontrol versi adalah mekanisme untuk melacak semua versi dari dokumentasi dan kode, memungkinkan rollback ke versi sebelumnya jika diperlukan, dan memfasilitasi kolaborasi antar anggota tim.

Implementasi manajemen perubahan yang efektif memerlukan struktur tata kelola yang jelas, proses yang terdefinisi dengan baik, dan alat yang mendukung alur kerja mulai dari permintaan perubahan hingga implementasi dan penutupan. Dewan penasihat perubahan (CAB) yang terdiri dari perwakilan berbagai fungsi dapat membantu dalam evaluasi dan penentuan prioritas permintaan perubahan untuk memastikan bahwa perubahan yang dilakukan selaras dengan prioritas bisnis dan tidak memiliki konsekuensi yang tidak diinginkan.

11.5.4 Alat dan Dokumentasi Praktik Terbaik

Dokumentasi implementasi yang efektif memerlukan penggunaan alat yang tepat dan adopsi dari praktik terbaik yang telah terbukti berhasil di berbagai organisasi. Alat dokumentasi dapat berkisar dari wiki sederhana dan platform kolaborasi hingga sistem manajemen pengetahuan yang canggih dengan kemampuan pencarian, penandaan, dan kontrol versi tingkat lanjut. Pilihan alat

harus didasarkan pada kebutuhan spesifik organisasi, dengan mempertimbangkan faktor-faktor seperti kemudahan penggunaan, skalabilitas, kemampuan integrasi dengan sistem yang ada, dan total biaya kepemilikan.

Praktik terbaik dalam dokumentasi mencakup pemeliharaan satu sumber kebenaran untuk menghindari inkonsistensi dan informasi yang ketinggalan jaman, menetapkan kepemilikan dan akuntabilitas yang jelas untuk menjaga dokumentasi tetap terkini, menetapkan pedoman format dan struktur untuk memastikan konsistensi, dan siklus peninjauan rutin untuk mengidentifikasi dan menghapus dokumentasi usang. Alat otomasi dapat dimanfaatkan untuk menghasilkan dokumentasi dari kode atau konfigurasi sistem, mengurangi upaya manual dan meningkatkan akurasi. Budaya dokumentasi yang solid, di mana anggota tim menyadari pentingnya dokumentasi dan memiliki insentif untuk menjaga kualitas dokumentasi, adalah kunci keberlanjutan upaya dokumentasi dalam jangka panjang.

BAB 12: Informasi Sistem Tata Kelola

Fajar Agustini, M.Kom.

12.1 Konsep Integrasi Sistem Informasi

12.1.1 Definisi dan Pentingnya Sistem Integrasi

Integrasi sistem informasi merupakan proses penggabungan berbagai aplikasi, platform, dan infrastruktur teknologi informasi menjadi satu ekosistem yang terkoneksi, tersinkronisasi, dan berfungsi secara harmonis untuk mendukung seluruh proses bisnis organisasi. Dalam era transformasi digital yang sedang berlangsung, sistem integrasi bukan lagi pilihan tetapi kebutuhan strategi bagi organisasi yang ingin tetap kompetitif dan relevan. Setiap organisasi modern yang menghadapi tantangan kompleks dimana sistem warisan dan aplikasi-aplikasi modern harus bekerja sama untuk menghasilkan informasi yang akurat, tepat waktu, dan dapat diandalkan. Integrasi sistem informasi tidak hanya mencakup aspek teknis seperti penghubungan berbagai sistem melalui *antarmuka* atau *middleware* , tetapi juga aspek organisasi seperti sinkronisasi proses bisnis, keselarasan data, dan penyelarasan tujuan antar unit organisasi (Santosa & Mulyana, 2023).

Pentingnya sistem integrasi terletak pada kemampuannya untuk menghilangkan fragmentasi data yang sering menjadi kendala

utama dalam operasional bisnis modern. Ketika sistem berjalan secara terpisah, organisasi mengalami duplikasi data, inkonsistensi informasi, keterlambatan pelaporan, dan kesulitan dalam mengambil keputusan strategi yang berbasis data secara komprehensif. Dengan mengintegrasikan sistem, organisasi dapat memastikan bahwa informasi yang sama dapat diakses oleh semua pihak yang membutuhkan dalam format yang konsisten dan waktu yang tepat. Hal ini meningkatkan efisiensi operasional, mengurangi kesalahan manual, mempercepat respons waktu terhadap kebutuhan bisnis, dan memfasilitasi pengambilan keputusan yang lebih tepat dan strategis di semua tingkat organisasi.

12.1.2 Arsitektur Sistem Integrasi

Arsitektur integrasi sistem mengacu pada desain keseluruhan dan struktur teknis yang memungkinkan berbagai komponen sistem bekerja secara sinergis dan efisien. Pendekatan *konektivitas yang dipimpin API* telah menjadi paradigma modern yang paling banyak diadopsi dalam membangun arsitektur integrasi yang terukur, dapat dipelihara, dan fleksibel untuk organisasi di berbagai industri. Arsitektur berbasis *API* mengorganisir integrasi dalam tiga lapisan yang saling melengkapi: *System APIs* yang menyediakan akses terhadap data dan fungsionalitas sistem inti organisasi, *Process APIs* yang mengorkestrasi logika bisnis dan proses lintas sistem, serta *Experience APIs* yang memfasilitasi interaksi langsung dengan pengguna akhir melalui berbagai *touchpoint* . Setiap lapisan dirancang dengan tujuan spesifik yang jelas dan mampu digunakan kembali dalam berbagai konteks aplikasi yang berbeda.

Selain pendekatan berbasis *API*, teknologi *middleware* dan *Integration Platform as a Service* (*iPaaS*) juga memainkan peran penting dalam arsitektur kontemporer. *Middleware* berfungsi sebagai perantara lapisan yang mengkoordinasikan komunikasi antara berbagai sistem, mengubah format data, mengelola keamanan, dan menangani *pemulihan kesalahan*. *iPaaS* menyediakan platform yang berpusat di cloud yang mengintegrasikan fungsionalitas transformasi data, perutean pesan, integrasi manajemen, dan *pemantauan* dalam satu solusi yang dapat diakses melalui web. Arsitektur integrasi yang dirancang dengan baik harus mampu mengatasi tantangan mendasar seperti duplikasi data yang tidak perlu, keterlambatan pelaporan yang signifikan, dan inkonsistensi informasi yang menyulitkan dalam pengambilan keputusan strategi.

12.1.3 Pola dan Strategi Integrasi

Organisasi dapat menerapkan berbagai pola integrasi yang berbeda tergantung pada konteks spesifik, kompleksitas lanskap teknologi, dan kebutuhan bisnis yang ingin dipenuhi. Pola *point-to-point* merupakan pendekatan tradisional dimana setiap sistem terhubung langsung dengan sistem lain, namun pendekatan ini memiliki keterbatasan yang signifikan dalam hal skalabilitas karena jumlah koneksi yang diperlukan tumbuh secara eksponensial seiring bertambahnya jumlah sistem. Pola *Enterprise Service Bus* (*ESB*) menawarkan peningkatan transmisi dengan mengarahkan semua komunikasi melalui bus yang mengatur perutean, transformasi data, dan *orkestrasi* proses bisnis. *Arsitektur* pola layanan mikro

memberikan tingkat otonomi yang lebih tinggi kepada setiap layanan sambil tetap mempertahankan kemampuan interoperabilitas, memungkinkan tim untuk mengembangkan, melakukan *penerapan*, dan melakukan *penskalaan* secara independen untuk setiap layanan.

Strategi migrasi juga merupakan aspek penting dalam sistem integrasi perencanaan, terutama ketika organisasi perlu memodernisasi sistem warisan tanpa mengganggu operasional bisnis yang berkelanjutan. Strategi *pencekik pola* telah terbukti sangat efektif dalam konteks ini, yaitu sistem baru secara bertahap menggantikan fungsi sistem lama sampai pada akhirnya sistem lama dapat dihentikan sepenuhnya. Organisasi juga dapat mempertimbangkan strategi *lift and shift* untuk migrasi cepat ke cloud, atau strategi *refactoring* yang lebih mendalam untuk memanfaatkan kemampuan teknologi cloud secara maksimal. Pemilihan strategi integrasi yang tepat memerlukan analisis mendalam terhadap keseimbangan antara kecepatan implementasi, biaya investasi, utilitas sistem, dan kemudahan pemeliharaan dalam jangka panjang.

12.1.4 Manfaat dan Tantangan Integrasi

Manfaat utama dari sistem integrasi informasi mencakup peningkatan yang signifikan dalam efisiensi operasional melalui pengurangan redundansi data dan penyederhanaan alur kerja bisnis. Sistem integrasi memungkinkan organisasi untuk meningkatkan kualitas informasi yang tersedia, mengurangi respons waktu terhadap kebutuhan bisnis yang mendesak, dan menciptakan fondasi yang solid untuk pengambilan keputusan strategi yang berbasis data.

Organisasi yang berhasil mengintegrasikan sistemnya secara komprehensif melaporkan peningkatan signifikan dalam produktivitas karyawan karena berkurangnya tugas manual yang berulang, pengurangan kesalahan data yang berasal dari entri manual yang tidak konsisten, dan peningkatan kepuasan pelanggan melalui layanan yang lebih responsif (Palade et al., 2024).

Meskipun manfaat yang ditawarkan sangat menarik, sistem informasi integrasi juga menghadapi berbagai tantangan teknis, organisasi, dan finansial yang perlu diatasi dengan strategi yang cermat dan terencana. Tantangan teknis mencakup kompleksitas dalam mencapai kompatibilitas antar sistem yang beragam, manajemen kualitas data di lingkungan terdistribusi, dan implementasi keamanan informasi yang konsisten di seluruh ekosistem sistem yang terintegrasi. Tantangan organisasi meliputi perubahan mendasar dalam proses bisnis yang sering menimbulkan resistensi dari karyawan yang terbiasa dengan cara kerja lama, kebutuhan untuk melatih ulang karyawan agar memahami sistem baru, dan pembangunan organisasi untuk mendukung sistem operasi terintegrasi. Tantangan finansial mencakup investasi modal awal yang besar untuk infrastruktur dan implementasi, biaya operasional keberlanjutan untuk pemeliharaan dan perbaikan sistem, dan terlibat dalam memproyeksikan *laba atas investasi* dari inisiatif sistem integrasi.

12.2 Arsitektur Perusahaan

12.2.1 Konsep dan Definisi Arsitektur Enterprise

Enterprise Architecture (*EA*) merupakan kerangka kerja komprehensif yang mendefinisikan, merancang, dan mengorganisir struktur operasional, proses bisnis, sistem informasi, infrastruktur teknologi, dan sumber daya organisasi secara holistik untuk mencapai tujuan strategis jangka panjang. *Arsitektur Perusahaan* berfungsi sebagai cetak biru strategi yang komprehensif yang memandu perencanaan, pengembangan, implementasi, dan pemantauan berkelanjutan dari infrastruktur teknologi organisasi dengan memastikan bahwa setiap investasi teknologi informasi memberikan kontribusi nyata dan terukur terhadap pencapaian tujuan bisnis organisasi (Hidayat et al., 2026). Pendekatan *EA* yang matang dan teruji memastikan bahwa investasi teknologi informasi yang seringkali bernilai jutaan hingga miliaran rupiah sejalan dengan strategi organisasi bisnis dan memberikan nilai jangka panjang yang berkelanjutan dan terukur. *Arsitektur Perusahaan* tidak hanya mengenai desain teknis infrastruktur teknologi semata, tetapi juga menciptakan keselarasan yang kuat, konsisten, dan berkelanjutan antara kebutuhan bisnis yang dinamis dan solusi teknologi yang inovatif dan responsif.

Konsep *EA* telah berkembang secara signifikan dari fokus yang semula sangat teknis dan bersifat *bottom-up* menjadi pendekatan yang lebih holistik dan strategis yang mempertimbangkan dengan dimensi serius manusia, proses bisnis,

dan budaya organisasi. *Arsitektur Perusahaan* yang efektif membantu organisasi dalam mencapai *keselarasan bisnis-TI* yang seringkali menjadi tantangan strategi dalam konteks transformasi digital yang kompleks. Melalui pendekatan *EA* yang sistematis dan terstruktur, organisasi dapat memahami lebih dalam lanskap teknologi yang kompleks, mengidentifikasi redundansi dan inefisiensi yang membuang sumber daya, serta merancang solusi teknologi yang terintegrasi, terukur, dan fleksibel untuk memenuhi kebutuhan bisnis yang terus berkembang.

12.2.2 Kerangka TOGAF dan ArchiMate

TOGAF (*The Open Group Architecture Framework*) merupakan kerangka *Enterprise Architecture* yang paling banyak diadopsi di seluruh dunia oleh organisasi dari berbagai ukuran dan industri, menyediakan metodologi yang terstruktur, komprehensif, dan terbukti untuk mengembangkan, mengimplementasikan, dan mengelola *Enterprise Architecture* dalam organisasi secara sistematis dan sistematis (Hidayat et al., 2026). *TOGAF Architecture Development Method* (*ADM*) menawarkan pendekatan iteratif dan inkremental yang mencakup berbagai fase yang saling terkait, dimulai dari fase *pendahuluan* yang mempersiapkan organisasi untuk proses *EA* , fase *architecture vision* yang menetapkan visi dan ruang lingkup *EA* , fase *arsitektur bisnis* yang merancang arsitektur bisnis, dan fase-fase berikutnya untuk merancang arsitektur sistem informasi dan teknologi. Kerangka kerja ini mengarahkan organisasi dalam melakukan analisis mendalam terhadap kondisi yang ada yang disebut *arsitektur*

As-Is , menentukan kondisi yang diinginkan di masa depan yang disebut *arsitektur To-Be* , dan menyusun peta jalan implementasi yang realistis dan terukur.

ArchiMate merupakan bahasa pemodelan standar yang dirancang secara khusus untuk menggambarkan, menganalisis, dan mengkomunikasikan struktur dan perilaku dari *Enterprise Architecture* kepada berbagai pemangku kepentingan dengan cara yang pemahaman dan mudah dipahami. Bahasa ini menggunakan notasi visual yang konsisten dan intuitif untuk merepresentasikan komponen bisnis, aplikasi, dan teknologi serta hubungan-hubungan kritis di antara mereka. Kombinasi yang kuat antara *TOGAF* sebagai metodologi proses dan *ArchiMate* sebagai bahasa pemodelan memberikan pendekatan organisasi yang komprehensif untuk mendokumentasikan, mengomunikasikan, dan memanfaatkan *Arsitektur Perusahaan* dalam pengambilan keputusan strategi di tingkat organisasi.

12.2.3 Dimensi dan Komponen Arsitektur Perusahaan

Arsitektur Enterprise terdiri dari empat dimensi utama yang saling terhubung, saling melengkapi, dan bersama-sama membentuk gambaran lengkap dari organisasi dan teknologinya. Dimensi bisnis (*arsitektur bisnis*) mendefinisikan dan mendeskripsikan strategi organisasi, struktur organisasi, proses bisnis utama, fungsi bisnis, dan proposisi nilai yang ditawarkan kepada pelanggan. Dimensi data (*arsitektur data*) menggambarkan struktur logistik dan fisik aset organisasi data, menggambarkan bagaimana data dikelola, disimpan, diakses, dan digunakan untuk mendukung proses bisnis. Dimensi

aplikasi (*arsitektur aplikasi*) menggambarkan sistem informasi individu dan aplikasi yang digunakan oleh organisasi, mengidentifikasi hubungan-hubungan kritis di antara mereka, dan menunjukkan bagaimana aplikasi-aplikasi tersebut secara bersama-sama mendukung fungsi dan proses bisnis (Lestari et al., 2025).

Dimensi teknologi (*teknologi arsitektur*) mencakup infrastruktur teknologi, platform komputasi, layanan cloud, jaringan komunikasi, sistem keamanan, dan komponen teknis lainnya yang memberikan landasan untuk menjalankan aplikasi dan mengelola data organisasi. Setiap dimensi *EA* memiliki komponen-komponen spesifik yang perlu dirancang dengan hati-hati, konsisten, dan selaras dengan dimensi lainnya. Integrasi yang harmonis antar dimensi keempat ini menciptakan *Enterprise Architecture* yang kohesif, logis, dan selaras dengan strategi bisnis organisasi.

12.2.4 Implementasi dan Evolusi Arsitektur Perusahaan

Implementasi *Arsitektur Perusahaan* memerlukan komitmen organisasi yang kuat, dukungan kepemimpinan yang konsisten, dan pendekatan yang terstruktur dengan *tata kelola yang jelas* , *akuntabilitas* , dan manajemen perubahan yang efektif. Organisasi harus memulai dengan penilaian mendalam terhadap kondisi yang ada, mengidentifikasi *kesenjangan* antara kondisi saat ini dan visi masa depan, mengidentifikasi peluang perbaikan, dan kemudian mengembangkan implementasi peta jalan yang realistis dengan tonggak sejarah yang jelas dan terukur serta alokasi sumber daya yang memadai. Proses implementasi *Arsitektur Perusahaan* adalah pekerjaan yang bersifat *berkelanjutan* dan

berulang, bukan proyek satu kali yang berakhir dengan penyerahan akhir. Organisasi perlu membangun kemampuan dan kematangan organisasi yang progresif dalam mengelola dan mengevolusi *EA* seiring dengan perubahan kebutuhan bisnis yang dinamis dan lanskap teknologi yang terus berkembang.

Evolusi *Enterprise Architecture* harus responsif dan tangkas terhadap perubahan lanskap teknologi yang cepat, kebutuhan bisnis yang dinamis dan seringkali tidak terduga, dan peluang baru yang ditawarkan oleh teknologi baru seperti kecerdasan buatan, komputasi awan, dan Internet of Things. Organisasi yang sukses dalam mengelola *EA* menunjukkan kemampuan untuk menciptakan keseimbangan yang sehat antara stabilitas infrastruktur yang ada dengan ketangkasan untuk mengadopsi inovasi teknologi yang bermanfaat. Struktur *tata kelola* yang jelas dengan tanggung jawab dan *akuntabilitas* yang terdefinisi dengan baik di setiap tingkat organisasi, kerangka kerja *pengambilan keputusan* yang transparan dan efisien, dan proses peninjauan berkala terhadap *EA* memastikan bahwa evolusi *EA* tetap selaras dengan strategi bisnis organisasi dan terus memberikan nilai yang berkelanjutan dan terukur kepada semua pemangku kepentingan organisasi.

12.3 Tata Kelola TI (Tata Kelola TI)

12.3.1 Definisi dan Tujuan Tata Kelola TI

Tata Kelola TI (*Tata Kelola TI*) Merujuk pada sistem keseluruhan yang komprehensif dan terintegrasi yang memastikan

penggunaan sumber daya informasi teknologi secara efektif, efisien, dan sesuai dengan standar regulasi dan *praktik terbaik* yang berlaku untuk mendukung tercapainya tujuan organisasi secara optimal. *Tata Kelola TI* menetapkan struktur organisasi yang jelas, proses-proses yang terdefinisi dengan baik, kebijakan-kebijakan yang konsisten, dan mekanisme kontrol yang efektif yang memungkinkan organisasi untuk mengelola investasi TI dalam jumlah besar, mengurangi risiko operasional dan strategi, dan memaksimalkan pembangkitan nilai dari informasi teknologi yang telah diterapkan. Tujuan utama *IT Governance* adalah menciptakan keselarasan yang kuat antara strategi bisnis organisasi dan strategi informasi teknologi, memastikan bahwa setiap keputusan investasi teknologi memberikan kontribusi nyata terhadap pencapaian tujuan bisnis organisasi, dan menciptakan lingkungan yang kondusif untuk pengambilan keputusan yang terinformasi, transparan, dan akuntabel pada tingkat kepemimpinan organisasi.

IT Governance berfungsi sebagai mekanisme penting untuk memastikan bahwa keputusan-keputusan strategi tentang teknologi informasi yang sering mencakup investasi jutaan rupiah dibuat dengan melibatkan seluruh pemangku kepentingan yang relevan dan berdasarkan informasi yang akurat, lengkap, dan komprehensif. Melalui *Tata Kelola TI* yang efektif dan diterapkan dengan baik, organisasi dapat memastikan bahwa *manajemen risiko* terhadap teknologi dilakukan secara proaktif dan sistematik, kepatuhan terhadap regulasi, standar industri, dan *praktik terbaik* terjaga dengan baik, dan nilai dari investasi TI dapat diukur dan

dioptimalkan secara berkelanjutan melalui *pemantauan* dan evaluasi yang ketat.

12.3.2 Tata Kelola TI Domain dan Praktik Terbaik

Tata Kelola TI mencakup beberapa domain utama yang saling terkait, saling melengkapi, dan bersama-sama membentuk sistem tata kelola yang komprehensif dan terintegrasi. Perencanaan strategi dan penyelarasan domain memastikan bahwa rencana pengembangan TI jangka menengah dan panjang sejalan dengan strategi bisnis organisasi dan mendukung pencapaian tujuan strategi. Pengelolaan domain risiko dan keamanan informasi memastikan bahwa aset informasi organisasi yang dilindungi secara efektif dari *ancaman* , kerentanan, dan ancaman keamanan cyber yang terus berkembang. Pengelolaan domain investasi TI dan realisasi nilai memastikan bahwa setiap investasi teknologi yang dilakukan memberikan return yang sesuai dengan ekspektasi dan menciptakan nilai bisnis yang nyata dan terukur. Pengelolaan domain sumber daya TI dan manajemen kinerja memastikan bahwa sumber daya teknologi digunakan secara optimal, anggaran dikelola secara efisien, dan metrik kinerja dipantau secara berkelanjutan untuk memastikan efektivitas operasional (Santosa & Mulyana, 2023).

Praktik terbaik dalam *Tata Kelola TI* mengacu pada pendekatan dan metodologi yang telah terbukti efektif dalam berbagai organisasi dan konteks industri. *ITIL* (*Perpustakaan Infrastruktur Teknologi Informasi*) menyediakan kerangka kerja komprehensif untuk *manajemen layanan* dan *keunggulan operasional* dalam pengelolaan layanan TI. *COBIT* (*Control*

Objectives for Information and Associated Technologies) menawarkan pedoman yang komprehensif untuk tata kelola dan manajemen TI perusahaan yang selaras dengan tujuan bisnis organisasi. *ISO/IEC 27001* memberikan standar internasional untuk manajemen keamanan informasi yang dapat diadopsi oleh organisasi di berbagai industri. Implementasi *praktik terbaik* ini membantu organisasi dalam membangun kematangan kemampuan organisasi yang progresif, meningkatkan efisiensi operasional secara signifikan, dan secara efektif mengurangi risiko informasi dan ancaman yang terkait dengan penggunaan teknologi dalam operasional organisasi.

12.3.3 Struktur Organisasi dan Mekanisme Kontrol Tata Kelola TI

Struktur organisasi yang efektif untuk *Tata Kelola TI* harus dengan jelas mendefinisikan peran, tanggung jawab, dan *garis wewenang* untuk berbagai keputusan strategi dan operasional tentang teknologi informasi di seluruh organisasi. Organisasi memerlukan komite tata kelola tingkat eksekutif yang terdiri dari representasi dari unit bisnis, unit TI, dan fungsi pendukung lainnya untuk memastikan kolaborasi dan keselarasan dalam proses *pengambilan keputusan* yang kompleks. *Chief Information Officer* (*CIO*) memiliki peran penting dalam *IT Governance*, bertugas memastikan keselarasan yang berkelanjutan antara strategi TI dan strategi bisnis organisasi, mengelola risiko TI secara proaktif, dan mengoptimalkan nilai dari investasi TI yang telah dilakukan organisasi. Struktur tata kelola yang terdefinisi dengan baik

memfasilitasi komunikasi yang lebih baik, lebih transparan, dan lebih konstruktif antar unit organisasi dan memastikan bahwa keputusan-keputusan strategi dibuat dengan melibatkan perspektif dari berbagai pemangku kepentingan yang relevan.

Mekanisme pengendalian dalam *IT Governance* mencakup kebijakan yang jelas, prosedur yang terstruktur, *mekanisme pemantauan* yang sistematis, dan indikator kinerja yang terukur yang digunakan untuk memastikan bahwa tujuan tercapai dan risiko diminimalkan secara efektif. *Pemantauan* dan pengukuran yang berkelanjutan terhadap metrik kinerja TI, seperti ketersediaan sistem yang tinggi, waktu respons yang cepat, kepuasan pengguna yang meningkat, dan status kepatuhan terhadap regulasi, memungkinkan organisasi untuk mengidentifikasi masalah dengan cepat dan mengambil tindakan perbaikan yang diperlukan sebelum masalah berkembang menjadi krisis. Audit rutin baik internal maupun eksternal terhadap sistem, proses, dan pengendalian TI memastikan bahwa kontrol yang dirancang benar-benar berfungsi efektif dalam praktik dan kepatuhan terhadap kebijakan, standar, dan regulasi terjaga dengan baik di seluruh organisasi.

12.3.4 Tantangan dan Peningkatan Tata Kelola TI

Organisasi menghadapi berbagai tantangan yang signifikan dalam mengimplementasikan dan memelihara *IT Governance* yang efektif dan memberikan nilai nyata. Salah satu tantangan utama adalah mencapai keselarasan yang berkelanjutan antara strategi bisnis yang dinamis dan strategi TI dalam konteks perubahan teknologi yang cepat dan kebutuhan bisnis yang terus berkembang

yang seringkali tidak terduga. Tantangan lainnya meliputi resistensi terhadap perubahan dari budaya organisasi yang sudah mapan dan mapan, kesulitan dalam mengukur dan menunjukkan nilai bisnis dari investasi TI yang sering diukur dalam jangka panjang, dan kompleksitas yang tinggi dalam mengelola berbagai kerangka tata kelola yang mungkin berlaku di berbagai unit bisnis atau yurisdiksi geografis tempat organisasi beroperasi.

Untuk meningkatkan efektivitas *Tata Kelola TI*, organisasi harus secara kontinyu dan konsisten melakukan perbaikan dalam berbagai aspek praktik tata kelola. Hal ini meliputi memperkuat komunikasi dan kolaborasi antara pemangku kepentingan bisnis dan TI melalui forum saluran komunikasi reguler dan transparan, berinvestasi dalam pelatihan dan pengembangan yang berkelanjutan untuk meningkatkan kemampuan dan pengetahuan tim tata kelola, mengadopsi alat dan teknologi yang canggih untuk mendukung proses tata kelola, dan secara berkala meninjau dan memperbarui kebijakan serta prosedur tata kelola untuk memastikan relevansi dan efektivitas yang berkelanjutan terhadap kondisi bisnis yang berubah. Budaya organisasi yang mendukung dan mendorong tata kelola yang baik, dengan penekanan yang kuat pada transparansi, akuntabilitas, dan perbaikan berkelanjutan, adalah penting dan kritis untuk mencapai jangka panjang dari inisiatif *Tata Kelola TI* di organisasi.

12.4 Standar dan Kebijakan Sistem Informasi

12.4.1 Pentingnya Standardisasi dalam Sistem Informasi

Standardisasi dalam sistem informasi merupakan hal mendasar dan kritis untuk menciptakan konsistensi yang berkelanjutan, interoperabilitas yang lancar, dan kepatuhan yang efektif dalam penggunaan teknologi informasi di seluruh organisasi dari berbagai tingkatan. Standar yang jelas, terdokumentasi dengan baik, dan memfasilitasi komunikasi yang lebih baik antar tim teknis dan non-teknis, mengurangi ambiguitas dan kebingungan dalam implementasi, dan memastikan bahwa semua pemangku kepentingan memiliki pemahaman yang sama dan konsisten tentang bagaimana sistem informasi harus digunakan, dikonfigurasi, dirawat, dan dikelola (Palade et al., 2024). Standar mencakup berbagai aspek yang luas mulai dari spesifikasi teknis yang detail hingga perangkat keras dan perangkat lunak, hingga aspek non-teknis seperti format data yang konsisten, protokol keamanan yang ketat, persyaratan pelatihan pengguna yang komprehensif, dan prosedur operasional yang terstandarisasi.

Implementasi standardisasi yang konsisten dan disiplin membantu organisasi dalam mencapai skala ekonomi yang signifikan, mengurangi vendor lock-in yang dapat membatasi keharmonisan organisasi, dan memfasilitasi berbagi pengetahuan yang lebih mudah antar tim yang berbeda di organisasi. Standardisasi juga memastikan memainkan peran penting dan strategis dalam keamanan yang ketat dan integritas data yang terjaga

dalam sistem informasi organisasi yang semakin kompleks. Standar keamanan didefinisikan dengan persyaratan jelas yang harus dipenuhi untuk melindungi aset informasi organisasi dari akses tidak sah, modifikasi yang tidak sah, atau penghancuran yang disengaja. Standar pengelolaan data memastikan kualitas data terjaga pada tingkat yang dapat diterima, konsistensi data terjaga di seluruh sistem, dan aksesibilitas data dapat dikelola secara efektif dan aman.

12.4.2 Kerangka dan Standar Internasional

Organisasi memiliki akses terhadap berbagai kerangka kerja dan standar internasional yang telah ditetapkan dan diakui yang dapat diadopsi atau disesuaikan sesuai dengan konteks spesifik dan kebutuhan unik masing-masing organisasi. *Rangkaian standar ISO/IEC* menyediakan pedoman komprehensif untuk berbagai aspek dari manajemen dan tata kelola teknologi informasi yang mencakup *ISO/IEC 20000* untuk manajemen layanan TI dan keunggulan operasional, *ISO/IEC 27001* untuk manajemen keamanan informasi yang komprehensif, dan *ISO/IEC 38500* untuk tata kelola dan manajemen TI di tingkat dewan direksi. *NIST Cybersecurity Framework* dari Institut Standar dan Teknologi Nasional Amerika Serikat menyediakan pedoman fleksibel untuk mengelola risiko keamanan siber dengan pendekatan yang berbasis risiko dan sistematis. Kerangka kerja *ITIL* menyediakan *praktik terbaik* yang telah terbukti untuk manajemen layanan TI dan keunggulan operasional dalam penyampaian layanan TI kepada organisasi. Kerangka kerja *COBIT* memberikan pedoman komprehensif untuk tata kelola dan manajemen TI perusahaan yang

diselaraskan dengan tujuan bisnis organisasi dan tujuan strategis jangka panjang.

Pemilihan kerangka kerja atau standar yang tepat tergantung dari beberapa faktor penting seperti spesifik industri tempat organisasi beroperasi, ukuran dan kompleksitas organisasi, kompleksitas lanskap TI dan infrastruktur teknologi organisasi, dan persyaratan peraturan serta kewajiban kepatuhan yang berlaku di jurisdiksi tempat organisasi beroperasi. Organisasi besar dengan infrastruktur TI yang kompleks dan beberapa unit bisnis mungkin akan memilih untuk mengimplementasikan beberapa kerangka kerja yang saling melengkapi, seperti kombinasi *TOGAF* untuk arsitektur perusahaan yang komprehensif, *ITIL* untuk manajemen layanan TI yang terstruktur, dan *ISO 27001* untuk keamanan informasi yang ketat. Organisasi dan organisasi yang lebih kecil dengan sumber daya yang lebih terbatas mungkin akan fokus pada subset standar yang paling relevan dan kritis untuk operasional mereka. Fleksibilitas dan adaptabilitas dalam memilih dan mengkombinasikan kerangka kerja ini memungkinkan organisasi untuk menciptakan sistem tata kelola yang disesuaikan dengan kebutuhan spesifik dan kemampuan yang dimiliki organisasi.

12.4.3 Kebijakan Sistem Informasi dan Manajemen Kebijakan

Kebijakan sistem informasi menguraikan prinsip, aturan, pedoman, dan prosedur yang mengatur penggunaan, pengelolaan, dan keamanan dari sumber daya teknologi informasi dan aset informasi dalam organisasi. Kebijakan yang dibuat dengan baik dan seimbang harus mencapai keseimbangan yang sehat antara kontrol

ketat yang bertujuan untuk melindungi aset organisasi dan informasi sensitif dengan fleksibilitas yang cukup untuk memungkinkan karyawan melakukan pekerjaan mereka secara produktif dan efisien. Kebijakan harus secara komprehensif mencakup berbagai aspek penting termasuk penggunaan sumber daya TI dan akses internet yang dapat diterima, klasifikasi data dan standar perlindungan yang ketat, mekanisme kontrol akses dan otentikasi, pelaporan insiden dan prosedur manajemen, perencanaan keberlangsungan bisnis dan strategi pemulihan bencana, serta kepatuhan komprehensif terhadap persyaratan hukum dan peraturan yang berlaku (Lestari et al., 2025) .

Kebijakan yang terdokumentasi dengan baik harus mudah diakses oleh semua karyawan melalui berbagai saluran, ditinjau secara berkala untuk memastikan relevansinya terhadap perubahan konteks bisnis, dan diperbarui dengan cepat ketika ada perubahan signifikan dalam lanskap teknologi atau persyaratan peraturan. Proses pengelolaan kebijakan meliputi memastikan beberapa tahapan penting yang saling terkait seperti pengembangan kebijakan yang seimbang, komprehensif, dan dapat dicapai dengan melibatkan pemangku kepentingan yang relevan, komunikasi yang efektif kepada semua pemangku kepentingan untuk memastikan kesadaran dan pemahaman yang mendalam, pelatihan dan pendidikan untuk kepatuhan yang efektif, pemantauan dan penegakan hukum yang konsisten untuk memastikan kebijakan dipatuhi secara nyata dalam operasional, dan tinjauan berkala untuk mengidentifikasi area untuk perbaikan dan adaptasi.

12.4.4 Penerapan dan Penegakan Standar serta Kebijakan

Implementasi standar dan sistem informasi kebijakan dalam organisasi memerlukan manajemen perubahan yang bijaksana, perencanaan terperinci, dan komitmen berkelanjutan dari kepemimpinan di semua tingkat organisasi. Organisasi harus mengalokasikan sumber daya yang memadai untuk mengembangkan rencana implementasi yang terperinci dan dapat ditindaklanjuti, memberikan pelatihan komprehensif kepada karyawan yang akan terkena dampak, menetapkan mekanisme pemantauan yang jelas untuk melacak kepatuhan terhadap standar, dan membangun sistem dukungan untuk memfasilitasi penerapannya. *Pendekatan peluncuran bertahap* memungkinkan organisasi untuk mengelola risiko secara efektif, belajar dari implementasi di unit bisnis yang lebih kecil sebelum melakukan peluncuran di seluruh organisasi secara besar-besaran, dan menyesuaikan strategi berdasarkan pembelajaran dari fase sebelumnya. Metrik keberhasilan harus ditetapkan sebelum implementasi dimulai untuk memungkinkan pengukuran efektivitas yang objektif dan identifikasi dari area yang memerlukan perbaikan atau penyesuaian.

Penegakan standar dan kebijakan sangat penting untuk memastikan bahwa standar dan kebijakan tersebut tidak hanya ada di atas kertas saja tetapi benar-benar diikuti dan diimplementasikan dalam operasi organisasi sehari-hari. Hal ini meliputi audit rutin untuk memeriksa kepatuhan terhadap standar, konsekuensi yang jelas atas ketidakpatuhan yang diterapkan secara konsisten, dan

dukungan nyata dari kepemimpinan yang menunjukkan bahwa organisasi serius dan berkomitmen terhadap kepatuhan terhadap standar. *Teknologi pemantauan* seperti alat pemeriksaan kepatuhan otomatis dan mekanisme pemindaian keamanan dapat membantu secara signifikan dalam mendeteksi pelanggaran dan memberikan peringatan dini tentang potensi masalah sebelum menjadi masalah yang serius. Pendekatan yang seimbang antara penegakan hukum yang ketat dan pendidikan suportif serta komunikasi membantu dalam menumbuhkan budaya organisasi dimana karyawan memahami pentingnya standar dan kebijakan dan secara sukarela mematuhi mereka dalam praktik kerja mereka.

12.5 Strategi Pengembangan Sistem Berkelanjutan

12.5.1 Konsep Pengembangan Sistem Berkelanjutan

Pengembangan sistem berkelanjutan Merujuk pada pendekatan holistik, terintegrasi, dan berwawasan ke depan dalam merancang, mengimplementasikan, dan mengevolusi sistem informasi organisasi sedemikian rupa sehingga dapat terus memberikan nilai dan manfaat jangka panjang sambil secara agile mengadaptasi terhadap perubahan kebutuhan bisnis yang dinamis dan lanskap teknologi yang terus berkembang dan berubah. Konsep ini menekankan pentingnya menciptakan keseimbangan yang sehat antara inovasi yang berkelanjutan dan stabilitas dari infrastruktur yang ada, kelincahan dalam merespons perubahan dan tata kelola

yang kuat, optimalisasi biaya yang cerdas dan pemeliharaan kualitas yang prima. Pengembangan sistem berkelanjutan bukan hanya tentang membangun sistem baru dari nol, tetapi juga secara aktif dan strategi tentang terus meningkatkan dan menyempurnakan sistem yang ada, mengelola utang teknis yang terakumulasi dari waktu ke waktu, dan memastikan bahwa infrastruktur teknologi tetap relevan, mendukung, dan selaras dengan tujuan bisnis organisasi seiring waktu yang berlalu (Hidayat et al., 2026).

Keberlanjutan dalam konteks sistem informasi mencakup berbagai dimensi yang saling terkait dan saling memperkuat termasuk keberlanjutan teknis, keberlanjutan organisasi, keberlanjutan finansial, dan kelestarian lingkungan. Dari perspektif teknis, keberlanjutan berarti menjaga kualitas kode yang tinggi, mengelola ketergantungan dengan hati-hati dan terencana, dan merencanakan peningkatan dan migrasi di masa depan dengan tinjauan ke masa depan. Dari perspektif organisasi, keberlanjutan berarti membangun kapabilitas dan pengetahuan kelembagaan dalam tim, menetapkan tata kelola yang jelas dan proses pengambilan keputusan yang efektif, serta menumbuhkan budaya pembelajaran dan perbaikan berkelanjutan. Dari perspektif finansial, keberlanjutan berarti mengoptimalkan biaya melalui pemanfaatan sumber daya yang efisien, menunjukkan ROI dan nilai bisnis yang jelas dari investasi, dan melakukan investasi strategis dalam teknologi yang selaras dengan strategi organisasi jangka panjang.

12.5.2 Transformasi Digital dan Modernisasi Sistem

Transformasi digital telah menjadi strategi penting yang mendorong dan membutuhkan modernisasi sistem informasi bagi organisasi di hampir semua industri dan sektor ekonomi, menghadirkan kebutuhan mendesak untuk modernisasi sistem informasi yang seringkali merupakan kombinasi kompleks dari *sistem warisan* yang menua dan teknologi modern yang mutakhir (Hidayat et al., 2026). Strategi modernisasi yang efektif dan terencana memerlukan pemahaman yang jelas dan mendalam terhadap keadaan saat ini dari lanskap sistem organisasi, identifikasi yang sistematis terhadap utang teknis dan inefisiensi yang menghambat, dan pengembangan peta jalan yang realistis dan dapat dicapai untuk transisi menuju keadaan sasaran yang diinginkan. Pendekatan seperti *pola pencekik* memungkinkan organisasi untuk secara bertahap mengganti *sistem lama* dengan solusi modern tanpa menyebabkan gangguan bisnis yang mengganggu dan merugikan. *Strategi migrasi cloud* , *containerization* , *arsitektur layanan mikro* , dan *integrasi berbasis API* adalah taktik modernisasi umum yang digunakan oleh organisasi untuk mencapai transformasi digital yang sukses dan berkelanjutan.

Transformasi digital juga melibatkan pemikiran ulang proses bisnis dan struktur organisasi secara mendasar untuk memanfaatkan kemampuan platform teknologi modern. Hal ini mungkin memerlukan perubahan signifikan dalam cara pengorganisasian kerja, cara pengambilan keputusan, dan cara kolaborasi dilakukan antar tim. Organisasi yang berhasil menjalani transformasi digital

biasanya mengkombinasikan inisiatif teknologi dengan manajemen perubahan organisasi yang komprehensif, program pelatihan yang ekstensif, dan pergeseran budaya yang mendalam untuk memastikan bahwa orang dan proses selaras dengan kemampuan teknologi yang baru. Sponsor eksekutif yang komunikasinya terlihat dan jelas dari visi transformasi sangat penting untuk menjaga momentum dan mengamankan dukungan dari pemangku kepentingan di seluruh organisasi.

12.5.3 Ketangkasan dan Responsif terhadap Perubahan

Agility dalam pengembangan dan penggelolaan sistem informasi merupakan kemampuan krusial organisasi untuk merespons dengan cepat terhadap perubahan kebutuhan bisnis, peluang teknologi yang muncul, dan dinamika pasar yang terus berubah dengan cepat. *Metodologi tangkas* seperti *Scrum* dan *Kanban* telah menjadi arus utama dalam pengembangan perangkat lunak, memungkinkan tim untuk memberikan nilai secara bertahap, mengumpulkan umpan balik dari pengguna, dan menyesuaikan pendekatan berdasarkan pembelajaran yang diperoleh. *Scaled Agile Framework* (*SAFe*) memperluas prinsip-prinsip agile ke tingkat perusahaan, memfasilitasi koordinasi yang efektif antar banyak tim dan penyelarasan yang kuat dengan strategi organisasi. *Praktik DevOps* menghilangkan silo tradisional antara tim pengembangan dan operasi, memungkinkan penerapan lebih cepat dan pengiriman berkelanjutan dari peningkatan yang bermanfaat bagi organisasi.

Membangun ketangkasan organisasi memerlukan lebih dari sekadar mengadopsi metodologi tangkas sebagai alat saja. Hal ini memerlukan investasi yang signifikan dalam pengembangan kemampuan dan pengembangan organisasi, membangun struktur tata kelola yang mendukung yang fleksibel, dan menumbuhkan budaya organisasi yang secara aktif mencakup eksperimen, pembelajaran, dan perbaikan berkelanjutan. Organisasi harus secara hati-hati menyeimbangkan antara kelincahan yang responsif terhadap perubahan dan stabilitas yang memastikan keandalan, keamanan, dan kualitas operasional (Palade et al., 2024). Hal ini mungkin memerlukan penetapan pedoman yang jelas tentang apa yang dapat berubah dengan cepat dan apa yang memerlukan perencanaan serta pengujian yang lebih cermat sebelum implementasi. *Pemantauan* berkelanjutan dan putaran umpan balik memungkinkan organisasi untuk mengidentifikasi masalah sejak dini dan menyesuaikan strategi jika diperlukan untuk mencapai tujuan bisnis.

12.5.4 Investasi dalam Inovasi dan Pembangunan Kapabilitas

Strategi pengembangan sistem berkelanjutan memerlukan investasi berkelanjutan dalam inovasi untuk memastikan bahwa organisasi tidak tertinggal dalam lanskap teknologi yang berkembang pesat yang penuh dengan kompetisi dan perubahan. Hal ini meliputi alokasi anggaran dan sumber daya untuk penelitian dan pengembangan, eksperimen yang diukur dengan teknologi baru, dan proyek percontohan untuk menguji pendekatan baru sebelum diadopsi secara lebih luas di seluruh organisasi. Organisasi harus

hati-hati menyeimbangkan antara investasi dalam inovasi yang berdampak tinggi dan strategis dengan mempertahankan serta meningkatkan sistem yang sudah ada yang telah memberikan nilai bisnis yang nyata. *Laboratorium inovasi* , hackathon internal, dan kemitraan yang strategis dengan vendor teknologi serta institusi akademis adalah pendekatan yang dapat membantu organisasi untuk tetap mengikuti tren teknologi yang paling relevan.

Pembangunan kemampuan sama pentingnya untuk keberlanjutan dari sistem informasi dalam jangka panjang yang akan datang. Organisasi harus secara konsisten berinvestasi dalam pelatihan dan pengembangan berkelanjutan untuk tim teknis, memastikan bahwa mereka memiliki keterampilan dan pengetahuan yang diperlukan untuk mengelola dan mengembangkan sistem yang semakin kompleks dan canggih. Mengembangkan praktik arsitektur yang kuat, mempromosikan tinjauan kode dan berbagi pengetahuan yang sistematis, dan membangun pusat keunggulan untuk domain teknologi utama dapat membantu secara signifikan dalam menyebarkan praktik terbaik dan meningkatkan kematangan kemampuan organisasi secara keseluruhan. Retensi pengetahuan melalui dokumentasi yang komprehensif, program pendampingan yang terstruktur, dan sistem manajemen pengetahuan memastikan bahwa pengetahuan institusional tidak hilang ketika staf berpengalaman meninggalkan organisasi. Dengan demikian, investasi berkelanjutan dalam inovasi dan pengembangan kapabilitas menjadi landasan yang penting untuk membangun sistem

informasi yang tidak hanya kuat dan efisien saat ini, tetapi juga adaptif, tangguh, dan berharga bagi organisasi di masa depan.

Profil Penulis



Kiky Rizky Nova Wardani, M.Kom., lahir di Palembang pada 25 November 1987. Ia menyelesaikan pendidikan Magister Teknik Informatika dan memiliki pengalaman sebagai dosen Sistem Informasi dan Teknik Elektro. Saat ini, ia berprofesi sebagai dosen di Universitas Bina Darma. Melalui karyanya, ia berharap buku ini tidak hanya menjadi sumber pengetahuan, tetapi juga menumbuhkan semangat belajar, membentuk adab dan etika, serta memperkuat moral dalam mengamalkan ilmu. Di tengah pesatnya perkembangan teknologi, pengetahuan hendaknya digunakan secara bijak, bertanggung jawab, dan memberi manfaat bagi diri sendiri maupun masyarakat.



Ali Haidir, M.Kom., lahir di Jakarta pada 9 November 1971. Saat ini, ia berdomisili di Jakarta Selatan. Ia memiliki hobi *traveling* dan *badminton*. Kepada para pembaca, ia menyampaikan bahwa teknologi terus berubah, namun prinsip dasar desain sistem yang berpusat pada pengguna (*user-centered*) tetaplah krusial. Ia mengajak pembaca untuk terus bereksplorasi, tidak takut salah dalam proses iterasi, dan selalu mengingat bahwa sistem terbaik adalah sistem yang benar-benar mempermudah kehidupan manusia. Selamat berkarya!



Fintri Indriyani, M.Kom., lahir di Rembang pada 3 Maret 1979. Ia menyelesaikan pendidikan magister (S2) dalam bidang Ilmu Komputer. Ia memiliki pengalaman sebagai dosen bersertifikasi sejak 2015, aktif di konsorsium program studi, serta mengampu mata kuliah Analisis dan Desain Sistem Informasi. Saat ini, ia berprofesi sebagai dosen. Kepada para pembaca, ia menyampaikan bahwa belajar sistem informasi bukan hanya tentang teknologi, tetapi juga tentang bagaimana membangun solusi nyata untuk kehidupan sehari-hari. Buku ini hadir untuk menemani perjalanan pembaca dalam memahami langkah demi langkah, mulai dari analisis kebutuhan hingga tata kelola sistem. Ia berharap setiap bab dapat menjadi bekal untuk masa depan karena ilmu yang dipelajari hari ini akan menjadi fondasi inovasi esok. Teruslah membaca, teruslah mencoba, dan jangan pernah berhenti berkreasi.



Euis Widanengsih, M.Kom., lahir di Tangerang pada 7 Juli 1985. Ia menyelesaikan pendidikan magister (S2) dalam bidang Sistem Informasi. Ia memiliki pengalaman sebagai dosen sejak tahun 2008 hingga saat ini. Saat ini, ia berprofesi sebagai dosen di Universitas Bina Sarana Informatika. Kepada para pembaca, ia berpesan agar tidak bosan untuk terus membaca.



Agus Salim, M.Kom., lahir di Jakarta pada 11 Agustus 1976. Ia menyelesaikan pendidikan magister (S2) dalam bidang Ilmu Komputer. Ia memiliki pengalaman sebagai instruktur laboratorium pada tahun 2000–2010, *programmer* pada tahun 2004–2023, serta dosen di Universitas Bina Sarana Informatika sejak tahun 2010 hingga sekarang. Saat ini, ia berprofesi sebagai dosen. Kepada para pembaca, ia menyampaikan bahwa kualitas informasi yang dihasilkan sangat bergantung pada bagaimana data dirancang, diorganisasi, serta dikelola dengan baik. Perancangan basis data bukan sekadar membuat tabel dan relasi, melainkan tentang membangun fondasi informasi yang kuat, andal, dan bermakna bagi organisasi.



Bilal Abdul Wahid, M.Kom., lahir di Pontianak pada 25 Juli 1975. Saat ini, ia berdomisili di Depok, Jawa Barat. Ia memiliki hobi sepak bola dan memancing. Kepada para pembaca, ia berpesan, “Belajarlah dari sebuah kegagalan karena usaha dan doa adalah kunci menuju keberhasilan.”



Henny Destiana, M.Kom., lahir di Jakarta pada 29 Desember 1980. Ia menyelesaikan pendidikan magister (S2) dalam bidang Ilmu Komputer. Ia memiliki pengalaman selama 23 tahun dan saat ini berprofesi sebagai dosen. Kepada para pembaca, ia menyampaikan pesan, “Semoga bermanfaat, ya.”



Hj. A. Irmayani Pawelloi, S.T., M.T., lahir di Rappang pada 19 Desember 1970. Ia menyelesaikan pendidikan sarjana (S1) Teknik Elektro di Universitas Muslim Indonesia dan pendidikan magister (S2) Teknik Elektro di Universitas Hasanuddin. Ia telah mengabdikan diri selama 25 tahun sebagai pengajar pada Program Studi Teknik Elektro Universitas Muhammadiyah Parepare. Saat ini, ia berprofesi sebagai dosen yayasan pada Program Studi Teknik Elektro Universitas Muhammadiyah Parepare. Kepada para pembaca, ia berpesan, "Jelajahi dunia sains dan rekayasa, temukan inovasi yang mengubah dunia, dan aplikasikan pengetahuan untuk kemajuan masa depan."



Ghofar Taufik, M.Kom., lahir di Jakarta pada 11 September 1978. Saat ini, ia berdomisili di Kabupaten Bogor, Jawa Barat. Ia memiliki hobi *traveling* dan olahraga. Kepada para pembaca, ia berharap buku ini dapat bermanfaat serta menambah ilmu pengetahuan dan wawasan tentang Ilmu Komputer Dasar.



Indra Riyana Rahadjeng, S.T., M.Kom., lahir di Surabaya pada 27 Mei 1969. Saat ini, ia berdomisili di Depok. Ia memiliki hobi membaca, *traveling*, dan berkreasi. Kepada para pembaca, ia berpesan bahwa membaca adalah investasi masa depan dan sumber inspirasi terbaik. Oleh karena itu, jangan pernah bosan membaca.



Aryo Tunjung Kusumo, M.Kom., lahir di Surakarta pada 7 Maret 1986. Ia menyelesaikan pendidikan pascasarjana di STMIK Nusa Mandiri Jakarta. Ia memiliki pengalaman dalam mengajar, menjadi *programmer*, dan analis sistem. Saat ini, ia berprofesi sebagai dosen.

Kepada para pembaca, ia berpesan agar memperbanyak membaca untuk menambah pengetahuan.



Fajar Agustini, M.Kom., lahir di Jakarta pada 22 Agustus 1989. Ia menyelesaikan pendidikan magister (S2) dalam bidang Ilmu Komputer. Ia memiliki pengalaman mengajar Matematika IPA tingkat SD dan SMP saat masih kuliah, menjadi asisten instruktur pada laboratorium komputer, serta mengajar mata kuliah teori dan pemrograman komputer sejak 2012. Saat ini, ia berprofesi sebagai dosen. Kepada para pembaca, ia berpesan, “Ikatlah ilmu dengan menuliskannya.”

Daftar Pustaka

Hidayat, S., Kirana, H. C., Wicaksono, H., & Nugroho, K. (2026). Perancangan Enterprise Architecture Berbasis Togaf ADM Untuk Integrasi Sistem Menggunakan ODOO ERP pada PT Bromindo Mekar Mitra. SENTRI: Jurnal Riset Ilmiah. https://doi.org/10.55681/sentri.v5i1.5348

Lestari, M., Wijaya, A. F., & Chandra, M. M. D. (2025). Enterprise Architecture Model for Smart Government Implementation. Journal of Information Systems and Informatics. https://doi.org/10.51519/journalisi.v7i1.978

Palade, D., Møller, C., & Hansen, A. (2024). Enterprise Integration as a Driving Factor for Guiding Digitalization in a Manufacturing Small and Medium Enterprise. Complex Systems Informatics and Modeling Quarterly. https://doi.org/10.7250/csimq.2024-39.03

Santosa, I., & Mulyana, R. (2023). The IT Services Management Architecture Design for Large and Medium-sized Companies based on ITIL 4 and TOGAF Framework. State Polytechnics of Andalas. https://doi.org/https://doi.org/10.30630/joiv.7.1.1590

Goyal, A. (2022). Optimising Software Lifecycle Management through Predictive Maintenance: Insights and Best Practices.

International Journal of Science and Research Archive. https://doi.org/10.30574/ijsra.2022.7.2.0348

Naik, M. L., & Kirkire, M. (2025). Maintenance 4.0 implementation in manufacturing industries: an analysis of enablers. International Journal of Quality & Reliability Management. https://doi.org/10.1108/ijqrm-01-2025-0001

Valencia, A., & Pinzón, J. D. (2021). Asset Management Model of SCADA Infrastructure of Power Control Centers based on Indicators. IEEE PES Innovative Smart Grid Technologies Conference. https://doi.org/10.1109/ISGTLatinAmerica52371.2021.9543089

Vilsmeier, J., & Salehi, V. (2025). Engineering change management in complex system networks. Journal of Intelligent System of Systems Lifecycle Management. https://doi.org/10.71015/0kr2tg27

Fayayola, O. A., Olorunfemi, O. L., & Shoetan, P. O. (2024). DATA PRIVACY AND SECURITY IN IT: A REVIEW OF TECHNIQUES AND CHALLENGES. Fair East Publishers. https://doi.org/https://doi.org/10.51594/csitrj.v5i3.909

Fayayola, O. A., Olorunfemi, O. L., & Shoetan, P. O. (2024). DATA PRIVACY AND SECURITY IN IT: A REVIEW OF TECHNIQUES AND CHALLENGES. Fair East Publishers. https://doi.org/https://doi.org/10.51594/csitrj.v5i3.909

Naiho, H. N. N., Layode, O., Adeleke, G. S., Udeh, E. O., & Labake, T. T. (2024). Addressing cybersecurity challenges in smart

grid technologies: Implications for sustainable energy infrastructure. Engineering Science & Technology Journal. https://doi.org/10.51594/estj.v5i6.1218

Swanzy, P. N., Abukari, A. M., & Ansong, E. D. (2024). Data Security Framework for Protecting Data in Transit and Data at Rest in the Cloud. Sciencedomain International. https://doi.org/https://doi.org/10.9734/cjast/2024/v43i64387

Hamidah, I., Haromain, I., & Drehem, I. M. (2025). Evaluasi Pengujian Kinerja Menggunakan JMeter untuk Menunjang Stabilitas Aplikasi Layanan Perbankan pada PT Bank Rakyat Indonesia Tbk. DBESTI: Journal of Digital Business and Technology Innovation. https://doi.org/10.54914/dbesti.v2i1.1621

Hidayat, M. A. R., Izzati, N. N., Prirhatama, A. R. P., Patria, Y., & Kurmilasari, S. (2025). PENGUJIAN PERANGKAT LUNAK PADA WEBSITE KA'CAKE: IMPLEMENTASI UNIT TESTING, INTEGRATION TESTING, SYSTEM TESTING, DAN VALIDATION TESTING UNTUK MENJAMIN KUALITAS DAN KEANDALAN SISTEM. JATI (Jurnal Mahasiswa Teknik Informatika). https://doi.org/10.36040/jati.v9i4.13969

Qystiar, L., & Wijayanto, D. (2025). DOKUMENTASI PENGUJIAN APLIKASI CLEONTECH SUPPORT MENGGUNAKAN METODE SOFTWARE TESTING LIFE CYCLE BERSTANDAR IEEE 829. Jurnal

Informatika, Teknologi Dan Sains. https://doi.org/10.51401/jinteks.v7i3.6197

Salshabila, A. D., Sutanto, I., Hosizah, H., & Pertiwi, T. S. (2026). Pengujian Kualitas Sistem Informasi Kesehatan Periksa.ID Berbasis Autentikasi Menggunakan Software Testing Life Cycle. Jurnal JTIK (Jurnal Teknologi Informasi Dan Komunikasi). https://doi.org/10.35870/jtik.v10i2.5477

Mahomedov, K. (2025). Information Security Challenges in an Enterprise-Grade Software Development Lifecycle. Theoretical and Applied Cybersecurity. https://doi.org/10.20535/tacs.2664-29132025.2.338771

Panguraj, A. R. R. (2021). End-to-End Automation of Software Development Lifecycle (SDLC) Tools and Processes. International Journal of Multidisciplinary Research and Growth Evaluation. https://doi.org/10.54660/.ijmrge.2021.2.6.438-441

Rajapakse, R., Zahedi, M., Babar, M., & Shen, H. (2021). Challenges and solutions when adopting DevSecOps: A systematic review. Information and Software Technology. https://doi.org/10.1016/j.infsof.2021.106700

Khairunnisa, K., & Emriadi, H. (2026). Design of User Interface (UI) and User Experience (UX) for the Visual Communication Design Student Works Data Center. Sainstek : Jurnal Sains Dan Teknologi. https://doi.org/10.31958/js.v17i2.15985

Masveta, D., & Manyangara, M. (2025). The UX-UI continuum: exploring the interplay between user experience and user interface in e-learning platforms. Cogent Education. https://doi.org/10.1080/2331186X.2025.2536531

Mustafa, H. M., Alsbaihi, O., Jahameh, S. S., Sayed, A., & Othman, M. (2025). User interface, usability, and user experience. Information Design Journal. https://doi.org/10.1075/idj.24006.mus

Riski, R., Hafsari, A. D., & Abdillah, R. (2025). HUMAN-AI COLLABORATION IN USER INTERFACE & USER EXPERIENCE EVLUATION USING DESIGN THINKING. Prosiding Seminar Nasional Teknologi Dan Sistem Informasi. https://doi.org/10.33005/sitasi.v5i1.2544

Chen, L., Ali Babar, M., & Nuseibeh, B. (2020). "Characterizing Architecturally Significant Requirements: A Systematic Literature Review." *Information and Software Technology*, 117, 106211.

Taibi, D., Lenarduzzi, V., & Pahl, C. (2019). "Architectural Patterns for Microservices: A Systematic Mapping Study." *Journal of Systems and Software*, 150, 77–91.

Weyns, D., et al. (2021). "Architectural Patterns for Self-Adaptive Systems: A Systematic Literature Review." *ACM Computing Surveys*, 54(3), 1–41.

Ali, S., et al. (2022). "A Systematic Review of Software Architecture Recovery Approaches." *IEEE Transactions on Software Engineering*, 48(8), 2563–2586.

Li, Y., & Liang, P. (2023). “A Survey on Software Architecture Evaluation Methods: State of the Art and Future Directions.” *Journal of Systems Architecture*, 140, 102932.

Delplanque, J., Etien, A., Anquetil, N., & Auverlot, O. (2018). Relational Database Schema Evolution: An Industrial Case Study. IEEE International Conference on Software Maintenance and Evolution. https://doi.org/10.1109/ICSME.2018.00073

Jena, R., Nadukuru, S., Scholar, Singiri, S., Goel, O., Kumar, Dr. L., & Jain, Dr. A. (2020). Leveraging AWS and OCI for Optimized Cloud Database Management. International Journal for Research Publication and Seminar. https://doi.org/10.36676/jrps.v11.i4.1587

Mandapuram, M., & Hosen, Md. F. (2018). The Object-Oriented Database Management System versus the Relational Database Management System: A Comparison. Global Disclosure of Economics and Business. https://doi.org/10.18034/gdeb.v7i2.657

Natarajan, K., & Shaik, V. (2020). Transparent Data Encryption: Comparative Analysis and Performance Evaluation of Oracle Databases. 2020 Fifth International Conference on Research in Computational Intelligence and Communication Networks (ICRCICN). https://doi.org/10.1109/ICRCICN50933.2020.9296168

Pokorný, J. (2018). Integration of Relational and NoSQL Databases. Asian Conference on Intelligent Information and Database Systems. https://doi.org/10.1007/978-3-319-75420-8_4

Amran, N., Mohamed, H., & Bahry, F. D. S. (2019). Developing Human Resource Training Management (HRTM) Conceptual Model Using Entity Relationship Diagram (ERD). International Journal of Academic Research in Business and Social Sciences. https://doi.org/10.6007/IJARBSS/V8-I12/5249

Hayat, R., & Kuswandi, K. (2025). RANCANG BANGUN SISTEM MANAJEMEN DATA MENGGUNAKAN QR CODE BERBASIS WEB PADA PT DENSO MANUFACTURING INDONESIA. JURNAL LENTERA ICT. https://doi.org/10.34127/jrict.v10i2.1933

Mo, W. (2024). Design of an Efficient Hospital Management Database System for Enhanced Data Management. Proceedings of the 2nd International Conference on Data Analysis and Machine Learning. https://doi.org/10.5220/0013516300004619

Lumat, Y. (2025). Data Flow Diagrams of Purchases Process and Internal Control: A Case Study. International Journal of Academic Research in Business and Social Sciences. https://doi.org/10.6007/ijarbss/v15-i5/25222

M, M. K. B., & Djamaluddin. (2022). Perancangan Proses Bisnis Sistem Informasi Pendaftaran Tugas Akhir Menggunakan Business Prosess Modelling Notation (BPMN). Bandung

Conference Series Industrial Engineering Science. https://doi.org/10.29313/bcsies.v2i1.2447

Reawaruw, V. A., Tuaputty, J., Bahtiar, S. R., Aprilia, F. P., Julianti, A., Tuhumury, A. M., Theris, A., Rehatta, G., Mustamu, M., & Baker, M. (2025). Desain Data Flow Diagram dan Flowchart Pada Siklus Pengeluaran Kas Percetakan Maleo Printing di Kota Ambon. Jumat Ekonomi: Jurnal Pengabdian Masyarakat. https://doi.org/10.32764/abdimasekon.v6i2.5944

Ridwan, R., Kustian, N., & Ambarsari, E. W. (2022). PERAN DATA STORE DALAM MEMPRESENTASIKAN HUBUNGAN DATA FLOW DIAGRAM SSADM DENGAN ENTITY RELATIONSHIP DIAGRAM. Jurnal Ilmiah Teknik Mesin Elektro Dan Komputer. https://doi.org/10.51903/juritek.v2i2.412

Teixeira, A., Ferreira, J., & Ramos, A. (2024). Optimization of Business Processes Through BPM Methodology: A Case Study on Data Analysis and Performance Improvement. Inf. https://doi.org/10.3390/info15110724

Tuasamu, Z., M.Lewaru, N. A. I., Idris, M. R., Syafaat, A. B. N., Faradilla, F., Fadlan, M., Nadiva, P., & Efendi, R. (2023). Analisis Sistem Informasi Akuntansi Siklus Pendapatan Menggunakan DFD dan Flowchart Pada Bisnis Porobico. Jurnal Bisnis Dan Manajemen. https://doi.org/10.61930/jurbisman.v1i2.181

Almeida, F., Simões, J., & Lopes, S. F. S. F. (2022). Exploring the Benefits of Combining DevOps and Agile. Multidisciplinary Digital Publishing Institute. https://doi.org/https://doi.org/10.3390/fi14020063

Cao, H. (2025). Comparative Analysis of Software Development Methodologies. Applied and Computational Engineering. https://doi.org/10.54254/2755-2721/2026.tj27796

Chahar, S., & Singh, S. (2024). Analysis of SDLC Models with Web Engineering Principles. None. https://doi.org/https://doi.org/10.1109/akgec62572.2024.10868694

Hnatchuk, Y., & Lebedovska, M. (2025). DECISION SUPPORT SYSTEM FOR PROJECT RESOURCE PLANNING BASED ON THE RANDOM FOREST METHOD. Computer Systems and Information Technologies. https://doi.org/10.31891/csit-2025-4-4

Maksymov, A. (2025). Analysis of risk management standards and their application in IT projects. Management of Development of Complex Systems. https://doi.org/10.32347/2412-9933.2025.61.66-75

Pargaonkar, S. (2023). A Comprehensive Research Analysis of Software Development Life Cycle (SDLC) Agile & Waterfall Model Advantages, Disadvantages, and Application Suitability in Software Quality Engineering. International Journal of Scientific and Research Publications. https://doi.org/10.29322/ijsrp.13.08.2023.p14015

Yas, Q. M., Alazzawi, A., & Rahmatullah, B. (2023). A Comprehensive Review of Software Development Life Cycle methodologies: Pros, Cons, and Future Directions. Iraqi Journal for Computer Science and Mathematics. https://doi.org/10.52866/ijcsm.2023.04.04.014

Subarkah, A., Abimanyu, M. D., Fahiri, S. A., Purwani, F., Informasi, S., Raden, U., Palembang, F., Akademik, S. I., Kebutuhan, U. A., & Sistem, P. (2025). PERANCANGAN SISTEM INFORMASI AKADEMIK MENGGUNAKAN UML PADA SEKOLAH SMA ABC. Jurnal Mahasiswa Sistem Informasi (JMSI). https://doi.org/10.24127/jmsi.v7i1.10739

Syafitri, R. D., Susilowati, M., & Kurniawan, Y. (2023). ANALISIS KEBUTUHAN SOFTWARE UNTUK SISTEM INFORMASI MANAJEMEN LAYANAN RSUD XYZ. Kurawal - Jurnal Teknologi, Informasi Dan Industri. https://doi.org/10.33479/kurawal.v6i1.840

Tjahyanti, L. P. A. S., & Sutama, G. R. (2025). Peran Analisis Kebutuhan Dalam Menciptakan Sistem Informasi yang Responsif dan Berkelanjutan. KOMTEKS. https://doi.org/10.37637/komteks.v3i2.2232

Widyastuti, Y. M., Oktiarso, T., & Putrianto, N. K. (2024). PERENCANAAN DAN ANALISIS KEBUTUHAN PENGGUNA DALAM PENGEMBANGAN SISTEM INFORMASI HUBUNGAN PELANGGAN. Kurawal -

Jurnal Teknologi, Informasi Dan Industri. https://doi.org/10.33479/kurawal.v7i1.1064

Catatan

PENERBIT BUKULOKA

Buku **Desain dan Pengembangan Sistem Informasi** mengulas proses merancang sistem informasi yang sesuai dengan kebutuhan pengguna dan organisasi. Pembahasan mencakup pemahaman kebutuhan, perancangan alur data, pengelolaan tampilan, keamanan, hingga penerapan sistem agar dapat digunakan secara efektif.

Dengan penyajian yang sederhana dan mudah diikuti, buku ini membantu pembaca memahami bagaimana teknologi informasi dibangun untuk mendukung pekerjaan, layanan, dan pengambilan keputusan. Buku ini dijual untuk masyarakat umum, khususnya bagi siapa saja yang ingin mengenal dunia sistem informasi secara praktis dan relevan dengan kebutuhan masa kini.

ISBN 978-634-324-017-4 (PDF)
9 786343 240174