ISBN 978-623-198-551-4

ANALISIS DAN DESAIN SISTEM INFORMASI:
PENDEKATAN TERSTRUKTUR DAN BERORIENTASI OBJEK

Muhammad Fairuzabadi, Irani Hoeronis, Zen Munawar, Johni S Pasaribu, Sarji, Irmawati, Rita Komalasari, Nurul Aini, Santo Fernandi Wijaya, Angga Aditya Permana, Melissa Indah Fianty

ANALISIS DAN DESAIN SISTEM INFORMASI: PENDEKATAN TERSTRUKTUR DAN BERORIENTASI OBJEK

Muhammad Fairuzabadi
Irani Hoeronis
Zen Munawar
Johni S Pasaribu
Sarji
Irmawati
Rita Komalasari
Nurul Aini
Santo Fernandi Wijaya
Angga Aditya Permana
Melissa Indah Fianty

GET PRESS INDONESIA

ANALISIS DAN DESAIN SISTEM INFORMASI: PENDEKATAN TERSTRUKTUR DAN BERORIENTASI OBJEK

Penulis :
Muhammad Fairuzabadi
Irani Hoeronis
Zen Munawar
Johni S Pasaribu
Sarji
Irmawati
Rita Komalasari
Nurul Aini
Santo Fernandi Wijaya
Angga Aditya Permana
Melissa Indah Fianty

ISBN : 978-623-198-551-4

Editor : Ari Yanto, M.Pd
Penyunting : Tri Putri Wahyuni,S.Pd
Desain Sampul dan Tata Letak : Atyka Trianisa, S.Pd

Penerbit : GET PRESS INDONESIA
Anggota IKAPI No. 033/SBA/2022

Redaksi :
Jln. Palarik Air Pacah No 26 Kel. Air Pacah
Kec. Koto Tangah Kota Padang Sumatera Barat
Website : www. getpress.co.id
Email : adm.getpress@gmail.com

Cetakan pertama, 1 Agustus 2023

KATA PENGANTAR

Segala Puji dan syukur atas kehadirat Allah SWT dalam segala kesempatan. Sholawat beriring salam dan doa kita sampaikan kepada Nabi Muhammad SAW. Alhamdulillah atas Rahmat dan Karunia-Nya penulis telah menyelesaikan Buku Analisis Dan Desain Sistem Informasi: Pendekatan Terstruktur Dan Berorientasi Objek Ini.

Buku Ini Membahas Konsep Dasar Analisis dan Desain Sistem Informasi, Analisis Kebutuhan Sistem Berorientasi Objek, Desain Model Proses Bisnis Pendekatan Terstruktur, Desain Model Proses Bisnis Berorientasi Objek, Desain Basis Data Pendekatan Terstruktur, Desain Basis Data Berorientasi Objek, Desain Antarmuka Pengguna, Desain Keamanan Sistem Informasi, Implementasi Sistem, Pemeliharaan Sistem.

Proses penulisan buku ini berhasil diselesaikan atas kerjasama tim penulis. Demi kualitas yang lebih baik dan kepuasan para pembaca, saran dan masukan yang membangun dari pembaca sangat kami harapkan.

Penulis ucapkan terima kasih kepada semua pihak yang telah mendukung dalam penyelesaian buku ini. Terutama pihak yang telah membantu terbitnya buku ini dan telah mempercayakan mendorong, dan menginisiasi terbitnya buku ini. Semoga buku ini dapat bermanfaat bagi masyarakat Indonesia.

Padang, 1 Agustus 2023

Penulis

DAFTAR ISI

DAFTAR GAMBAR

DAFTAR TABEL

BAB 1
KONSEP DASAR ANALISIS DAN DESAIN SISTEM INFORMASI

Oleh Muhammad Fairuzabadi

1.1 Pendahuluan

Analisis dan Desain Sistem Informasi (ADSI) adalah kerangka kerja yang digunakan dalam mengembangkan sistem informasi yang efektif dan efisien. Sistem informasi memiliki peran yang sangat penting dalam dunia bisnis dan organisasi modern, karena mereka membantu dalam pengumpulan, penyimpanan, pengolahan, dan penyebarluasan informasi yang relevan untuk mendukung pengambilan keputusan dan operasional sehari-hari.

Analisis dan Desain Sistem Informasi melibatkan pemahaman yang mendalam tentang kebutuhan bisnis, proses bisnis, serta teknologi informasi yang tersedia. Dalam proses ADSI, seorang analis sistem bertugas untuk memahami masalah atau peluang bisnis yang harus dipecahkan melalui sistem informasi, dan merancang solusi yang optimal untuk memenuhi kebutuhan tersebut.

Penerapan konsep dasar ADSI secara efektif dapat membantu organisasi untuk mencapai keunggulan kompetitif, meningkatkan efisiensi operasional, mengoptimalkan penggunaan sumber daya, dan meningkatkan kualitas pengambilan keputusan. Namun, untuk mencapai hal tersebut, diperlukan pemahaman yang baik tentang konsep dasar ADSI serta kemampuan untuk menerapkannya dengan tepat.

1.2 Definisi, Tujuan dan Manfaat

Analisis dan Desain Sistem Informasi (ADSI) adalah disiplin yang melibatkan pemahaman dan perancangan sistem informasi yang efektif dan efisien (Senn 2012). Sistem informasi memiliki peran sentral dalam organisasi modern, memungkinkan pengumpulan, penyimpanan, pengolahan, dan penyebarluasan informasi yang relevan untuk mendukung pengambilan keputusan dan operasional sehari-hari.

Analisis dan Desain Sistem Informasi berfokus pada proses pengembangan sistem informasi yang optimal, yang memenuhi kebutuhan bisnis dan memberikan nilai tambah bagi organisasi. Dalam konteks ini, analisis mengacu pada pemahaman mendalam tentang kebutuhan bisnis dan permasalahan yang perlu dipecahkan melalui sistem informasi. Sedangkan desain melibatkan perancangan struktur sistem yang meliputi arsitektur sistem, pemodelan data, antarmuka pengguna, dan proses bisnis.

Tujuan dan Manfaat ADSI meliputi beberapa aspek yang penting. Berikut ini adalah tujuan dan manfaat utama ADSI (Tilley 2020):

Tujuan ADSI:

1. Memahami kebutuhan bisnis
   Tujuan utama ADSI adalah memahami kebutuhan bisnis organisasi dan mengidentifikasi permasalahan yang dapat diselesaikan melalui sistem informasi. Dengan pemahaman yang baik tentang kebutuhan bisnis, analis sistem dapat merancang solusi yang tepat dan efektif.
2. Meningkatkan efisiensi operasional
   ADSI bertujuan untuk meningkatkan efisiensi operasional organisasi dengan mengoptimalkan proses bisnis, menghilangkan redundansi, dan mengurangi biaya operasional. Melalui analisis dan desain yang baik, sistem informasi dapat mempercepat aliran informasi, mengotomatisasi tugas-tugas rutin, dan meningkatkan produktivitas.

3. Meningkatkan kualitas pengambilan keputusan
   Sistem informasi yang baik dapat memberikan data yang akurat, terkini, dan relevan untuk pengambilan keputusan. ADSI bertujuan untuk merancang sistem informasi yang dapat menghasilkan informasi yang tepat waktu dan mendukung pengambilan keputusan yang lebih baik dan lebih terinformasi.
4. Mendukung pertumbuhan dan inovasi
   Dengan merancang sistem informasi yang adaptif dan fleksibel, ADSI dapat membantu organisasi dalam mengatasi tantangan yang berkaitan dengan pertumbuhan bisnis dan perubahan lingkungan. Sistem informasi yang baik juga dapat mendorong inovasi dengan memberikan platform untuk pengembangan dan implementasi ide-ide baru.

Manfaat ADSI:

Efisiensi dan produktivitas yang meningkat: Implementasi sistem informasi yang dirancang dengan baik dapat meningkatkan efisiensi operasional dan produktivitas karyawan dengan mengotomatisasi tugas-tugas rutin, mengurangi kesalahan manusia, dan mempercepat aliran informasi.

1. Pengambilan keputusan yang lebih baik
   ADSI dapat memberikan informasi yang akurat, tepat waktu, dan terintegrasi, yang memungkinkan pengambilan keputusan yang lebih baik dan lebih terinformasi. Data yang terorganisir dengan baik dan analisis yang mendalam dapat membantu manajemen dalam mengambil keputusan strategis yang tepat.
2. Pengendalian yang lebih baik
   Sistem informasi yang baik membantu dalam mengendalikan operasi bisnis dan memantau kinerja organisasi secara efektif. ADSI dapat menyediakan

mekanisme pengendalian yang memungkinkan identifikasi masalah secara cepat dan pengambilan tindakan yang diperlukan.

3. Keunggulan kompetitif
   Dengan menerapkan ADSI secara efektif, organisasi dapat mencapai keunggulan kompetitif melalui penggunaan sistem informasi yang inovatif. Sistem informasi yang baik dapat membantu organisasi dalam membedakan diri dari pesaing, meningkatkan kepuasan pelanggan, dan menanggapi perubahan pasar dengan lebih responsif.
4. Pengurangan risiko
   Dengan analisis yang teliti dan perancangan yang baik, ADSI dapat membantu mengidentifikasi dan mengurangi risiko yang terkait dengan implementasi sistem informasi. Identifikasi risiko potensial dapat memungkinkan langkah-langkah mitigasi yang sesuai untuk diambil sebelum masalah terjadi.

1.3 Tren Bisnis untuk Sistem Informasi Masa Kini

Tren bisnis yang mempengaruhi pengembangan dan penerapan sistem informasi masa kini adalah sebagai berikut (Whitten dan Bentley D. 2008):

1. Digitalisasi
   Organisasi mengadopsi digitalisasi untuk mengubah proses bisnis tradisional menjadi proses yang berbasis teknologi. Ini melibatkan penggunaan solusi digital untuk mengotomatisasi tugas, meningkatkan efisiensi, dan memberikan akses informasi yang lebih cepat.
2. Analitik Data
   Organisasi semakin mengandalkan analitik data untuk mendapatkan wawasan yang berharga dari data yang dihasilkan. Dengan memanfaatkan teknologi seperti big data dan analisis prediktif, organisasi dapat

mengidentifikasi tren, pola, dan peluang bisnis yang dapat membantu pengambilan keputusan yang lebih baik.

3. Kecerdasan Buatan (*Artificial Intelligence/AI*)
   AI telah menjadi faktor kunci dalam sistem informasi masa kini. Dengan menggabungkan kecerdasan mesin, pembelajaran mesin, dan teknik pengenalan pola, organisasi dapat mengembangkan sistem yang dapat melakukan tugas-tugas yang biasanya memerlukan intervensi manusia.
4. Cloud Computing
   Penggunaan cloud computing telah memungkinkan organisasi untuk menyimpan, mengelola, dan mengakses data dan aplikasi secara fleksibel melalui internet. Cloud computing juga menyediakan skala yang mudah disesuaikan, keamanan yang ditingkatkan, dan biaya yang lebih efisien dibandingkan dengan infrastruktur on-premise.
5. Internet of Things (IoT)
   IoT menghubungkan perangkat fisik ke jaringan internet, memungkinkan pertukaran data dan komunikasi antar perangkat. Organisasi dapat memanfaatkan IoT untuk mengumpulkan data secara real-time dari sensor dan perangkat yang terhubung, yang dapat digunakan untuk pemantauan, analisis, dan pengambilan keputusan yang lebih baik.
6. Pengalaman Pengguna yang Personal
   Fokus pada pengalaman pengguna yang personal dan terkustomisasi semakin penting dalam pengembangan sistem informasi. Organisasi harus dapat menyediakan antarmuka yang intuitif, interaktif, dan mudah digunakan agar pengguna dapat berinteraksi dengan sistem dengan nyaman.

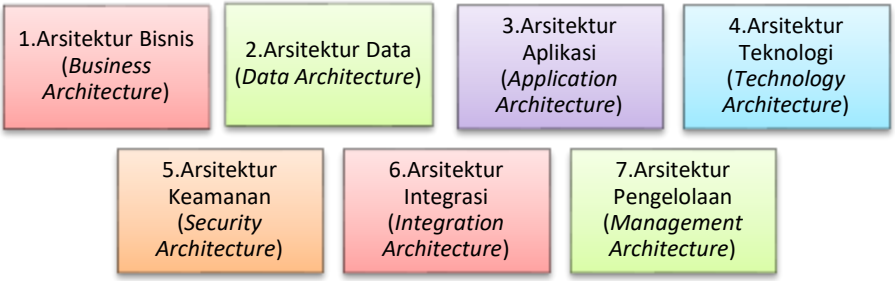
7. Keamanan Data dan Privasi
   Dalam era digital, keamanan data dan privasi pengguna menjadi perhatian utama. Organisasi harus memprioritaskan perlindungan data dan melaksanakan praktik keamanan yang kuat untuk melindungi informasi sensitif dan mencegah pelanggaran privasi.
8. Integrasi dan Kolaborasi
   Organisasi menghadapi tantangan dalam mengintegrasikan sistem informasi yang berbeda dan memungkinkan kolaborasi yang efektif antara departemen dan mitra bisnis. Solusi yang mendukung integrasi dan kolaborasi menjadi penting untuk meningkatkan efisiensi dan produktivitas.
9. Responsif terhadap Perubahan
   Lingkungan bisnis yang cepat berubah membutuhkan fleksibilitas dan responsivitas dari sistem informasi. Organisasi harus dapat dengan cepat menyesuaikan dan mengubah sistem informasi mereka untuk menghadapi perubahan kebutuhan bisnis, teknologi, dan pasar.

Dengan memahami tren bisnis ini, organisasi dapat merancang dan mengimplementasikan sistem informasi yang sesuai dengan kebutuhan masa kini, meningkatkan daya saing, efisiensi, dan inovasi bisnisnya.

1.4 Kerangka Arsitektur Sitem Informasi

Kerangka arsitektur sistem informasi (SI) adalah struktur konseptual yang digunakan untuk merencanakan, merancang, dan mengelola sistem informasi dalam suatu organisasi. Kerangka arsitektur SI menyediakan panduan dan pedoman untuk mengorganisasi komponen sistem informasi, interaksi antara komponen, serta hubungan dengan komponen lain dalam lingkungan organisasi.

Berikut adalah kerangka arsitektur sistem informasi yang umum digunakan (Fairuzabadi et al. 2023; Kustiyahningsih dan Rahmanita 2022):



Gambar 1.1. Kerangka Arsitektur Sistem Informasi

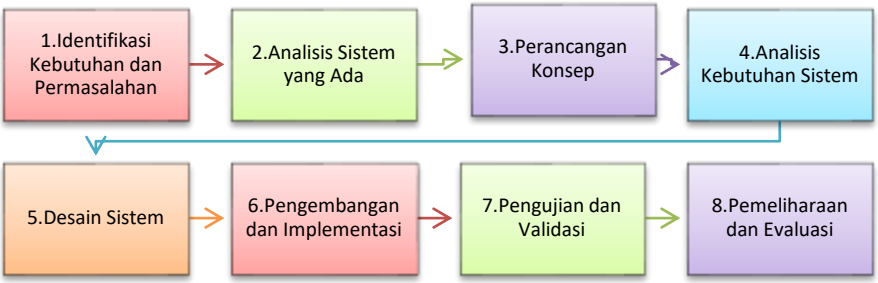
1. Arsitektur Bisnis (*Business Architecture*)
   Merupakan pandangan terhadap struktur organisasi, proses bisnis, dan kebutuhan bisnis yang harus diakomodasi oleh sistem informasi. Arsitektur bisnis mengidentifikasi tujuan bisnis, fungsi organisasi, proses bisnis, dan ketergantungan antar proses.
2. Arsitektur Data (*Data Architecture*)
   Merupakan pandangan terhadap struktur data yang digunakan dalam sistem informasi. Arsitektur data melibatkan perencanaan, perancangan, dan pengelolaan basis data, serta definisi struktur data yang digunakan untuk menyimpan, mengatur, dan mengakses informasi.
3. Arsitektur Aplikasi (*Application Architecture*)
   Merupakan pandangan terhadap aplikasi atau sistem perangkat lunak yang digunakan dalam sistem informasi. Arsitektur aplikasi mencakup desain aplikasi, integrasi aplikasi, pemilihan teknologi, serta pola dan prinsip desain yang digunakan dalam pengembangan perangkat lunak.

4. Arsitektur Teknologi (*Technology Architecture*)
   Merupakan pandangan terhadap infrastruktur teknologi yang digunakan dalam sistem informasi. Arsitektur teknologi melibatkan perencanaan dan pengaturan perangkat keras, perangkat lunak, jaringan komunikasi, serta keamanan sistem yang mendukung operasional sistem informasi.
5. Arsitektur Keamanan (*Security Architecture*)
   Merupakan pandangan terhadap keamanan sistem informasi. Arsitektur keamanan mencakup kebijakan keamanan, pengendalian akses, perlindungan data, dan tindakan pencegahan terhadap ancaman keamanan seperti serangan siber.
6. Arsitektur Integrasi (*Integration Architecture*)
   Merupakan pandangan terhadap integrasi antara komponen sistem informasi yang berbeda. Arsitektur integrasi mencakup cara komponen sistem informasi berinteraksi, pertukaran data antara aplikasi, dan pengaturan aliran informasi antara sistem yang terhubung.
7. Arsitektur Pengelolaan (*Management Architecture*)
   Merupakan pandangan terhadap pengelolaan dan pemeliharaan sistem informasi. Arsitektur pengelolaan mencakup kebijakan, proses, dan alat yang digunakan untuk mengelola siklus hidup sistem informasi, termasuk pengembangan, implementasi, pengujian, pemeliharaan, dan peningkatan sistem.

Kerangka arsitektur sistem informasi ini membantu dalam merencanakan dan merancang sistem informasi yang koheren, terintegrasi, dan sesuai dengan kebutuhan organisasi. Kerangka arsitektur juga memungkinkan perubahan dan evolusi sistem informasi seiring waktu dengan tetap menjaga konsistensi dan keselarasan antara komponen-komponen yang ada.

1.5 Tahapan dalam Analisis dan Desain Sistem Informasi

Tahapan dalam Analisis dan Desain Sistem Informasi (ADSI) meliputi serangkaian langkah yang dilakukan untuk mengembangkan sistem informasi yang efektif. Berikut adalah tahapan umum dalam ADSI (Fatta 2007):



Gambar 1.2. tahapan umum dalam ADSI

1. Identifikasi Kebutuhan dan Permasalahan
   Tahap pertama dalam ADSI adalah mengidentifikasi kebutuhan bisnis dan permasalahan yang perlu dipecahkan melalui sistem informasi. Ini melibatkan wawancara dengan pemangku kepentingan, analisis dokumen, dan observasi langsung untuk memahami proses bisnis dan masalah yang ada.
2. Analisis Sistem yang Ada
   Pada tahap ini, sistem informasi yang sudah ada dievaluasi untuk memahami kelebihan, kekurangan, dan kesenjangan yang perlu diatasi. Analisis dilakukan dengan mempelajari dokumentasi, mewawancarai pengguna sistem, dan mengumpulkan data tentang kinerja sistem yang ada.
3. Perancangan Konsep
   Setelah pemahaman yang mendalam tentang kebutuhan dan permasalahan, tahap perancangan konsep dilakukan. Pada tahap ini, solusi yang mungkin diidentifikasi dan

dipertimbangkan. Berbagai alternatif dan strategi dirancang untuk memenuhi kebutuhan bisnis yang telah diidentifikasi.

4. Analisis Kebutuhan Sistem
   Tahap ini melibatkan pengumpulan dan dokumentasi kebutuhan sistem yang spesifik. Hal ini melibatkan identifikasi fungsi dan fitur yang diperlukan oleh sistem informasi yang baru.
5. Desain Sistem
   Tahap desain sistem melibatkan merancang struktur sistem yang mencakup arsitektur sistem, pemodelan data, antarmuka pengguna, dan proses bisnis. Rancangan ini berfungsi sebagai panduan untuk pengembangan dan implementasi sistem informasi yang baru.
6. Pengembangan dan Implementasi
   Pada tahap ini, sistem informasi yang dirancang dibangun dan diimplementasikan. Ini melibatkan pemrograman, pengujian, dan penerapan sistem. Pengguna juga dilibatkan dalam tahap ini untuk pelatihan dan adaptasi sistem baru.
7. Pengujian dan Validasi
   Tahap ini melibatkan pengujian sistem informasi untuk memastikan bahwa sistem berjalan dengan baik dan memenuhi persyaratan yang telah ditetapkan. Tes dilakukan untuk memverifikasi fungsi, kehandalan, dan kinerja sistem.
8. Pemeliharaan dan Evaluasi
   Setelah implementasi, sistem informasi memerlukan pemeliharaan rutin untuk memastikan kinerjanya yang optimal. Evaluasi dilakukan untuk mengukur keberhasilan sistem dan mengidentifikasi area perbaikan atau peningkatan yang mungkin diperlukan.

Tahapan dalam ADSI dapat bervariasi tergantung pada metodologi dan pendekatan yang digunakan, namun prinsip dasarnya tetap sama. Tahapan ini membantu memastikan bahwa sistem informasi yang dikembangkan memenuhi kebutuhan bisnis, efektif, dan memberikan manfaat yang diharapkan.

1.6 Stakeholder

Stakeholder dalam sistem informasi adalah individu, kelompok, atau entitas yang memiliki kepentingan dan pengaruh terhadap sistem informasi yang sedang dikembangkan atau digunakan. Stakeholder dapat terlibat dalam berbagai tahap siklus hidup sistem informasi, mulai dari analisis kebutuhan hingga implementasi dan pemeliharaan. Berikut adalah beberapa contoh stakeholder yang umum dalam sistem informasi yang berinteraksi dengan analis sistem (Kadir 2014):

Gambar 1.3. Stake Holder dalam Sistem Informasi

1. Pemilik atau Pemangku Kepentingan Bisnis
   Individu atau kelompok yang memiliki kepemilikan atau tanggung jawab terhadap organisasi atau unit bisnis yang akan menggunakan sistem informasi. Pemilik bisnis sering kali memiliki tujuan dan kebutuhan bisnis yang harus dipenuhi oleh sistem informasi.
2. Pengguna Akhir
   Pengguna akhir adalah individu atau kelompok yang akan menggunakan langsung sistem informasi. Pengguna akhir dapat termasuk karyawan, manajer, pelanggan, atau mitra bisnis yang menggunakan sistem untuk menjalankan tugas sehari-hari, mengambil keputusan, atau berinteraksi dengan organisasi.
3. Tim Manajemen Proyek
   Tim manajemen proyek terdiri dari individu yang bertanggung jawab untuk mengelola pengembangan sistem informasi. Mereka dapat mencakup manajer proyek, analis sistem, arsitek, pengembang perangkat lunak, dan spesialis keamanan. Tim manajemen proyek bertanggung jawab untuk memastikan bahwa sistem informasi dikembangkan sesuai dengan persyaratan dan kebutuhan bisnis.
4. Tim Teknis
   Tim teknis meliputi para ahli IT dan profesional teknis yang terlibat dalam pengembangan, implementasi, dan pemeliharaan sistem informasi. Tim teknis dapat mencakup administrator basis data, pengembang perangkat lunak, administrator jaringan, dan spesialis keamanan. Tim teknis bertanggung jawab untuk memastikan infrastruktur teknis yang diperlukan tersedia dan sistem informasi berfungsi dengan baik.
5. Pemasok dan Vendor
   Pemasok dan vendor adalah entitas eksternal yang menyediakan solusi perangkat keras, perangkat lunak, atau

layanan yang digunakan dalam sistem informasi. Mereka dapat termasuk produsen perangkat keras, pengembang perangkat lunak, penyedia layanan cloud, atau konsultan teknologi. Pemasok dan vendor memiliki peran penting dalam menyediakan solusi dan mendukung implementasi sistem informasi.

6. Regulator dan Pihak Eksternal
   Pihak eksternal, seperti regulator, pihak berwenang, atau mitra bisnis, juga dapat menjadi stakeholder dalam sistem informasi. Mereka mungkin memiliki persyaratan khusus yang harus dipenuhi oleh sistem informasi, atau mereka dapat mempengaruhi kebijakan dan regulasi terkait penggunaan sistem.

Peran dan tanggung jawab seorang analis sistem sangat penting dalam ADSI. Analisi sistem bertanggung jawab untuk memahami kebutuhan bisnis, menganalisis proses bisnis yang ada, merancang solusi yang sesuai, dan memastikan implementasi yang berhasil. Berikut adalah peran dan tanggung jawab utama seorang analis sistem:

Tabel 1.2. Peran dan Tanggung Jawab Analis Sistem

Peran	**Tanggung Jawab**
Memahami Kebutuhan Bisnis	Mampu memahami kebutuhan bisnis organisasi dengan berinteraksi dan mendengarkan pemangku kepentingan. Identifikasi tantangan bisnis, masalah, atau peluang yang dapat dipecahkan melalui sistem informasi.
Menganalisis Proses Bisnis	Melakukan analisis mendalam terhadap proses bisnis yang ada dalam organisasi. Memahami alur kerja, kebutuhan data, dan interaksi antara pemangku kepentingan yang terlibat dalam proses tersebut.

Peran	Tanggung Jawab
Merancang Solusi Sistem Informasi	Merancang solusi sistem informasi yang sesuai berdasarkan pemahaman tentang kebutuhan bisnis dan analisis proses bisnis. Termasuk desain struktur sistem, antarmuka pengguna, pemodelan data, dan proses bisnis.
Dokumentasi Persyaratan dan Spesifikasi	Mendokumentasikan persyaratan sistem secara jelas dan terperinci. Termasuk kebutuhan fungsional dan non-fungsional, aliran data, kebutuhan antarmuka pengguna, serta persyaratan keamanan dan performa.
Berinteraksi dengan Tim Proyek	Bekerja sama dengan pemangku kepentingan dan anggota tim proyek, seperti pengembang, arsitek, dan desainer. Berkomunikasi secara efektif untuk memastikan pemahaman yang sama dan mengoordinasikan upaya pengembangan sistem informasi.
Pengujian dan Validasi	Membantu merencanakan dan melaksanakan uji coba sistem. Memastikan sistem informasi berfungsi sesuai persyaratan yang ditetapkan dan memvalidasi bahwa solusi yang dirancang memenuhi kebutuhan bisnis.
Pelatihan dan Pendampingan Pengguna	Melatih pengguna tentang penggunaan sistem yang baru setelah implementasi. Memberikan dukungan dan pendampingan selama periode transisi.
Pemeliharaan dan Perbaikan	Bertanggung jawab dalam pemeliharaan sistem informasi yang ada. Memantau kinerja sistem, mengidentifikasi masalah, dan mengusulkan perbaikan atau peningkatan yang diperlukan.

Sebagai seorang analis sistem, terdapat beberapa keahlian yang penting untuk dimiliki guna melakukan tugas-tugas analisis

dan desain sistem. Berikut adalah beberapa keahlian yang diperlukan:

Gambar 1.4. Keahlian yang Diperlukan Analis Sistem

1. Pemahaman Bisnis
   Seorang analis sistem harus memiliki pemahaman yang kuat tentang bisnis atau industri tempat sistem akan digunakan. Ini termasuk pemahaman tentang proses bisnis, tujuan organisasi, dan kebutuhan pengguna. Kemampuan untuk menerjemahkan kebutuhan bisnis menjadi persyaratan sistem yang jelas sangat penting.
2. Kemampuan Analitis
   Analis sistem harus memiliki kemampuan analitis yang baik untuk menganalisis masalah dan situasi bisnis, mengidentifikasi kebutuhan, dan merumuskan solusi yang tepat. Ini melibatkan kemampuan dalam memecah masalah besar menjadi komponen yang lebih kecil, melakukan

analisis kebutuhan, dan mengidentifikasi aspek-aspek kritis yang mempengaruhi desain sistem.

3. Pemodelan dan Notasi
   Kemampuan untuk memodelkan dan mewakili informasi dengan jelas menggunakan notasi yang tepat adalah keterampilan penting bagi seorang analis sistem. Ini meliputi pemodelan proses bisnis menggunakan alat seperti Diagram Alir Data (Data Flow Diagram/DFD), pemodelan struktur menggunakan Diagram *Entity-Relationship* (ERD), dan pemodelan objek menggunakan *Unified Modeling Language* (UML).
4. Komunikasi dan Kolaborasi
   Seorang analis sistem harus memiliki kemampuan komunikasi yang baik untuk berinteraksi dengan pemangku kepentingan yang beragam, termasuk pengguna bisnis, manajemen, dan tim teknis. Kemampuan untuk mendengarkan dengan baik, mengajukan pertanyaan yang tepat, dan menjelaskan konsep-konsep kompleks secara sederhana sangat penting. Kolaborasi dengan tim proyek dan pemangku kepentingan lainnya juga diperlukan.
5. Pengetahuan Teknis
   Pengetahuan teknis yang solid tentang konsep dan prinsip sistem informasi, infrastruktur teknologi, serta perkembangan dan tren terkini dalam bidang teknologi informasi sangat penting. Ini membantu analis sistem dalam memahami keterkaitan antara kebutuhan bisnis dan solusi teknologi yang ada atau yang dapat dikembangkan.
6. Keterampilan Manajemen Proyek
   Kemampuan manajemen proyek membantu analis sistem dalam merencanakan, mengelola, dan mengarahkan proyek pengembangan sistem informasi. Ini melibatkan pemahaman tentang pengelolaan risiko, perencanaan jadwal, anggaran, dan sumber daya, serta kemampuan

untuk memprioritaskan tugas dan mengelola perubahan dalam lingkungan proyek.

7. Keahlian dalam Alat dan Teknologi
   Analis sistem harus memiliki pemahaman dan keahlian dalam penggunaan alat dan teknologi yang relevan untuk analisis dan desain sistem. Ini termasuk alat pemodelan, alat manajemen persyaratan, alat pengujian, dan alat kolaborasi. Pengetahuan tentang perkembangan terbaru dalam teknologi juga penting untuk memahami potensi solusi yang dapat diterapkan.

1.7 Pendekatan-Pendekatan

Dalam Analisis dan Desain Sistem Informasi (ADSI), terdapat beberapa pendekatan yang digunakan untuk memahami, merancang, dan mengimplementasikan sistem informasi. Berikut adalah beberapa pendekatan yang umum digunakan dalam ADSI (Muslihudin dan Oktafianto 2016):

1. Pendekatan Terstruktur (*Structured Approach*)
   Pendekatan terstruktur berfokus pada pemecahan masalah secara sistematis dengan mengidentifikasi langkah-langkah yang jelas dan berurutan. Pendekatan ini menggunakan metode analisis dan desain yang terstruktur, seperti Diagram Alir Data (Data Flow Diagram/DFD) dan Diagram Hierarchy of Modules (Hierarchy Input Process Output/HIPO). Pendekatan terstruktur mengedepankan pemisahan antara logika proses dan data(Dixit 2007).
2. Pendekatan Berorientasi Obyek (*Object-Oriented Approach*)
   Pendekatan berorientasi obyek memandang sistem informasi sebagai kumpulan objek yang saling berinteraksi untuk mencapai tujuan bisnis. Pendekatan ini menggunakan konsep-konsep seperti kelas, objek, pewarisan, dan polimorfisme. Pemodelan berorientasi obyek yang umum

digunakan adalah Unified Modeling Language (UML)(Fairuzabadi 2021).

3. Pendekatan Campuran (*Hybrid Approach*)
   Pendekatan campuran menggabungkan elemen-elemen dari pendekatan terstruktur dan berorientasi obyek. Pendekatan ini memanfaatkan kelebihan keduanya untuk merancang sistem informasi yang lebih fleksibel dan efisien. Pendekatan campuran memungkinkan penggunaan DFD untuk menganalisis aliran data dan UML untuk memodelkan struktur obyek.
4. Pendekatan Berbasis Komponen (*Component-Based Approach*):
   Pendekatan berbasis komponen memandang sistem informasi sebagai kumpulan komponen yang dapat digunakan kembali. Komponen-komponen ini memiliki fungsionalitas yang terdefinisi dengan baik dan dapat digabungkan untuk membentuk sistem yang lebih kompleks(Cesare, Lycett, dan Macredie 2015).

Pilihan pendekatan ADSI tergantung pada karakteristik proyek, kebutuhan bisnis, dan preferensi organisasi. Beberapa proyek mungkin lebih cocok dengan pendekatan terstruktur, sementara yang lain mungkin memerlukan pendekatan berorientasi obyek atau campuran.

DAFTAR PUSTAKA

Cesare, Sergio de, Mark Geoffrey Lycett, dan Robert Macredie. 2015. *The Development of Component-based Information Systems*. New York City, Amerika Serikat: Taylor & Francis.

Dixit, J.B. 2007. *Structured System Analysis and Design*. New Delhi, India: Laxmi Publications Pvt. Limited.

Fairuzabadi, Muhammad. 2021. "Pengembangan Project Based E-Learning dengan Analisis PIECES dan Desain UML." 10(2): 39–50.

———. 2023. *Enterprise Information System: Integrasi dan Pengelolaan Bisnis yang Efektif*. Medan: Yayasan Kita Menulis.

Fatta, Hanif Al. 2007. *Analisis dan Perancangan Sistem Informasi untuk Keunggulan Bersaing Perusahaan dan Organisasi Modern*. Yogyakarta, Indonesia: Penerbit Andi.

Kadir, Abdul. 2014. *Pengenalan Sistem Informasi*. Revisi. Yogyakarta: ANDI.

Kustiyahningsih, Yeni, dan Eza Rahmanita. 2022. *Arsitektur Sistem Informasi Perusahaan*. Malang, Indonesia: Media Nusa Creative.

Muslihudin, Muhamad, dan Oktafianto. 2016. *Analisis dan Perancangan Sistem Informasi Menggunakan Model Terstruktur dan UML*. Yogyakarta, Indonesia: Penerbit Andi.

Senn, James A. 2012. *Analisis dan Desain Sistem Informasi*. 6 ed. Penerbit Erlangga.

Tilley, Scott. 2020. *Systems Analysis and Design*. 4 ed. Boston, MA: Cengage Learning.

Whitten, Jeffrey L., dan Lonnie Bentley D. 2008. *Introduction to Systems Analysis and Design*. New York City, Amerika Serikat: McGraw Hill Irwin.

BAB 2
ANALISIS KEBUTUHAN SISTEM BERORIENTASI OBJEK

Oleh Irani Hoeronis

2.1 Pengertian Analisis Kebutuhan Sistem Berorientasi Objek

Sistem berorientasi objek didasarkan pada pendekatan dalam memandang permasalahan sistem sebagai sekumpulan objek yang saling berkorespondensi dengan berbagai macam objek di dunia nyata (Nazaruddin et al., 2022). Abstraksi dan model objek-objek tersebut dienkapsulasi menjadi sebuah kesatuan. Komponen yang dapat dienkapsulasi mencakup atribut, sifat, dan bentuk interaksi terhadap komponen lainnya. Beberapa sifat atau karakteristik dari sistem berorientasi objek diantaranya adalah:

1. Abstraksi
   Representasi dari dunia nyata dalam bentuk model sesuai dengan aspek-aspek permasalahan yang akan diselesaikan
2. Enkapsulasi
   Proses pengikatan data atau metode yang berbeda sehingga dapat memudahkan dalam proses pembacaan dan penyajian informasi. Detail sebuah objek disembunyikan dari akses luar untuk mencegah modifikasi data dan membuat kode menjadi lebih aman.
3. Pewarisan
   Teknik yang menunjukkan bahwa objek memiliki anak induk yang mewarisi data/atribut. Objek dan anak objek memiliki hubungan hirarki dan membentuk kelas dan subkelas. Hubungan hirarki antar subkelas dapat ditentukan dengan

melihat sifat unik yang dimiliki subkelas anak objek tersebut ditambah dengan sifat yang diwariskan oleh induk objek.

4. *Reusability*
   Penggunaan ulang kelas-kelas atau bagian-bagian yang sudah dibuat untuk dimanfaatkan pada permasalahan sejenis. Salah satu kelebihan sistem berbasis objek adalah *reusability*.
5. Generalisasi dan spesialisasi
   Pendekatan yang menunjukkan hubungan antar kelas dan objek. Generalisasi dapat menghasilkan superkelas dengan pendekatan bottom-up, sedangkan spesialisasi dapat menghasilkan himpunan subkelas dengan pendekatan top-down.
6. Komunikasi antar objek
   Objek yang satu dengan yang lain berkomunikasi melalui pesan (*message*) yang dikirimkan.
7. *Polymorphism*
   Sebuah konsep pada kelas banyak yang saling berkorespondensi dengan mewarisi atribut, metode, atau fungsi dari kelas. *Polymorphism* dapat menggunakan fungsi pewarisan untuk melakukan aksi yang berbeda tanpa membuat fungsi khusus baru dengan cara-cara yang berbeda pula.

Kebutuhan sistem berorientasi objek didasarkan pada sekumpulan objek yang merupakan representasi dunia nyata berupa struktur data dan perilaku entitas tertentu. Kebutuhan tersebut dianalisis dalam konsep orientasi objek dengan membedakan konsep struktural yang sebelumnya berkembang. Analisis kebutuhan sistem berorientasi objek merupakan metode analisis yang memuat kebutuhan (*requirement*) sistem dalam membangun struktur data berdasarkan kelas dan objek sesuai dengan ruang lingkup permasalahan. Analisis kebutuhan sistem berorientasi objek ditekankan pada proses analisis sistem

informasi yang efektif dengan membuat konsep sistem baru yang dapat menjelaskan kebutuhan sistem (Fergina & Gisni Ariyanti, 2022).

2.2 Kebutuhan Dasar dalam Analisis Berorientasi Objek

Kebutuhan dalam analisis sistem berorientasi objek mencakup kebutuhan fungsional dan non fungsional. Kebutuhan fungsional merupakan kebutuhan yang berkaitan dengan fungsi sistem, sedangkan kebutuhan non fungsional berkaitan dengan karakteristik dan batasan kemampuan sistem.

Kebutuhan fungsional sistem disesuaikan dengan perkembangan teknologi informasi yang bersifat dinamis. Hal tersebut memaksa sistem untuk bisa beradaptasi dengan meningkatkan kemampuan teknologi dalam pengolahan informasi kompleks. Sistem berorientasi objek dapat mengurangi kompleksitas melalui notasi yang digunakan pada setiap tahapan pengembangan sistem. Mulai dari tahap analisis, perancangan, dan implementasi. Tingkat abstraksi tinggi yang mendekati dunia nyata juga dimiliki sistem berorientasi objek.

Sistem berorientasi objek memerlukan komponen dasar pembentuk utama berupa kelas dan objek. Objek merupakan abstraksi yang merepresentasikan dunia nyata seperti orang, benda, tempat atau hal abstrak lainnya. Sedangkan kelas merupakan sekumpulan objek yang memiliki sifat atau karakteristik sama. Sebuah kelas memiliki atribut, metode, hubungan, dan arti yang mempunyai sifat pewarisan.

Objek merepresentasikan perpaduan antara struktur data dan perilaku pada suatu entitas dengan pembeda nilai antar entitas tertentu. Setiap objek memiliki identitas tertentu yang memiliki nilai pembeda antar objek. Hal tersebut menjadikan objek sebagai identitas yang dapat diukur dan bersifat konkrit.

Kedua komponen dasar membentuk hubungan antar objek dan kelas. Keterhubungan tersebut menunjukkan hirarki dan menjelaskan relasi yang dapat dianalisis perilakunya dalam sistem.

Objek memiliki tipe objek yang dibangun dalam sistem komputer. Sebuah sistem komputer dapat memiliki lebih dari satu tipe objek. Beberapa tipe objek di dalam sistem komputer meliputi objek *User Interface* (UI), objek domain sistem, dan objek domain masalah (Pelawi, 2013). Objek UI merupakan objek yang langsung berinteraksi dengan user melalui tombol dalam sistem. Objek domain sistem merupakan objek yang operasi dalam sistem yang menunjukkan perilaku atau operasi yang dapat dilakukan objek. Objek domain masalah merupakan objek yang diperuntukkan bagi penyelesaian masalah tertentu (Randa, 2022).

Proses analisis berorientasi objek dilakukan dalam beberapa tahap analisis, diantaranya dengan mendokumentasikan fungsional sistem menggunakan *use case,* pengelompokkan objek, membangun hirarki kelas, dan menggambarkan keterhubungan antar objek dan kelas, serta menganalisis perilaku sistem (Anardani, 2019).

Dalam pengembangan sistem, pendekatan berorientasi objek tidak memandang *software* sebagai fungsi dan data saja. Pendekatan berorientasi objek menggabungkan data dan fungsionalitas secara bersamaan. Abstraksi dan konsep berorientasi objek dapat dilakukan pada keseluruhan proses sistem (Maria et al., 2019).

2.3 Teknik dalam Analisis Kebutuhan Sistem Berorientasi Objek

Pendekatan berorientasi objek mengalami perkembangan. Pendekatan atau model berorientasi objek dilakukan dengan memodularisasi masalah dalam pemodelan konseptual (Pastor et al., 2001). Proses-proses penyederhanaan model dalam memandang dunia nyata tidak hanya dipahami secara konseptual

tetapi juga bagaimana dapat diterjemahkan dalam sistem perangkat lunak (Nerson, 1992). Model objek dan kelas yang dibangun memuat aktivitas utama dalam sistem berorientasi objek, diantaranya adalah:

1. Analisis domain masalah
2. Penjelasan proses sistem
3. Identifikasi objek
4. Penentuan atribut
5. Definisi operasi
6. Komunikasi antar objek

Analisis kebutuhan dalam sistem berorientasi objek dibedakan menjadi 4 aktifitas utama (Colbert, 1989), yaitu:

1. Spesifikasi interaksi-objek
   Aktifitas dalam spesifikasi interaksi-objek mengidentifikasi objek, interaksi antar objek, dan hirarki hubungan antar objek. Analisis yang dibangun meliputi analisis interaksi-objek dan analisis hirarki objek. Analisis interaksi-objek mendeskripsikan sekumpulan objek dan interaksi antar objek tersebut. Analisis hirarki-objek dihasilkan secara otomatis dari pembentukan analisis relasi antar objek.
   Dalam pendekatan spesifikasi interaksi-objek, adaptasi dilakukan tidak hanya proses yang digambarkan secara *outside-in* tetapi juga memperhatikan *behavioral partitioning* dengan menganalisis masalah perilaku yang terjadi dan menggunakan informasi tersebut untuk menentukan objek.
2. Spesifikasi kelas-objek
   Aktifitas dalam spesifikasi kelas-objek mengidentifikasi kelas objek dan hubungan antar kelas. Analisis kelas-objek mengidentifikasi kelas setiap objek dan informasi interaksi-objek dalam sistem. Identifikasi kelas sebuah objek, struktur dan modularitas perilaku, dan propertis objek dapat menentukan ketepatan kelas.

3. Spesifikasi perilaku
   Aktifitas dalam spesifikasi perilaku diidentifikasi pada perilaku dinamik dan setiap objek pada sistem. Terdapat tiga (3) tahap utama dalam menentukan spesifikasi perilaku:
   a. Mendeskripsikan perilaku sistem dengan mendefinisikan masalah dan menganalisis ketepatan dan kelengkapannya.
   b. Untuk setiap objek dalam model, mendefinisikan perilaku internal dan eksternal. Perilaku eksternal menunjukkan relasi objek terhadap objek lain diluar operasi objek tersebut. Perilaku internal menunjukkan relasi objek terhadap operasi dalam objek tersebut.
   c. Menunjukkan perilaku seluruh objek dalam model tersebut benar. Perilaku yang benar dapat ditunjukkan untuk setiap objek jika implementasi objek tersebut juga benar. Proses implementasi objek ditunjukkan pada perilaku objek yang berkorelasi dengan eksternal dan internal. Selain itu juga, perilaku objek pada perilaku eksternal dan internal secara tepat menunjukkan korelasi dan dapat diperbandingkan. Kebutuhan perilaku yang didefinisikan oleh deskripsi masalah juga secara tepat diimplementasikan dalam perilaku internal dan eksternal objek.
4. Spesifikasi atribut
   Aktifitas dalam spesifikasi atribut mengidentifikasi pengukuran kuantitatif, pengukuran kualitatif, dan sumber daya untuk setiap objek dalam sistem.

Selain aktifitas yang dapat digunakan untuk memahami keseluruhan proses sistem, kebutuhan sistem berorientasi objek juga dapat diketahui melalui analisis metode. Metodologi orientasi objek dalam menganalisis kebutuhan sistem dapat menggunakan beberapa metode, diantaranya:

1. Metode Booch
   Merupakan metode dengan proses analisis yang dilakukan berdasarkan empat (4) tahap iteratif berupa identifikasi kelas dan objek, identifikasi semantik, relasi antar objek dan kelas tersebut, rincian interface dan implementasi. Metode ini dikenal juga dengan nama Metode Desain Objek Oriented.
2. Metode Rumbaurgh (*Object Modelling Technique*-OMT)
   Merupakan metode yang didasarkan pada analisis terstruktur dengan pemodelan entity-relationship. Metode ini dapat digunakan pada tahap analisis dan perancangan. Tahapan dalam metode ini adalah analisis, desain sistem dan objek, serta implementasi. Salah satu keunggulan metode ini adalah bentuk penotasian dengan konsep object oriented.
3. Metode Jacobson (*Object Oriented Software Engineering*-OOSE)
   Merupakan metode yang memuat elemen-elemen lain dari object oriented dan lebih fokus pada use-case. Tahapan dalam metode ini adalah pembuatan kebutuhan model dan analisis model, rancangan dan implementasi, serta pengujian model. Metode ini memiliki keunggulan berupa notasi yang sederhana sehingga lebih mudah untuk dipelajari.
4. Metode Coad dan Yourdan
   Merupakan metode yang berdasarkan pada pemodelan object oriented dan entity-relationship. Tahapan perancangan meliputi empat (4) komponen, yaitu komponen domain masalah, komponen interaksi manusia, komponen manajemen data, dan komponen manajemen tugas.
5. Metode Wirfs-Brock
   Merupakan metode pada proses perancangan tetapi dapat memunculkan ide sehingga dapat digunakan dalam tahap analisis. Salah satu keunggulan metode ini adalah terdapatnya identifikasi hirarki kelas dan subsistem. Hal tersebut dapat memudahkan ketika proses perancangan dan analisis dilakukan.

6. Metode Shlair-Mellor *Object Oriented Analysis/Design* (OOA/D) Merupakan metode dengan menggunakan teknik pemodelan tradisional berupa penjelasan entitas di dalam sistem. State diagram digunakan untuk memodelkan suatu keadaan state entitas. Data flow diagram digunakan untuk memodelkan alur data dalam sistem. Terdapat tiga (3) jenis model yang dihasilkan dari metode ini, diantaranya adalah: model informasi, model state, dan model proses. Beberapa proses yang dilakukan dapat memudahkan dalam menganalisis masalah pada view of point yang berbeda serta lebih mudah dilakukan konversi dari model struktural.

2.4 Perbandingan dengan Pendekatan Sistem Terstruktur dalam Pendekatan Sistem Berorientasi Objek

Perbedaan antara pendekatan sistem terstruktur dan pendekatan sistem berorientasi objek dijelaskan pada tabel 2.1.

Tabel 2.1. Perbedaan Pendekatan Terstruktur dan Pendekatan Berorientasi Objek

Pendekatan Terstruktur	**Pendekatan Berorientasi Objek**
Sistem berfokus pada proses	Sistem berfokus pada objek-objek yang berinteraksi dengan dunia nyata
Proses tersebut menerima input yang saling berinteraksi dengan entitas data, kemudian menghasilkan output	Objek merespon dan mengirim pesan
Pendekatan fungsional	Pendekatan objek
Dekomposisi masalah dilakukan secara hirarki	Dekomposisi masalah dilakukan berdasarkan objek

Pendekatan Terstruktur	Pendekatan Berorientasi Objek
berdasarkan proses atau fungsi	dalam sistem
Tidak fokus pada coding	Fokus pada coding
Tools: DFD, Kamus Data, Entity Relationship Diagram (ERD), State Transition Diagram (STD)	Tools: Rational Unified Process (RUP), Fusion, STD Development Method 3, Berard's Object-Oriented Design, Booch's Object-Oriented Design, Coad and Yourdon's Object-Oriented Analysis, Jacobson's Objectory, Rumbaugh's Object Modelling Technique (OMT), Object-Oriented System Analysis (OOA)

DAFTAR PUSTAKA

Anardani, S. 2019. *Perancangan Sistem Berorientasi Objek dengan Pemodelan UML (Unified Modelling Language) Tools*. UNIPMA Press.

Colbert, E. 1989. The object-oriented software development method: A practical approach to object-oriented development. *Proceedings of the Conference on Tri-Ada 1989: Ada Technology in Context: Application, Development, and Deployment, TRI-Ada 1989*, 400–415. https://doi.org/10.1145/74261.74291

Fergina, A., & Gisni Ariyanti. 2022. Perancangan Sistem Peminjaman Buku Perpustakaan Mdta Attawakkal Dengan Metodologi Berorientasi Obyek. *Jurnal RESTIKOM : Riset Teknik Informatika Dan Komputer*, *2*(2), 79–84. https://doi.org/10.52005/restikom.v2i2.68

Maria, Y. A., Fajar, M., & Afifah. 2019. Implementasi Pendekatan Berorientasi Objek Terpadu Untuk Pengembangan Sistem Informasi Yang Terintegrasi Pada PT. Sapta Karya. *Jurnal Ilmu Komputer*, 24–37.

Nazaruddin, A., Krisnanik, E., Muliawati, A., & Syamsiah, N. 2022. Analisa Perancangan Sistem Informasi Berorientasi Objek. In *Widina Media Utama*. Widina Media Utama.

Nerson, J. 1992. Applying Analysis and Design. *Communications of the ACM*, *35*(9).

Pastor, O., Gómez, J., Insfrán, E., & Pelechano, V. 2001. The OO-Method approach for information systems modeling: From object-oriented conceptual modeling to automated programming. *Information Systems*, *26*(7), 507–534.

Pelawi, D. 2013. *Pembuatan Sistem Informasi… (Dewan Pelawi)*. *9*, 213–224.

Randa, D. D. 2022. Analisa dan perancangan berorientasi objek pada sistem informasi geografis untuk menentukan lokasi irigasi sungai. *JRTI (Jurnal Riset Tindakan Indonesia)*, *7*(2), 84. https://doi.org/10.29210/30031525000

BAB 3
DESAIN MODEL PROSES BISNIS PENDEKATAN TERSTRUKTUR

Oleh Zen Munawar

3.1 Pendahuluan

Desain model proses bisnis pendekatan terstruktur merupakan pendekatan pengembangan sistem informasi yang paling tersebar luas dan populer saat ini. Ini adalah metode sederhana. Dengan mempelajari konsep dasar dapat membantu seseorang mengembangkan keterampilan dan pengetahuan yang diperlukan untuk memulai dan mengelola bisnis yang sukses, terutama dalam bidang teknologi dan inovas (Munawar, Pratomo, *et al.*, 2023). Mudah dipahami, dan menangkap sebagian besar fitur yang relevan dari pemrosesan data berorientasi transaksi konvensional. Pendekatan Terstruktur adalah metode pengembangan yang memungkinkan analis untuk memahami sistem dan aktivitasnya secara logis. Ada dua fase proyek utama: Analisis Terstruktur dan Desain Terstruktur. Fase analisis sering dilihat sebagai pengembangan apa yang disebut spesifikasi kebutuhan fungsional. Fase desain dilihat sebagai pengembangan perangkat lunak dan solusi basis data yang memenuhi persyaratan fungsional, serta persyaratan non-fungsional, misalnya kinerja dan keamanan (Sølvberg, 1993). Dalam kehidupan sehari-hari, orang dapat mendapatkan kemudahan dengan penggunaan komputer yang mudah dibawa kemana-mana (Munawar, Hasnawi, *et al.*, 2023). Dengan bantuan komputer maka dapat melakukan kegiatan analisis dan desain dengan lebih mudah.

Analisis dan desain terstruktur adalah metode yang berorientasi proses, karena titik awal pengembangan sistem informasi adalah analisis proses informasi sistem. Proses ini dijelaskan dengan menggunakan apa yang disebut Data Flow Diagram (DFD). Dari titik awal ini, komponen sistem utama lainnya dari data, aturan, dan objek program ditemukan dan dirancang. Ada alat deskripsi berbeda yang akan digunakan untuk masing-masing komponen sistem utama ini.

Metode ini terutama didasarkan pada penggunaan alat diagram. Metode analisis dan desain terstruktur dikembangkan selama tahun 1970-an, sebelum workstation dan PC menjadi hal biasa. Alat diagram dirancang untuk lingkungan kertas dan pensil. Asal-usul sederhana ini dapat dengan mudah terlihat di alat CASE (*Computer Aided Software Engineering*) yang kuat yang digunakan saat ini untuk mendukung metode yang tersebar luas ini (DeMarco and Plauger, 1979).

Dalam metode asli tidak ada langkah desain eksplisit untuk mengembangkan antarmuka pengguna. Ini dapat ditafsirkan sebagai indikasi yang jelas bahwa antarmuka pengguna tidak dilihat sebagai bagian desain yang sangat penting di awal tahun 1970-an. Masalah desain lainnya mengambil preferensi. Masalah desain antarmuka pengguna saat ini jauh lebih penting, dan didukung oleh sebagian besar alat CASE.

3.2 Analisis Terstruktur

Spesifikasi fungsional dimaksudkan untuk mendeskripsikan semua yang dilihat pengguna tentang perangkat lunak dan interaksi antara perangkat lunak dan lingkungannya. Deskripsi fungsional harus diberikan dalam terminologi yang akrab bagi pengguna. Artinya, properti sistem harus dijelaskan dengan menggunakan terminologi area aplikasi komputer tertentu yang sedang diselidiki. Jika area aplikasinya adalah perbankan, spesifikasi fungsionalnya harus diungkapkan dalam terminologi

bankir; jika itu adalah pembuatan widget maka terminologi komunitas widget tertentu harus digunakan. Deskripsi fungsional juga harus menahan diri untuk tidak menjelaskan detail implementasi yang tidak dilihat pengguna. Itu urusan pelaksana. Spesifikasi implementasi harus diutarakan dalam bahasa komputer dan berisi sejumlah detail tambahan. Detail ini mungkin kurang menarik bagi pengguna dan juga sulit dipahami dalam waktu terbatas yang ingin dihabiskan pengguna untuk itu.

3.2.1 Spesifikasi Fungsional

Spesifikasi fungsional dimaksudkan untuk mendeskripsikan semua yang dilihat pengguna tentang perangkat lunak dan interaksi antara perangkat lunak dan lingkungannya. Deskripsi fungsional harus diberikan dalam terminologi yang akrab bagi pengguna. Artinya, properti sistem harus dijelaskan dengan menggunakan terminologi area aplikasi komputer tertentu yang sedang diselidiki. Jika area aplikasinya adalah perbankan, spesifikasi fungsionalnya harus diungkapkan dalam terminologi bankir; jika itu adalah pembuatan widget maka terminologi komunitas widget tertentu harus digunakan. Deskripsi fungsional juga harus menahan diri untuk tidak menjelaskan detail implementasi yang tidak dilihat pengguna. Itu urusan pelaksana. Spesifikasi implementasi harus diutarakan dalam bahasa komputer dan berisi sejumlah detail tambahan. Detail ini mungkin kurang menarik bagi pengguna dan juga sulit dipahami dalam waktu terbatas yang ingin dihabiskan pengguna untuk itu.

Analisis Terstruktur adalah salah satu pendekatan khusus untuk mengabstraksi rincian implementasi dan berkonsentrasi pada penggambaran fitur-fitur sistem informasi yang sangat penting bagi pengguna. Spesifikasi fungsional yang dihasilkan dari penerapan pendekatan ini juga harus membentuk dasar yang efektif untuk desain dan implementasi selanjutnya dari perangkat lunak.

Jadi dokumen spesifikasi harus masuk akal baik bagi pengguna maupun perancang perangkat lunak. Pengalaman menunjukkan bahwa ini adalah tugas yang sangat sulit yang dalam keadaan seni saat ini tidak dilakukan dengan memuaskan. Alat Analisis Terstruktur adalah: Diagram aliran data (DFD); Alat untuk menjelaskan aturan pemrosesan; Diagram akses data. Alat-alat ini memberi analis sistem sarana untuk melakukan analisis disiplin, dan menyediakan produk akhir analisis - spesifikasi fungsional. Tujuan penggunaan alat-alat ini adalah persiapan spesifikasi fungsional yang dipahami dengan baik oleh pengguna, menetapkan persyaratan fungsional sistem tanpa mendikte implementasinya, mengungkapkan preferensi pengguna dan trade-off sistem. Kami akan menggunakan contoh untuk mendukung deskripsi pendekatan Analisis Terstruktur (Gane and Sarson, 1979).

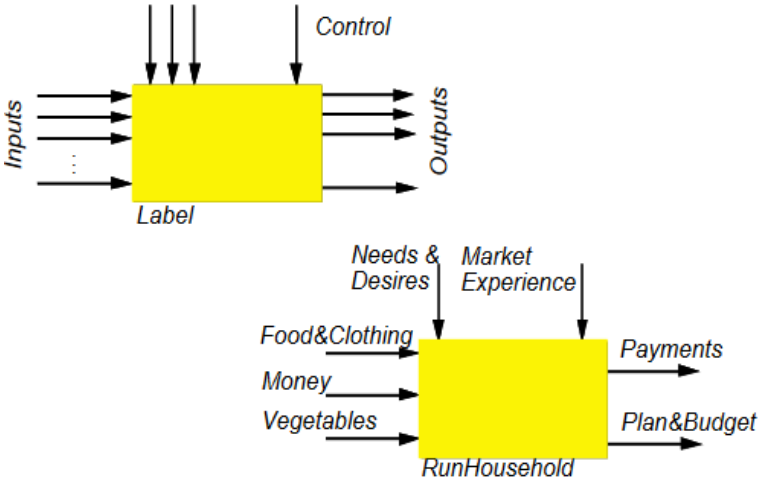
3.2.2 Analisis Terstruktur dan Desain Terstruktur

Analisis Terstruktur dan Desain Terstruktur adalah notasi diagram yang dirancang untuk membantu orang memahami sistem. Tujuan dasar analisis terstruktur dan desain analisis adalah untuk meningkatkan kualitas dan mengurangi risiko kegagalan sistem. Ini menetapkan spesifikasi dan dokumentasi manajemen yang konkret. Ini berfokus pada soliditas, kelenturan, dan pemeliharaan sistem. Konsep teoretis desain terstruktur dan membuatnya berlaku untuk pengembangan perangkat lunak dunia nyata (Page-Jones, 1988).

Analisis Terstruktur dan Desain Terstruktur adalah notasi diagram untuk membuat sketsa aplikasi. Menawarkan kotak untuk mewakili entitas dan aktivitas. Menawarkan berbagai panah untuk menghubungkan kotak. Kotak dan panah memiliki semantik (informal) terkait; pengguna dibantu oleh label kotak dan panah, dokumentasi informal lainnya. Telah menginspirasi banyak alat komersial lainnya. Telah digunakan sejak pertengahan tahun tujuh

puluhan (Ross, 1977). Analisis terstruktur dan desain terstruktur tersedia sebagai alat kasus komersial dengan nama IDEF0.

Analisis terstruktur dan desain analisis adalah metode pengembangan perangkat lunak yang populer pada tahun 1970-an dan 1980-an. Metode ini didasarkan pada prinsip pemrograman terstruktur, yang menekankan pentingnya memecah sistem perangkat lunak menjadi komponen yang lebih kecil dan lebih mudah diatur.



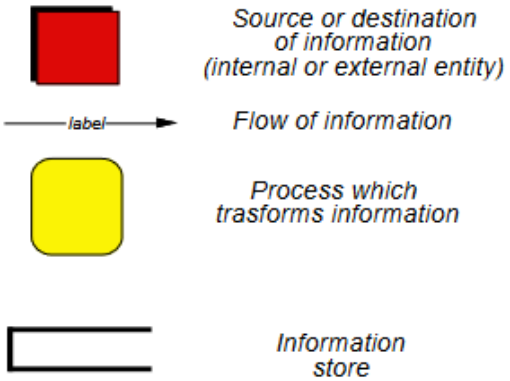
Gambar 3.1. Contoh Notasi Analisis Terstruktur Dan Desain Analisis
Sumber : (Mylopoulopoulos, 2004)

Dalam analisis terstruktur dan desain analisis, proses pengembangan perangkat lunak dibagi menjadi dua fase: Analisis Terstruktur dan Desain Terstruktur. Selama fase Analisis Terstruktur, masalah yang akan dipecahkan dianalisis dan persyaratan dikumpulkan. Fase Desain Terstruktur melibatkan perancangan sistem untuk memenuhi persyaratan yang dikumpulkan dalam fase Analisis Terstruktur.

Analisis Terstruktur dan Desain Terstruktur adalah metodologi pengembangan perangkat lunak tradisional yang populer pada 1980-an dan 1990-an. Ini melibatkan serangkaian teknik untuk merancang dan mengembangkan sistem perangkat lunak dengan cara yang terstruktur dan sistematis. Berikut adalah beberapa konsep utama analisis terstruktur dan desain analisis:

Dekomposisi Fungsional: Analisis terstruktur dan desain analisis menggunakan dekomposisi fungsional untuk memecah sistem yang kompleks menjadi subsistem yang lebih kecil dan lebih mudah dikelola. Teknik ini melibatkan identifikasi fungsi utama sistem dan memecahnya menjadi fungsi yang lebih kecil yang dapat diimplementasikan secara mandiri.

Diagram Aliran Data (DAD): Analisis terstruktur dan desain analisis menggunakan DAD untuk memodelkan aliran data melalui sistem. DAD adalah representasi grafis dari sistem yang menunjukkan bagaimana data bergerak di antara berbagai komponen sistem.



Gambar 3.2. Notasi Diagram Aliran Data
Sumber : (Mylopoulopoulos, 2004)

Kamus Data: Kamus data adalah pusat penyimpanan yang berisi deskripsi dari semua elemen data yang digunakan dalam

sistem. Ini memberikan definisi elemen data yang jelas dan konsisten, membuatnya lebih mudah untuk memahami cara kerja sistem.

Desain Terstruktur: Analisis terstruktur dan desain analisis menggunakan teknik desain terstruktur untuk mengembangkan arsitektur dan komponen sistem. Ini melibatkan identifikasi komponen utama sistem, merancang antarmuka di antara mereka, dan menentukan struktur data dan algoritma yang akan digunakan untuk mengimplementasikan sistem.

Pemrograman Modular: Analisis terstruktur dan desain analisis menggunakan teknik pemrograman modular untuk memecah kode sistem menjadi modul yang lebih kecil dan lebih mudah dikelola. Ini membuatnya lebih mudah untuk mengembangkan, menguji, dan memelihara sistem.

Beberapa keunggulan analisis terstruktur dan desain analisis mencakup penekanannya pada desain dan dokumentasi terstruktur, yang dapat membantu meningkatkan kejelasan dan pemeliharaan sistem. Namun, analisis terstruktur dan desain analisis memiliki beberapa kelemahan, termasuk kekakuan dan ketidakfleksibelannya, yang membuatnya sulit untuk beradaptasi dengan perubahan kebutuhan bisnis atau tren teknologi. Selain itu, analisis terstruktur dan desain analisis mungkin tidak cocok untuk sistem yang kompleks dan dinamis, yang mungkin memerlukan metodologi pengembangan yang lebih gesit.

3.2.3 Proses Analisis Terstruktur dan Desain Analisis

Pengumpulan persyaratan: Langkah pertama dalam proses analisis terstruktur dan desain analisis adalah mengumpulkan persyaratan dari pemangku kepentingan, termasuk pengguna, pelanggan, dan mitra bisnis.

Analisis Terstruktur: Selama fase Analisis Terstruktur, persyaratan dianalisis untuk mengidentifikasi komponen utama

sistem, hubungan antara komponen tersebut, dan aliran data dalam sistem.

Pemodelan Data: Selama fase ini, model data dibuat untuk mewakili data yang digunakan dalam sistem dan hubungan antara elemen data.

Pemodelan Proses: Selama fase ini, proses dalam sistem dimodelkan menggunakan diagram alur dan diagram aliran data.
Desain Input/Output: Selama fase ini, input dan output sistem dirancang, termasuk antarmuka pengguna dan laporan.

Desain Terstruktur: Selama fase Desain Terstruktur, sistem dirancang untuk memenuhi persyaratan yang dikumpulkan dalam fase Analisis Terstruktur. Ini mungkin termasuk memilih platform perangkat keras dan perangkat lunak yang sesuai, merancang basis data, dan menentukan struktur data.

Implementasi dan Pengujian: Setelah desain selesai, sistem diimplementasikan dan diuji.

Analisis terstruktur dan desain analisis sebagian besar telah digantikan oleh metodologi pengembangan perangkat lunak yang lebih modern, tetapi prinsip analisis dan desain terstrukturnya terus memengaruhi praktik pengembangan perangkat lunak saat ini. Metode ini dikenal dengan fokusnya untuk memecah sistem yang kompleks menjadi komponen yang lebih kecil, yang membuatnya lebih mudah untuk memahami dan mengelola sistem secara keseluruhan.

Pada dasarnya, pendekatan analisis terstruktur dan desain analisis didasarkan pada Data Flow Diagram. Analisis terstruktur dan desain analisis mudah dipahami tetapi berfokus pada batas sistem yang terdefinisi dengan baik sedangkan pendekatan JSD terlalu rumit dan tidak memiliki representasi grafis.

Analisis terstruktur dan desain analisis digabungkan dikenal sebagai SAD dan terutama berfokus pada 3 poin berikut: Sistem, Proses, Teknologi

3.3 Dua Fase Analisis Terstruktur dan Desain Analisis

Tahap Analisis: Menggunakan Data Flow Diagram, Data Dictionary, State Transition diagram dan ER diagram.
Fase Desain: Menggunakan Bagan Struktur dan Kode Pseudo.

3.3.1 Tahap Analisis

Terdapat beberapa bagian dalam proses perencanaan, selanjutnya harus memilih tujuan, memutuskan metrik, membuat prediksi mengenai parameter penting ddan membangun konsensus dari semua pihak yang terlibat . Tidak akan terlalu mendalami hal ini, tetapi ada beberapa poin yang penting yaitu : Estimasi itu penting, tetapi seringkali dibuat tanpa dasar apa pun dalam kenyataan (dan tidak ada yang pandai membuatnya). Analis terutama bertugas mencegah kegagalan (daripada memastikan kesuksesan). Kegagalan sering terjadi karena memiliki masalah dengan analisis yang tidak terlihat sampai terlambat politik interpersonal merupakan faktor yang sangat penting dalam keberhasilan dan tidak boleh diremehkan. Domain bergeser, dan rencana harus bergeser bersama, dan rencana harus dapat dipertahankan / cukup beradaptasi untuk melakukannya.

Tahap Analisis melibatkan diagram aliran data, kamus data, diagram transisi keadaan, dan diagram hubungan entitas.

Diagram aliran data: Dalam diagram aliran data, model menggambarkan bagaimana data mengalir melalui sistem. Kita dapat menggabungkan operator Boolean dan & atau menautkan aliran data ketika lebih dari satu aliran data dapat menjadi input atau output dari suatu proses.

Misalnya, jika kita harus memilih antara dua jalur dari suatu proses, kita dapat menambahkan operator atau dan jika dua aliran data diperlukan untuk suatu proses, kita dapat menambahkan operator. Input dari proses “check-order” membutuhkan informasi kredit dan informasi order sedangkan output dari proses tersebut adalah cash-order atau good-credit-order.

Kamus data: Konten yang tidak dijelaskan dalam DFD dijelaskan dalam kamus data. Ini mendefinisikan penyimpanan data dan makna yang relevan. Kamus data fisik untuk elemen data yang mengalir antar proses, antar entitas, dan antara proses dan entitas dapat disertakan. Ini juga akan mencakup deskripsi elemen data yang mengalir di luar penyimpanan data.

Kamus data logis juga dapat disertakan untuk setiap elemen data tersebut. Semua nama sistem, apakah itu nama entitas, tipe, relasi, atribut, atau layanan, harus dimasukkan ke dalam kamus.

Diagram Transisi Status: Diagram transisi keadaan mirip dengan model dinamis. Ini menentukan berapa banyak waktu yang dibutuhkan fungsi untuk mengeksekusi dan akses data dipicu oleh peristiwa. Ini juga menjelaskan semua keadaan yang dapat dimiliki suatu objek, peristiwa di mana suatu objek mengubah keadaan, kondisi yang harus dipenuhi sebelum transisi akan terjadi dan aktivitas yang dilakukan selama hidup suatu objek.

Diagram ER : Diagram ER menentukan hubungan antara penyimpanan data. Ini pada dasarnya digunakan dalam desain basis data. Ini pada dasarnya menggambarkan hubungan antara entitas yang berbeda.

3.3.2 Fase Desain

Tahap Desain melibatkan struktur bagan dan pseudocode. Pada desain diperlukan input agar dapat diproses (Munawar *et al.*, 2022).

Bagan Struktur: Ini dibuat oleh diagram aliran data. Bagan Struktur menentukan bagaimana proses DFS dikelompokkan ke dalam tugas dan dialokasikan ke CPU. Bagan terstruktur tidak menunjukkan struktur kerja dan internal dari proses atau modul dan tidak menunjukkan hubungan antara data atau aliran data. Mirip dengan alat analisis terstruktur dan desain terstruktur lainnya, ini tidak tergantung waktu dan biaya dan tidak ada teknik pengecekan kesalahan yang terkait dengan alat ini. Modul-modul

bagan terstruktur disusun secara sewenang-wenang dan setiap proses dari DFD dapat dipilih sebagai transformasi pusat tergantung pada persepsi analis sendiri. Bagan terstruktur sulit untuk diubah, diverifikasi, dipelihara, dan diperiksa kelengkapan dan konsistensinya.

Kode Pseudo: Ini adalah implementasi sebenarnya dari sistem. Ini adalah cara pemrograman informal yang tidak memerlukan bahasa atau teknologi pemrograman tertentu.

3.3.3 Keuntungan Analisis Terstruktur dan Desain Terstruktur

Kejelasan dan Kesederhanaan: Metode analisis terstruktur dan desain terstruktur menekankan penguraian sistem kompleks menjadi komponen yang lebih kecil dan lebih mudah dikelola, yang membuat sistem lebih mudah dipahami dan dikelola.

Komunikasi yang Lebih Baik: Metode analisis terstruktur dan desain terstruktur menyediakan bahasa dan kerangka kerja yang sama untuk mengkomunikasikan desain sistem, yang dapat meningkatkan komunikasi antara pemangku kepentingan dan membantu memastikan bahwa sistem memenuhi kebutuhan dan harapan mereka.

Pemeliharaan yang ditingkatkan: Metode analisis terstruktur dan desain terstruktur memberikan struktur yang jelas dan terorganisir untuk sistem, yang dapat mempermudah pemeliharaan dan pembaruan sistem dari waktu ke waktu.

Testabilitas yang Lebih Baik: Metode analisis terstruktur dan desain terstruktur memberikan definisi yang jelas tentang input dan output sistem, yang membuatnya lebih mudah untuk menguji sistem dan memastikannya memenuhi persyaratannya.

3.3.4 Kekurangan Analisis Terstruktur dan Desain Terstruktur

Memakan Waktu: Metode analisis terstruktur dan desain terstruktur dapat memakan waktu, terutama untuk sistem yang besar dan kompleks, karena metode ini memerlukan dokumentasi dan analisis dalam jumlah yang signifikan.

Tidak fleksibel: Setelah sistem dirancang menggunakan metode analisis terstruktur dan desain terstruktur, akan sulit untuk membuat perubahan pada desain, karena prosesnya sangat terstruktur dan padat dokumentasi.

Iterasi Terbatas: Metode analisis terstruktur dan desain terstruktur tidak cocok untuk pengembangan iteratif, karena dirancang untuk diselesaikan dalam sekali jalan.

DAFTAR PUSTAKA

DeMarco, T. and Plauger, P. J. 1979. *Structured Analysis and System Specification*. Facsimile. Prentice Hall.

Gane, C. and Sarson, T. 1979. *Structured Systems Analysis: Tools and Techniques*. Prentice Hall.

Munawar, Z. *et al.* 2022. 'Aplikasi Peninjauan Tempat Parkir Menggunakan Arduino dan Sensor Cahaya Berbasis Web di Miko Mall', *Darma Abdi Karya*, 1(1), pp. 1–12. Available at: https://jurnal.plb.ac.id/index.php/darmaabdikarya/article/view/1045.

Munawar, Z., Pratomo, A. B., *et al.* 2023. *Konsep Dasar Technopreneurship*. 1st edn. Edited by I. P. Kusuma. Batam: Yayasan Cendikia Mulia Mandiri.

Munawar, Z., Hasnawi, M., *et al.* 2023. *Visi Komputer*. 1st edn. Edited by R. Komalasari. Bandung: Kaizen Media Publishing.

Mylopoulopoulos, J. 2004. *Teknik Analisis Terstruktur dan Teknik Perancangan*. Available at: https://www.cs.toronto.edu/~jm/2507S/Notes04/SADT.pdf.

Page-Jones, M. 1988. *Practical Guide to Structured Systems Design*. Subsequent. Prentice Hall.

Ross, D. . 1977. 'Structured Analysis (SA): A Language for Communicating Ideas', *IEEE Transactions on Software Engineering*, 3(1), pp. 16–34. doi: 10.1109/TSE.1977.229900.

Sølvberg, A. 1993. *Structured Analysis and Design. In: Information Systems Engineering*. Berlin: Springer. doi: 10.1007/978-3-642-78001-1_2.

BAB 4
DESAIN MODEL PROSES BISNIS BERORIENTASI OBJEK

Oleh Johni S Pasaribu

4.1 Pendahuluan

Analisis dan Desain Sistem (SAD, *Systems Analysis and Design*) adalah hal yang menarik dan dinamis di mana para analis sistem terus mempelajari teknik dan pendekatan baru untuk mengembangkan sistem secara lebih efektif dan efisien. Namun, ada keterampilan yang perlu diketahui oleh semua analis sistem—apa pun pendekatan atau metodologi pengembangan perangkat lunak yang digunakan. Semua proyek sistem informasi bergerak melalui empat fase berikut: perencanaan, analisis, desain, dan implementasi; semua proyek tersebut membutuhkan analis sistem untuk mengumpulkan persyaratan, memodelkan kebutuhan bisnis, dan membuat cetak biru (*blueprint*) bagaimana sistem harus dibangun; dan semua proyek memerlukan pemahaman tentang konsep perilaku organisasi seperti mengatur pengelolaan proyek dan pembangunan tim. Sekarang ini, biaya pengembangan perangkat lunak modern utamanya terkait dengan biaya-biaya pengembangan perangkat lunak itu sendiri dan bukan biaya perangkat-perangkat komputer. Pendekatan berorientasi objek untuk pengembangan sistem informasi memegang peran penting dalam pengurangan biaya pengembangan sistem informasi.

Sekarang ini, pendekatan analisis dan desain sistem beralih dari tradisional ke berorientasi objek, pandangan dimana sistem dipandang sebagai kumpulan dari objek-objek yang memiliki data dan proses. Perubahan ini semakin dipermudah melalui penemuan

Unified Modeling Language (UML). UML menyediakan kosakata (*vocabulary*) berorientasi objek dan pembuatan diagram untuk model apapun dari pengembangan sistem perangkat lunak baik mulai dari tahap analisis hingga tahap implementasi.

4.2 Apa itu Desain Berorientasi Objek?

Paradigma berorientasi objek terbentuk dari konsep awal pendekatan pemrograman baru, sedangkan ketertarikan pada metode analisis dan desain berorientasi objek muncul belakangan.

1. Simula adalah nama dari dua bahasa pemrograman simulasi, Simula I dan Simula 67, dimana Simula dianggap sebagai bahasa pemrograman berorientasi objek pertama yang dikembangkan pada tahun 1960 oleh para peneliti di Pusat Komputasi Norwegia.
2. Pada tahun 1970, Alan Kay dan kelompok risetnya di Xerox PARK menciptakan komputer pribadi bernama Dynabook dan bahasa pemrograman berorientasi objek (OOPL, *Object Oriented Programming Language*) yaitu Smalltalk, untuk memprogram Dynabook.
3. Pada tahun 1980-an, Grady Booch menerbitkan sebuah makalah berjudul Desain Berorientasi Objek yang menyajikan desain untuk bahasa pemrograman, Ada. Dalam edisi berikutnya, dia memperluas idenya ke metode desain berorientasi objek yang lengkap

Pada tahun 1990-an, Peter Coad memasukkan ide perilaku ke metode berorientasi objek. Inovasi penting lainnya adalah Teknik Pemodelan Objek (OMT, *Object Modelling Techniques*) oleh James Rumbaugh dan Rekayasa Perangkat Lunak Berorientasi Objek (OOSE, *Object-Oriented Software Engineering*) oleh Ivar Jacobson (Tutorialspoint, 2023a).

4.2.1 Dua Pandangan Pengembangan Perangkat Lunak

Metodologi pengembangan perangkat lunak adalah serangkaian proses yang mengarah pada pengembangan perangkat lunak. Dua pandangan pengembangan perangkat lunak:

1. Metodologi pengembangan sistem tradisional. Pendekatan ini berfokus pada "Fungsi"
2. Metodologi pengembangan sistem berorientasi objek. Pendekatan ini berfokus pada "Objek" yang menggabungkan "Data dan Fungsionalitas".

Dalam desain berorientasi fungsi, sistem terdiri dari banyak sub-sistem yang lebih kecil yang dikenal sebagai fungsi. Fungsi-fungsi ini mampu melakukan tugas penting dalam sistem. Sistem dianggap sebagai tampilan atas (*top view*) dari semua fungsi.
Proses desain berorientasi fungsi:

1. Seluruh sistem dilihat sebagai data yang mengalir dalam sistem melalui diagram aliran data (DFD, *data flow diagram*).
2. DFD menggambarkan bagaimana fungsi mengubah data dan keadaan seluruh sistem.
3. Seluruh sistem secara logis dipecah menjadi unit-unit yang lebih kecil yang dikenal sebagai fungsi-fungsi berdasarkan operasinya dalam sistem.
4. Setiap fungsi kemudian dijelaskan secara luas.

Desain berorientasi objek bekerja di sekitar entitas dan karakteristiknya dibandingkan dengan fungsi yang terlibat dalam sistem perangkat lunak. Strategi desain berorientasi objek ini fokus pada entitas dan karakteristiknya. Seluruh konsep solusi perangkat lunak berputar di sekitar entitas yang terlibat.

Konsep penting dari Desain Berorientasi Objek: objek, kelas, Enkapsulasi, Warisan, Polimorfisme (lebih detail akan dijelaskan pada sub bab). Proses desain perangkat lunak dapat dianggap sebagai rangkaian langkah-langkah yang terdefinisi dengan baik.

Meskipun bervariasi sesuai dengan pendekatan desain (berorientasi fungsi atau berorientasi objek), namun melibatkan langkah-langkah berikut (Tutorialspoint, 2023b):

1. Sebuah desain solusi dibuat dari kebutuhan atau sistem yang digunakan sebelumnya dan/atau diagram sequence sistem.
2. Objek diidentifikasi dan dikelompokkan ke dalam kelas atas nama kesamaan karakteristik atribut.
3. Hirarki kelas dan hubungan di antara kelas didefinisikan.
4. Kerangka aplikasi didefinisikan.

Pandangan tradisional tentang program komputer adalah proses yang telah dikodekan dalam bentuk yang dapat dijalankan di komputer. Pandangan ini berasal dari fakta bahwa komputer pertama dikembangkan terutama untuk mengotomatiskan proses yang terdefinisi dengan baik (yaitu, algoritme) untuk perhitungan numerik, dan berasal dari komputer program tersimpan pertama. Dengan demikian, proses pembuatan perangkat lunak dilihat sebagai terjemahan dari deskripsi dalam beberapa bahasa 'alami' ke urutan operasi yang dapat dijalankan di komputer. Seperti banyak yang berpendapat, paradigma ini masih merupakan cara terbaik untuk memperkenalkan gagasan pemrograman kepada pemula, tetapi karena sistem menjadi lebih kompleks, keefektifannya dalam mengembangkan solusi menjadi curiga. Perubahan perspektif pada bagian dari pengembang perangkat lunak ini terjadi selama periode waktu tertentu dan didorong oleh beberapa faktor termasuk tingginya biaya pengembangan dan upaya terus-menerus untuk menemukan kegunaan perangkat lunak di domain baru. Seseorang dapat dengan aman berargumen bahwa aplikasi perangkat lunak yang dikembangkan di tahun-tahun berikutnya memiliki dua karakteristik yang berbeda:

1. Perilaku yang sulit dicirikan sebagai suatu proses
2. Persyaratan keandalan, kinerja, dan biaya yang tidak dihadapi oleh pengembang asli.

Pada Pendekatan Tradisional, di sini Algoritma + Struktur data = Program. “Sistem perangkat lunak adalah seperangkat mekanisme untuk melakukan tindakan tertentu pada data tertentu.” Pengembangan Berorientasi Objek menawarkan model yang berbeda dari pendekatan pengembangan perangkat lunak tradisional, yang didasarkan pada fungsi dan prosedur. Dalam sistem Berorientasi objek, semuanya adalah objek dan setiap objek bertanggung jawab atas dirinya sendiri. Objek jendela bertanggung jawab untuk hal-hal seperti membuka, mengukur, dan menutup sendiri.

4.2.2 Mengapa Berorientasi Objek ?

Sistem lebih mudah beradaptasi dengan perubahan persyaratan, lebih mudah dirawat, lebih kuat, desain yang lebih besar dan penggunaan kembali kode. Pengembangan berorientasi objek memungkinkan kita membuat modul fungsionalitas. Berikut adalah beberapa alasan lain mengapa Orientasi objek berfungsi:

1. Tingkat abstraksi yang lebih tinggi. Pendekatan top-down mendukung abstraksi pada level fungsi. Pendekatan berorientasi objek mendukung abstraksi pada tingkat objek. Karena objek merangkum data (atribut) dan fungsi (metode), mereka bekerja pada tingkat abstraksi yang lebih tinggi.
2. Transisi mulus di antara berbagai fase pengembangan perangkat lunak. Pendekatan berorientasi fungsi dalam pengembangan perangkat lunak memerlukan metodologi yang berbeda untuk setiap bagian dari proses.
3. Mendorong teknik pemrograman yang baik. Kelas dalam sistem berorientasi objek menggambarkan antara antarmuka (spesifikasi apa yang dapat dilakukan kelas) dan implementasi antarmuka itu (bagaimana kelas melakukan apa yang dilakukannya).

4. Penggunaan ulang. Objek dapat digunakan kembali karena dimodelkan langsung dari domain masalah dunia nyata. Setiap objek berdiri sendiri atau dalam hubungan dengan objek lain.

Tabel 4.1. Pendekatan Berorientasi Fungsi versus Berorientasi Objek

Pendekatan Berorientasi Fungsi	**Pendekatan Berorientasi Objek**
Pengumpulan prosedur	Kombinasi data dan fungsi
Berfokus pada fungsi dan prosedur gaya dan metodologi yang berbeda untuk setiap langkah proses	Berfokus pada objek, kelas, modul yang dapat dengan mudah diganti, dimodifikasi, dan digunakan kembali
Berpindah dari satu fase ke fase lain itu kompleks	Berpindah dari satu fase ke fase lainnya lebih mudah
Meningkatkan kompleksitas	Mengurangi kompleksitas dan redundansi
Meningkatkan durasi proyek	Mengurangi durasi proyek

4.3 Analisis dan Desain Berorientasi Objek

Analisis Berorientasi Objek (OOA, *Object-Oriented Analysis*) adalah prosedur untuk mengidentifikasi persyaratan rekayasa perangkat lunak dan mengembangkan spesifikasi perangkat lunak dalam hal model objek sistem perangkat lunak, yang terdiri dari objek yang berinteraksi. Perbedaan utama antara analisis berorientasi objek dan bentuk analisis lainnya adalah bahwa dalam pendekatan berorientasi objek, persyaratan diatur di sekitar objek, yang mengintegrasikan data dan fungsi. Mereka dimodelkan setelah objek dunia nyata yang berinteraksi dengan sistem. Dalam metodologi analisis tradisional, dua aspek - fungsi dan data -

dipertimbangkan secara terpisah. Grady Booch mendefinisikan OOA sebagai metode analisis yang memeriksa persyaratan dari perspektif kelas dan objek yang ditemukan dalam kosa kata (*vocabulary*) domain masalah (Grady Booch, Robert Maksimchuk, Michael Engle, Jim Conallen, Kelli Houston, 2007).
Tugas utama analisis berorientasi objek (OOA) adalah:

1. Mengidentifikasi objek
2. Pengorganisasian objek dengan membuat diagram model objek
3. Mendefinisikan internal objek atau atribut objek
4. Mendefinisikan perilaku objek atau metode objek
5. Mendeskripsikan bagaimana objek-objek berinteraksi

Model umum yang digunakan dalam OOA adalah use case dan model objek.

Desain Berorientasi Objek (OOD, *Object-Oriented Design*) melibatkan implementasi model konseptual yang dihasilkan pada analisis berorientasi objek. Detail implementasi umumnya meliputi:

1. Restrukturisasi data kelas (bila perlu),
2. Implementasi metode, yaitu struktur data internal dan algoritma,
3. Implementasi kontrol, dan
4. Implementasi asosiasi

Grady Booch telah mendefinisikan desain berorientasi objek sebagai metode desain yang mencakup proses dekomposisi berorientasi objek dan notasi untuk menggambarkan model logis dan fisik serta model statis dan dinamis dari sistem yang dirancang (Grady Booch, Robert Maksimchuk, Michael Engle, Jim Conallen, Kelli Houston, 2007).

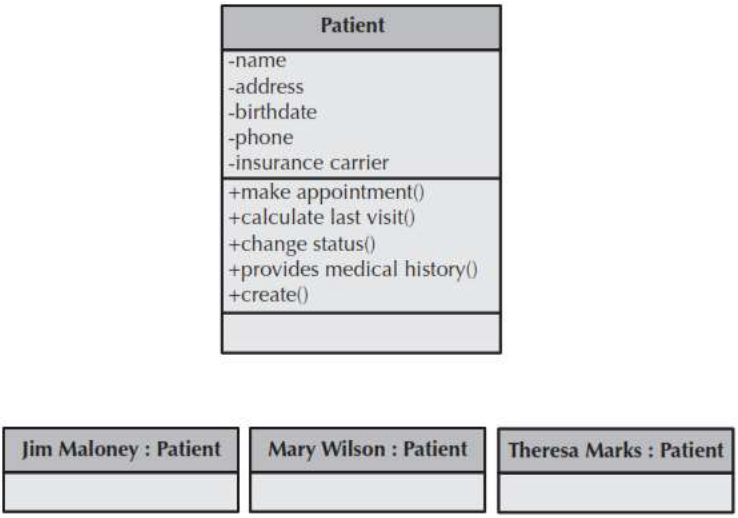
Analisis dan desain berorientasi objek (OOAD, *Object-Oriented Analysis and Design*) adalah pendekatan rekayasa

perangkat lunak yang memodelkan sistem sebagai sekelompok objek yang berinteraksi. Setiap objek mewakili beberapa entitas dalam sistem, dicirikan oleh kelasnya, statusnya (elemen data), dan perilakunya. Berbagai model dapat dibuat untuk menunjukkan struktur statis, perilaku dinamis, dan implementasi dari objek yang berkolaborasi ini. Ada sejumlah notasi yang berbeda untuk merepresentasikan model ini, salah satu model tersebut adalah *Unified Modeling Language* (UML) (Castagna, 2012).

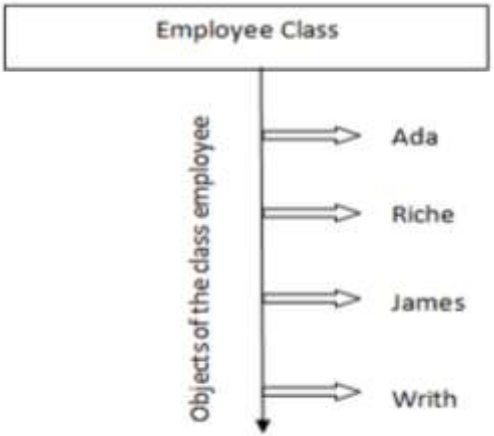
4.4 Beberapa Istilah Desain Berorientasi Objek

4.4.1 Objek

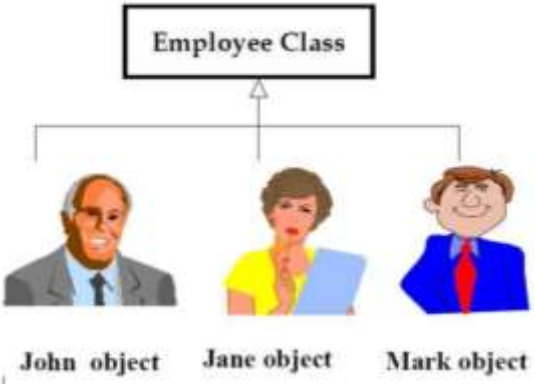
Objek adalah semua entitas yang terlibat dalam desain solusi. Misalnya, orang, bank, perusahaan, dan pelanggan diperlakukan sebagai objek. Setiap entitas memiliki beberapa atribut yang terkait dengannya dan memiliki beberapa metode untuk tampil pada atribut tersebut. Setiap objek adalah turunan dari kelas, lihat pada Gambar 4.2 dan 4.3.



Gambar 4.1. Kelas dan Objek.
(Sumber: (Dennis, Wixom, 2015))



Gambar 4.2. Objek-objek: Ada, Riche, James dan Writh dari Kelas Employee



Gambar 4.3. Kelas Employee

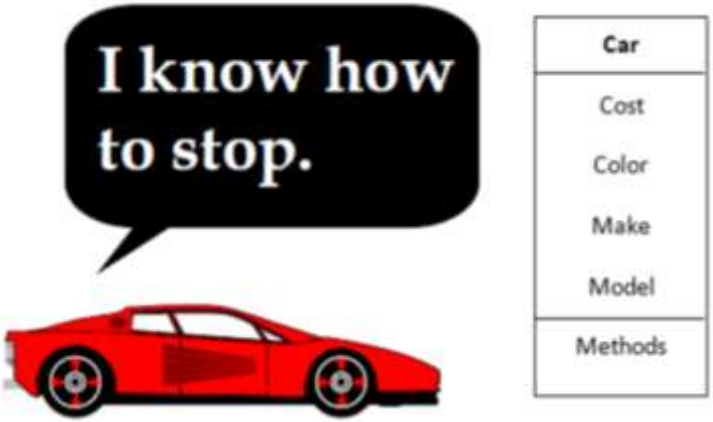
4.4.2 Atribut

Atribut adalah semua informasi dan data yang digunakan untuk menggambarkan karakteristik objek. Diwakili oleh tipe data, mendeskripsikan keadaan suatu objek. Misalnya atribut mobil, lihat Gambar 4.4.

Gambar 4.4. Atribut mobil

4.4.3 Perilaku

Perilaku pada dasarnya disebut sebagai metode, prosedur, operasi atau layanan yang dapat dilakukan oleh objek dengan beroperasi pada atribut atau data objek. Misalnya, Metode: mengendarai mobil, lihat Gambar 4.5.



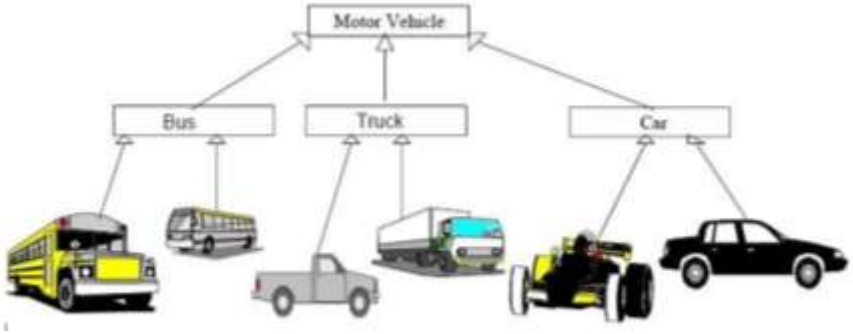
Gambar 4.5. Metode mobil

4.4.4 Kelas

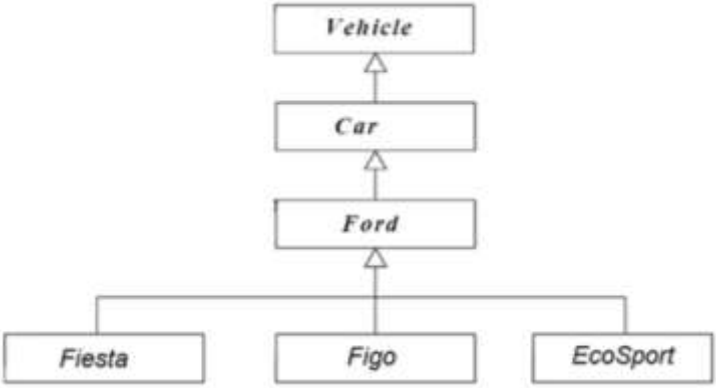
Kelas adalah kumpulan objek atau kumpulan objek berbeda yang berbagi atribut dan perilakunya. Objek adalah turunan dari kelas. Kelas mendefinisikan semua atribut (yang dapat dimiliki objek) dan metode (yang mendefinisikan fungsionalitas objek). Dalam desain, atribut disimpan sebagai variabel dan fungsionalitas didefinisikan melalui metode atau prosedur.

4.4.5 Hirarki Kelas

Sistem berorientasi objek mengatur kelas menjadi hierarki kelas *Subclass-Super*. Di bagian atas hierarki adalah kelas (dasar) paling umum dan di bagian bawah adalah kelas paling spesifik (sub-kelas). Kelas induk juga dikenal sebagai kelas basis atau kelas super. Subclass mewarisi semua properti dan metode (prosedur) yang didefinisikan dari kelas supernya, lihat Gambar 4.6 dan 4.7.



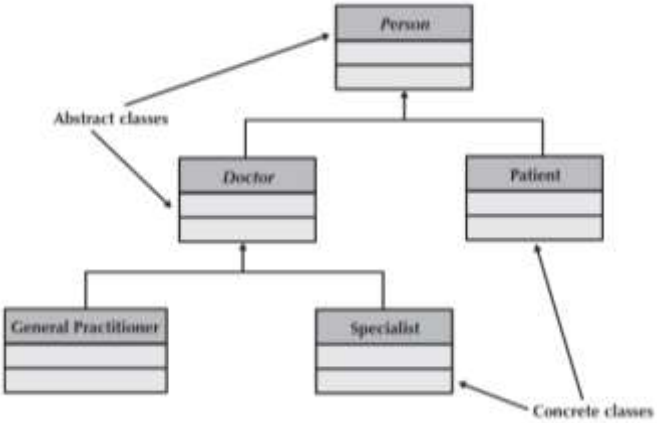
Gambar 4.6. Hierarki kelas Super / Sub kelas



Gambar 4.7. Hierarki kelas untuk kelas Ford

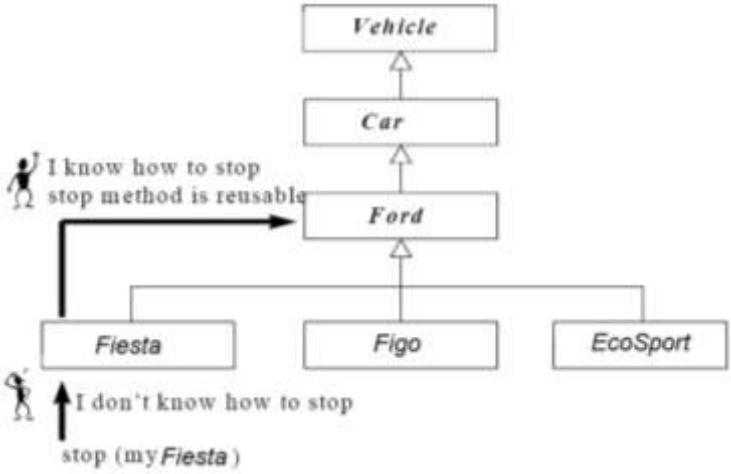
4.4.6 Pewarisan

OOD (*Object-Oriented Design*) memungkinkan kelas serupa untuk menumpuk secara hierarkis di mana kelas bawah atau sub-kelas dapat mengimpor, mengimplementasikan, dan menggunakan kembali variabel dan metode yang diizinkan dari kelas super (induk). Properti OOD ini dikenal sebagai pewarisan. Ini membuatnya lebih mudah untuk mendefinisikan kelas tertentu dan membuat kelas umum dari yang spesifik.



Gambar 4.8. Hierarki kelas dengan kelas abstrak dan konkrit

Pewarisan (*Inheritance*) adalah hubungan antar kelas dimana satu kelas merupakan induk (*parent*) kelas dari kelas lain (turunan), lihat Gambar 4.9. Pewarisan memungkinkan kelas untuk berbagi dan menggunakan kembali perilaku dan atribut. Keuntungan nyata dari pewarisan adalah kita dapat membangun dan menggunakan kembali apa yang sudah kita miliki.

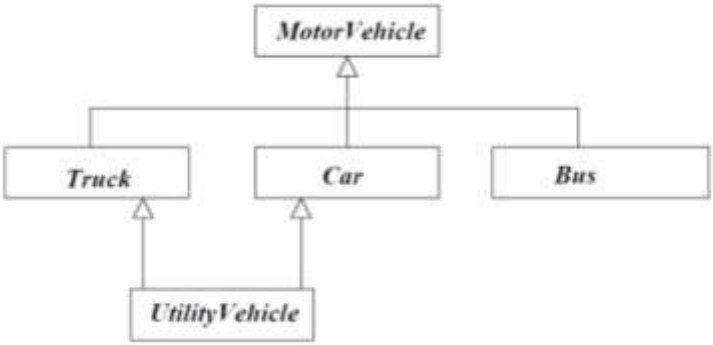


Gambar 4.9. Pewarisan memungkinkan penggunaan kembali

Warisan Dinamis:

1. Memungkinkan objek untuk berubah dan berkembang seiring waktu.
2. Warisan dinamis mengacu pada kemampuan untuk menambah, menghapus, atau mengubah induk dari objek saat dijalankan.

Sistem berorientasi objek memungkinkan kelas untuk mewarisi statusnya (atribut) dan perilaku dari lebih dari satu kelas super. Warisan semacam ini disebut sebagai "Warisan Berganda". Lihat Gambar 4.10.

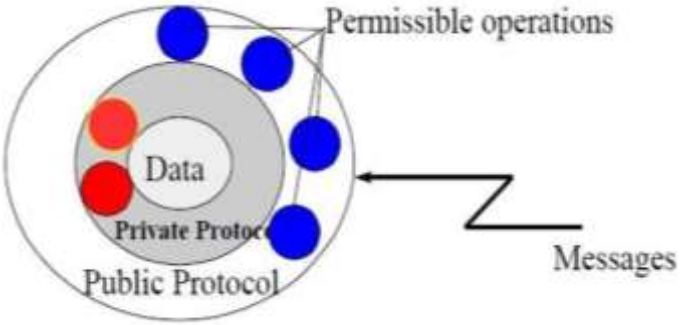


Gambar 4.10. Kendaraan utilitas mewarisi dari kelas mobil dan truk

4.4.7 Enkapsulasi

Dalam OOD, atribut (variabel data) dan metode (operasi pada data) digabungkan bersama disebut enkapsulasi. Enkapsulasi tidak hanya menyatukan informasi penting dari suatu objek, tetapi juga membatasi akses data dan metode dari dunia luar. Ini disebut penyembunyian informasi.

Penyembunyian informasi adalah prinsip penyembunyian data internal dan prosedur suatu objek, lihat Gambar 4.11.

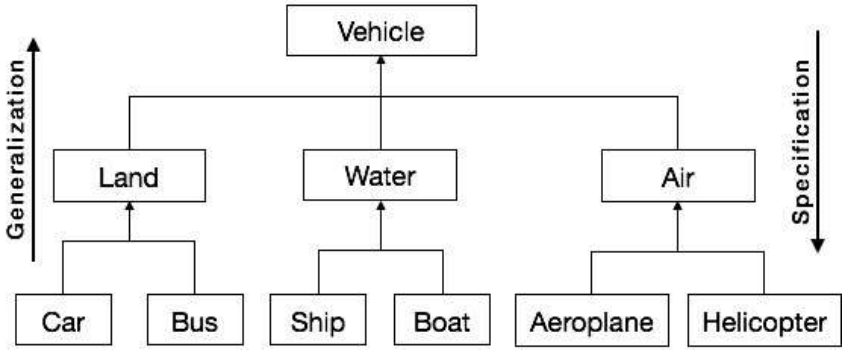


Gambar 4.11. Penentu akses

Enkapsulasi melindungi data dari kerusakan. Menyediakan antarmuka untuk setiap objek sedemikian rupa untuk mengungkapkan sedikit mungkin tentang cara kerja bagian dalam. Enkapsulasi secara umum berarti mekanisme proteksi dengan menggunakan penentu akses seperti *public, protected* dan *private*. Data diabstraksi ketika dilindungi atau disembunyikan oleh sekumpulan metode lengkap dan hanya metode tersebut yang dapat mengakses bagian data dari suatu objek.

4.4.8 Generalisasi/Spesialisasi

Ini pada dasarnya adalah proses atau teknik untuk penggunaan kembali pewarisan secara luas, di mana atribut dan karakteristik umum dari beberapa kelas objek digabungkan bersama untuk membentuk kelas umum. Kadang-kadang disebut sebagai kelas generalisasi. Metode dan atribut tertentu dari kelas objek induk dikenal sebagai spesialisasi.



Gambar 4.12. Generalisasi dan Spesialisasi

4.4.9 Hubungan objek/kelas

Konsep asosiasi mewakili hubungan antara objek dan kelas. Asosiasi adalah dua arah dimana dapat dilintasi melalui dua arah. Misalnya, seorang pilot dapat menerbangkan pesawat atau pesawat diterbangkan oleh pilot, lihat Gambar 4.13.

Gambar 4.13. Asosiasi merepresentasikan hubungan antar objek yang bersifat dua arah

4.4.10 Kardinalitas atau Multiplisitas

Multiplisitas atau Kardinalitas mendefinisikan berapa banyak objek yang berpartisipasi dalam suatu hubungan dan itu adalah jumlah dari satu kelas yang terkait dengan kelas lainnya. Misalnya, hubungan klien-akun berikut, lihat Gambar 4.14.

Cardinality	
Customer	Account no.
1	1
1	*
*	*

Note: * means Many

Gambar 4.14. Kardinalitas dalam asosiasi

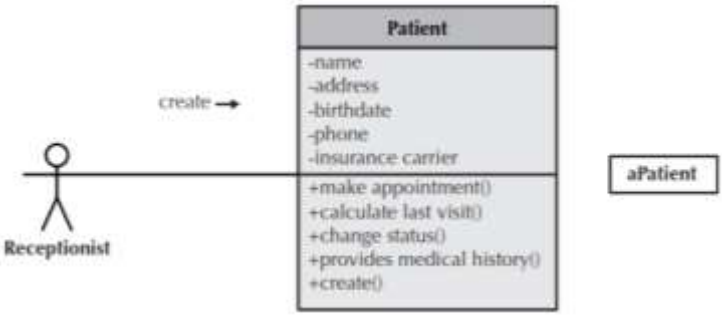
4.4.11 Agregasi

Agregasi atau komposisi adalah hubungan antar kelas dimana kelas dapat terdiri dari kombinasi objek dari kelas lain. Hal ini memungkinkan objek ditempatkan langsung di dalam tubuh kelas lain. Agregasi disebut sebagai hubungan "*part of*" atau "*has a*", dengan kemampuan memandu dari keseluruhan ke bagian-bagiannya. Objek agregat adalah objek yang terdiri dari satu atau lebih objek lain. Contoh: dalam hubungan, "*a car has a motor*", mobil adalah keseluruhan objek atau agregat, dan motor adalah "*part of*" mobil. Ini mungkin digambarkan sebagai jenis hubungan khusus, yang menunjukkan bahwa beberapa objek atau kelas disiapkan dengan menggunakan objek atau kelas lain. Dengan mengenali agregasi dan interaksi yang tepat, seseorang dapat

memisahkan objek yang rumit dengan menetapkan karakteristik dan atribut yang berbeda ke kelas atau objek individual. Ada dua jenis hubungan agregasi: (i) hubungan agregat komposisi di antara kelas-kelas objek, semua objek bagian membentuk dan hidup dalam keseluruhan objek, (ii) hubungan agregasi bersama, yang menjelaskan bahwa bagian dapat dibagi dengan yang lain.

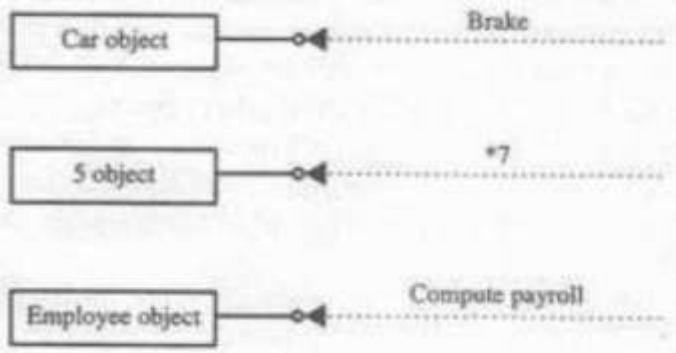
4.4.12 Pesan

Pesan harus lewat terutama ketika salah satu objek meminta informasi dari satu atau lebih objek yang berbeda untuk beberapa tindakan atau serangkaian tindakan tertentu.



Gambar 4.15. Pesan dan metode

Pesan jauh lebih umum daripada pemanggilan fungsi. Objek melakukan operasi dalam menanggapi pesan yang diterimanya. Objek menanggapi pesan sesuai dengan metode yang didefinisikan dalam kelasnya. Contoh lihat Gambar 4.16:

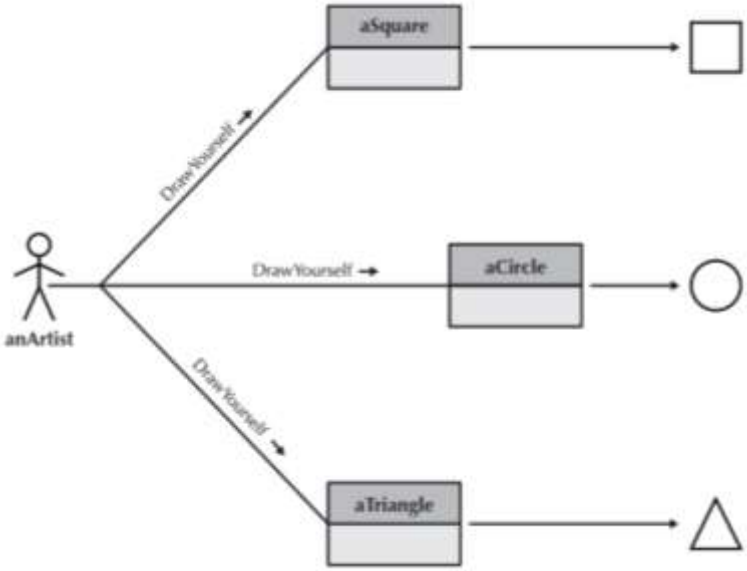


Gambar 4.16. Pesan dan metode

1. Kirim pesan rem ke objek mobil
2. Kirim pesan perkalian *7 ke 5 objek
3. Kirim pesan Compute payroll ke objek Employee

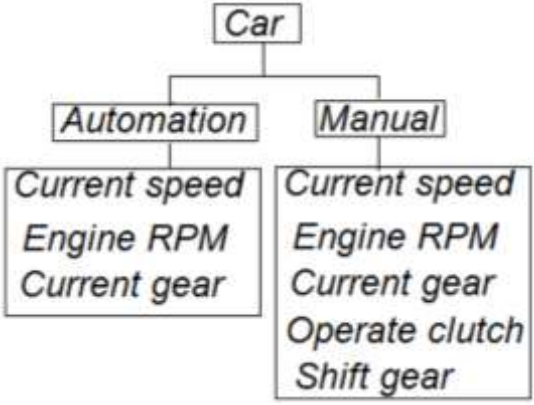
4.4.13 Polimorfisme

Polimorfisme berasal dari kata Yunani, Poly berarti "banyak" dan morph berarti "bentuk", yang berarti kemampuan untuk mengambil berbagai bentuk. Dalam paradigma berorientasi objek, polimorfisme menyiratkan penggunaan operasi dengan cara yang berbeda, tergantung pada contoh operasinya. Polimorfisme memungkinkan objek dengan struktur internal yang berbeda memiliki antarmuka eksternal yang sama. Polimorfisme sangat efektif saat menerapkan pewarisan.

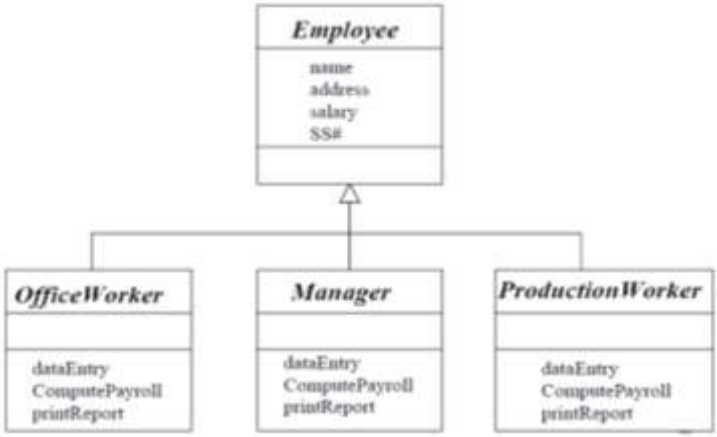


Gambar 4.17. Polimorfisme

Polimorfisme berarti bahwa operasi yang sama dapat berperilaku berbeda pada kelas yang berbeda. Lihat Gambar 4.18 dan 4.19.



Gambar 4.18. Objek mobil menunjukkan polimorfisme



Gambar 4.19. Compute Payroll merepresentasikan polimorfisme

4.5 Diagram UML

UML (*Unified Modeling Language*) adalah bahasa standar untuk menspesifikasikan, membangun, memvisualisasikan, dan mendokumentasikan sistem perangkat lunak dan komponennya. UML adalah bahasa grafis dengan seperangkat aturan dan semantik. Aturan dan semantik model diekspresikan dalam bahasa Inggris dengan bentuk yang dikenal sebagai *object constraint language* (OCL). OCL adalah bahasa spesifikasi yang menggunakan logika sederhana untuk menentukan properti sistem. UML tidak dimaksudkan untuk menjadi bahasa pemrograman visual dalam arti memiliki semua dukungan visual dan semantik yang diperlukan untuk menggantikan bahasa pemrograman. Namun, UML memang memiliki pemetaan yang ketat dalam keluarga bahasa berorientasi objek, sehingga semua orang bisa mendapatkan yang terbaik dari kedua dunia tersebut.

UML dibuat oleh *Object Management Group* (OMG) dan draf spesifikasi UML 1.0 diajukan ke OMG pada Januari 1997. Awalnya menggambarkan perilaku perangkat lunak yang kompleks dan sistem non-perangkat lunak dan sekarang telah menjadi standar

OMG. Ada dua kategori utama yaitu dibagi menjadi subkategori: Diagram Struktural (*Structural Diagrams*) dan Diagram Perilaku (*Behavioral Diagrams*) (Tutorialspoint, 2023c). Empat diagram struktural adalah: *Class Diagram, Object Diagram, Component Diagram* dan *Deployment Diagram*. UML memiliki lima jenis diagram perilaku berikut: *Use Case Diagram*, *Sequence Diagram, Collaboration Diagram, Statechart Diagram* dan *Activity Diagram* (Tutorialspoint, 2023c). Berbagai jenis diagram UML dan objeknya dibahas secara singkat di sini:

1. *Use Case Diagram*: Diagram ini menggambarkan hubungan antara sistem internal dan juga sistem eksternal yang berbeda dengan pengguna akhir. *Use case diagram* adalah sekumpulan use case, aktor, dan hubungannya. Use case diagram mewakili tampilan use case dari suatu sistem. Sebuah use case merepresentasikan fungsionalitas tertentu dari sebuah sistem. Oleh karena itu, *use case diagram* digunakan untuk menggambarkan hubungan antara fungsionalitas dan pengontrol internal/eksternalnya. Pengontrol ini dikenal sebagai aktor.
2. *Class Diagram*: diagram yang paling umum digunakan dalam UML. Class diagram terdiri dari kelas, antarmuka, asosiasi, dan kolaborasi. *Class diagram* pada dasarnya mewakili pandangan berorientasi objek dari suatu sistem, yang bersifat statis. *Class diagram* mewakili struktur dasar dari tujuan sistem. Ini mungkin menunjukkan bagaimana kelas objek disusun dengan cerdas dan merepresentasikan hubungan antar kelas objek yang berbeda.
3. *Object Diagram*: Ini cukup analog dengan diagram kelas tetapi bukannya menggambarkan kelas objek, ia mereplikasi karakteristik objek yang sebenarnya. *Object Diagram* dapat digambarkan sebagai turunan dari diagram kelas. Dengan demikian, diagram ini lebih mirip dengan skenario kehidupan nyata di mana implementasi system tersebut.

4. *Sequence Diagram*: adalah diagram interaksi. Dari namanya, jelas bahwa diagram tersebut berhubungan dengan beberapa urutan, yaitu urutan pesan yang mengalir dari satu objek ke objek lainnya. *Sequence diagram* menggambarkan secara grafis bagaimana objek yang berbeda dapat berinteraksi atau berkomunikasi di antara mereka melalui penyampaian berbagai pesan pada saat eksekusi untuk setiap operasi tipe use case. *Sequence diagram* juga menjelaskan bagaimana pesan ditransmisikan dan bagaimana diterima dengan mempertahankan urutan yang telah ditentukan sebelumnya.
5. *Collaboration Diagram*: adalah bentuk lain dari diagram interaksi. Ini mewakili organisasi struktural dari suatu sistem dan pesan yang dikirim/diterima. Organisasi struktural terdiri dari objek dan tautan/jaringan (*link*). Tujuan dari *Collaboration Diagram* mirip dengan *Sequence Diagram*. Namun, tujuan khusus dari *Collaboration Diagram* adalah untuk memvisualisasikan organisasi objek dan interaksinya.
6. *Activity Diagram*: Ini pada dasarnya digunakan untuk pemodelan grafis yang menggambarkan aliran data berurutan dari berbagai aktivitas, perkembangan bisnis atau kasus penggunaan apa pun. *Activity Diagram* menggambarkan aliran kontrol dalam suatu sistem. Ini terdiri dari kegiatan dan tautan. Aliran dapat berurutan, bersamaan, atau bercabang. Aktivitas tidak lain adalah fungsi dari sebuah sistem. Sejumlah *Activity Diagram* disiapkan untuk menangkap keseluruhan aliran dalam suatu sistem.
7. *Statechart Diagram*: Digunakan untuk memodelkan berbagai perilaku dinamis dari objek atau kelas tertentu. Itu juga dapat menggambarkan siklus hidup objek tertentu. Status objek yang berbeda dapat dianggap sebagai peristiwa yang menyebabkan transaksi status.
8. *Component Diagram*: Diagram ini sering digunakan untuk menggambarkan secara grafis setiap arsitektur fisik sistem.

Component Diagram mewakili sekumpulan komponen dan hubungannya. Komponen-komponen ini terdiri dari kelas, antarmuka, atau kolaborasi. Diagram komponen mewakili tampilan implementasi suatu sistem. Selama fase desain, artefak perangkat lunak (kelas, antarmuka, dan lain-lain) dari suatu sistem disusun dalam kelompok yang berbeda tergantung pada hubungannya. Sekarang, grup ini dikenal sebagai komponen. Akhirnya, dapat dikatakan diagram komponen digunakan untuk memvisualisasikan implementasi.

9. *Deployment Diagram*: Digunakan untuk mewakili arsitektur fisik dari perangkat lunak dan perangkat keras dari sistem tertentu. Deployment diagram adalah sekumpulan node dan hubungannya. Node ini adalah entitas fisik tempat komponen menyebar. Deployment diagram digunakan untuk memvisualisasikan tampilan penyebaran sistem. Ini umumnya digunakan oleh tim penerapan.

DAFTAR PUSTAKA

Castagna, G. 2012. *Object-Oriented Programming A Unified Foundation*. Springer Science & Business Media.

Dennis, Wixom, and T. 2015. *Systems Analysis and Design: An Object-Oriented Approach with UML*. 5th Editio. Wiley.

Grady Booch, Robert Maksimchuk, Michael Engle, Jim Conallen, Kelli Houston, B. Y. P. D. 2007. *Object-Oriented Analysis and Design with Applications*. 3rd Editio. Pearson Education.

Tutorialspoint. 2023a. *OOAD - Object Oriented Paradigm*, *tutorialspoint*. Available at: https://www.tutorialspoint.com/object_oriented_analysis_design/ooad_object_oriented_paradigm.htm (Accessed: 10 June 2023).

Tutorialspoint. 2023b. *Software Design Strategies*, *tutorialspoint*. Available at: https://www.tutorialspoint.com/software_engineering/software_design_strategies.htm (Accessed: 10 June 2023).

Tutorialspoint. 2023c. *UML - Standard Diagrams*, *tutorialspoint*. Available at: https://www.tutorialspoint.com/uml/uml_standard_diagrams.htm (Accessed: 13 June 2023).

BAB 5
DESAIN BASIS DATA PENDEKATAN TERSTRUKTUR

Oleh Sarji

5.1 Pengertian Desain Basis Data

Hector Garcia-Molina dalam buku "Database Sistems: The Complete Book" memberikan pengertian desain basis data sebagai proses merancang struktur dan organisasi suatu basis data untuk mengakomodasi kebutuhan informasi dari suatu sistem. Desain basis data melibatkan pemodelan data, pemilihan struktur data yang tepat, dan mengatur hubungan antara entitas dalam basis data(Pearson Education, 2008).

Menurut Michael J. Hernandez dalam bukunya "Database Design for Mere Mortals: A Hands-On Guide to Relational Database Design," desain basis data didefinisikan sebagai proses menciptakan struktur yang baik untuk basis data yang efektif dan efisien. Desain basis data melibatkan pemodelan data, pengorganisasian tabel, dan menentukan hubungan antara tabel dalam basis data(Addison-Wesley, 2020).

Desain Basis Data mengacu pada proses merencanakan dan mengatur struktur, format, dan hubungan antar data dalam sistem basis data. Basis data adalah kumpulan terorganisir dari data yang saling terkait, yang disimpan dan diakses menggunakan perangkat lunak *database management sistem* (DBMS).

Desain Basis Data melibatkan pemodelan data dan identifikasi kebutuhan informasi yang harus dipenuhi oleh sistem basis data. Tujuannya adalah untuk menciptakan struktur *database*

yang efisien, fleksibel, dan mudah digunakan untuk manajemen informasi yang efektif.

Proses desain basis data mencakup beberapa langkah penting. Tahap awal melibatkan analisis kebutuhan bisnis dan memahami tentang jenis data apa yang akan disimpan. Langkah selanjutnya adalah membuat skema *database*, yang berisi entitas (objek) dan atribut-atribut yang berkaitan dengan entitas tersebut. Selain itu, relasi antara entitas juga didefinisikan, seperti relasi *one-to-one*, *one-to-many,* atau *many-to-many*.

Selanjutnya, tahap desain meliputi pemodelan fisik *database*, yaitu desain struktur penyimpanan fisik *database*. Ini termasuk memilih tipe data yang sesuai untuk setiap atribut, menentukan indeks, dan mengatur strategi data di dalam media penyimpanan seperti tabel atau file.

Pertimbangan yang berkaitan dengan keamanan informasi, integritas referensial, dan kinerja sistem basis data juga harus diperhitungkan selama proses desain. Keamanan data melibatkan perlindungan terhadap akses tidak sah atau manipulasi data, sedangkan integritas referensial berarti memastikan konsistensi data dalam hubungan antar entitas. Kinerja sistem basis data harus dipertimbangkan untuk memastikan waktu respons yang cepat dan operasi basis data yang efisien.

Desain basis data juga mencakup pemilihan metode pengindeksan dan penyimpanan yang sesuai, pemilihan teknik pengoptimalan kueri, dan pemilihan strategi cadangan dan pemulihan data. Tujuan dari semua ini adalah untuk membuat *database* yang handal, efisien dan memenuhi kebutuhan bisnis.

Singkatnya, desain basis data adalah proses merancang struktur dan hubungan data dalam sistem basis data. Ini termasuk pemodelan data, pemilihan tipe data, identifikasi entitas dan atribut, pemilihan metode penyimpanan, dan pengoptimalan kueri. Perancangan basis data penting dalam menciptakan basis

data yang efisien, fleksibel, dan memenuhi kebutuhan pengelolaan data pengguna.

5.2 Pendekatan Terstruktur dalam Desain Basis Data

Pendekatan terstruktur dalam desain basis data mengacu pada metode yang menggunakan struktur data hierarkis atau jaringan untuk mengatur dan menyimpan data. Pendekatan ini dikenal sebagai pendekatan awal dalam desain *database* sebelum munculnya pendekatan relasional yang lebih umum digunakan saat ini.

Dalam pendekatan terstruktur, data disimpan dalam format hierarkis atau jaringan, di mana entitas utama memiliki anak entitas yang terhubung secara hierarkis. Pendekatan ini didasarkan pada konsep induk-anak, dimana satu entitas bisa menjadi anak dari entitas yang lain. Misalnya, dalam struktur data hierarkis, satu entitas dapat menjadi induk dari beberapa entitas anak, tetapi setiap entitas anak hanya memiliki satu induk.

Salah satu contoh penerapan pendekatan terstruktur adalah dengan menggunakan *Hierarchical Data Model* (HDM). Dalam HDM, data diatur ke dalam struktur pohon, di mana satu entitas tingkat atas (*root*) merupakan induk dari entitas yang lain. Setiap entitas dapat memiliki beberapa entitas anak yang terhubung secara hierarkis.

Keuntungan dari pendekatan terstruktur adalah kemudahan dalam menggambarkan hubungan hierarkis antara entitas dan kemampuan untuk mengakses data dengan cepat melalui navigasi struktur hierarki. Namun, pendekatan terstruktur memiliki beberapa keterbatasan. Pertama, pendekatan ini kurang fleksibel dalam mengubah struktur *database* karena perubahan pada struktur hierarki dapat mempengaruhi sistem secara keseluruhan. Kedua, pendekatan terstruktur cenderung kurang

efisien dalam mengelola hubungan kompleks antar entitas, seperti hubungan *many-to-many*.

Dengan berkembangnya pendekatan desain basis data relasional, pendekatan terstruktur menjadi kurang umum digunakan dalam desain basis data saat ini. Pendekatan berbasis relasional menggunakan model data relasional yang menggunakan tabel dan hubungan antar tabel untuk mengatur dan menyimpan data. Pendekatan ini lebih fleksibel dan lebih mudah untuk mengubah struktur basis data dan mengelola hubungan kompleks antar entitas

Meskipun pendekatan terstruktur jarang digunakan dalam desain basis data modern, pemahaman tentang pendekatan ini sangat penting untuk dipelajari ketika ingin melihat sejarah perkembangan desain basis data dan sebagai dasar untuk pengetahuan umum dalam bidang *database*.

5.3 Langkah-langkah dalam Desain Basis Data Pendekatan Terstruktur

Ada beberapa langkah yang harus diikuti dalam pendekatan terstruktur untuk desain database. Langkah-langkah umum untuk merancang basis data pendekatan terstruktur adalah sebagai berikut:

Identifikasi kebutuhan data: Langkah pertama adalah mengidentifikasi kebutuhan data yang akan disimpan dalam basis data. Identifikasi entitas-entitas yang relevan dan atribut-atribut yang berkaitan dengan entitas tersebut.

Analisis hubungan hierarkis: Selanjutnya, analisis hubungan hierarkis antara entitas-entitas yang telah diidentifikasi. Tentukan entitas-entitas yang akan menjadi entitas induk (*parent entity*) dan entitas-entitas yang akan menjadi entitas anak (*child entity*).

Sketsa struktur hierarkis: Buatlah sketsa atau diagram yang menggambarkan struktur hierarkis dari entitas-entitas yang telah

diidentifikasi. Gunakan simbol-simbol yang sesuai untuk menunjukkan hubungan *parent-child antara* entitas.

Normalisasi data: Normalisasi adalah proses untuk mengatur dan mengurangi redundansi data dalam basis data. Dalam pendekatan terstruktur, normalisasi data tidak diperlukan karena struktur hierarkis sudah mengatur relasi antara entitas secara inheren.

Implementasi struktur hierarkis: Selanjutnya, implementasikan struktur hierarkis yang telah dirancang ke dalam basis data menggunakan bahasa *query* atau perangkat lunak basis data terstruktur yang sesuai.

Validasi dan pengujian: Setelah implementasi, lakukan validasi dan pengujian basis data untuk memastikan bahwa data dapat disimpan, diakses, dan dimanipulasi dengan benar sesuai dengan struktur hierarkis yang telah ditentukan.

Perbaikan dan perubahan: Jika diperlukan, lakukan perbaikan dan perubahan pada struktur hierarkis basis data. Hal ini mungkin diperlukan saat kebutuhan data berubah atau ketika ada kebutuhan untuk menyesuaikan struktur dengan kebutuhan bisnis yang baru.

Pemeliharaan dan pembaruan: Terakhir, lakukan pemeliharaan rutin dan pembaruan basis data untuk memastikan integritas data dan performa yang optimal.

Penting untuk diingat bahwa langkah-langkah ini bersifat umum dan dapat disesuaikan dengan kebutuhan dan kompleksitas proyek desain basis data yang sedang dilakukan. Selain itu, seiring perkembangan teknologi dan pendekatan desain basis data yang lebih modern, seperti pendekatan berbasis relasional, langkah-langkah dalam desain basis data terstruktur mungkin tidak lagi relevan.

5.4 Perancangan Basis Data dengan Pendekatan Terstruktur

Perancangan basis data dengan pendekatan terstruktur melibatkan langkah-langkah berikut ini, bersama dengan contoh-contoh yang relevan:

5.4.1 Identifikasi Entitas Utama:

Identifikasi entitas utama yang akan disimpan dalam basis data. Misalnya, dalam basis data perpustakaan, entitas utama dapat mencakup "Buku", "Anggota", dan "Pinjaman".

Contoh:

Entitas utama "Buku" memiliki atribut seperti "Judul", "Pengarang", dan "Tahun Terbit".

Entitas utama "Anggota" memiliki atribut seperti "Nama", "Alamat", dan "Nomor Telepon".

Entitas utama "Pinjaman" memiliki atribut seperti "Tanggal Pinjam", "Tanggal Kembali", dan "Status".

5.4.2 Pengorganisasian Data dalam Struktur Hierarkis:

Organisasikan data dalam struktur hierarkis dengan menentukan entitas utama dan entitas anak yang terkait. Misalnya, dalam basis data perpustakaan, entitas "Buku" dapat menjadi entitas utama, dengan entitas anak seperti "Penulis", "Penerbit", dan "Kategori".

Contoh:

Entitas utama "Buku" memiliki entitas anak seperti "Penulis" dan "Penerbit".

Entitas utama "Penulis" memiliki atribut seperti "Nama" dan "Negara".

Entitas utama "Penerbit" memiliki atribut seperti "Nama" dan "Alamat".

5.4.3 Penentuan Relasi Antara Entitas:

Tentukan relasi antara entitas dalam basis data untuk menggambarkan hubungan antara entitas tersebut. Misalnya, hubungan "Penulis" dengan "Buku" adalah relasi *one-to-many*, di mana satu penulis dapat menulis banyak buku, tetapi satu buku hanya ditulis oleh satu penulis.

Contoh:

Relasi antara "Penulis" dan "Buku" adalah *one-to-many*, di mana satu penulis dapat menulis banyak buku, tetapi satu buku hanya ditulis oleh satu penulis.

Relasi antara "Buku" dan "Penerbit" adalah *one-to-one,* di mana satu buku diterbitkan oleh satu penerbit, dan satu penerbit dapat menerbitkan banyak buku.

5.4.4 Pemilihan Kunci dan Penentuan Keterkaitan Data:

Pilih kunci utama untuk setiap entitas, yang merupakan atribut unik yang mengidentifikasi setiap baris dalam tabel entitas. Misalnya, dalam entitas "Buku", kunci utama dapat berupa "ISBN".

Tetapkan keterkaitan antara entitas menggunakan kunci utama dan kunci asing untuk menghubungkan entitas anak dengan entitas utama. Misalnya, dalam entitas "Pinjaman", kunci asing "ID_Anggota" menghubungkan entitas dengan entitas "Anggota".

Contoh:

Dalam entitas "Buku", kunci utama "ISBN" mengidentifikasi setiap buku secara unik.

Dalam entitas "Penulis", kunci utama dapat berupa "ID_Penulis" yang unik untuk setiap penulis.

Dalam entitas "Buku", atribut "ID_Penulis" adalah kunci asing yang menghubungkan entitas dengan entitas "Penulis".

Pada tahap perancangan basis data dengan pendekatan terstruktur, Anda dapat menggunakan perangkat lunak desain basis data seperti perangkat lunak manajemen basis data (DBMS)

yang mendukung fitur dan alat untuk memodelkan struktur hierarkis dan mengatur entitas dan relasinya.

Penting untuk mencatat bahwa perancangan basis data dengan pendekatan terstruktur cocok untuk sistem dengan hierarki yang jelas dan data yang relatif sederhana. Namun, jika kompleksitas data meningkat atau ada kebutuhan untuk fleksibilitas yang lebih besar, perlu dipertimbangkan pendekatan desain basis data lainnya seperti desain basis data relasional atau berorientasi objek.

5.5 Kelebihan Pendekatan Terstruktur dalam Desain Basis Data

Kelebihan Pendekatan Terstruktur dalam Desain Basis Data adalah sebagai berikut:

1. Organisasi yang Teratur: Pendekatan terstruktur dalam desain basis data memastikan adanya struktur dan organisasi yang teratur. Data disimpan dalam tabel yang terpisah dan saling terhubung melalui kunci atau relasi. Hal ini memudahkan dalam mengelola dan mengakses data dengan lebih efisien.
2. Integritas Data yang Tinggi: Dalam pendekatan terstruktur, aturan dan batasan yang jelas diterapkan pada basis data. Hal ini membantu menjaga integritas data dengan mencegah adanya duplikasi atau inkonsistensi data. Misalnya, dengan penggunaan kunci unik dan kunci asing, integritas referensial dapat dipertahankan di antara tabel yang berhubungan.
3. Pemeliharaan yang Mudah: Desain basis data terstruktur memisahkan data menjadi entitas atau objek yang berbeda dan saling terkait. Ini memudahkan dalam pemeliharaan basis data karena jika ada perubahan pada entitas tertentu, hanya perlu dilakukan perubahan pada tabel yang terkait tanpa mempengaruhi tabel lainnya. Hal ini meminimalkan dampak perubahan dan mempercepat proses pemeliharaan.

4. Pengolahan Data yang Efisien: Pendekatan terstruktur memungkinkan operasi pengolahan data yang efisien. *Query* dan manipulasi data dapat dilakukan dengan cepat dan tepat menggunakan bahasa *query* seperti SQL (*Structured Query Language*). Indeks dapat digunakan untuk meningkatkan kecepatan pencarian data, dan optimasi *query* dapat dilakukan untuk mempercepat eksekusi *query* yang kompleks.
5. Skalabilitas: Desain basis data terstruktur memungkinkan penambahan dan perubahan data dengan mudah. Jika ada kebutuhan untuk menambahkan entitas baru atau mengubah struktur basis data, dapat dilakukan dengan mempertahankan integritas data yang ada. Hal ini memungkinkan skala basis data untuk tumbuh sesuai kebutuhan tanpa mengganggu fungsi dan konsistensi keseluruhan sistem.
6. Keseragaman dan Standarisasi: Pendekatan terstruktur dalam desain basis data memungkinkan adanya keseragaman dan standarisasi dalam penyimpanan dan pengelolaan data. Aturan dan konvensi desain yang konsisten diterapkan di seluruh basis data, yang mempermudah penggunaan dan pemahaman oleh pengembang dan pengguna sistem.
7. Dukungan untuk Analisis dan Pelaporan: Dalam desain basis data terstruktur, data disimpan dalam bentuk yang sesuai untuk analisis dan pelaporan. Dengan menggunakan *query* dan alat analisis yang tepat, data dapat diekstraksi dengan mudah dan digunakan untuk membuat laporan atau mendapatkan wawasan yang berharga.

Pendekatan terstruktur dalam desain basis data memiliki kelebihan-kelebihan tersebut di atas, tetapi juga perlu mempertimbangkan kebutuhan dan kompleksitas proyek tertentu. Terkadang, pendekatan lain seperti pendekatan berbasis objek atau pendekatan berbasis dokumen mungkin lebih cocok

tergantung pada jenis data dan persyaratan aplikasi yang sedang dikerjakan.

5.6 Keterbatasan Pendekatan Terstruktur dan Batasan Penggunaannya

Berikut adalah beberapa keterbatasan Pendekatan Terstruktur dalam Desain Basis Data:

1. Ketidakfleksibelan: Salah satu keterbatasan pendekatan terstruktur adalah kurangnya fleksibilitas dalam mengakomodasi perubahan yang tidak terduga dalam kebutuhan data. Jika terjadi perubahan dalam struktur data atau persyaratan bisnis, mungkin diperlukan perubahan yang signifikan dalam desain basis data terstruktur, yang dapat menjadi rumit dan memakan waktu.
2. Sulit untuk Mengatasi Data yang Kompleks: Pendekatan terstruktur mungkin tidak efektif dalam mengelola data yang sangat kompleks atau memiliki struktur yang dinamis. Misalnya, jika ada data yang memiliki hierarki yang kompleks atau hubungan yang rumit antara entitas, desain basis data terstruktur mungkin tidak mampu menggambarkannya dengan baik.
3. Masalah Kinerja: Dalam beberapa kasus, desain basis data terstruktur mungkin menghadapi masalah kinerja, terutama ketika menghadapi kueri yang kompleks atau ketika jumlah data menjadi sangat besar. Karena data disimpan dalam tabel terpisah dan saling terhubung, beberapa kueri dapat memerlukan penggabungan yang mahal secara komputasi atau pemindaian penuh tabel, yang dapat mengurangi kinerja.
4. Repetisi Data: Pendekatan terstruktur cenderung mengandung beberapa tingkat duplikasi data. Misalnya, jika terdapat entitas yang terkait dan informasi tentang entitas tersebut disimpan di beberapa tabel terpisah, maka ada kemungkinan terjadinya

duplikasi data. Hal ini dapat mempengaruhi integritas data dan menghasilkan masalah konsistensi.

5. Kesulitan dalam Mengelola Perubahan Skema: Jika ada perubahan pada skema basis data terstruktur, seperti menambahkan kolom baru atau mengubah tipe data, dapat sulit untuk mengelola perubahan tersebut secara konsisten di seluruh sistem. Hal ini terutama berlaku jika basis data terstruktur tersebut digunakan oleh banyak aplikasi yang tergantung pada skema yang konsisten.
6. Kurangnya Keterbacaan: Desain basis data terstruktur, terutama jika skemanya kompleks, mungkin sulit dibaca dan dipahami oleh orang yang tidak terbiasa dengan struktur basis data tersebut. Hal ini dapat menyulitkan dalam pengembangan, pemeliharaan, dan pemecahan masalah.
7. Kurangnya Dukungan untuk Objek Kompleks: Pendekatan terstruktur mungkin tidak memiliki dukungan yang memadai untuk menyimpan dan mengelola objek kompleks seperti gambar, audio, video, atau struktur data yang kompleks. Jika jenis data semacam ini penting dalam aplikasi, pendekatan terstruktur mungkin tidak efisien dalam menangani kebutuhan tersebut.

Penting untuk mempertimbangkan keterbatasan-keterbatasan ini ketika memilih pendekatan desain basis data. Terkadang, pendekatan lain seperti pendekatan berbasis objek atau pendekatan berbasis dokumen mungkin lebih cocok untuk mengatasi tantangan yang spesifik.

DAFTAR PUSTAKA

Garcia-Molina, Hector. 2008. *Database Sistems: The Complete Book.* India: Pearson Education.

Hernandez, M. J. 2020. *Database Design for Mere Mortals: A Hands-on Guide to Relational Database Design.* Netherlands: Addison-Wesley.

BAB 6
DESAIN BASIS DATA BERORIENTASI OBYEK

Oleh Irmawati

6.1 Pendahuluan

Desain Basis Data Berorientasi Obyek adalah proses merancang struktur dan organisasi basis data yang berdasarkan paradigma pemrograman berorientasi obyek (Dittrich, Dayal and Buchmann, 2012). Paradigma ini mencakup konsep-konsep seperti objek, kelas, pewarisan, asosiasi, dan polimorfisme. Basis data berorientasi obyek memungkinkan penyimpanan dan manipulasi data dalam bentuk yang lebih dekat dengan cara kita memahami dunia nyata.

Pendekatan obyek dalam desain basis data memungkinkan integrasi yang lebih baik antara struktur data dan fungsi yang beroperasi pada data tersebut. Dalam desain basis data berorientasi obyek, objek menjadi unit dasar penyimpanan data dan memiliki atribut yang mendefinisikan karakteristiknya. Objek dapat memiliki metode yang merepresentasikan perilaku atau operasi yang dapat dilakukan terhadap objek tersebut.

Tujuan utama desain basis data berorientasi obyek adalah menciptakan model data yang akurat, fleksibel, dan mudah dipahami. Dengan memanfaatkan konsep-konsep obyek, desain basis data berorientasi obyek dapat menggambarkan hubungan antar objek secara langsung, menyediakan mekanisme pewarisan untuk memfasilitasi perluasan fungsionalitas, dan memungkinkan abstraksi data yang lebih baik.

6.1.1 Perbedaan Basis Data Berorientasi Obyek dan Relasional

Basis Data Berorientasi Obyek dan Basis Data Relasional adalah dua model yang digunakan untuk mengorganisir dan mengelola data dalam sistem basis data. Meskipun keduanya memiliki tujuan yang sama, yaitu menyimpan, mengelola, dan mengambil data, ada perbedaan signifikan dalam cara mengatur dan memanipulasi data.

Berikut ini adalah penjelasan beserta perbedaan-perbedaan utamanya (Elmasri *et al.*, 2000):

1. Basis Data Berorientasi Obyek

   Basis Data Berorientasi Obyek adalah model yang didasarkan pada konsep pemrograman berorientasi obyek. Data disimpan dalam bentuk objek atau instance, yang memiliki atribut dan metode yang terkait dengannya.

   Beberapa karakteristiknya adalah sebagai berikut:

   a. Pewarisan: Objek dapat mewarisi atribut dan metode dari objek lain, memungkinkan hierarki kelas dan pewarisan sifat.
   b. Encapsulation (Pembungkusan): Data dan metode yang berhubungan ditempatkan dalam sebuah objek, yang memungkinkan kontrol akses terhadap data tersebut.
   c. Polimorfisme: Objek dapat memiliki tindakan atau operasi yang berbeda tergantung pada konteks penggunaannya.
   d. Identitas: Setiap objek memiliki identitas unik yang membedakannya dari objek lain.
   e. Relasi Objek: Objek dapat memiliki hubungan dengan objek lain melalui referensi atau pointer.

   Contoh kasus :

   Misalkan kita memiliki sistem manajemen perpustakaan. Kita dapat mewakili buku sebagai objek dengan atribut seperti judul, penulis, dan penerbit. Objek buku ini dapat berinteraksi dengan objek lain seperti objek peminjam, dan dapat memiliki metode seperti "pinjam" atau "kembalikan".

2. Basis Data Relasional :
   Basis Data Relasional adalah model basis data yang didasarkan pada teori hubungan atau relasi. Data disimpan dalam bentuk tabel yang terdiri dari baris dan kolom. Setiap tabel merepresentasikan sebuah entitas, sedangkan baris dan kolom mewakili instansi dan atribut dari entitas tersebut. Beberapa karakteristiknya adalah sebagai berikut:
   a. Struktur Tabel: data disimpan dalam bentuk tabel dua dimensi, dimana setiap kolom memiliki tipe data yang didefinisikan dan setiap baris mewakili entitas atau instansi.
   b. *Primary Key*: setiap tabel memiliki satu atau lebih kolom yang berperan sebagai *primary key*, yang memberikan identitas unik pada setiap baris.
   c. *Foreign Key* : relasi antar tabel dapat dibentuk menggunakan *foreign key*, yang menghubungkan kolom dalam satu tabel dengan kolom dalam tabel lain.
   d. Normalisasi: menerapkan prinsip normalisasi untuk mengurangi redundansi data dan menjaga integritas data.

   Contoh kasus :
   Misalkan kita memiliki sistem manajemen penjualan. Kita dapat menggunakan tabel "Produk" dengan kolom seperti ID Produk, Nama Produk, dan Harga. Kemudian kita dapat menggunakan tabel "Penjualan" dengan kolom seperti ID Penjualan, ID Produk (kunci asing), dan Jumlah.

Perbedaan utama antara basis data berorientasi obyek dan basis data relasional adalah sebagai berikut:

1. Representasi Data: basis data berorientasi obyek menggunakan objek sebagai unit dasar untuk merepresentasikan data sedangkan basis data relasional menggunakan tabel yang terdiri dari baris dan kolom.

2. Konsep Dasar: basis data berorientasi obyek berfokus pada konsep pemrograman berorientasi obyek seperti pewarisan, polimorfisme, dan encapsulation, sedangkan basis data relasional berfokus pada teori hubungan dan normalisasi.
3. Pengaksesan Data: basis data berorientasi obyek menggunakan metode pemanggilan objek untuk mengakses data, sedangkan basis data relasional menggunakan bahasa kueri seperti SQL.
4. Skalabilitas: basis data berorientasi obyek lebih cocok untuk aplikasi dengan kompleksitas tinggi yang membutuhkan pengelolaan objek yang kompleks, sedangkan basis data relasional lebih umum digunakan untuk aplikasi bisnis yang membutuhkan konsistensi dan keamanan data yang tinggi.
5. Penanganan Data Terstruktur: basis data berorientasi obyek lebih mampu mengelola data terstruktur yang kompleks, seperti gambar, suara, dan video, sedangkan basis data relasional lebih terfokus pada data terstruktur dalam bentuk tabel.
6. Pemasangan: basis data berorientasi obyek memerlukan sistem manajemen basis data berorientasi obyek, sedangkan basis data relasional memerlukan sistem manajemen basis data relasional (RDBMS).

Masing-masing model basis data ini memiliki kelebihan dan kelemahan tergantung pada kebutuhan aplikasi. Pemilihan antara basis data berorientasi obyek dan basis data relasional tergantung pada sifat data, kompleksitas aplikasi, dan kebutuhan bisnis yang dihadapi.

Keuntungan dan tantangan dalam desain basis data berorientasi obyek merujuk pada manfaat dan kendala yang muncul ketika menggunakan pendekatan berorientasi obyek dalam merancang dan implementasi basis data (Brown, 2000). Berikut adalah keuntungan dan tantangan tersebut:

Keuntungan Desain Basis Data Berorientasi Obyek:

1. Representasi yang dekat dengan dunia nyata: dalam desain basis data berorientasi obyek, entitas dunia nyata direpresentasikan secara langsung sebagai objek. Hal ini memungkinkan pemodelan yang lebih akurat dan semantik dari data, karena objek dalam basis data mencerminkan struktur dan perilaku entitas dalam dunia nyata.
2. Fleksibilitas dan perluasan: desain basis data berorientasi obyek memungkinkan fleksibilitas dalam menambahkan dan mengubah struktur data. Dengan menggunakan konsep pewarisan, kelas dapat diwarisi dan diperluas untuk menciptakan hirarki kelas yang lebih kaya. Hal ini memudahkan pengembangan dan perubahan sistem secara modular.
3. Hubungan antar objek yang kuat: basis data berorientasi obyek memungkinkan penggunaan hubungan yang lebih kuat antara objek. Konsep asosiasi, komposisi, dan agregasi memungkinkan penggabungan objek yang saling terkait secara langsung, mencerminkan hubungan nyata dalam dunia nyata.
4. Abstraksi dan pemrograman lebih baik: dalam desain basis data berorientasi obyek, pengguna dapat memanfaatkan abstraksi objek yang kuat untuk mempermudah pemrograman dan pengelolaan data. Konsep-konsep seperti enkapsulasi, polimorfisme, dan pewarisan memungkinkan penggunaan kode yang lebih efisien dan reusable.
5. Integritas dan keamanan: desain basis data berorientasi objek memungkinkan penerapan kendala integritas dan keamanan lebih baik. Konsep enkapsulasi memungkinkan kontrol akses yang lebih baik terhadap data dan metode objek, sehingga meningkatkan keamanan dan integritas data.

Tantangan Desain Basis Data Berorientasi Obyek:

1. Kompleksitas desain: desain basis data berorientasi obyek dapat menjadi lebih kompleks dibandingkan dengan desain basis data relasional. Konsep-konsep seperti pewarisan, hubungan antar objek, dan polimorfisme memerlukan pemahaman yang mendalam dan perencanaan yang matang untuk mencapai desain yang efektif.
2. Performa: performa bisa menjadi tantangan dalam desain basis data berorientasi obyek. Operasi kompleks seperti pengambilan data, operasi gabungan objek, dan navigasi yang dalam dapat mempengaruhi kinerja sistem, terutama jika skala data sangat besar.
3. Kesulitan integrasi: integrasi dengan sistem yang menggunakan basis data relasional atau paradigma non-obyek dapat menjadi tantangan. Migrasi dari basis data relasional ke obyek atau integrasi dengan sistem lama dapat melibatkan upaya yang signifikan dan memerlukan perencanaan yang matang.
4. Keterbatasan standar: standar dan bahasa query untuk basis data berorientasi obyek mungkin belum seumum atau sebaku dalam basis data relasional. Hal ini dapat menyulitkan interoperabilitas dan pertukaran data antara sistem yang menggunakan pendekatan yang berbeda.
5. Keahlian dan pengembangan: desain basis data berorientasi obyek membutuhkan keahlian khusus dalam konsep obyek dan metodologi desain. Sumber daya manusia dan pelatihan mungkin diperlukan untuk mengembangkan keahlian yang diperlukan untuk desain dan pengembangan basis data berorientasi objek.

6.2 Konsep Dasar Obyek dalam Basis Data

Konsep dasar obyek dalam basis data merujuk pada elemen-elemen fundamental yang terkait dengan representasi dan pengelolaan objek dalam basis data berorientasi obyek. Berikut adalah beberapa konsep dasar obyek yang penting dalam basis data berorientasi obyek.

1. Objek: unit dasar dalam basis data berorientasi obyek. Objek mewakili entitas nyata atau konsep dalam dunia nyata, seperti orang, barang, atau transaksi. Setiap objek memiliki identitas yang unik dan dapat memiliki atribut dan metode terkait.
2. Kelas: panduan atau *blueprint* untuk menciptakan objek. Kelas mendefinisikan struktur dan perilaku yang dimiliki oleh objek yang berasal dari kelas tersebut. Kelas mendefinisikan atribut (variabel) yang dimiliki oleh objek dan metode (fungsi) yang dapat dijalankan oleh objek tersebut.
3. Atribut: karakteristik atau properti dari objek. Atribut mewakili data yang terkait dengan objek. Misalnya, dalam kelas "Mobil", atribut dapat mencakup "merek", "warna", dan "tahun produksi". Setiap objek yang berasal dari kelas tersebut akan memiliki nilai yang spesifik untuk setiap atribut.

4. Metode: tindakan yang dapat dilakukan oleh objek. Metode adalah fungsi yang terkait dengan objek dan digunakan untuk memanipulasi objek atau mengakses atributnya. Misalnya, dalam kelas "Mobil", metode mungkin termasuk "menyalakan mesin" atau "mengganti warna mobil".
5. Pewarisan: konsep yang memungkinkan kelas untuk mewarisi atribut dan metode dari kelas lain. Dalam pewarisan, kelas yang mewarisi disebut "kelas turunan" atau "subkelas", sedangkan kelas yang memberikan warisan disebut "kelas induk" atau "superkelas". Pewarisan memungkinkan penciptaan hierarki kelas dan memfasilitasi penggunaan kembali kode.

6. Asosiasi: menggambarkan hubungan antara dua atau lebih objek. Asosiasi dapat memiliki kardinalitas, yaitu menentukan berapa banyak objek yang terlibat dalam hubungan tersebut. Misalnya, dalam basis data toko, asosiasi antara "Pelanggan" dan "Pesanan" bisa berarti bahwa satu pelanggan dapat memiliki banyak pesanan, tetapi satu pesanan hanya dapat dimiliki oleh satu pelanggan.
7. Komposisi dan Agregasi: bentuk hubungan yang lebih kuat antara objek. Komposisi menggambarkan hubungan yang bersifat sebagian-keseluruhan, di mana objek yang lebih besar terdiri dari objek yang lebih kecil. Agregasi menggambarkan hubungan yang bersifat kepemilikan, di mana objek yang lebih besar memiliki referensi ke objek yang lebih kecil.

Konsep dasar objek dalam basis data berorientasi obyek ini membentuk fondasi untuk merancang skema dan struktur data yang fleksibel dan intuitif.

6.2.1 Objek dan Kelas dalam Paradigma Obyek

Objek dan kelas adalah dua konsep fundamental dalam paradigma pemrograman berorientasi objek (OOP). Dalam OOP, program direpresentasikan sebagai kumpulan objek yang saling berinteraksi melalui pertukaran pesan. Berikut adalah penjelasan tentang objek dan kelas :

1. Objek

   Objek adalah instansi konkret dari suatu kelas. Setiap objek memiliki keadaan yang diwakili oleh atribut (data) dan perilaku yang diwakili oleh metode (fungsi). Atribut mewakili data yang dimiliki objek, sedangkan metode memungkinkan objek melakukan operasi atau tindakan tertentu. Objek dalam OOP mencerminkan entitas nyata atau konsep abstrak dalam dunia nyata atau dalam konteks perangkat lunak.

   Misalnya, dalam sebuah sistem perbankan, objek "Rekening" dapat diwakili oleh atribut seperti nomor rekening, saldo, dan

pemilik, serta metode seperti setoran, penarikan, dan transfer. Setiap objek "Rekening" yang dibuat dalam sistem akan memiliki keadaan dan perilaku yang unik.

2. Kelas
   Kelas adalah sebuah *blueprint* atau *template* yang mendefinisikan struktur, atribut, dan perilaku yang dimiliki oleh objek yang dibuat berdasarkan kelas tersebut. Kelas berfungsi sebagai entitas yang menggambarkan karakteristik yang umum untuk semua objek yang akan dibuat berdasarkan kelas tersebut. Dalam kelas, kita mendefinisikan atribut dan metode yang akan dimiliki oleh objek.
   Misalnya, dalam kasus objek "Rekening" di atas, kelas "Rekening" akan mendefinisikan atribut seperti nomor rekening, saldo, dan pemilik, serta metode seperti setoran, penarikan, dan transfer. Kelas ini memberikan struktur dan kerangka kerja yang diperlukan untuk membuat objek "Rekening" dalam sistem.

Dalam contoh sistem perpustakaan, kita dapat membuat objek "Buku" bernama "BukuA" dengan atribut judul "Harry Potter", penulis "J.K. Rowling", dan tahun terbit "1997". Objek ini kemudian memiliki kemampuan untuk dipinjam dan dikembalikan sesuai dengan metode yang didefinisikan dalam kelas "Buku".

Dalam paradigma pemrograman berorientasi objek, konsep kelas dan objek memungkinkan kita untuk memodelkan sistem dengan cara yang lebih terstruktur dan modular. Kita dapat menggunakan kelas sebagai sarana untuk mengelompokkan data dan fungsi yang terkait bersama, sehingga memudahkan pengembangan, pemeliharaan, dan pemahaman kode. Objek-objek yang dibuat dari kelas-kelas ini dapat saling berinteraksi, mengirim pesan satu sama lain, dan bekerja sama untuk mencapai tujuan sistem yang lebih besar.

6.2.2 Notasi dan Metode Desain Basis Data Berorientasi Objek

Desain basis data berorientasi objek melibatkan penggunaan notasi dan metode yang khusus untuk merancang basis data yang berfokus pada objek-objek yang ada dalam sistem. Berikut ini adalah penjelasan tentang bahasa pemodelan objek yang digunakan dalam desain basis data berorientasi objek.

1. UML (*Unified Modeling Language*): UML adalah bahasa pemodelan yang sering digunakan dalam desain basis data berorientasi objek. UML menyediakan notasi grafis untuk menggambarkan kelas, atribut, metode, relasi, dan konsep-konsep lain yang terkait dengan desain basis data berorientasi objek.
2. ORM (*Object-Relational Mapping*): ORM adalah metode yang digunakan untuk memetakan objek-objek dalam basis data berorientasi objek ke dalam struktur relasional yang dapat disimpan dalam sistem manajemen basis data relasional (RDBMS). Ini memungkinkan penggunaan fitur-fitur basis data relasional dengan menggunakan objek dalam kode aplikasi.

Notasi dan metode yang digunakan dalam desain basis data berorientasi objek memiliki beberapa aspek penting. Berikut adalah penjelasan tentang notasi dan metode yang digunakan.

1. Notasi Objek
    a. Kelas: Notasi kelas untuk mendefinisikan objek dalam basis data. Biasanya, notasi kelas mengikuti format yang mirip dengan notasi kelas dalam pemrograman berorientasi objek, seperti UML (*Unified Modeling Language*).

Gambar 6.1. Contoh notasi kelas dalam UML
(Sumber : Pribadi)

b. Atribut: properti dari sebuah objek. Biasanya, atribut memiliki tipe data yang menentukan jenis nilai yang dapat disimpan di dalamnya. Berikut ini adalah contoh atribut dalam UML.
   - id: int
   - name: string

c. Metode: perilaku yang dapat dilakukan oleh objek. Biasanya didefinisikan dengan menentukan nama, parameter, dan tipe data kembalian (jika ada). Berikut adalah contoh metode dalam UML.
   + getAddress(): string
   + setAddress(address: string): void

2. Metode dalam Desain Basis Data Berorientasi Obyek

a. Identifikasi Objek: melibatkan identifikasi objek-objek yang ada dalam sistem dan mencatat atribut dan perilaku yang relevan untuk setiap objek. Pendekatan yang digunakan adalah pemodelan berbasis skenario, dimana skenario penggunaan sistem digunakan untuk mengidentifikasi objek-objek yang terlibat.

b. Pemodelan Kelas: kelas-kelas dibuat untuk mewakili objek-objek tersebut. Pemodelan kelas melibatkan penentuan atribut-atribut dan metode-metode yang akan dimiliki oleh kelas tersebut. Notasi UML digunakan untuk menggambarkan kelas dan hubungannya dengan kelas lain.

c. Relasi Antar Kelas: menggambarkan hubungan antara objek-objek dalam basis data. Beberapa jenis relasi yang umum digunakan adalah hubungan satu-ke-satu, satu-ke-banyak, dan banyak-ke-banyak. Notasi UML menyediakan notasi khusus untuk menggambarkan relasi antar kelas.
d. Pewarisan: konsep yang memungkinkan kelas untuk mewarisi atribut dan metode dari kelas lain. Dalam desain basis data berorientasi objek, pewarisan dapat digunakan untuk mengatur hierarki kelas dan membangun struktur objek yang kompleks.
e. Agregasi dan Komposisi: relasi dimana objek dari kelas satu dapat berisi objek dari kelas lain. Agregasi memungkinkan pembentukan objek yang kompleks dengan menggabungkan beberapa objek yang lebih sederhana. Komposisi adalah bentuk khusus dari agregasi di mana objek yang dibuat secara dinamis hanya ada selama objek yang menciptakannya masih ada.
f. Polimorfisme: memungkinkan penggunaan metode yang sama dengan implementasi yang berbeda oleh kelas-kelas yang berbeda. Hal ini memungkinkan fleksibilitas dalam pemanggilan metode dan penggunaan objek dalam basis data berorientasi objek.

Dalam desain basis data berorientasi objek, notasi dan metode ini membantu dalam merancang struktur objek yang terorganisir dengan baik, memodelkan hubungan antar objek, dan menggambarkan perilaku objek dalam sistem. Dengan menggunakan pendekatan ini, desain basis data dapat menjadi lebih fleksibel, modular, dan dapat diubah dengan mudah ketika kebutuhan sistem berubah.

6.3 Model Data Berorientasi Obyek

Model Data Berorientasi Objek adalah sebuah paradigma pemodelan yang digunakan dalam pengembangan perangkat lunak untuk merepresentasikan dan mengelola data sebagai kumpulan objek yang saling berinteraksi. Model ini didasarkan pada konsep-konsep dasar dalam pemrograman berorientasi objek, seperti enkapsulasi, pewarisan, dan polimorfisme.

Dalam Model Data Berorientasi Objek, data dianggap sebagai objek yang memiliki atribut (data) dan perilaku (metode). Objek-objek ini bisa memiliki hubungan dan interaksi antara satu sama lain melalui pewarisan, asosiasi, dan komposisi. Pendekatan ini memungkinkan pemodelan yang lebih intuitif dan fleksibel, dengan kemampuan untuk merepresentasikan hubungan dan struktur data yang kompleks.

Berikut adalah beberapa elemen kunci dalam Model Data Berorientasi Obyek:

1. Kelas (*Class*): Kelas mendefinisikan atribut dan perilaku yang dimiliki oleh objek. Contohnya, dalam sebuah sistem manajemen pegawai, kita bisa memiliki kelas "Pegawai" dengan atribut seperti nama, alamat, dan gaji, serta metode untuk mengelola data tersebut.
2. Objek (*Object*): setiap objek memiliki atribut yang unik dan dapat melakukan operasi yang didefinisikan oleh kelasnya. Sebagai contoh, kita dapat menciptakan objek "Pegawai1" dari kelas "Pegawai" dengan atribut tertentu.
3. Pewarisan (Inheritance): konsep dimana sebuah kelas dapat mewarisi atribut dan metode dari kelas lain yang disebut "kelas induk" atau "superclass". Pewarisan memungkinkan kita untuk membuat hierarki kelas, di mana kelas anak dapat mewarisi sifat-sifat kelas induk dan menambahkan sifat-sifat baru. Misalnya, kita dapat memiliki kelas "Manajer" yang mewarisi atribut dan metode dari kelas "Pegawai", namun juga

memiliki atribut dan metode tambahan yang spesifik untuk peran manajerial.

4. Asosiasi (*Association*): hubungan antara dua atau lebih objek. Objek-objek tersebut dapat berinteraksi dan saling bergantung satu sama lain. Misalnya, dalam sistem pemesanan online, ada hubungan asosiasi antara objek "Pemesanan" dan "Pelanggan", di mana satu pelanggan dapat memiliki banyak pemesanan.
5. Komposisi (*Composition*): bentuk hubungan antara objek dimana satu objek menjadi bagian penting dari objek lain. Dalam hubungan komposisi, jika objek utama dihapus, objek bagian juga akan terhapus. Contohnya, dalam sistem perpustakaan, objek "Buku" dapat berkomposisi dari objek "Halaman", di mana jika buku dihapus, halaman-halaman juga akan dihapus.

Contoh Kasus:
Misalkan Anda ingin membuat aplikasi manajemen perpustakaan. Anda dapat menggunakan model data berorientasi objek untuk mengorganisasi data dan operasi terkait. Beberapa kelas yang mungkin ada dalam kasus ini adalah:

1. Kelas "Buku" dengan atribut seperti "judul", "penulis", dan "tahun terbit". Metode pada kelas ini dapat mencakup peminjaman, pengembalian, dan penambahan buku ke perpustakaan.
2. Kelas "Anggota" dengan atribut seperti "nama", "alamat", dan "nomor anggota". Metode pada kelas ini dapat mencakup pendaftaran anggota baru, pencarian anggota, dan penghapusan anggota.
3. Kelas "Peminjaman" dengan atribut seperti "tanggal pinjam", "tanggal kembali", dan "status". Metode pada kelas ini dapat mencakup perhitungan denda keterlambatan pengembalian dan pengelolaan status peminjaman.

Dengan menggunakan model data berorientasi objek, Anda dapat membuat objek-objek yang mewakili buku, anggota, dan peminjaman, dan mengatur hubungan dan interaksi antara objek-objek tersebut. Misalnya, Anda dapat membuat objek "BukuA" dari kelas "Buku" dengan atribut "judul = 'Harry Potter'" dan metode "pinjam()" yang mengubah status buku menjadi dipinjam ketika buku tersebut dipinjam oleh anggota tertentu.

Dalam kasus ini, model data berorientasi objek memungkinkan pengorganisasian data perpustakaan menjadi entitas terpisah dengan atribut dan metode terkait yang mempermudah pemrograman dan pengelolaan sistem.

6.4 Proses Desain Basis Data Berorientasi Objek

Proses desain basis data berorientasi obyek melibatkan langkah-langkah yang terstruktur untuk merancang struktur basis data yang sesuai dengan paradigma pemrograman berorientasi obyek (Booch *et al.*, 2008). Dalam proses ini, fokus diberikan pada pengorganisasian data dalam bentuk objek yang saling terkait dan memanfaatkan konsep seperti kelas, objek, pewarisan, enkapsulasi, dan polimorfisme.

Berikut ini adalah langkah-langkah umum dalam proses desain basis data berorientasi obyek:

1. Identifikasi Kebutuhan: tahap ini melibatkan pemahaman yang mendalam tentang kebutuhan bisnis atau kasus penggunaan. Identifikasi aktor utama, fungsi sistem, dan persyaratan data yang harus dipenuhi.
2. Analisis Kasus: proses ini berfokus pada pemodelan objek dan hubungan antara objek-objek yang ada dalam konteks kasus penggunaan. Ini melibatkan pemahaman tentang atribut, metode, dan relasi antara objek-objek tersebut.
3. Identifikasi Kelas: pada tahap ini, kelas-kelas utama yang mewakili objek-objek dalam kasus penggunaan diidentifikasi. Setiap kelas memiliki atribut dan metode yang sesuai.

4. Hubungan Antar Kelas: hubungan antara kelas-kelas yang ada ditentukan pada tahap ini. Hubungan ini dapat berupa hubungan pewarisan (inheritance), asosiasi (association), atau komposisi (composition) antara kelas-kelas.
5. Skema Basis Data: skema basis data dirancang berdasarkan kelas-kelas yang diidentifikasi sebelumnya. Setiap kelas akan menjadi tabel dalam skema basis data dan atribut kelas akan menjadi kolom dalam tabel tersebut.
6. Normalisasi: digunakan untuk menghilangkan anomali data dan memastikan integritas data. Prinsip normalisasi diterapkan pada skema basis data untuk mengurangi redundansi dan meningkatkan efisiensi.
7. Implementasi: tahap ini melibatkan implementasi skema basis data yang dirancang menggunakan bahasa pemrograman berorientasi obyek seperti Java, C++, atau Python. Objek-objek diinstantiasi dan data disimpan dalam basis data.
8. Pengujian: skema basis data dan aplikasi yang terkait diuji untuk memastikan bahwa data disimpan dan diakses dengan benar. Pengujian ini mencakup uji fungsionalitas, performa, dan kehandalan basis data.

6.4.1 Identifikasi Kebutuhan Pengguna dan Analisis Sistem

Proses Identifikasi Kebutuhan Pengguna dan Analisis Sistem merupakan langkah penting dalam desain sistem berorientasi objek. Tujuannya adalah memahami kebutuhan pengguna dan menganalisis sistem yang akan dikembangkan sebelum merancang solusi berorientasi obyek yang sesuai. Berikut adalah penjelasan tentang kedua proses tersebut:

1. Identifikasi Kebutuhan Pengguna
   a. Mengumpulkan Informasi: Proses dimulai dengan mengumpulkan informasi tentang kebutuhan dan persyaratan pengguna. Ini melibatkan wawancara dengan pengguna, observasi langsung, survei, atau analisis

dokumen terkait. Tujuannya adalah untuk mendapatkan pemahaman yang jelas tentang masalah yang harus diselesaikan dan tujuan yang ingin dicapai.

b. Mengidentifikasi Kebutuhan: Berdasarkan informasi yang terkumpul, langkah selanjutnya adalah mengidentifikasi kebutuhan pengguna secara sistematis. Ini melibatkan pemodelan kebutuhan dengan menggunakan alat seperti use case diagram atau user stories. Kebutuhan tersebut harus spesifik, terukur, dapat dicapai, relevan, dan terbatas oleh waktu (SMART).
c. Menganalisis Prioritas: Setelah kebutuhan diidentifikasi, prioritas harus ditentukan. Ini melibatkan memahami urgensi, pentingnya, dan dampak dari setiap kebutuhan. Hal ini membantu dalam menentukan urutan pengembangan dan pengalokasian sumber daya yang tepat.

2. Analisis Sistem
   a. Mendefinisikan Lingkup: Penting untuk mendefinisikan batasan dan lingkup sistem yang akan dikembangkan. Ini melibatkan mengidentifikasi entitas yang terlibat, fungsi yang harus dilakukan, dan batasan sistem dalam hal waktu, anggaran, dan teknologi yang tersedia.
   b. Identifikasi Objek dan Hubungan: Dalam desain berorientasi obyek, langkah ini melibatkan identifikasi objek yang akan menjadi bagian dari sistem dan hubungan antara objek tersebut. Analisis ini membantu dalam memahami struktur sistem dan hubungan antara objek-objek tersebut.
   c. Menentukan Fungsi dan Metode: Proses ini melibatkan menentukan fungsi yang harus dilakukan oleh setiap objek dan metode yang akan digunakan untuk memanipulasi objek atau mengakses atributnya. Ini melibatkan pemodelan aliran data dan aliran kontrol dalam sistem.

d. Menganalisis Keamanan dan Kinerja: Selama analisis sistem, perhatian juga diberikan pada keamanan dan kinerja sistem. Keamanan melibatkan identifikasi ancaman potensial dan penentuan langkah-langkah keamanan yang tepat. Analisis kinerja melibatkan memeriksa bagaimana sistem akan beroperasi dalam skenario yang berbeda dan mengidentifikasi area yang mungkin perlu dioptimalkan.

Dalam proses ini, dokumentasi yang lengkap dan berkelanjutan sangat penting untuk mengkomunikasikan kebutuhan pengguna dan hasil analisis sistem kepada tim pengembang dan pemangku kepentingan lainnya.

Melalui proses identifikasi kebutuhan pengguna dan analisis sistem, tim pengembang dapat memahami kebutuhan pengguna dengan baik, menganalisis sistem secara menyeluruh, dan merancang solusi yang sesuai dalam desain berorientasi obyek. Hal ini membantu dalam menciptakan sistem yang memenuhi kebutuhan pengguna dengan efektif dan efisien.

6.4.2 Normalisasi Kelas dan Atribut

Normalisasi kelas dan atribut dalam desain basis data berorientasi objek melibatkan pengorganisasian kelas dan atribut dalam struktur yang efisien, menghilangkan redundansi data, dan memastikan integritas data (Bruegge and H Dutoit, 2010). Tujuannya adalah untuk memperbaiki desain basis data, meningkatkan kualitas kode, dan memudahkan pemeliharaan.

1. Normalisasi kelas dalam basis data berorientasi obyek melibatkan pemisahan kelas yang memiliki tanggung jawab yang terkait namun berbeda-beda. Setiap kelas sebaiknya memiliki tanggung jawab yang terdefinisi dengan jelas dan tidak bertanggung jawab atas terlalu banyak hal. Normalisasi kelas membantu dalam menciptakan desain basis data yang

modular, dengan kelas-kelas yang memiliki peran dan tanggung jawab yang terfokus.

Sebagai contoh, dalam sebuah sistem manajemen universitas, kita dapat memiliki kelas "Mahasiswa" yang bertanggung jawab untuk informasi seperti nama, alamat, dan nomor telepon mahasiswa. Namun, jika kita ingin memisahkan tanggung jawab pengelolaan mata kuliah yang diambil oleh mahasiswa, kita dapat melakukan normalisasi dengan memisahkan kelas "Mata Kuliah" yang berfungsi untuk menyimpan informasi seperti kode mata kuliah, nama mata kuliah, dan sks.

2. Normalisasi atribut dalam basis data berorientasi obyek melibatkan pemisahan atribut yang terlalu banyak bergantung satu sama lain atau memiliki tanggung jawab yang berbeda-beda. Setiap kelas sebaiknya memiliki atribut yang relevan dan berkaitan langsung dengan tanggung jawabnya. Normalisasi atribut membantu dalam menciptakan struktur basis data yang terorganisir dengan baik, menghindari redundansi, dan memastikan integritas data.

   Misalnya, dalam sistem pemesanan online, kita dapat memiliki kelas "Pesanan" dengan atribut seperti nomor pesanan, tanggal pesanan, dan total harga. Namun, jika atribut "Total Harga" bergantung pada atribut "Jumlah Produk" dan "Harga Satuan", kita dapat melakukan normalisasi dengan memisahkan atribut "Total Harga" ke dalam metode atau kelas terpisah yang bertanggung jawab untuk menghitung total harga.

6.5 Integritas dan Keamanan

Integritas dan keamanan basis data berorientasi objek adalah dua aspek penting dalam desain dan pengelolaan basis data yang berfokus pada objek (Hassan, 2005). Integritas basis data berkaitan dengan kebenaran, konsistensi, dan validitas data yang

disimpan dalam basis data, sementara keamanan basis data berkaitan dengan perlindungan data terhadap akses yang tidak sah, kerusakan, atau penggunaan yang tidak sesuai.

1. Integritas Basis Data Berorientasi Obyek:
    a. Integritas Entitas: Memastikan setiap entitas atau objek dalam basis data memiliki atribut yang sesuai dan nilai yang valid. Misalnya, dalam basis data mahasiswa, pastikan setiap entitas mahasiswa memiliki atribut nama, nim, dan alamat yang tidak boleh kosong.
    b. Integritas Referensial: Menjaga hubungan referensial antara entitas dalam basis data. Misalnya, jika ada hubungan antara entitas "Pesanan" dan "Pelanggan", pastikan bahwa setiap pesanan memiliki referensi ke pelanggan yang valid.
    c. Integritas Domain: Memastikan nilai atribut dalam basis data sesuai dengan domain yang ditentukan. Misalnya, jika atribut "Usia" dalam basis data memiliki batasan nilai 0-100, pastikan tidak ada data dengan nilai usia di luar rentang tersebut.

    Contoh kasus:

    Misalkan kita memiliki basis data untuk sistem manajemen perpustakaan. Dalam desain basis data tersebut, kita perlu memastikan integritas data yang tepat. Sebagai contoh:

    a. Integritas entitas: Setiap entitas "Buku" harus memiliki atribut seperti "Judul", "Penulis", dan "Penerbit" yang tidak boleh kosong.
    b. Integritas referensial: Setiap entitas "Peminjaman" harus memiliki referensi yang valid ke entitas "Anggota Perpustakaan" dan "Buku" yang ada dalam basis data.
    c. Integritas domain: Atribut "Jumlah Salinan" dalam entitas "Buku" harus memiliki nilai yang lebih besar dari 0.

2. Keamanan Basis Data Berorientasi Obyek:
    a. Keamanan Akses: Melibatkan pengaturan hak akses pengguna dan peran untuk mencegah akses yang tidak sah ke data dalam basis data. Misalnya, memastikan hanya pengguna terotorisasi yang memiliki akses untuk membaca atau mengubah data.
    b. Keamanan Transaksi: Melindungi integritas data selama proses transaksi dengan menggunakan mekanisme seperti locking atau penguncian untuk mencegah konflik dan kehilangan data.
    c. Keamanan Fisik: Melibatkan langkah-langkah untuk melindungi basis data dari kerusakan fisik atau bencana seperti kebakaran atau banjir.

    Contoh kasus:

    Dalam sistem e-commerce, keamanan basis data menjadi kritis. Contoh langkah-langkah keamanan yang dapat diimplementasikan:

    a. Keamanan akses: Menerapkan otentikasi dan otorisasi yang kuat untuk mengontrol akses pengguna ke data sensitif seperti informasi pembayaran.
    b. Keamanan transaksi: Menggunakan protokol enkripsi untuk melindungi data transaksi saat dikirim melalui jaringan.
    c. Keamanan fisik: Memiliki kebijakan cadangan data yang rutin dan melindungi basis data dengan firewall dan solusi keamanan jaringan.

6.6 Tantangan dan Perkembangan

Tantangan dan Perkembangan Desain Basis Data Berorientasi Objek merupakan topik yang menarik dalam pengembangan basis data. Desain Basis Data Berorientasi Objek adalah pendekatan dalam mengorganisir dan mengelola data dengan menggunakan konsep dan prinsip yang terkait dengan

pemrograman berorientasi objek (Harrington, 2000). Berbeda dengan pendekatan tradisional basis data relasional, desain basis data berorientasi obyek menggabungkan data dan operasi yang berkaitan dengannya menjadi satu kesatuan yang dikenal sebagai objek. Objek-objek ini dapat memiliki properti, perilaku, dan hubungan dengan objek lainnya. Desain basis data berorientasi obyek mencakup konsep seperti pewarisan, enkapsulasi, polimorfisme, dan asosiasi antara objek.

Salah satu tantangan dalam desain basis data berorientasi obyek adalah mencocokkan model data dengan kebutuhan bisnis. Mengidentifikasi objek dan relasi antara objek-objek tersebut dapat menjadi kompleks, terutama dalam sistem yang kompleks dengan banyak entitas dan hubungan di antara mereka. Selain itu, integrasi dengan sistem dan teknologi lainnya juga dapat menjadi tantangan, terutama jika ada kebutuhan untuk berinteraksi dengan sistem yang menggunakan pendekatan basis data yang berbeda, seperti basis data relasional.

Namun, desain basis data berorientasi obyek juga memiliki beberapa keuntungan dan perkembangan penting. Salah satu manfaat utama adalah kemampuan untuk mewakili dunia nyata dengan lebih akurat. Objek dalam desain basis data berorientasi obyek dapat mencerminkan entitas dan hubungan di dunia nyata dengan lebih baik daripada model relasional tradisional. Hal ini dapat membantu dalam pemodelan dan analisis domain bisnis yang kompleks. Selain itu, desain basis data berorientasi obyek juga mendukung pengembangan perangkat lunak yang lebih modular dan terstruktur. Dengan menggunakan konsep pemrograman berorientasi objek, objek dalam basis data dapat memiliki metode dan perilaku yang terkait dengannya. Hal ini memungkinkan pengembang untuk mengelompokkan fungsi-fungsi terkait bersama-sama dalam objek, yang dapat meningkatkan keberlanjutan, pemeliharaan, dan pengembangan perangkat lunak.

Perkembangan terkini dalam desain basis data berorientasi obyek melibatkan beberapa tren yang signifikan. Berikut adalah beberapa tren utama dalam desain basis data berorientasi obyek:

1. Objek Relasional (ORDBMS): Salah satu tren penting dalam desain basis data obyek adalah integrasi antara konsep-konsep berorientasi objek dengan basis data relasional. Objek Relasional Database Management Systems (ORDBMS) menggabungkan fitur-fitur berorientasi objek seperti pewarisan, tipe kompleks, dan metode dengan model basis data relasional. Hal ini memungkinkan penggunaan kombinasi yang kuat antara kedua paradigma tersebut, yang menggabungkan kelebihan dari keduanya.
2. Penyimpanan Dokumen: Dalam beberapa tahun terakhir, basis data berorientasi objek juga telah mengadopsi model penyimpanan dokumen yang fleksibel. Basis data dokument-oriented menyimpan data dalam format dokumen (seperti JSON atau XML) dan menggunakan struktur semi-terstruktur untuk mengorganisir informasi. Pendekatan ini memungkinkan penyimpanan dan pengambilan data yang lebih fleksibel dan tidak memerlukan skema yang ketat seperti pada basis data relasional.
3. Penggunaan NoSQL: Basis data NoSQL (Not Only SQL) juga telah mendapatkan popularitas dan penggunaan yang luas dalam desain basis data obyek. NoSQL merupakan pendekatan yang menghindari model relasional tradisional dan menggunakan skema fleksibel untuk mengelola data yang tidak terstruktur dan semi-terstruktur. NoSQL memberikan fleksibilitas dalam skema dan memungkinkan skalabilitas horizontal yang baik untuk mengatasi volume data yang besar.
4. Desain Berorientasi Layanan (Service-Oriented Design): Dalam lingkungan yang didorong oleh arsitektur layanan, desain basis data obyek telah berkembang untuk mendukung integrasi dan interoperabilitas yang lebih baik antara layanan dan aplikasi

yang terkait. Basis data obyek digunakan sebagai penyimpanan untuk objek-objek yang digunakan dalam layanan, dan mereka dapat digunakan untuk berbagi data dengan cara yang terstandarisasi dan terstruktur.

5. Pemodelan Big Data: Desain basis data obyek juga telah berevolusi untuk memenuhi kebutuhan pemodelan dan analisis Big Data. Ketersediaan jumlah data yang besar, termasuk data semi-terstruktur seperti sensor data, data jejaring sosial, dan data log, telah mendorong pengembangan basis data berorientasi objek yang mampu menangani volume, kecepatan, dan keragaman data tersebut.
6. Penerapan Konsep Pemrograman Berorientasi Objek : Salah satu tren dalam desain basis data obyek adalah penerapan konsep pemrograman berorientasi objek ke dalam desain dan pemodelan basis data. Konsep seperti pewarisan, enkapsulasi, polimorfisme, dan asosiasi antara objek diterapkan untuk memodelkan hubungan dan perilaku didalam basis data.

Perkembangan terkini ini dalam desain basis data obyek bertujuan untuk meningkatkan fleksibilitas, skalabilitas, dan efisiensi dalam mengelola data yang semakin kompleks. Dengan adopsi tren ini, desain basis data obyek terus berkembang dan beradaptasi dengan kebutuhan sistem yang semakin maju dan tuntutan data yang semakin besar.

DAFTAR PUSTAKA

Booch, G. *et al.* 2008. 'Object-oriented analysis and design with applications', *ACM SIGSOFT software engineering notes*, 33(5), p. 29.

Brown, P.G. 2000. *Object-Relational Database Development: A Plumber's Guide with Cdrom*. Prentice Hall PTR.

Bruegge, B. and H Dutoit, A. 2010. *Object-Oriented Software Engineering Using UML, Patterns, and Java™ Third Edition*. by Pearson Education, Inc.,.

Dittrich, K.R., Dayal, U. and Buchmann, A.P. 2012. *On object-oriented database systems*. Springer Science & Business Media.

Elmasri, R. *et al.* 2000. *Fundamentals of Database Systems*. Springer.

Harrington, J.L. 2000. *Object-oriented database design clearly explained*. Morgan Kaufmann.

Hassan, A.A. 2005. 'Database Security and Auditing: Protecting data integrity and accessibility', *Paperback-2005* [Preprint].

BAB 7
DESAIN ANTARMUKA PENGGUNA

Oleh Rita Komalasari

7.1 Pendahuluan

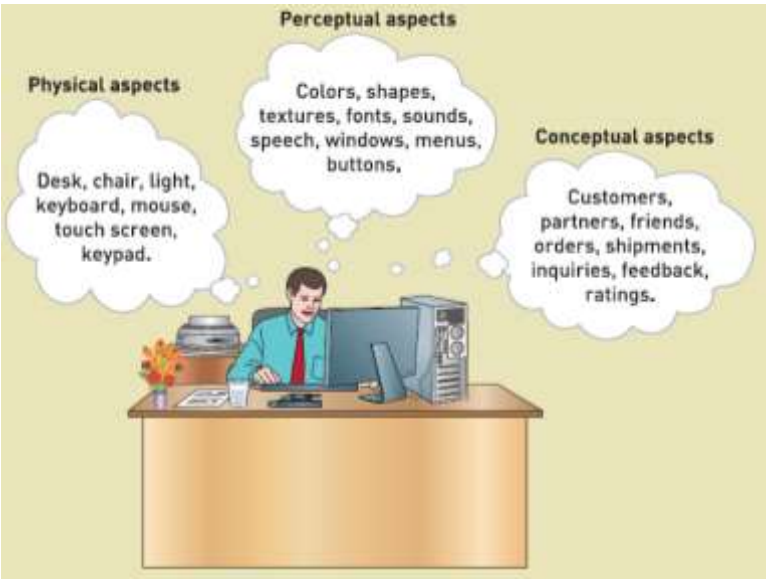
Antarmuka pengguna adalah bagian dari sistem yang berinteraksi dengan pengguna. Dari sudut pandang pengguna, antarmuka pengguna adalah sistem. Ini mencakup tampilan layar yang menyediakan navigasi melalui sistem, layar dan formulir yang menangkap data, dan laporan yang dihasilkan sistem (baik di atas kertas, di layar, atau melalui media lain). Desain antarmuka adalah proses mendefinisikan bagaimana sebuah sistem akan berinteraksi dengan entitas eksternal (misalnya, pelanggan, pemasok, dan sistem lain). Pada bab ini, kita fokus pada desain antarmuka pengguna, namun penting juga untuk diingat bahwa terkadang ada antarmuka sistem, yang bertukar informasi dengan sistem lain. Antarmuka sistem biasanya dirancang sebagai bagian dari upaya integrasi sistem. Secara umum, antarmuka sistem didefinisikan sebagai bagian dari arsitektur fisik dan lapisan manajemen data. Lapisan interaksi manusia-komputer mendefinisikan cara pengguna berinteraksi dengan sistem dan sifat input dan output yang diterima dan dihasilkan oleh sistem. yang diterima dan dihasilkan oleh sistem (Dennis, Wixom, and Tegarden 2015).

Hingga saat ini, seluruh proses pengembangan telah difokuskan untuk menyelesaikan masalah lapisan domain dan penyimpanannya pada lapisan manajemen data dengan benar. Namun, dari sudut pandang pengguna pengguna, antarmuka pengguna pada lapisan interaksi manusia-komputer adalah sistem. Pengguna tidak tidak terlalu peduli tentang bagaimana objek domain masalah disimpan. Namun, mereka sangat peduli dengan

bagaimana mereka dapat menggunakan sistem untuk mendukung aktivitas mereka. Berdasarkan pendekatan desain berlapis kami, antarmuka pengguna dari lapisan interaksi manusia-komputer tidak bergantung pada lapisan manajemen data. Tetapi itu tergantung pada domain masalah dan arsitektur fisik lapisan. Tergantung pada jenis perangkat tempat lapisan interaksi manusia-komputer diterapkan akan menentukan peluang dan kendala mengenai fitur antarmuka pengguna yang dapat disertakan. Sebagai contoh, menerapkan lapisan interaksi manusia-komputer pada ponsel cerdas dan komputer desktop akan menyebabkan dua antarmuka pengguna yang berbeda untuk dirancang.

Antarmuka pengguna menyediakan sarana komunikasi antara sistem dan pengguna dan ini disebut sebagai interaksi manusia-komputer (HCI). Sangatlah penting untuk memastikan kegunaan - merancang antarmuka pengguna (UI) yang efektif yang memungkinkan pengguna merasa nyaman dan untuk memastikan mereka seproduktif mungkin saat menggunakan sistem (Gould 2016).

Antarmuka pengguna adalah segala sesuatu yang berhubungan dengan pengguna akhir saat menggunakan sistem-secara fisik, persepsi, dan konseptual (lihat Gambar 7.1).



Gambar 7.1. Desain yang berpusat pada pengguna
(Sumber : (Satzinger and Jackson 2012))

Dari perspektif pengguna, antarmuka pengguna adalah keseluruhan sistem. Program, skrip, basis data, dan perangkat keras di balik antarmuka tidak relevan.

Teknik desain yang mewujudkan pandangan antarmuka pengguna ini secara kolektif disebut desain yang berpusat pada pengguna, yang menekankan tiga prinsip penting:

1. Fokus sejak awal pada pengguna dan pekerjaan mereka.
2. Mengevaluasi desain untuk memastikan kegunaannya.
3. Gunakan pengembangan berulang.

7.2 Panduan umum

Namun, ada beberapa panduan umum yang berguna untuk diingat ketika merancang antarmuka pengguna apa pun; beberapa di antaranya tercantum di bawah ini (Britton and Doake 2005).

1. Sistem harus konsisten, misalnya, tata letak layar harus mengikuti pola umum yang sama, tugas-tugas yang sama harus dilakukan dengan cara yang sama, dan pesan dari sistem ke pengguna harus selalu dalam format yang sama.
2. Tugas pengguna yang membosankan dan rentan terhadap kesalahan harus diminimalkan; ini berarti bahwa sistem harus menyediakan jalan pintas untuk pengguna yang berpengalaman dan, jika memungkinkan, data harus data harus dimasukkan dengan memilih dari menu dan bukan dengan mengetik.
3. Layar harus bebas dari kekacauan, berisi semua informasi yang relevan dan tidak lebih. Layar yang penuh sesak akan melelahkan untuk dilihat dan menjengkelkan untuk digunakan. Kombinasi warna yang dramatis dan sinyal yang berkedip harus dihindari dan penyorotan hanya boleh digunakan untuk memilih yang penting penting, bukan untuk menambah dekorasi pada layar. Layar harus harus mandiri dan cukup jelas. Seharusnya tidak perlu terus-menerus merujuk ke bantuan online atau panduan pengguna untuk mengetahui apa yang harus dilakukan selanjutnya atau cara keluar dari layar.
4. Dialog dengan sistem harus mudah diikuti dan harus jelas bagi pengguna bagaimana menavigasi jalan mereka melalui berbagai layar. Hirarki menu sederhana yang mudah dipahami dan digunakan jauh lebih efektif daripada sistem navigasi canggih yang mungkin mengesankan tetapi sulit bagi pengguna untuk menemukan jalan mereka.
5. Umpan balik harus informatif, tetapi tidak mengganggu, sehingga pengguna selalu tahu apa yang sedang terjadi, tetapi tidak dibebani dengan rincian tentang apa yang dilakukan sistem. Umpan balik dapat berupa berbagai bentuk dalam antarmuka pengguna, termasuk:
   Umpan balik yang dapat didengar, seperti bunyi klik saat tombol ditekan dan bunyi bip saat tombol di layar ditekan.

Umpan balik yang dapat dilihat, seperti perubahan ikon yang ditunjukkan pada Gambar 8.2(a) dan 8.2(b), atau pengukur kemajuan yang ditampilkan selama pengunduhan file berukuran besar.

(a)

(b)

Gambar 7.2. Visibilitas dan keterjangkauan dalam kontrol pemutar media
(Sumber : (Satzinger and Jackson 2012))

6. Umpan balik memberi pengguna rasa konfirmasi dan perasaan bahwa bahwa sistem responsif dan berfungsi dengan benar. Kurangnya umpan balik membuat pengguna pengguna bertanya-tanya apakah perintah atau input dikenali atau apakah sistem tidak berfungsi. Ketika pemrosesan selanjutnya tertunda lebih dari satu atau dua detik, pengguna mungkin berulang kali menekan kontrol atau memasukkan kembali informasi, mengakibatkan kesalahan pemrosesan dan frustrasi pengguna.
7. Bahasa yang digunakan dalam instruksi dan pesan kepada pengguna harus jelas, ringkas, dan bebas dari jargon. Pesan kesalahan harus menyatakan penyebab kesalahan, jika memungkinkan menunjukkan bagaimana kesalahan dapat diperbaiki, dan menunjukkan di mana bantuan lebih lanjut dapat ditemukan.
8. Harus ada dukungan pengguna yang memadai dengan instruksi yang jelas, pesan, dan bantuan online yang komprehensif.

Antarmuka pengguna adalah area di mana pengguna benar-benar yang paling tahu. Mereka adalah orang-orang yang akan

menggunakan sistem dan sangat penting bagi mereka untuk terlibat secara dekat dengan desain antarmuka. Antarmuka yang menurut pengguna menarik dan mudah digunakan serta sesuai dengan pandangan mereka tentang tugas yang sedang dikerjakan tidak hanya berarti pengguna yang lebih bahagia, namun juga lebih sedikit kesalahan dan kinerja yang lebih efisien dan efektif.

7.3 Proses Desain Antarmuka Pengguna

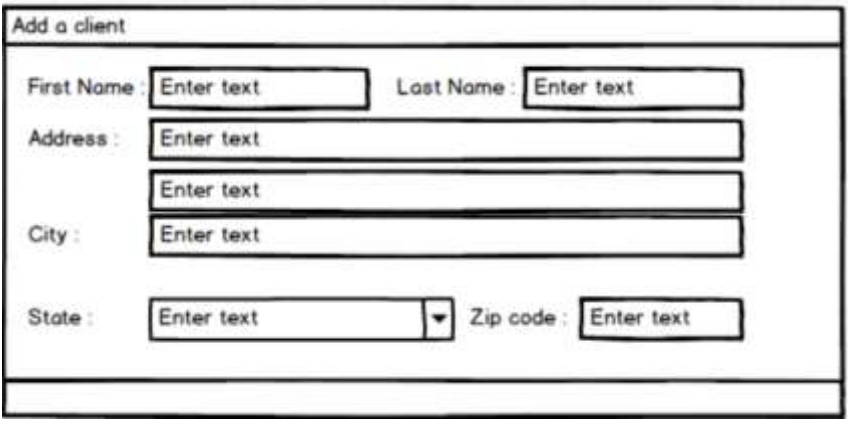
Desain antarmuka pengguna adalah proses yang digerakkan oleh kasus penggunaan, bertahap, dan berulang. Analis sering kali bergerak bolak-balik di antara bagian-bagian yang berbeda (navigasi, input, dan output) dari antarmuka pengguna, daripada melanjutkan secara berurutan dari satu bagian ke bagian lain. Mengingat bahwa proses desain didorong oleh kasus penggunaan, analis memulai proses desain antarmuka pengguna dengan memeriksa use case dan sequence diagram terkait yang dikembangkan dalam analisis. Analis kemudian biasanya bekerja sama dengan pengguna untuk mengembangkan skenario penggunaan yang menggambarkan pola tindakan yang umum dilakukan pengguna sehingga antarmuka memungkinkan pengguna untuk melakukan skenario ini dengan cepat dan lancar. Dalam beberapa kasus, persyaratan tambahan ditemukan. Tergantung pada pentingnya persyaratan yang baru ditemukan, hal ini dapat menyebabkan lapisan domain masalah dimodifikasi, yang pada gilirannya dapat menyebabkan lapisan manajemen data dimodifikasi. Namun, sering kali, persyaratan baru ini dapat ditunda hingga iterasi sistem berikutnya. Dengan pendekatan agile, desain antarmuka pengguna dan pemodelan kebutuhan saling terkait sehingga kebutuhan baru dapat ditemukan secara teratur. Akibatnya tergantung pada stabilitas pemodelan domain masalah, desain antarmuka pengguna dapat terjadi bersamaan dengan pemodelan fungsional. Meskipun pemodelan fungsional dan perilaku dikaitkan dengan alur kerja analisis dan desain antarmuka

pengguna dikaitkan dengan alur kerja desain, tingkat aktivitas yang terkait dengan kedua alur kerja tersebut saling tumpang tindih.

7.3.1 Pembuatan Prototipe Desain Antarmuka

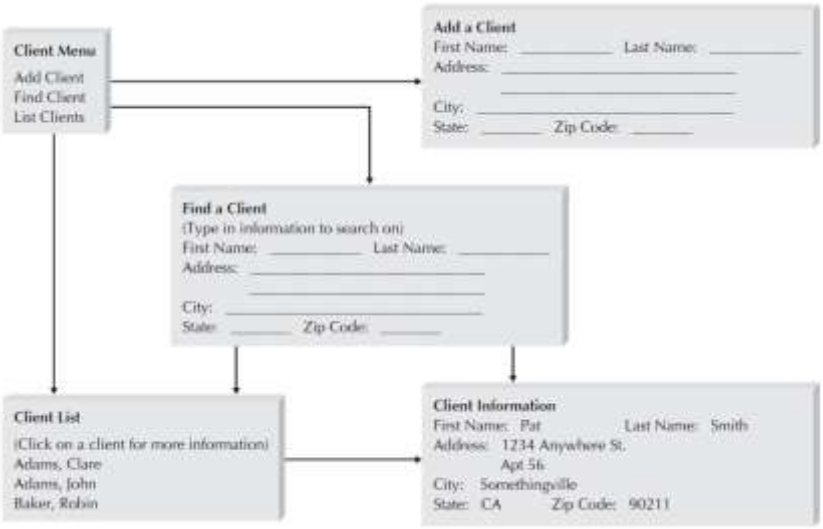
Prototipe desain antarmuka adalah tiruan atau simulasi layar komputer, formulir, atau laporan. Prototipe disiapkan untuk setiap antarmuka dalam sistem untuk menunjukkan kepada pengguna dan pemrogram bagaimana sistem akan bekerja. Di "masa lalu", prototipe desain antarmuka biasanya ditentukan pada formulir kertas yang menunjukkan apa yang akan ditampilkan pada setiap bagian layar. Formulir kertas masih digunakan saat ini, tetapi semakin banyak prototipe desain antarmuka dibuat dengan menggunakan alat bantu komputer, bukan kertas. Empat pendekatan yang paling umum untuk pembuatan prototipe desain antarmuka adalah papan cerita, diagram tata letak jendela, dan bahasa prototipe.

1. Diagram Tata Letak Windows hanyalah sebuah gambar yang menyerupai antarmuka pengguna yang sebenarnya yang secara bertahap akan diterima oleh pengguna. Biasanya, diagram ini dibuat dengan menggunakan alat bantu seperti Microsoft Visio. Dengan menggunakan alat bantu jenis ini, perancang dapat dengan cepat menarik dan melepaskan komponen antarmuka pengguna ke kanvas untuk meletakkan desain antarmuka pengguna. Sebagai contoh, Gambar 7.3 menggambarkan diagram tata letak Windows yang sederhana. Meskipun tidak ada kemampuan yang dapat dieksekusi yang terkait dengan diagram tata letak Windows, diagram ini memungkinkan pengguna untuk dengan cepat merasakan tampilan antarmuka pengguna yang akan disampaikan.



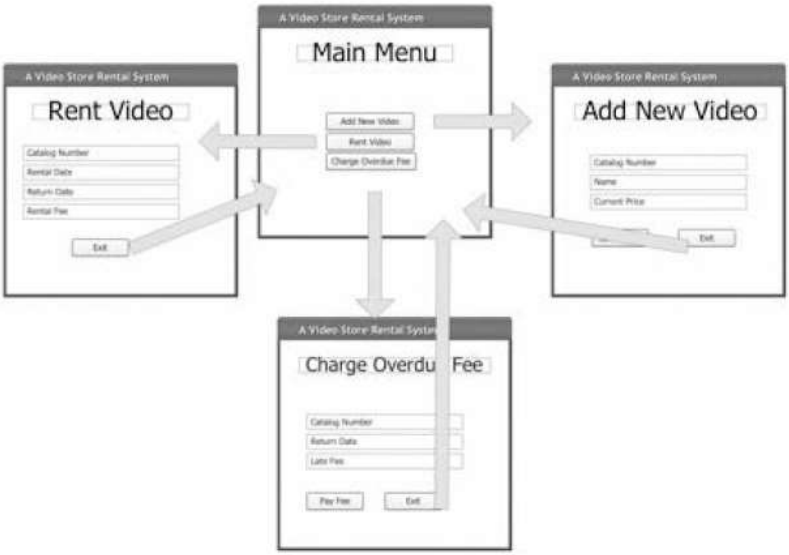
Gambar 7.3. Contoh Jendela Diagram Tata Letak
(Sumber : (Dennis et al. 2015))

2. Storyboard, pada dasarnya, prototipe desain antarmuka adalah storyboard berbasis kertas. Storyboard menunjukkan gambar yang digambar tangan tentang seperti apa tampilan layar dan bagaimana mereka mengalir dari satu layar ke layar lainnya, dengan cara yang sama, storyboard untuk kartun menunjukkan bagaimana aksi akan mengalir dari satu adegan ke adegan berikutnya (lihat Gambar 7.4).



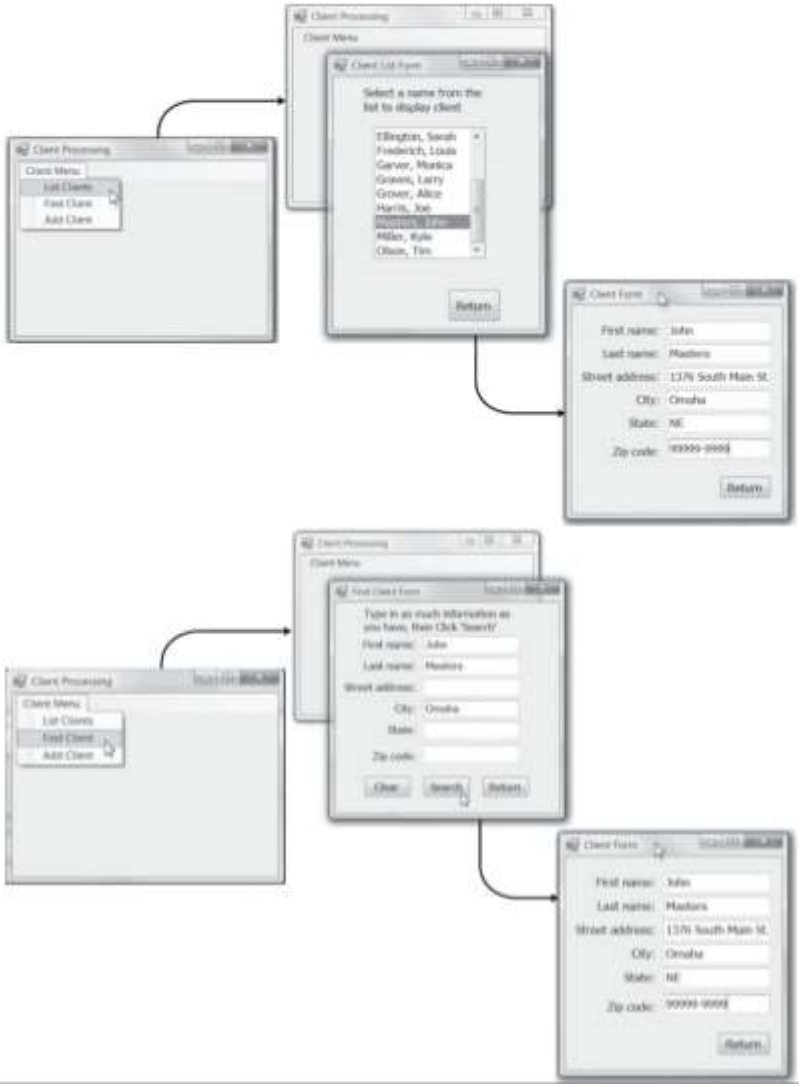
Gambar 7.4. Contoh Storyboard
(Sumber : (Dennis et al. 2015))

Storyboard adalah teknik yang paling sederhana karena yang dibutuhkan hanyalah kertas (biasanya berupa flip chart) dan pena-dan seseorang yang memiliki kemampuan artistik. Storyboard juga menggabungkan informasi navigasi dari diagram navigasi Windows dan pada tingkat tertentu informasi tata letak dari diagram tata letak Windows. Namun, dengan alat grafis saat ini, perancang dapat bekerja secara efektif dengan sekumpulan pengguna untuk merancang tampilan dan nuansa sistem yang sedang berkembang tanpa harus benar-benar mengimplementasikan apa pun, dengan menggabungkan diagram tata letak windows dengan diagram navigasi windows ke dalam satu jenis diagram storyboard yang lebih baik (lihat Gambar 7.5).



Gambar 7.5. Contoh Gabungan Navigasi Windows dan Tata Letak Diagram. (Sumber : (Dennis et al. 2015))

3. Prototipe Antarmuka Pengguna, dengan lingkungan pemrograman saat ini, cukup mudah untuk mengembangkan prototipe yang dapat dieksekusi (lihat Gambar 7.8) dari antarmuka pengguna yang memungkinkan pengguna untuk dapat berinteraksi dengan antarmuka pengguna dengan mengklik tombol dan memasukkan data pura-pura ke dalam formulir (tetapi karena tidak ada sistem di belakang halaman, data tidak pernah diproses). Bagian-bagian yang berbeda dari antarmuka pengguna dihubungkan bersama sehingga ketika pengguna mengklik tombol, bagian yang diminta dari sistem akan muncul. Prototipe yang dapat dieksekusi ini membutuhkan waktu lebih lama untuk dikembangkan daripada diagram navigasi Windows, diagram tata letak Windows, dan storyboard, tetapi memiliki keuntungan yang berbeda dalam menunjukkan dengan tepat seperti apa tampilan layarnya.



Gambar 7.6. Contoh Pengguna Prototipe Antarmuka
(Sumber : (Dennis et al. 2015))

7.3 Desain Navigasi

Komponen navigasi pada antarmuka memungkinkan pengguna memasukkan perintah untuk menavigasi melalui sistem dan melakukan tindakan untuk memasukkan dan meninjau informasi yang ada di dalamnya. Komponen navigasi juga menyajikan pesan kepada pengguna tentang keberhasilan atau kegagalan tindakannya. Tujuan dari sistem navigasi adalah membuat sistem sesederhana mungkin untuk digunakan.

1. Mencegah Kesalahan, prinsip pertama dalam mendesain kontrol navigasi adalah mencegah pengguna melakukan kesalahan. Kesalahan akan menghabiskan waktu dan menyebabkan frustrasi. Lebih buruk lagi, serangkaian kesalahan dapat menyebabkan pengguna meninggalkan sistem. Kesalahan dapat dikurangi dengan memberi label pada perintah dan tindakan dengan tepat dan dengan membatasi pilihan.
2. Sederhanakan pemulihan dari kesalahan apa pun yang dilakukan perancang sistem, pengguna akan akan membuat kesalahan. Sistem harus membuatnya semudah mungkin untuk memperbaiki kesalahan ini.
3. Gunakan urutan tata bahasa yang konsisten. Salah satu keputusan yang paling mendasar adalah urutan tata bahasa. Sebagian besar perintah mengharuskan pengguna untuk menentukan objek (misalnya, file, catatan, kata), dan tindakan yang akan dilakukan pada objek tersebut (misalnya, salin, hapus). Urutan tata bahasa harus konsisten di seluruh sistem, baik pada elemen data elemen data dan pada tingkat menu secara keseluruhan.

7.4 Desain Input

Tujuan dari mekanisme input adalah untuk secara sederhana dan mudah menangkap informasi yang akurat untuk sistem. Prinsip-prinsip dasar untuk desain input mencerminkan

sifat input (apakah *batch* atau *online*) dan cara-cara untuk menyederhanakan pengumpulannya.

Setiap item data yang harus dimasukkan dihubungkan ke sebuah field pada formulir di mana nilainya diketikkan. Setiap field juga memiliki label field, yaitu teks di samping, di atas, atau di bawah field yang memberi tahu pengguna jenis informasi apa yang ada di dalam field tersebut. Sering kali label field mirip dengan nama elemen data, tetapi tidak harus sama persis. Dalam beberapa kasus, sebuah field menampilkan template di atas kotak entri untuk menunjukkan kepada pengguna bagaimana data harus diketik.

Semua data yang dimasukkan ke dalam sistem harus divalidasi untuk memastikan keakuratannya. Validasi input (juga disebut pemeriksaan edit) dapat dilakukan dalam berbagai bentuk. Idealnya, sistem komputer tidak boleh menerima data yang gagal dalam pemeriksaan validasi untuk mencegah informasi yang tidak valid masuk ke dalam sistem. Namun, hal ini bisa jadi sangat sulit, dan data yang tidak valid sering kali lolos dari operator pemasukan data dan pengguna yang memberikan informasi. Tergantung pada sistem untuk mengidentifikasi data yang tidak valid data yang tidak valid dan melakukan perubahan atau memberitahukan kepada seseorang yang dapat menyelesaikan masalah informasi tersebut. Ada enam jenis pemeriksaan validasi yang berbeda: pemeriksaan kelengkapan, pemeriksaan format, pemeriksaan rentang pemeriksaan rentang, pemeriksaan digit, pemeriksaan konsistensi, dan pemeriksaan basis data.

7.5 Desain Output

Output adalah apa yang dihasilkan oleh sistem, baik di layar, di atas kertas, atau di media lain, seperti Web. Keluaran mungkin merupakan bagian yang paling terlihat dari sistem apa pun karena alasan utama untuk menggunakan sistem informasi adalah untuk mengakses informasi yang dihasilkannya. Tujuan dari mekanisme

keluaran adalah untuk menyajikan informasi kepada pengguna sehingga mereka dapat memahaminya secara akurat dengan sedikit usaha. Prinsip-prinsip dasar untuk desain keluaran mencerminkan bagaimana output digunakan dan cara-cara untuk mempermudah pengguna memahaminya.

Prinsip pertama dalam mendesain laporan adalah memahami bagaimana laporan tersebut digunakan. Laporan dapat digunakan untuk berbagai tujuan. Dalam beberapa kasus-tetapi tidak terlalu sering-laporan dibaca dari awal sampai akhir karena semua informasi dibutuhkan. Pada kebanyakan kasus, laporan digunakan untuk mengidentifikasi item tertentu atau digunakan sebagai referensi untuk menemukan informasi, sehingga urutan item yang diurutkan pada laporan atau dikelompokkan dalam kategori sangat penting.

Frekuensi laporan juga dapat memainkan peran penting dalam desain dan distribusinya. Laporan waktu nyata memberikan data yang akurat hingga detik atau menit saat laporan tersebut dibuat (misalnya detik atau menit saat laporan tersebut dibuat (misalnya, harga pasar saham). Laporan batch adalah laporan yang melaporkan informasi historis yang mungkin sudah berumur beberapa bulan, hari, atau jam, dan sering kali memberikan informasi tambahan.

Mengelola Beban Informasi. Kebanyakan manajer mendapatkan terlalu banyak informasi, bukan terlalu sedikit (yaitu, beban informasi yang harus ditangani oleh manajer terlalu besar). Tujuan dari laporan yang dirancang dengan baik adalah untuk menyediakan semua informasi yang dibutuhkan untuk mendukung tugas yang dirancang.

Meminimalkan Bias. Tidak ada analis yang merancang laporan yang bias. Masalah dengan bias adalah bias bisa sangat halus; analis dapat memperkenalkannya secara tidak sengaja. Bias dapat diperkenalkan dengan cara daftar data diurutkan karena

entri yang muncul pertama kali dalam daftar dapat menerima lebih banyak perhatian daripada yang muncul belakangan dalam daftar.

Ada banyak jenis laporan yang berbeda, seperti laporan detail, laporan ringkasan, laporan pengecualian laporan pengecualian, dokumen perputaran, dan grafik. Mengklasifikasikan laporan adalah menantang karena banyak laporan memiliki karakteristik dari beberapa jenis yang berbeda. Sebagai contoh, beberapa laporan detail juga menghasilkan total ringkasan, menjadikannya laporan ringkasan.

DAFTAR PUSTAKA

Britton, Carol, and Jill Doake. 2005. *A Student Guide to Object-Oriented Development*. Burlington: Elsevier Butterworth-Heinemann.

Dennis, Alan, Barbara Haley Wixom, and David Tegarden. 2015. *System Analysis & Design: An Object-Oriented Approach with UML*. John Wiley & Sons, Inc.

Gould, Howard. 2016. *Systems Analysis and Design*. bookboon.com.

Satzinger, John W., and Robert B. Jackson. 2012. *Systems Analysis and Design in a Changing World*. Course Technology.

BAB 8
DESAIN KEAMANAN SISTEM INFORMASI

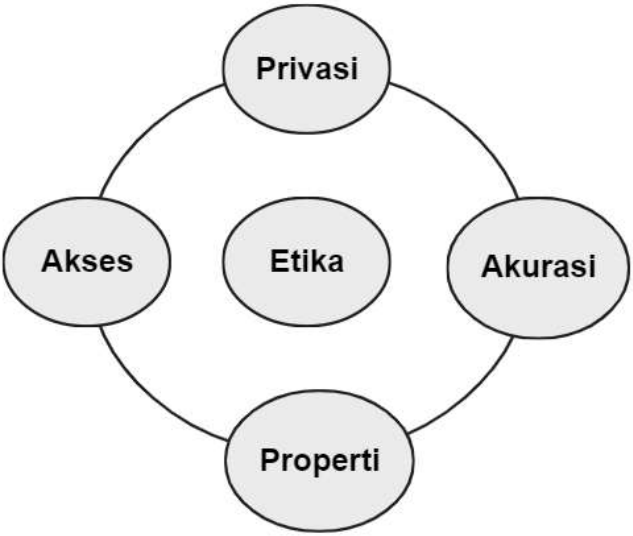
Oleh Nurul Aini

8.1 Kemananan Sistem Informasi

Berbicara tentang sistem informasi, maka tidak jauh juga membahas tentang sisi keamanan dari sistem informasi tersebut. Instansi maupun perusahaan di era digitalisasi dengan adanya sistem informasi yang diimplementasikan menjadi syarat mutlak dalam proses kegiatan. Tidak dapat dihindari keamanan sistem informasi menjadi suatu poin tertentu dalam tata kelola manajemen teknologi(Hananto, Rerung and Indonesia, 2023). Tata kelola teknologi informasi didefenisikan sebagai upaya menjamin pengelolaan informasi dalam mendukung stategis bisnis dalam perusahaan(Yustanti et al., 2020). Banyak sistem informasi yang dibangun maupun dalam pengembangan tidak ditemui keamanan sistem informasi yang kuat untuk suatu fungsi tertentu dalam sistem. Padahal kebutuhan keamanan sistem dapat diintegrasikan dalam siklus hidup sebuah sistem melalui proses dalam tahap pengembangan, sehingga permasalahan yang mungkin akan muncul seperti ganguan virus ataupun serangan dalam jaringan dapat diatasi. Terdapat teknologi yang dibutuhkan sampai saat ini yang dapat melindungi sistem yang biasanya disebut sebagai teknologi reaktif seperti firewall , aplikasi anti virus dan sistem deteksi (Zen Munawar, 2023) . Meskipun aplikasi tersebut masih umum dan diandalkan namun masih tidak cukup bekerja dengan baik.

Sistem informasi memberikan kemudahan dalam mengakses dan menyediakan data secara cepat dan akurat baik bagi perusahaan , instansi pemerintah, perguruan tinggi , maunpun pribadi. Keamanan informasi cara untuk melindungi data dan informasi agar tidak disalahgunakan oleh pihak yang tidak bertanggung jawab dan hal ini merupakan suatu yang sangat disadari. Keamanan informasi yang dimaksud yaitu perlindungan data, ketersediaan data yang benar dan asli, dan integritas sumber daya informasi.

Di era informasi ini terdapat etika dalam bidang teknologi informasi, dimana pelaku dunia teknologi informasi mengetahui cakupan etika yang terdapat didalam. Etika dalam sistem informasi ini pertama kali diangkat oleh Richard Mason ditahun 1986 dan Zwass ditahun 1998 dengan memberi singkatan PAPA (Alter, 2001).



Gambar 8.8. Etika Sistem Informasi
((ALBANNA, 2016)

Privasi yang dimaksud adalah privasi fisik dan privasi informasi, privasi fisik hak pemilik mencegah tindakan orang lain terhadap ruang, waktu dan properti, sedangkan untuk privasi informasi yaitu hak personal untuk mengatur jadwal informasi yang akan dikomunikasikan dengan orang lain. Akurasi dalam sistem informasi adalah data dan informasi dalam mendukung pengambilan keputusan harus benar adanya. Properti dapat berupa perlindungan hak cipta yang dikenal dengan sebutan HAKI yang dapat didaftakan melalui hak cipta copyright, paten ataupun trade secret. Dan yang terakhir akses yaitu menyediakan kemudahan bagi pengguna resmi dari sistem informasi.

8.2 Tiga Pilar Keamanan Sistem Informasi

Menggunakan tiga pilar keamanan dalam program keamanan sistem informasi dapat menghasilkan sistem yang masuk akal dan hemat biaya. Tiga pilar yang dimaksud adalah kebijakan dalam manajemen resiko, titik kontak keamanan dan pengetahuan. Tiga pilar tersebut dapat digambarkan sebagai berikut.

Gambar 8.9. Tiga Pilar Keamanan
(sumber: Zen Munawar, 2023)

1. Pilar 1 : Resiko Manajemen
   Tidak adanya pembahasan keamanan sistem tanpa menimbang manajemen resiko yang diterapkan, dan hal ini pun terjadi untuk keamanan sistem informasi. Untuk menciptakan susunan manajemen risk (resiko) yang terkait, perlunya pemahaman perbedaan analisis resiko pada desain arsitektur keamanan sistem dan ide palacakan dan pengurangan resiko dalam aktivitas siklus hidup dari sistem informasi, karena resiko keamanan ini dapat ditemui dalam siklus hidup(SDLC) sistem informasi. Manajemen resiko yang berhasil , sebagaimanapun tidak jauh dari adanya alat pendukung untuk mencapai bisnis seperti halnya ketika mengumpulkan data untuk suatu penilaian, yang diperhatikan dalam proses ini adalah rentan terhadap serangan, ancaman, dampak setelah tindakan dan probalitias. Sehingga perlu menerapkan definisi resiko pada desain sistem informasi untuk menghasilkan upaya yang baik dalam mengurangi resiko .
2. Pilar 2 : Software Security TouchPoint
   Keamanan perangkat lunak merupakan problem sistem informasi yang biasa ditemui dimana tim pengembang harus memperhitungkan mekanisme tambahan untuk keamanan seperti kontrol akses user dan menambah rancangan untuk keamanan. Penerapan ini diperlukan kesepakatan bersama bagi pembuat sistem dan pemesan sistem, bahwa keamanan sistem informasi bukan suatu program keamanan tersendiri, dan hal ini harus disetujui bersama ketika sistem informasi itu dipesan. Pemesan atau pembeli sistem kadang hanya cenderung fokus pada fungsional dari sistem saja.
   Touchpoint atau titik sentuhan keamanan sistem informasi diperlukan kesadaran akan hal tersebut, karena masalah keamanan kemungkinan juga muncul pada standar sistem dibandingkan dengan beberapa fitur kemanan. Ini adalah

alasan penting mengapa keamanan perangkat lunak harus menjadi bagian dari pendekatan SDLC. Seperti halnya tidak dapat menguji kualitas sistem, tidak berkemampuan memasang fitur keamanan di desain sistem dan mengharapkan semuanya menjadi aman.

3. Pilar 3 : Pengetahuan (Knowledge)
   Pilar ke 3 melibatkan berbagai pengetahuan keamanan yang dapat digunakan memberikan dasar yang kuat untuk praktik kemananan perangkat lunak. Seiring perkembangan latihan dan kegiatan yang ditetapkan, manajemen pengetahuan dan pelatihan sebagai peran utama dalam menyusun dan mendistibusi ketetapan yang muncul secara lebih ringkas. Informasi dan pengetahuan suatu hal yang berbeda dan perlu untuk dipahami perbedaaannya. Pengetahuan diartikan informasi dalam konteks yang digunakan untuk bekerja menggunakan sejumlah proses dan prosedures sedankang informasi merupakan sekumpulan data yang dikelola atau diproses menjadi suatu yang bermanfaat bagi penerimanya.
   Pengetahuan keamanan sistem informasi disusun dalam tujuh daftar pengetahuan yaitu prinsip, aturan, eksplotasi, pedoman, resiko historis , pola serangan dan kerentanan(Zen Munawar, 2023). Adapun yang mengelompokkan dalam 3 pengetahuan yaitu preskriptif, pengetahuan sejarah dan diagnostik. Sejumlah pengetahuan ini jika dapat diterapkan pada seluruh tahan SDLC dengan penggunaan *software security touchpoint*.

8.3 Strategi Desain Keamanan

Kemajuan teknologi jaringan berkembang dari jaringan lokak ke jaringan global tau tanpa batas dengan disebut teknologi internet. Sistem informasi yang berbasiskan jaringan internet sudah berkembang sedemikan besarnya, hal ini juga memicu jumlah peretas (*hacker)* semakin naik. Penanggung jawab pengelola sistem informasi tidak menerus akan merasa aman

dengan keadaan sistem masih status terkendali, dan mereka juga tidak menunggu apakah jaringan akan diserang oleh peretas hingga rusak.

Saat sistem informasi berbasis jaringan dan terhubungi ke jaringan lokal atau internet, maka disaat itulah seorang pengelola harus siaga dan menyusun strategi dengan metode untuk meningkatkan keamanan pada jaringan sistem yang dibangun. Dengan adanya keamanan jaringan dengan teknologi tinggi , pengelola sistem dapat menambah rasa aman ketika berkerja dengan jaringan komputer yang ada. Tujuan mendesain keamanan sistem informasi dapat tercapai jika disusun dalam sejumlah tahapan proses pengamanan dalam merekayasan keamanan teknologi informasi. Salah satu tujuannya adalah memberikan gambaran tentang tingkatatan atau level keamanan sistem yang akan menjadi penilaian dalam mendesain, mengembang dan mengimplementasikan sistem informasi. Prinsip dasar akan digunakan oleh user ketika sistem akan dikembangkan atau dievaluasi kebutuhan fugnsional dalam beroperasi, tenaga arsitek dan Enggineers yang mendesain dan mengimplementasikan atau memodifikasi sistem informasi, teknisi IT yang hadir disetiap siklus hidup sistem, dan manajer dan petugas keamanan sistem yang meyakinkan penerapan keamanan yang sesuai pada semua tahapan SDLC.

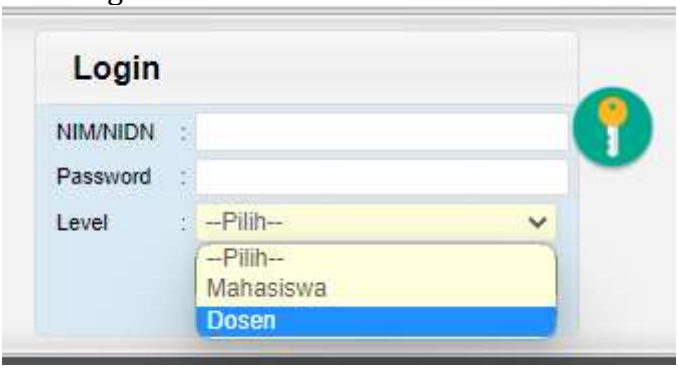
Prinsip dasar yang digunakan dalam keamanan sistem informasi yang dimaksud adalah merumuskan strategi pengamanan dan memaksimalkan sumber daya untuk dapat mengamankan sistem pada beberapa fitur sistem dengan tepat sehingga lebih efisien. Jika tanpa strategi yang disusun dengan serius, proses pengoperasian teknik keamanan dapat terbuang sumber daya secara percuma. Ini sesuai dengan suatu badan nasional yang disebut National Institute of Standards and Technology (NIST) dalam bukunya dengan judul ” *Generally*

Accepted Principles and Practices for Securing Information Technology Systems".

8.3.1 Prinsip Desain Keamanan Sistem

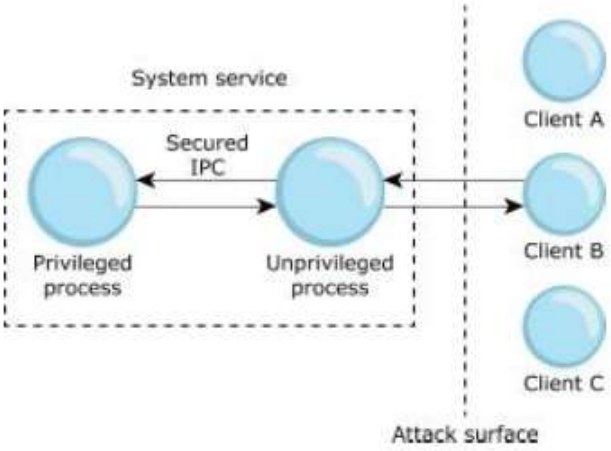
Terdapat 12 prinsip desain keamanan sistem informasi .

1. Least Privilage.
   Pemberian hak akses kepada para pengguna sistem infomasi . Prinsip ini dinyatakan bahwa setiap user yang berinteraksi pada sistem harus memiliki level dan dimulai dari level bawah. Pemberian hak akses ini sebagai dasar strategi keamanan, setiap pengguna yang menggunakan sistem apakah sebagai user biasa, administrator, software atau sistem itu sendiri , hal ini berguna dalam menunjang fungsi kerja masing-masing pengguna, hak akses yang dibatasi sehingga pengguna resmi lainnya tidak dapat melakukan tindakan yang merugikan suatu organisasi sendiri. Walau kadang pembatasan hak akses ini dapat menimbulkan ketidaknyamanan pengguna resmi (user) yang disebabkan kadangkala ada kebutuhan yang tidak dapat diprediksi. Pemberian hak akses dapat dilihat dari tampilan program sebagai berikut.



Gambar 8.10. Contoh pembatasan hak akses pada sistem informasi
(sumber : siaka.undipa.ac.id)

2. Mekanisme Sederhana
   Prinsip dasaranya pada poin ini adalah metode yang sederhana lebih muda dikuasai dan dikendalikan. Berikut beberapa contoh mekanisme keamanan sistem yang sederhana.
   Pembatasan Login : admin memberlakukan jadwal setiap user ke sistem pada waktu dan hari yang telah ditentukan.
   Pembatasan jumlah usaha login : setiap user yang berusahan untuk login ke sistem diberi kesempatan sebanyak 3 kali , jika gagal ke 3 kalinya maka sistem akan dikunci dan diinformasikan ke administrator.
3. Pemeriksaan Otorisasi
   Prinsip ini menyatakan setiap user yang berinteraksi ke sistem harus diperiksa kedalam kontrol akses informasi untuk otorisasi yang tepat. Pemeriksaan otorisasi dilakukan saat user mulai log-on ke sistem dan menggunakan menu sesuai hak aksesnya, kemudian sistem akan memeriksa otorisasi user setiap ada perubahan pada database.
4. *Open Design*
   Prinsip ini memiliki mekanisme keamanan dengan menginformasikan dengan baik dari sistem sehingga dapat menimbulkan umpan balik , untuk dapat digunakan dalam perbaikan sistem keamanan.
5. *Separation Of Priviledge*
   Prinsip ini menyatakan sistem dirancang dengan komponen-komponen dasar dengan beberapa progress dimana user dengan hak akses tertentu yang berbeda, beberapa bagian sistem akan memerlukan persetujuan dari user dengan priviledge yang tinggi untuk menyelesaikan aktivitas tertentu atau akses ke file pada sistem. Penguatan priviledge ini dapat muncul dari manajemen proses dari perusahaan, sehingga dapat pula diterapkan pada sistem informasi. Pada gambar dibawah memperlihatkan aktivitas istimewa untuk semua user pada permukaan serangan yang dapat terjadi.



Gambar 8.11. Pemisahaan Priviledge

Konsep pemisahan kewenangan berhubungan dengan kepercayaan manajemen yang mencakup pemisahan tugas dan batasan keperluan yang harus diketahui.

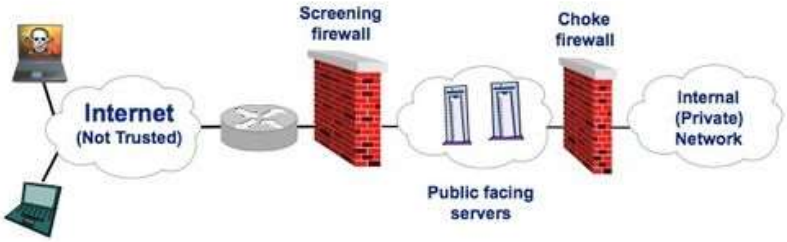
6. *Psychological Acceptability*

   Prinsip ini adalah mekanisme keamanan sistem diaman mudah digunakan oleh user biasa, yaitu dengan melakukan pendekakatan penerimaan psikologi. Hal yang dapat dilakukan adalah petugas keamanan sistem dapat mensurvei mengenai prilaku user terhadapa sistem, kegiatan ini dapat dilakukan secara berkala.

7. *Defense in Depth*

   Prinsip ini memberikan syarat bahwa pengguna dapat berbagi perangkat keamanan untuk saling memberi backup. Misal menggunakan multiple screening router, mirrorring hardisk, atau tersedia perangkat CDRW setiap kali proses backup maka hasil backup ada pada masing-masing pengguna, sehinga jika salah satu pengguna diretas maka yang satu masih memiliki data backup yang masih layak digunakan.

8. Choke point.
   Choke point didefenisikan sebagai jalur sempit , dimana jalur tersebut menghubungkan 2 jalur besar. Gagasan *choke point* telah diadopsi di dunia *cybersecurity*. Dalam kasus terakhir, *choke point* adalah tempat beberapa jalur serangan melintas tepat sebelum mencapai aset kritis. Untuk memahami choke point dalam konteks *cybersecurity*, pertama-tama kita harus memahami apa itu entitas dan aset kritis. Entitas adalah titik akhir, file, folder, atau sumber daya cloud apa pun di lingkungan Anda yang dapat digunakan penyerang untuk maju dalam jalur serangan menuju aset penting. Aset kritis adalah entitas dalam jaringan yang memiliki nilai potensial bagi penyerang atau organisasi (yang membuatnya menarik bagi penyerang). Choke point adalah entitas kunci tempat beberapa jalur serangan bertemu sebelum mencapai aset penting. Semakin banyak jumlah jalur serangan di mana entitas berperan, semakin banyak entitas yang menjadi choke point.



Gambar 8.12. Tampilan dari Choke Point
((Piscitello, 2011)

9. *Fail-Safe Stance*.
   Prinsip ini menyatakan jika salah satu perangkat keamanan menemui kendala, maka sistem akan secara default mengatur perangkat tersebut ke mode setting yang paling aman. Sebagai contoh seperti paket filtering pada firewall modem router ADSL

jika rusak maka akan mencegah masukkan paket-paket yang keluar dan masuk.

10. *Universal participation*
    Semua pengguna sistem dalam satu organisasi atau perusahaan sebaiknya terlibat dalam proses keamanan sistem. Dilakukan kegiatan rutin seperti tiga bulan sekali mengadakan pelatihan kesadaran dalam mengamankan perangkat keamanan sistem ataupun komputer. Dalam kegiatan tersebut dilakukan proses evaluasi untuk mengukur tingkat kinerja proses keamanan sistem.
11. *Diversity of Defense*
    Prinsip ini mnyatakan tidak mempercayakan hanya satu sistem keamanan atau sistem pertahanan, namuan mengenal dan mengetahui teknik dari sistem kemanan lainnya, sehingga jika terjadi pada suatu keamanan yng telah dipasang , maka tidak diperlukan untuk mencari atau baru mempelajari sistem keamanan lainnya.
12. *Simplicity*
    Prinsip ini menyatakan suatu sistem juga tidak boleh terlalu kompleks, karena jika ada kesalahan akan sulit untuk diselidiki kerusakan ada dimana. Dapat dilakukan jika sistem ada perubahan data pada database , maka sistem dapat merekam data user, waktu, aksi apa yang dilakukan dan lokasi komputer. Sehingga jika ada kesalahan proses data akan muncul trigger yang memberitahukan bahwa sistem dapat menolak aksi perubahn data yang dilakukan.

8.3.2 Model-Model Keamanan

Model keamanan adalah apa yang boleh dan tidak boleh secara teknik. Didefenisikan sebagai representasi secara simbolik dari kebijakan suatu organisasi atau perusahaan nyang menggunakan sistem informasi , yang juga harus dilaksanakan oleh

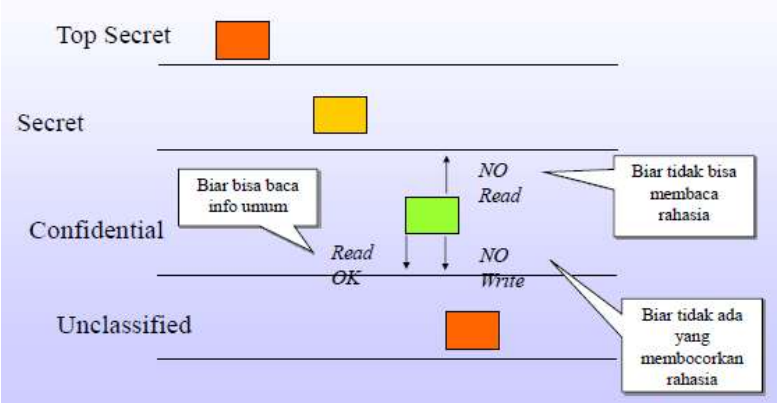
sistem komputer. Tujuannya adalah untuk menformalkan kebijakan kemanan sistem dalam organisasi.

1. Access Control Matrix Models.
   Mengatur atau menjadwalkan user akan aksi yang dilakukan pada sistem. Seperti contoh pengaturan pada tabel dibawah.

Tabel 8.2. Pengaturan transaksi user ke sistem

Objek Subjek	File : Income	File : Salaries	Process: Deductions	Print Server
Jim	R	R/W	X	W
Josh	R/W	R	-	W
Cek Prog.	R	R	X	-
Tax Prog.	R/W	R/W	X	W

2. Bell-LaPadula Model
   Model ini dibuat pada tahun 1970-an untuk pengamanan kerahasiaan informasi dibidang militer. Memiliki tinggakat status kerahasiaan seperti *Top Secret, Secret, Sensitive but unclassified, Unclassified*. Fokus pada *Confidential*.

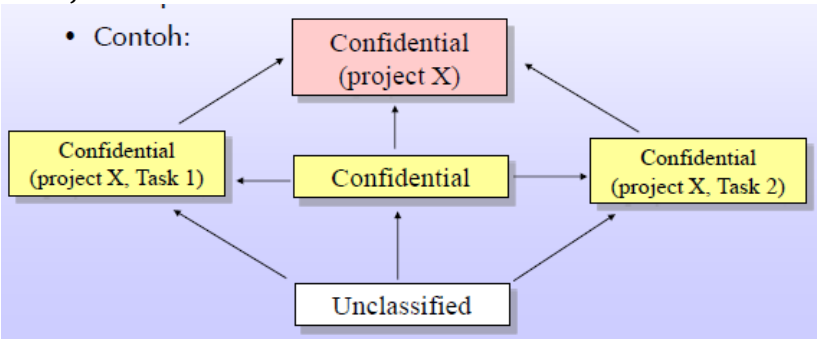


Gambar 8.13. Susunan Tingkat Kerahasian Informasi.

3. Biba.
   Model ini dibuat tahun 1977-an dan menjamin integritas data. Misalnya, seorang akuntan ingin agar data keuangan disistem akuran, dan tidak menginginkan data menjadi corrupt

dikarenakan operator sitem tidak bekerja seperti semestinya (misalnya: pembulatan, salah ketik, penambahan digit, dsb.)

4. Clark-Wilson Model.
   User tidak dapat secara langsung mengakses data langsung, tapi harus lewat suatu program. Sebuah file tidak dapat dibuka diprogram lainnya harus menggunakan program khusus. Dapat menerapkan prinsip *Separation Of Duties*, contohnya user yang melakukan input data (*write*) dan yang membuat laporan (*read*) adalah user yang berbeda.
5. Information Flow Model.
   Model ini memberikan setiap objek class keamanan dan diberikan nilai , informasi hanya dapat terkirim sesuai aturan kebijakan yang telah ditetapkan sebelumnya oleh pihak manajemen kemanan.



Gambar 8.14. Contoh Model Information Flow.

DAFTAR PUSTAKA

ALBANNA. 2016. *Etika Sistem Informasi*, *student. telkomuniversity.ac.id*. Available at: https://albanna.student.telkomuniversity.ac.id/2016/10/26/etika-sistem-informasi/.

Alter, S. 2001. 'Which Life Cycle--Work System, Information System, or Software?', *Communications of the Association for Information Systems*, 7(1), p. 17.

Hananto, A. L., Rerung, R. R. and Indonesia, M. S. 2023. *Tata Kelola Keamanan Sistem Informasi*. Media Sains Indonesia. Available at: https://books.google.co.id/books?id=cUamEAAAQBAJ.

Piscitello, D. 2011. *Choke and screen firewall: the modern day concentric castle - The Security Skeptic*. Available at: https://www.securityskeptic.com/2011/01/choke-and-screen-firewall-the-modern-day-concentric-castle.html (Accessed: 13 May 2023).

Yustanti, W. *et al.* 2020. *Keamanan Sistem Informasi*. Zifatama Jawara. Available at: https://books.google.co.id/books?id=w0IBEAAAQBAJ.

Zen Munawar, D. 2023. *KEAMANAN SISTEM INFORMASI: Prinsip Dasar, Teori, dan Rekayasa Penerapan Konsep*. Kaizen Media Publishing. Available at: https://books.google.co.id/books?id=_o-nEAAAQBAJ.

ArifinZainal. 2012. *Evaluasi Pembelajara.* Jakarta: Direktorat Jenderal Pendidikan Islam Kementrian Agama RI.

Arikunto Suharsimi. 2001. *Dasar-Dasar Evaluasi Pendidikan*. Jakarta: PT Bumi Aksara.

BAB 9
IMPLEMENTASI SISTEM

Oleh Santo Fernandi Wijaya

9.1 Pendahuluan

Tahapan penting setelah proses analisa dan desain dari suatu sistem informasi adalah tahapan untuk mengimplementasi sistem informasi. Proses mengimplementasi sistem merupakan bagian krusial dimana pada tahapan implementasi inilah untuk membuktikan hasil pemetaan dari kebutuhan para pengguna terutama kebutuhan pemangku kepentingan dan para pimpinan manajemen perusahaan telah sesuai dalam memperoleh berbagai informasi yang dibutuhkan rangka pengambilan keputusan untuk mengembangkan bisnis ke depan. Tahapan implementasi ini juga sebagai klarifikasi atau verifikasi dari para pengguna untuk melakukan pengujian terhadap rancangan sistem informasi yang telah dibangun. Pada tahapan mengimplementasi sistem informasi ini biasanya dikatakan dengan istilah User Acceptance Test. Pada bagian inilah seharusnya para pengguna, pemangku kepentingan dan para pimpinan manajemen perusahaan terlibat secara aktif dalam melakukan pengujian. Biasanya akan diusulan perbaikan sesuai kebutuhan apabila hasil pengujian ini belum sesuai kebutuhan terkait hal yang esensi dalam proses bisnis perusahaan.

9.2 User Acceptance Test

User Acceptance Test (UAT) merupakan salah satu cara jitu untuk membuktikan kesesuaian hasil pelatihan dengan membandingkan secara actual, sehingga bila masih terjadi penyimpangan dalam pemgujian penggunaan aplikasi sistem informasi oleh para pengguna, maka dapat dilakukan tindaklanjut

sesuai kebutuhan. Untuk itu, perlu keterlibatan aktif dari para pengguna dan pemahaman pemikiran yang menyeluruh dari para pengguna terhadap aplikasi sistem informasi bahwa aplikasi hanya sebagai alat yang berperan penting adalah para pengguna yang dapat menyelesaikan pekerjaan dengan menggunakan aplikasi sistem informasi. Untuk itu, peran para pengguna yang memiliki integritas tinggi dan dapat dipercaya sebagai role model (agen perubahan), berfokus menyelesaikan pekerjaaan dengan tuntas, memiliki kreatifitas dalam menghadapi tantangan dengan memberikan solusi. Semua pengolahan transaksi terkait bisnis perusahaan dilakukan dengan penggunaan aplikasi sistem informasi. Disamping itu, para pengguna yang dapat bekerjasama dalam tim dengan sikap positif untuk saling memberikan pemecahan masalah, bukan hanya menambah kerumitan permasalahan yang menyebabkan hasil kerja tidak produktif dan dapat menimbulkan kecurigaan atau prasangka buruk. Perilaku para pengguna seperti inilah yang dibutuhkan dan menjadi salah satu faktor kritis dalam menentukan keberhasilan dalam melakukan UAT terhadap aplikasi sistem informasi yang siap diimplementasikan lebih lanjut.

9.3 Menilai kesiapan perusahaan

Menilai kesiapan perusahaan dalam mengimplementasi sistem baru merupakan hal esensi sebagai pembuktian agar hasil implementasi dapat mencapai keberhasilan. Penilaian kesiapan perusahaan ini dapat ditinjau dari perseptif Teknologi, Proses, Organisasi, dan orang. Pengukuran kesiapan perusahaan dari perspektif teknologi, menilai kesiapan teknologi merupakan hal yang esensi untuk mencapai keberhasilan implementasi sistem informasi. Dengan mengidentifikasi potensi masalah teknologi, maka perusahaan dapat memastikan bahwa sistem informasi yang terintegrasi, dapat bekerja secara optimal, andal, dan aman, serta dapat meningkatkan kinerja perusahaan [Kirmizi et al., 2020].

Adapaun keuntungan melakukan menilai kesiapan dari perspektif teknologi adalah sebagai berikut:

1. Menilai kesiapan teknologi dapat mengidentifikasi potensi masalah kompatibilitas dan memastikan bahwa sistem informasi dapat diintegrasikan ke dalam infrastruktur yang ada.
2. Menilai kesiapan teknologi untuk memastikan infrastruktur teknologi yang dimiliki perusahaan dapat mampu mendukung penggunaan sistem informasi dengan memperhatikan kapasitas penampungan data yang relatif besar.
3. Menilai kesiapan teknologi dapat mengidentifikasi potensi masalah keandalan dan memastikan bahwa infrastruktur teknologi yang dimiliki perusahaan telah dirancang untuk mendukung tingkat keandalan yang diperlukan.
4. Menilai kesiapan teknologi dapat mengidentifikasi potensi risiko keamanan dan memastikan bahwa infrastruktur teknologi yang dimiliki perusahaan telah dirancang untuk mendukung tingkat keamanan yang diperlukan.
5. Menilai kesiapan teknologi dapat mengidentifikasi potensi masalah kinerja dan memastikan bahwa infrastruktur teknologi yang dimiliki perusahaan telah dirancang untuk mendukung tingkat kinerja yang diperlukan.

Pengukuran penilaian kesiapan perusahaan dari perspektif organisasi memiliki hal esensi untuk mencapai keberhasilan implementasi sistem informasi. Kesiapan perusahaan dari perspektif organisasi mengacu pada kesiapan perusahaan secara keseluruhan untuk mengimplementasi sistem informasi. Menilai kesiapan perusahaan dapat mengidentifikasi hambatan potensial untuk adopsi dan mengatasi, perusahaan dapat memastikan bahwa sistem informasi diadopsi dan digunakan secara efektif, mencapai manfaat dan meningkatkan kinerja perusahaan secara menyeluruh. Adapun beberapa hal yang menjadi keuntungan

dalam menilai kesiapan perusahaan dari perspektif organisasi adalah sebagai berikut:

1. Menilai kesiapan perusahaan dapat mengidentifikasi potensi resistensi terhadap tingkat perubahan, kesenjangan keterampilan atau pengetahuan, dan faktor yang memengaruhi kemampuan perusahaan untuk mengelola perubahan secara efektif dan efisien.
2. Menilai kesiapan perusahaan untuk memastikan bahwa sistem informasi telah mencapai tingkat keselarasan dan searah strategis bisnis perusahaan untuk jangka menengah dan jangka panjang.
3. Menilai kesiapan perusahaan dapat mengidentifikasi permasalahan terkait ketersediaan sumber daya dan memastikan bahwa perusahaan dapat mengalokasikan sumber daya yang diperlukan guna mendukung proses mengimplementasi sistem informasi.
4. Menilai kesiapan perusahaan dapat mengidentifikasi hambatan budaya potensial untuk adopsi dan memastikan bahwa sistem informasi sejalan dengan budaya perusahaan.
5. Menilai kesiapan perusahaan dapat membantu memastikan bahwa kepemimpinan yang memiliki komitmen pada pelaksanaan mengimplementasi sistem informasi dan memberikan dukungan yang diperlukan untuk mendukung untuk mencapai keberhasilan dalam mengimplementasi sistem informasi dapat golive secara tepat waktu.

Menilai kesiapan perusahaan dari perspektif proses memiliki esensi untuk mencapai keberhasilan dalam mengimplementasi sistem informasi. Kesiapan perusahaan dari perspektif proses mengacu pada kesiapan perbaikan proses bisnis, prosedur, dan alur kerja untuk mendukung mengimplementasi sistem informasi. Juga dapat mengidentifikasi masalah potensial dengan proses bisnis dan alur kerja yang dapat memastikan bahwa

sistem informasi terintegrasi dengan lancar, mendukung proses bisnis yang efisien dan efektif, dan mencapai manfaat yang diharapkan. Beberapa hal yang menjadi keuntungan menilai kesiapan perusahaan dari perspektif proses adalah sebagai berikut:

1. Menilai kesiapan proses dapat mengidentifikasi potensi ketidaksesuaian dan memastikan bahwa sistem terintegrasi dengan lancar ke dalam proses yang ada.
2. Menilai kesiapan proses dapat mengidentifikasi area untuk perbaikan dan memastikan bahwa mengimplementasi sistem informasi dapat mendukung proses yang efisien dan efektif.
3. Menilai kesiapan proses dapat mengidentifikasi peluang potensial untuk otomatisasi dan memastikan bahwa sistem informasi dirancang untuk mendukung perubahan proses bisnis.
4. Menilai kesiapan proses untuk memastikan bahwa sistem informasi mematuhi kebijakan, peraturan, dan standar perusahaan yang telah ditentukan.
5. Menilai kesiapan proses dapat mengidentifikasi potensi risiko dan memitigasi sebelum menjadi masalah yang lebih kompleks.

Menilai kesiapan perusahaan dari perspektif orang merupakan hal esensi untuk mencapai keberhasilan implementasi sistem informasi untuk memastikan bahwa sistem informasi yang diadopsi, digunakan secara efektif, dan mencapai manfaat yang diinginkan. Kesiapan perusahaan dari perspektif orang mengacu pada kesiapan individu, tim, dan departemen untuk mengadopsi dan mengimplementasi sistem informasi. Juga dapat mengidentifikasi hambatan potensial untuk adopsi dan mengatasi, perusahaan dapat memastikan bahwa sistem informasi yang diadopsi dapat digunakan secara efektif. efisien, guna mencapai manfaat yang diharapkan dan meningkatkan kinerja perusahaan.

Beberapa hal yang menjadi keuntungan menilai kesiapan perusahaan dari perspektif orang adalah sebagai berikut:

1. Menilai kesiapan orang dapat mengidentifikasi potensi penolakan terhadap perubahan, kesenjangan dalam keterampilan atau pengetahuan, dan faktor lain yang dapat memengaruhi adopsi.
2. Menilai kesiapan orang dapat mengidentifikasi masalah potensial yang dapat memengaruhi produktivitas, seperti kurangnya pelatihan atau dukungan yang tidak memadai.
3. 3.. Menilai kesiapan orang dapat mengidentifikasi potensi masalah yang dapat memengaruhi kepuasan pengguna, seperti kesulitan dalam menggunakan sistem atau kurangnya akses ke fitur utama.
4. Menilai kesiapan orang dapat membantu mengidentifikasi potensi risiko dan memitigasinya sebelum menjadi masalah.
5. Menilai kesiapan orang dapat membantu memastikan bahwa sistem digunakan secara efektif, memaksimalkan potensi laba atas investasi.

9.4 Kegagalan mengimplementasi sistem informasi

Dalam proses mengimplementasi sistem informasi dapat mengalami kegagalan dan keberhasilan. Kegagalan dapat disebaban karena berbagai alasan. Perusahaan dapat meningkatkan kemungkinan keberhasilan dengan menangani faktor-faktor ini melalui perencanaan yang efektif, keterlibatan pemangku kepentingan, pengujian, pelatihan, komunikasi, manajemen proyek, dan dokumentasi [Wijaya et al., 2017].

Faktor-faktor yang menjadi penyebab kegagalan dalam mengmplementasi sistem informasi adalah sebagai berikut:

1. Perencanaan yang tidak memadai dapat menyebabkan definisi ruang lingkup proyek yang buruk, garis waktu yang tidak realistis, anggaran yang tidak mencukupi, dan peran serta

tanggung jawab yang tidak jelas. Hal ini dapat mengakibatkan penundaan, pembengkakan biaya, dan ketidaksesuaian dengan harapan pemangku kepentingan.

2. Keterlibatan pemangku kepentingan penting untuk keberhasilan mengimplementasi sistem informasi. Tanpa keterlibatan aktif pemangku kepentingan, maka akan ada resistensi terhadap perubahan, karena kurangdukungan, dan mengalamai kesulitan dalam mencapai keberhasilan yang diinginkan.
3. Pengujian yang tidak memadai dapat menyebabkan kesalahan yang tidak terdeteksi, sehingga dapat mengakibatkan kegagalan sistem, kehilangan data, dan pelanggaran keamanan.
4. Pelatihan yang tidak efektif, dapat mengakibatkan kebingungan, kesalahan, dan frustrasi pengguna, yang menyebabkan kurangnya adopsi dan kurang dimanfaatkannya sistem.
5. Komunikasi yang tidak memadai dapat menyebabkan kesalahpahaman, ekspektasi yang tidak selaras, dan kurangnya kepercayaan antar pemangku kepentingan, yang mengakibatkan penundaan dan pengerjaan ulang.
6. Manajemen proyek yang buruk dapat mengakibatkan alokasi sumber daya yang tidak memadai, tenggang waktu yang terlewatkan, dan ruang lingkup yang tidak jelas, dapat menyebabkan kelebihan biaya, penundaan, dan penurunan kualitas.
7. Dokumentasi yang buruk dapat mengakibatkan kebingungan, ketidakjelasan, dan kesulitan dalam memelihara sistem dari waktu ke waktu, yang menyebabkan inefisiensi dan kesalahan.

9.5 Tujuan mengimplementasi sistem informasi

Mengimplementasi sistem informasi adalah proses perencanaan, pengujian, pelatihan, dokumentasi, komunikasi, dan manajemen proyek. Untuk itu, perusahaan dapat memastikan keberhasilan mengimplementasi sistem informasi yang memenuhi kebutuhan pemangku kepentingan. Mengmplementasi sistem informais mengacu pada proses menginstal, mengkonfigurasi, menguji, dan menyebarkan sistem baru atau memutakhirkan sistem yang ada. Beberapa faktor penting dari yang perlu mendapat perhatian dalam mengimplementasi sistem informasi adalah:

1. Sebelum tahapan implementasi, maka perlu untuk merencanakan dan menentukan tujuan dan sasaran mengimplementasi sistem informasi dengan melibatkan secara aktif dan menentukan ruang lingkup proyek, jadwal, dan anggaran, serta mengidentifikasi pemangku kepentingan.
2. Pengujian secara menyeluruh merupakan hal penting untuk memastikan bahwa hasil mengimplementasi sistem informasi telah dapat bekerja seperti yang diharapkan dan memenuhi kebutuhan pemangku kepentingan. Pengujian harus dilakukan pada setiap tahap implementasi, mulai dari konfigurasi awal hingga penerapan akhir dengan melibatkan para pengguna secara aktif.
3. Pelatihaan secara intens untuk para pengguna dan tim TI mengenai cara mengimplementasi sistem informasi untuk memastikan bahwa para pengguna dan tim TI dapat menggunakan secara efektif dan efisien. Pelatihan harus disesuaikan dengan kebutuhan para pengguna dan tim TI yang berbeda, dan harus berkelanjutan untuk memastikan bahwa para pengguna dan tim TI telah mengetahui secara menyeluruh.
4. Dokumentasi merupakan hal penting untuk memastikan bahwa hasil mengimplementasi sistem informasi telah dipahami dengan baik dan dapat dipertahankan dari waktu ke waktu.

Dokumentasi harus mencakup persyaratan sistem, spesifikasi desain, manual para pengguna, dan informasi relevan lainnya.

5. Komunikasi yang efektif untuk memastikan bahwa pemangku kepentingan telah mendapat informasi selama proses mengimplementasi sistem informasi, termasuk permasalahan yang ditemukan, dan meminta umpan balik dan mengatasi permasalahan tersebut.
6. Manajemen proyek yang efektif untuk memastikan bahwa mengimplementasi sistem informasi dapat selesai tepat waktu, sesuai anggaran, dan untuk mencapai kepuasan pemangku kepentingan, termasuk dalam mengelola ketersediaan sumber daya, jadwal, dan anggaran, serta mengidentifikasi dan mengurangi risiko.

9.6 Strategi dalam mengimplementasi sistem baru

Mengimplementasi sistem informasi memiliki risiko terhadap kegagalan. Untuk itu, perlu penentuan dan pemilihan metodologi implementasi sistem yang terdefinisi dengan baik, sehingga dapat mengurangi risiko kegagalan dalam melakukan implementasi sistem baru [Luvai et al., 2014]. Pemilihan metodologi implementasi sistem dapat mengatasi berbagai permasalahan terkait dalam mengimplementasi sistem informasi. Setiap metode implementasi memiliki keuntungan dan kerugian masing-masing tergantung pada kebutuhan bisnis dan sumber daya yang tersedia untuk mengimplementasi sistem baru. Metode implementasi sistem secara umum adalah sebagai berikut:

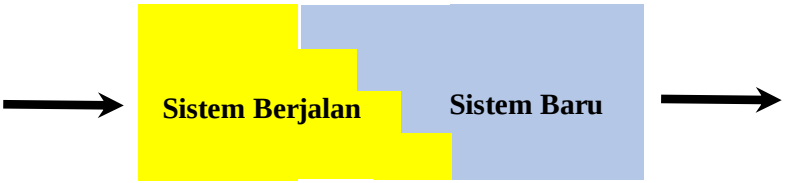
1. **Implementasi Langsung**. Metode big bang merupakan metode implementasi sistem informasi yang dilakukan secara langsung pada seluruh unit bisnis dalam waktu yang relatif singkat. Metode ini memiliki keuntungan dalam waktu implementasi yang cepat, tetapi juga memiliki risiko tinggi kegagalan dan memerlukan anggaran yang besar. Dengan kata lain bahwa

metode ini sederhana namun memiliki resiko dan berbahaya dengan matikan sistem lama dan menghidupkan yang baru

Gambar 9.1. Metode Implementasi Langsung

2. **Implementasi Bertahap (Modular)**. Metode Bertahap ini merupakan metode implementasi sistem yang dilakukan secara bertahap untuk setiap unit bisnis tertentu. Metode ini memiliki keuntungan dalam fleksibilitas, kontrol risiko, dan pengelolaan biaya, tetapi membutuhkan waktu yang lebih lama untuk menyelesaikan implementasi sistem informasi secara keseluruhan. Pada saat go live, para pengguna cukup hanya melakukan input data 1x pada sistem baru, dan pekerjaan dengan sistem berjalan sudah tidak dikerjakan atau sistem berjalan langsung dimatikan pada saat sistem baru berjalan digunakan. Pada saat proyek implementasi sistem informasi berjalan, para pengguna harus melakukan pengujian sistem baru lebih serius, karena saat pengujian/testing merupakan kesempatan bagi para pengguna untuk melakukan uji coba secara lengkap dari bagian master data, transaksi, dan pelaporan. Pada saat go live, maka para pengguna hanya menggunakan sistem baru sebagai sarana yang mendukung operasional. Untuk itu, harus dipastikan terkait perpindahan data pada saat tanggal Cut Off yang ditentukan, saldo dari sistem berjalan harus dipindahkan ke sistem baru, mulai dari stok,

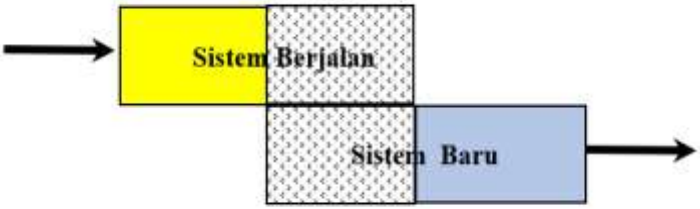
bank/kas, hutang, piutang dan aset. Perlu diantisipasi mitigasi yang harus dilakukan, bila terjadi kekacauan pada sistem baru, agar tidak terjadi hambatan operasional menggunakan sistem baru. Dengan kata lain, metode bertahap satu modul pada satu waktu, menggabungkan yang terbaik dari modul langsung dan modular sambil meminimalkan risiko. Metode Implementasi Bertahap dapat dilihat pada gambar sebagai berikut:



Gambar 9.2. Metode Implementasi Bertahap

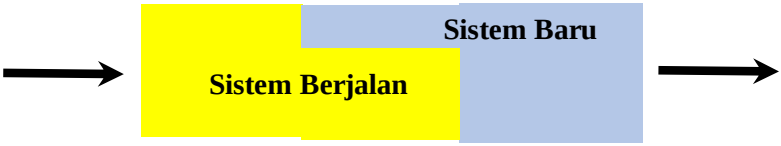
3. **Implementasi Paralel**. Metode parallel merupakan metode mengimplementasi sistem informasi yang dilakukan secara parallel/bersamaan dengan sistem berjalan. Hal ini dapat menjamin bahwa sistem baru dapat berfungsi dengan benar sesuai kebutuhan para pengguna. Metode ini memiliki rasaa man secara bisnis, karena bila terjadi sesuatu yang mengganggu operasional, maka masih ada sistem berjalan sehingga bisnis tidak akan terganggu. Biasanya para pengguna akan menggunakan sistem lama dalam menyelesaikan pekerjaan, baru menjalankan sistem baru. Hal ini tentu melelahkan bagi para pengguna, karena para pengguna harus melakukan penginputan data dua kali. Disamping itu, perlu melakukan perbandingan terhadap laporan dari sistem berjalan dan sistem baru untuk menguji apakah sistem baru sudah sama dengan sistem lama. Hal ini perlu pemeriksaan dua laporan dan memerlukan waktu kerja lembur yang juga mengalami kesibukan operasional. Dengan demikian, metode ini

membutuhkan biaya yang relative lebih tinggi, karena harus mempertahankan sistem berjalan untuk waktu tertentu dan membangun kepercayaan terhadap sistem baru yang stabil dan layak digunakan para pengguna. Dengan kata lain, metode paralel ini mahal tapi aman dimana perusahaan menjalankan kedua sistem sampai semua para pengguna puas, baru memmatikan sistem lama. Metode Implementasi Paralel dapat dilihat pada gambar sebagai berikut:



Gambar 9.3. Metode Implementasi Paralel

4. **Implementation Pilot**. Metode pilot merupakan metode implementasi sistem informasi yang dilakukan dengan cara melakukan implementasi terlebih dahulu pada unit bisnis terpilih sebagai pilot project. Metode pilot ini dapat membantu mengurangi risiko dan menyelesaikan permasalahan terkait kebutuhan para pengguna pada tahap awal sebelum dilakukan implementasi pada unit bisnis atau departemen lainnya. Dengan kata lain, metode pilot ini membiarkan hanya beberapa unit bisnis terpilih yang menggunakan sistem baru sampai para pengguna merasa puas, baru mengimplementasi sistem baru untuk semua unit bisnis, hal ini juga dapat membuktikan para pengguna terbaik yang dapat dipilih untuk melakukan uji coba dalam mengimplementasi pilot ini. Metode Implementasi Paralel dapat dilihat pada gambar sebagai berikut:



Gambar 9.4. Metode Implementasi Pilot

Memperhatikan kelebihan dan kekurangan dari metode implementasi Go Live sistem baru, dikembalikan kepada Manajemen Proyek yang menangani dengan melihat kepada kesiapan dan kondisi para pengguna, kompleksitas bisnis proses, dan kompleksitas migrasi data. Apapun metode implementasi sistem informasi yang akan diputuskan, perlu diperhatikan bahwa proses implementasi sistem informasi menjadi lebih efektif bila ada perubahan proses bisnis berjalan dengan mengikuti proses bisnis dari sistem informasi yang akan digunakan untuk meningkatkan cara kerja yang lebih efektif dan efisien, dan akhirnya dapat meningkatkan kinerja dan keunggulan bersaing.

9.7 Mengelola Kompleksitas mengimplementasi Sistem SAP

Proses Bisnis yang Jelas dalam mengimplementasi sistem informasi merupakan salah satu faktor kritis. Pemetaan bisnis proses harus dilakukan oleh tim fungsional tiap unit bisnis yang telah memiliki pengalaman mengenai cara kerja unit bisnis. Tahapan proses pemetaan bisnis proses dimulai dengan melakukan diskusi terkait cara kerja dengan unit bisnis terkait yang diterjemahkan dalam alur kerja dari sistem informasi yang diimplementasikan. Untuk itu, pada bagian ini penulis memberikan penjelasan terkait risiko yang mengakibatkan kompleksitas mengimplementasi suatu sistem SAP sebagai studi kasus dalam melakukan pemilihan metodologi sistem SAP. Salah satu

metodologi dalam mengimplementasi sistem SAP adalah Metode ASAP yaitu metode pengukuran SAP yang diusulkan adalah bagian dari manajemen kompleksitas umum. Dasar mengimplementasi sistem SAP adalah analisis proses bisnis dan kebutuhan bisnis unit dalam memanfaatkan sistem informasi untuk pengambilan keputusan [Holub and Bruckner,. 2016]. Prosedur implementasi standar dimulai dengan analisis proses bisnis dan persyaratan sistem informasi yang berdasarkan metodologi ASAP, konsep target dibuat, yang berfungsi sebagai tugas untuk menyiapkan sistem. metodologi ASAP ini menjelaskan struktur perusahaan dan mencakup struktur organisasi dasar dari masing-masing modul SAP, jenis dokumen, dan jenis data master. Pemilihan perangkat lunak standar. Keputusan penting yang memengaruhi kompleksitas TIK di suatu perusahaan adalah pilihan menggunakan berbagai platform teknologi yang difokuskan pada berbagai cabang industri, atau mencakup kelompok proses khusus dalam suatu perusahaan. Dalam konfigurasi dasar, sebelum menerapkan pengaturan pengguna, SAP ERP hanya berisi model struktur organisasi dan tipe dokumen. Bergantung pada deskripsi struktur dan proses bisnis, kustomisasi harus dilakukan, bersama dengan pemrograman tambahan yang potensial untuk kebutuhan pelanggan tertentu. Atribut utama dari sistem SAP adalah kustomisasi pengguna yang dapat mengadaptasi sistem ke berbagai proses bisnis dan varian, mulai dari pembelian, manufaktur, penjualan, manajemen gudang, manajemen mutu, dan akuntansi. Sistem dapat dikonfigurasi untuk semua ukuran dari perusahaan kecil atau menengah hingga perusahaan multinasional, yang juga dapat bekerja dengan SAP, tetapi hanya menyiapkan sistem saja sudah sangat menuntut. itu di luar kemampuan mereka.

Kompleksitas dalam sistem SAP dipengaruhi oleh perluasan sistem SAP oleh sistem lain, dan oleh fungsi duplikasi dan struktur data di luar sistem inti ERP. Metode menyediakan alat untuk mengamati kompleksitas dalam waktu dan selama perubahan dan

untuk mengurangi kompleksitas dan menjaga sistem sesederhana mungkin dengan mengikuti proses bisnis dari sistem SAP tanpa melakukan modifikasi sistem SAP. Hal ini mampu mengelola kompleksitas implementasi SAP dengan menghemat biaya dan waktu implementasi sistem ERP dan mempermudah pemeliharaan sistem.

Metode ASAP adalah metode standar SAP yang berkaitan dengan persiapan mengimplementasi sistem, dikembangkan dan diperbarui oleh konsultan yang terlibat dalam implementasi produk perangkat lunak SAP. Metode ASAP ini berguna bagi manajer proyek dan tim proyek dalam merancang implementasi SAP dengan cara yang efisien, dan mengoptimalkan waktu, orang, kualitas, dan sumber daya lainnya secara efektif. Metode ASAP menyediakan peta jalan sebagai kerangka kerja yang digunakan untuk mengimplementasi sistem SAP secara berkelanjutan dalam proses bisnis. Kerangka kerja metode ASAP terdiri dari lima fase implementasi, setiap fase menyediakan rencana yang terperinci untuk membantu dalam dokumentasi, rekomendasi, dan implementasi sistem SAP. Setiap fase diakhiri dengan pemeriksaan kualitas sebagai bukti memantau hasil dan faktor kritis mencapai keberhasilan [Dewangga et al., 2022]. Penjelasan fase adalah sebagai berikut:

Fase 1 : Persiapan Project

Fase ini berfokus pada perencanaan awal dan persiapan untuk menetapkan tujuan secara tepat untuk menentukan ruang lingkup implementasi, pembentukan tim inti, proyek, dan konsultasi, dan menentukan urutan implementasi proyek.

Memperoleh dukungan manajemen/pemangku kepentingan tingkat senior, Merancang proses pengambilan keputusan yang efisien,Menciptakan lingkungan yang cocok untuk perubahan dan rekayasa ulang, Membangun tim proyek yang memiliki berkapabilitas.

Fase 2 : Blueprint Bisnis

Fase ini mengenai prosedur untuk mendukung proyek, yang berisi persyaratan yang diperoleh selama rapat dengar pendapat dan inisiasi awal, dan menetapkan jadwal untuk masing-masing tim. Contoh: bagaimana proses pengadaan barang yang terjadi diperusahaan, bagaimana proses pencatatan suatu transaksi keuangan perusahaan dicatat mempunyai proses kerja yang berbeda dan dijadikan dalam dokumen *blueprint* yang berbeda.

Fase 3 : Realisasi

Fase ini untuk menerapkan persyaratan proses bisnis mengenai Baseline, terkonsentrasi pada ruang lingkup utama; Konfigurasi Akhir, berkaitan dengan ruang lingkup yang tersisa. Fungsional SAP akan menginisialisai konfigurasi dasar dari SAP dan menyesuaikannya dengan *blueprint* yang sudah ada. *Fine tunning* dilakukan untuk menambahkan kondisi yang tidak ada pada *blueprint* dan menampung kebutuhan-kebutuhan khusus di sistem SAP.

Fase 4 : Persiapan Akhir

Fase ini mengenai persiapan akhir untuk pengujian, manajemen sistem, pelatihan para pengguna, dan tugas lain yang diperlukan untuk melakukan go live proyek. Manajemen sistem, pengujian, pelatihan kepada pengguna, dan proses *cut-over* disertakan dalam langkah ini. Penyelesaian untuk memastikan bahwa sistem SAP baru siap digunakan. Fase ini tidak hanya melakukan pengujian konfigurasi, tetapi juga untuk memastikan migrasi data dari sistem lama ke yang baru. Untuk memastikan keakuratan data dan stabilitas sistem SAP, pengujian beban kerja, dan pengujian fungsional membantu dalam memberikan hasil yang baik. Penting untuk menjaga kinerja optimal sistem SAP. Ikhtisar metode ASAP disediakan pada tahap ini.

Fase 5 : Go Live dan Support

Fase ini merupakan perjalanan dari lingkungan pra-produksi ke operasi proyek secara langsung, dengan monitoring

transaksi sistem, dan pengoptimalan kinerja sistem. semua operasi dan sistem yang dilakukan untuk memenuhi persyaratan. Fase ini juga mewakili siklus perencanaan untuk aktivasi proyek. Elemen-elemen yang termasuk dalam langkah ini meliputi pemeriksaan dan pemantauan transaksi pada sistem, pengaturan dukungan produksi dan mengoptimalkan kinerja sistem. Fase ini juga mencakup pemeliharaan kontak, proses dokumentasi dan prosedur agar dapat mengoptimalkan kinerja bisnis.

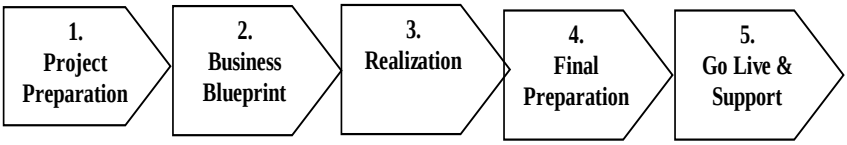
Metode ASAP mencakup seluruh siklus hidup proyek dari evaluasi awal hingga manajemen pasca-proyek. Peta jalan ASAP juga memastikan sistem delivery nilai yang transparan dan secara konsisten mencerminkan bagaimana perawatan bisnis yang dilakukan sambil bertindak sebagai panduan untuk SOA (*Service Oriented Architecture*) dan BPM (*Business Process Management*). Kelebihan dari metode ASAP adalah sebagai berikut:

1. Pendekatan yang dilakukan secara transparan terbukti dalam memfasilitasi tim proyek untuk berkolaborasi secara efisien dalam mengurangi biaya, waktu, dan risiko implementasi
2. Manajemen risiko yang masuk akal
3. Pengetahuan manajemen yang lebih baik : ASAP membantu project management dalam mengatur, mengimplementasikan, dan memenuhi tujuan bisnis
4. Tools untuk steering committee
5. Hasil proyek yang lebih baik karena metodologi ASAP didasarkan pada pengalaman orang lain sehingga dalam menyusun roadmap membutuhkan soft skill dan fakta yang jelas
6. Waktu pengerjaan proyek dapat dikurangi
7. Perencanaan komunikasi yang lebih baik
8. Kombinasi dari beberapa best practice tools.

Metode ASAP digunakan dalam proyek implementasi ERP SAP karena merupakan *best practice* dalam mengadopsi SAP ke perusahaan. Tujuan metode ASAP adalah untuk membantu *project*

management dalam merancang implementasi SAP dengan cara yang paling efisien serta dengan mengoptimalkan waktu, sumber daya manusia, kualitas proyek, dan sumber daya lainnya secara efektif.

Dengan menggunakan metode ASAP, maka *project management* menyediakan peta jalan yang digunakan untuk mengoptimalkan dan mengimplementasikan sistem SAP secara berkelanjutan dalam proses bisnis. Peta jalan metode ASAP implementasi ERP dapat dilihat pada gambar berikut:



Gambar 9.10. Metode ASAP Untuk Implementasi ERP

DAFTAR PUSTAKA

Holub, I., and Bruckner, T. 2016. Measuring complexity of SAP systems. Complex Systems Informatics and Modeling Quarterly, (8), 60-67.

Kirmizi, M., & Kocaoglu, B. 2020. The key for success in enterprise information systems projects: development of a novel ERP readiness assessment method and a case study. Enterprise Information Systems, 14(1),1-37. doi:10.1080/17517575.2019.1686656.

Luvai F. Motiwalla; Jeff Thompson. 2014. Enterprise Systems for Management second edition Pearson Education Limited, England ISBN: 978-1-29202-348-9.

Wijaya, S.F., Prabowo, H., Kosala, R.R., Meyliana. 2017. Identification of key success factors and challenges for ERP systems-A systematic literature review IEEE International Conference on Applied Computer and Communication Technologies. pp. 1–6. **doi:** 10.1109/COMCOM.2017.8167091

Dewangga, K., Emanuel, A. W. R., & Widhiyanti, K. 2022. Perancangan Gamifikasi Pada Proses Implementasi ERP Menggunakan Metode Accelerate SAP. Teknika, 11(3), 225-234. doi: 10.34148/teknika.v11i3.552

BAB 10
PENGUJIAN SISTEM

Oleh Angga Aditya Permana

10.1 Pendahuluan

Pengujian sistem adalah proses verifikasi dan validasi yang dilakukan untuk memastikan bahwa sistem atau perangkat lunak berfungsi sebagaimana yang diharapkan dan memenuhi persyaratan yang telah ditetapkan sebelumnya. Pengujian sistem melibatkan serangkaian langkah yang dirancang untuk mengidentifikasi kesalahan, cacat, atau ketidaksesuaian antara perilaku aktual sistem dengan perilaku yang diharapkan. (Romeo, 2003)

Gambar 10.1. Pengujian Sistem

Berikut adalah tahapan-tahapan umum yang terlibat dalam pengujian sistem:

1. Perencanaan Pengujian: Tahap perencanaan melibatkan identifikasi tujuan pengujian, penentuan sumber daya yang diperlukan, dan penyusunan rencana pengujian. Pada tahap ini, tim pengujian juga mengidentifikasi skenario pengujian, mengumpulkan data pengujian, dan membuat jadwal pengujian.
2. Analisis Persyaratan: Tim pengujian harus memahami persyaratan sistem secara mendalam. Mereka menganalisis persyaratan fungsional dan non-fungsional untuk mengidentifikasi perilaku yang diharapkan dari sistem.
3. Desain Pengujian: Pada tahap ini, tim pengujian merancang rencana pengujian yang terperinci. Mereka menentukan jenis pengujian yang akan dilakukan, seperti pengujian fungsional, pengujian integrasi, pengujian kinerja, pengujian keamanan, dan lain-lain. Selain itu, mereka juga merancang skenario pengujian yang mencakup langkah-langkah konkret yang akan diambil selama pengujian.
4. Pembuatan Kasus Uji: Tim pengujian membuat kasus uji yang berisi serangkaian langkah-langkah yang akan dijalankan selama pengujian. Kasus uji ini mencakup input yang diharapkan, langkah-langkah yang harus diambil, dan output yang diharapkan. Kasus uji dirancang untuk mencakup berbagai situasi yang mungkin terjadi saat sistem digunakan.
5. Eksekusi Pengujian: Pada tahap ini, tim pengujian menjalankan kasus uji yang telah dibuat. Mereka mengumpulkan data pengujian dan mencatat hasilnya. Jika ada kesalahan atau cacat yang ditemukan, tim pengujian akan mengidentifikasi sumber masalah dan mencatatnya untuk diperbaiki.
6. Analisis Hasil: Setelah pengujian selesai, tim pengujian menganalisis hasil pengujian. Mereka membandingkan output yang dihasilkan dengan output yang diharapkan dan

mengidentifikasi kesalahan atau ketidaksesuaian yang mungkin terjadi. Analisis ini membantu dalam penentuan apakah sistem telah memenuhi persyaratan yang ditetapkan.

7. Pelaporan: Tim pengujian menghasilkan laporan pengujian yang mencakup ringkasan hasil pengujian, kesalahan yang ditemukan, dan rekomendasi perbaikan. Laporan ini disampaikan kepada tim pengembangan atau pemangku kepentingan lainnya untuk mengambil tindakan yang diperlukan.
8. Perbaikan dan Retesting: Jika kesalahan atau cacat ditemukan selama pengujian, tim pengembangan akan melakukan perbaikan dan memperbaiki masalah yang ada. Setelah perbaikan selesai, pengujian ulang dilakukan untuk memastikan bahwa masalah telah diperbaiki dengan baik.
9. Pengujian Integrasi: Tahap ini melibatkan pengujian interaksi antara komponen-komponen sistem yang berbeda untuk memastikan bahwa mereka berfungsi dengan baik bersama-sama. Hal ini memerlukan pengujian integrasi antara modul atau komponen individual, antarmuka antara komponen, dan aliran data di antara mereka.
10. Pengujian Kinerja: Pengujian kinerja dilakukan untuk memeriksa sejauh mana sistem mampu mengatasi beban kerja yang tinggi dan memenuhi persyaratan kinerja yang ditetapkan. Pengujian ini mencakup pengujian respons waktu, penggunaan sumber daya (misalnya, CPU, memori), skalabilitas, dan kestabilan sistem saat beroperasi dalam kondisi beban yang tinggi.
11. Pengujian Keamanan: Tahap ini melibatkan pengujian sistem untuk mengidentifikasi kerentanan keamanan dan memastikan bahwa langkah-langkah keamanan yang tepat telah diimplementasikan. Pengujian keamanan mencakup pengujian penetrasi, pengujian kerentanan, dan pengujian keamanan perangkat lunak untuk mengidentifikasi celah

keamanan yang mungkin dieksploitasi oleh pihak yang tidak berwenang.

12. Pengujian Pengguna: Pengujian pengguna (user acceptance testing) melibatkan pengguna akhir atau pemangku kepentingan yang terlibat dalam sistem untuk memverifikasi apakah sistem memenuhi kebutuhan pengguna dan tujuan bisnis yang telah ditetapkan. Pengguna melakukan pengujian dengan menggunakan skenario dan kasus uji yang relevan dengan kebutuhan mereka, serta memberikan umpan balik tentang pengalaman pengguna dan kemampuan sistem untuk memenuhi harapan mereka.
13. Pengujian Pemulihan Bencana (Disaster Recovery Testing): Jika sistem memiliki strategi pemulihan bencana atau rencana darurat, pengujian ini dilakukan untuk memastikan bahwa rencana tersebut berfungsi seperti yang diharapkan dalam situasi pemulihan. Pengujian pemulihan bencana melibatkan simulasi bencana atau kegagalan sistem yang memerlukan pemulihan, dan memverifikasi bahwa pemulihan dilakukan dengan sukses dan sistem berfungsi dengan benar setelah pemulihan dilakukan.
14. Pengujian Regresi: Pengujian regresi dilakukan setelah perubahan atau pembaruan pada sistem untuk memastikan bahwa perubahan tersebut tidak mengganggu fungsionalitas yang ada. Dalam pengujian regresi, kasus uji yang telah ada sebelumnya dijalankan kembali untuk memastikan bahwa tidak ada perubahan yang tidak diinginkan atau kesalahan yang muncul karena perubahan tersebut.
15. Validasi Akhir: Tahap terakhir dari pengujian sistem melibatkan validasi akhir untuk memastikan bahwa sistem siap untuk diimplementasikan atau dirilis. Tim pengujian melakukan evaluasi menyeluruh terhadap hasil pengujian, laporan, dan metrik pengujian yang dikumpulkan selama seluruh proses pengujian. Validasi akhir juga melibatkan

verifikasi bahwa semua persyaratan sistem telah terpenuhi dan sistem telah melalui pengujian yang memadai.

Setiap langkah dalam pengujian sistem harus didokumentasikan dengan baik, termasuk hasil pengujian, kesalahan yang ditemukan, tindakan perbaikan yang diambil, dan rekomendasi untuk pengembangan masa depan. Dokumentasi ini sangat penting untuk melacak kemajuan pengujian, berbagi informasi dengan tim pengembangan, dan memastikan bahwa sistem memenuhi standar kualitas yang diharapkan sebelum diterapkan atau dirilis ke pengguna akhir.

10.2 Macam – macam Pengujian Sistem Bagi Quality Assurance

Dalam bidang *Quality Assurance* (QA), berikut ini adalah beberapa macam pengujian sistem yang umum dilakukan untuk memastikan kualitas produk atau sistem: (Jorgensen, 2014)

Gambar 10.2. QA Icon

1. Pengujian Fungsional: Pengujian fungsional dilakukan untuk memverifikasi apakah sistem memenuhi persyaratan fungsional yang telah ditentukan. Hal ini melibatkan pengujian fitur-fitur dan fungsi-fungsi sistem, serta memastikan bahwa sistem beroperasi sesuai dengan harapan.

2. Pengujian Non-Fungsional: Pengujian non-fungsional bertujuan untuk menguji aspek-aspek sistem yang tidak terkait dengan fungsionalitasnya, seperti performa, keamanan, kehandalan, dan skalabilitas. Pengujian ini membantu memastikan bahwa sistem memenuhi persyaratan non-fungsional yang ditetapkan.
3. Pengujian Pengguna (User Acceptance Testing/UAT): Pengujian pengguna dilakukan oleh pengguna akhir atau pemangku kepentingan untuk memastikan bahwa sistem memenuhi kebutuhan dan harapan mereka. UAT melibatkan penggunaan kasus uji yang relevan dengan kebutuhan pengguna dan memberikan umpan balik tentang pengalaman pengguna serta kemampuan sistem untuk memenuhi harapan mereka.
4. Pengujian Regresi: Pengujian regresi dilakukan setelah perubahan atau pembaruan pada sistem untuk memastikan bahwa perubahan tersebut tidak mengganggu fungsionalitas yang sudah ada. Pengujian ini melibatkan menjalankan kembali kasus uji yang relevan untuk memastikan tidak ada efek samping atau kesalahan yang muncul akibat perubahan tersebut.
5. Pengujian Integrasi: Pengujian integrasi dilakukan untuk menguji interaksi antara komponen-komponen sistem yang berbeda setelah unit-unit tersebut diuji secara individu. Tujuannya adalah memastikan bahwa komponen-komponen tersebut dapat berkomunikasi dengan baik dan berfungsi secara terintegrasi.
6. Pengujian Penyusunan Kembali (Reconfiguration Testing): Pengujian ini dilakukan ketika ada perubahan pada konfigurasi sistem, seperti pembaruan perangkat keras atau perubahan lingkungan operasional. Pengujian ini membantu memastikan bahwa sistem beroperasi dengan baik setelah perubahan konfigurasi dilakukan.

7. Pengujian Kinerja: Pengujian kinerja dilakukan untuk menguji kemampuan sistem dalam menangani beban kerja yang tinggi dan memastikan performa yang memadai. Ini melibatkan pengujian respons waktu, kecepatan pemrosesan, penggunaan sumber daya, dan skalabilitas sistem dalam kondisi beban yang berbeda.
8. Pengujian Keamanan: Pengujian keamanan dilakukan untuk mengidentifikasi dan menguji kerentanan keamanan dalam sistem. Hal ini melibatkan pengujian penetrasi, pengujian kerentanan, dan pengujian keamanan perangkat lunak untuk memastikan sistem tahan terhadap serangan dan melindungi data yang sensitif.
9. Pengujian Pemulihan Bencana (*Disaster Recovery Testing*): Pengujian pemulihan bencana dilakukan untuk menguji kemampuan sistem dalam pemulihan setelah terjadinya kegagalan atau bencana. Pengujian ini memverifikasi keberhasilan rencana pemulihan bencana dan memastikan bahwa sistem dapat pulih dengan sukses.
10. Pengujian Kompatibilitas: Pengujian kompatibilitas melibatkan pengujian sistem pada berbagai platform, perangkat keras, atau lingkungan operasional yang berbeda. Tujuannya adalah memastikan bahwa sistem berfungsi dengan baik di berbagai kombinasi yang mungkin digunakan oleh pengguna.

Pengujian sistem dalam konteks *Quality Assurance* membantu memastikan bahwa produk atau sistem memenuhi standar kualitas yang ditetapkan, memenuhi persyaratan pengguna, dan berfungsi dengan baik sesuai dengan yang diharapkan.

10.3 Macam – macam Pengujian Fungsional

Gambar 10.3. Pengujian Fungsional

Berikut ini adalah beberapa macam pengujian fungsional yang umum dilakukan dalam pengembangan perangkat lunak: (Jorgensen, 2014)

1. Pengujian Unit: Pengujian unit fokus pada pengujian komponen individual atau unit terkecil dari sistem, seperti fungsi atau metode. Tujuannya adalah untuk memastikan bahwa setiap unit berfungsi dengan benar secara terisolasi.
2. Pengujian Integrasi: Pengujian integrasi melibatkan pengujian interaksi antara komponen-komponen sistem yang berbeda setelah unit-unit tersebut diuji secara individu. Pengujian ini memastikan bahwa komponen-komponen tersebut dapat berinteraksi dengan baik dan saling berkomunikasi.
3. Pengujian Sistem: Pengujian sistem dilakukan pada sistem secara keseluruhan untuk memverifikasi fungsi-fungsi sistem yang lebih kompleks. Tujuannya adalah untuk memastikan bahwa sistem beroperasi sesuai dengan persyaratan yang ditetapkan.
4. Pengujian Fungsionalitas Fungsional: Pengujian fungsionalitas fungsional berfokus pada pengujian fitur-fitur sistem dan fungsi-fungsi yang telah ditentukan dalam persyaratan

fungsional. Tujuannya adalah untuk memastikan bahwa setiap fitur berfungsi dengan benar dan menghasilkan output yang diharapkan.

5. Pengujian Kasus Uji Positif: Pengujian kasus uji positif melibatkan pengujian fitur dengan memberikan input yang valid dan diharapkan menghasilkan output yang benar. Tujuannya adalah untuk memastikan bahwa sistem dapat mengatasi situasi yang diharapkan dengan benar.
6. Pengujian Kasus Uji Negatif: Pengujian kasus uji negatif dilakukan dengan memberikan input yang tidak valid atau tidak diharapkan, dan memeriksa apakah sistem menghasilkan output yang sesuai dengan yang diharapkan dalam situasi tersebut. Tujuannya adalah untuk memastikan bahwa sistem dapat mengatasi situasi yang tidak diinginkan dengan benar, seperti menangani kesalahan atau memberikan pesan kesalahan yang tepat.
7. Pengujian Interaksi Pengguna: Pengujian interaksi pengguna melibatkan pengujian antarmuka pengguna (UI) untuk memastikan bahwa sistem dapat digunakan dengan mudah dan intuitif oleh pengguna. Pengujian ini mencakup navigasi, input data, validasi, respons UI, dan kesesuaian dengan prinsip-prinsip desain antarmuka pengguna yang baik.
8. Pengujian Keberlanjutan: Pengujian keberlanjutan dilakukan untuk memastikan bahwa sistem dapat berfungsi secara terus-menerus dalam jangka waktu yang lama. Tujuannya adalah untuk memverifikasi kestabilan sistem, pengelolaan sumber daya yang efisien, dan pemulihan dari kegagalan yang mungkin terjadi.
9. Pengujian Ketersediaan: Pengujian ketersediaan dilakukan untuk memverifikasi bahwa sistem dapat diakses dan digunakan sesuai dengan kebutuhan pengguna. Pengujian ini mencakup pengujian toleransi kesalahan, pemulihan dari

kegagalan, dan upaya untuk meminimalkan waktu tidak tersedia.

10. Pengujian Kompatibilitas: Pengujian kompatibilitas melibatkan pengujian sistem pada berbagai platform, konfigurasi, perangkat keras, dan lingkungan operasional yang berbeda. Tujuannya adalah untuk memastikan bahwa sistem berfungsi dengan baik dan kompatibel dengan berbagai kombinasi yang mungkin digunakan oleh pengguna.

Pengujian fungsional sangat penting dalam memastikan bahwa sistem memenuhi persyaratan fungsional yang telah ditetapkan dan dapat berfungsi sesuai dengan yang diharapkan oleh pengguna. Melalui pengujian ini, cacat atau kesalahan dalam fitur dan fungsi sistem dapat diidentifikasi dan diperbaiki sebelum sistem diterapkan atau dirilis kepada pengguna akhir.

10.4 Performance Testing

Gambar 10.4. Performance Testing Icon

Performance testing adalah jenis pengujian sistem yang dilakukan untuk mengukur dan mengevaluasi kinerja suatu sistem atau aplikasi dalam berbagai kondisi dan beban kerja yang berbeda. Tujuannya adalah untuk memastikan bahwa sistem mampu beroperasi dengan performa yang memadai dan

memenuhi persyaratan kinerja yang telah ditetapkan. (Jorgensen, 2014)

Berikut ini adalah beberapa aspek yang diperhatikan dalam *performance testing*:

1. Respons Waktu (*Response Time*): Pengujian respons waktu mengukur waktu yang dibutuhkan oleh sistem untuk merespons suatu permintaan atau tindakan pengguna. Hal ini mencakup waktu yang diperlukan untuk memproses permintaan, mengambil data, dan menghasilkan respons yang diharapkan. Performa yang baik akan ditunjukkan oleh respons waktu yang cepat dan responsif.
2. Throughput: Pengujian throughput mengukur sejauh mana sistem mampu menangani jumlah transaksi atau permintaan dalam satu waktu tertentu. Hal ini mencakup pengukuran jumlah transaksi yang berhasil dilakukan dalam interval waktu tertentu. Performa yang baik ditandai dengan tingginya throughput, yang menunjukkan kemampuan sistem dalam menangani beban kerja yang tinggi.
3. Beban Kerja (*Workload*): Pengujian beban kerja melibatkan memberikan beban atau volume pekerjaan yang tinggi pada sistem untuk menguji kinerjanya di bawah tekanan. Hal ini dapat mencakup simulasi pengguna yang banyak, permintaan yang intensif, atau jumlah transaksi yang tinggi. Pengujian beban kerja membantu mengevaluasi kemampuan sistem dalam menangani situasi beban kerja yang tinggi secara efisien.
4. Skalabilitas (*Scalability*): Pengujian skalabilitas bertujuan untuk mengukur kemampuan sistem dalam menangani peningkatan beban kerja atau pertambahan pengguna. Pengujian ini membantu mengidentifikasi apakah sistem dapat diubah atau diperluas untuk menangani pertumbuhan yang lebih besar di masa depan tanpa mengorbankan performa.

5. Stabilitas: Pengujian stabilitas melibatkan menjalankan sistem dalam waktu yang lama untuk memastikan bahwa tidak ada penurunan performa yang signifikan atau kegagalan yang terjadi seiring waktu. Hal ini penting untuk memastikan bahwa sistem dapat beroperasi secara konsisten dan dapat diandalkan dalam jangka waktu yang lama.
6. Penggunaan Sumber Daya (*Resource Utilization*): Pengujian penggunaan sumber daya melibatkan pengukuran dan evaluasi penggunaan sumber daya oleh sistem, seperti penggunaan CPU, memori, jaringan, dan penyimpanan. Hal ini membantu mengidentifikasi apakah sistem mengoptimalkan penggunaan sumber daya dan memiliki keseimbangan yang baik antara kinerja dan penggunaan sumber daya.
7. Pemulihan dari Kegagalan: Pengujian pemulihan dari kegagalan dilakukan untuk menguji kemampuan sistem dalam pulih dari kegagalan atau situasi darurat. Hal ini melibatkan simulasi kegagalan, seperti kehilangan koneksi jaringan atau mati listrik, dan memverifikasi bahwa sistem dapat pulih dan berfungsi dengan baik setelah situasi tersebut.
8. Pemantauan (*Monitoring*): Pengujian performa juga melibatkan pemantauan secara kontinu terhadap sistem selama pengujian untuk mengumpulkan data performa dan mengidentifikasi potensi masalah. Pemantauan dapat melibatkan penggunaan alat pemantauan performa, pengumpulan metrik performa, dan analisis data untuk mendapatkan wawasan yang lebih baik tentang kinerja sistem.

Pengujian performa sangat penting untuk memastikan bahwa sistem atau aplikasi mampu beroperasi dengan performa yang baik, responsif, dan dapat menangani beban kerja yang tinggi. Dengan melakukan pengujian performa, dapat diidentifikasi dan diperbaiki masalah performa sebelum sistem diimplementasikan atau dirilis ke pengguna akhir.

10.5 System Testing

System testing adalah jenis pengujian yang dilakukan untuk mengevaluasi keseluruhan sistem atau perangkat lunak yang telah dikembangkan. Tujuan utama dari system testing adalah memverifikasi apakah sistem telah memenuhi persyaratan fungsional dan non-fungsional yang telah ditetapkan sebelumnya. (Jorgensen, 2014)

Berikut ini adalah beberapa hal yang dilakukan dalam *system testing*:

1. Verifikasi Persyaratan Sistem: Pada tahap awal system testing, tim pengujian memeriksa apakah semua persyaratan sistem telah diimplementasikan dengan benar. Mereka memeriksa apakah fitur-fitur yang diharapkan telah ada dan berfungsi sesuai yang diharapkan.
2. Pengujian Fungsional: Pengujian fungsional dalam system testing dilakukan untuk memastikan bahwa setiap fitur dan fungsi sistem beroperasi dengan benar. Tim pengujian menjalankan kasus uji yang dirancang sebelumnya untuk menguji fungsionalitas sistem dan membandingkan hasilnya dengan yang diharapkan.
3. Pengujian Integrasi: Pengujian integrasi juga menjadi bagian dari system testing, di mana komponen-komponen sistem yang berbeda diuji untuk memastikan bahwa mereka dapat berinteraksi dengan baik dan mengirimkan data secara efektif di antara mereka. Pengujian ini melibatkan pengujian antarmuka dan aliran data antara komponen-komponen sistem.
4. Pengujian Kinerja: Dalam system testing, pengujian kinerja dilakukan untuk mengevaluasi kinerja sistem dalam berbagai situasi dan kondisi beban kerja. Hal ini mencakup pengujian respons waktu, throughput, penggunaan sumber daya, dan stabilitas sistem di bawah beban yang tinggi. Tujuan pengujian

kinerja adalah memastikan bahwa sistem berkinerja sesuai dengan persyaratan kinerja yang ditetapkan sebelumnya.

5. Pengujian Keamanan: Pengujian keamanan dilakukan untuk mengidentifikasi kerentanan keamanan dalam sistem dan memastikan bahwa langkah-langkah keamanan yang sesuai telah diimplementasikan. Pengujian ini melibatkan pengujian penetrasi, pengujian kerentanan, dan pengujian keamanan perangkat lunak untuk mengevaluasi tingkat keamanan sistem.
6. Pengujian Usability: System testing juga mencakup pengujian usability, yang bertujuan untuk mengevaluasi sejauh mana sistem dapat digunakan dengan mudah oleh pengguna. Pengujian usability melibatkan pengujian antarmuka pengguna, navigasi, pengalaman pengguna, dan kemudahan penggunaan sistem secara keseluruhan.
7. Pengujian Pemulihan Bencana (Disaster Recovery Testing): Pengujian pemulihan bencana dilakukan untuk menguji kemampuan sistem dalam pemulihan setelah terjadinya bencana atau kegagalan. Pengujian ini memverifikasi efektivitas rencana pemulihan bencana dan kemampuan sistem untuk pulih dan beroperasi kembali dengan benar.
8. Pengujian Kompatibilitas: Pengujian kompatibilitas melibatkan pengujian sistem pada berbagai platform, konfigurasi, perangkat keras, atau lingkungan operasional yang berbeda. Tujuan dari pengujian ini adalah memastikan bahwa sistem berfungsi dengan baik dan kompatibel di berbagai kombinasi yang mungkin digunakan oleh pengguna.
9. Pengujian Pembaruan (Regression Testing): Pengujian pembaruan dilakukan setelah perubahan atau pembaruan pada sistem untuk memastikan bahwa perubahan tersebut tidak mempengaruhi fungsionalitas yang sudah ada. Pengujian ini melibatkan menjalankan kembali kasus uji yang relevan

untuk memverifikasi bahwa tidak ada efek samping atau kesalahan yang muncul akibat perubahan tersebut.

Pada akhir system testing, tim pengujian akan menganalisis hasil pengujian, melaporkan temuan, dan memberikan rekomendasi perbaikan jika ditemukan masalah. Hasil dari system testing digunakan untuk mengambil keputusan apakah sistem telah siap untuk diimplementasikan atau dirilis kepada pengguna akhir.

10.6 Unit Testing

Unit testing adalah jenis pengujian perangkat lunak yang dilakukan pada tingkat unit terkecil, yaitu pada level komponen individual atau unit dalam sistem. Tujuannya adalah untuk memverifikasi kebenaran dan korektibilitas fungsi atau metode individual. (Jorgensen, 2014)

Berikut ini adalah beberapa hal yang dilakukan dalam unit testing:

1. Unit yang Diuji: Unit testing berfokus pada pengujian komponen individual atau unit, seperti fungsi, metode, kelas, atau modul yang independen dalam sistem. Unit ini biasanya merupakan bagian terkecil yang dapat diuji secara terisolasi.
2. Kasus Uji: Kasus uji atau test case dibuat untuk menguji setiap unit. Kasus uji ini dirancang untuk mencakup berbagai skenario yang mungkin terjadi, termasuk kondisi normal dan eksepsi. Kasus uji mengidentifikasi input yang diberikan ke unit dan memverifikasi output yang diharapkan.
3. *Framework* Pengujian: Biasanya, unit testing menggunakan *framework* pengujian seperti JUnit (untuk Java), NUnit (untuk NET), atau PyTest (untuk Python). Framework ini menyediakan struktur dan alat untuk membuat dan menjalankan kasus uji secara otomatis.
4. Menjalankan Unit Testing: Unit testing dilakukan dengan menjalankan kasus uji menggunakan framework pengujian.

Setiap kasus uji menjalankan unit secara terisolasi dan memverifikasi bahwa output yang dihasilkan sesuai dengan yang diharapkan. Pemantauan dan pengujian debugging juga dapat dilakukan untuk memeriksa nilai-nilai dalam unit selama proses pengujian.

5. Simulasi Dependensi: Unit testing sering melibatkan penggunaan teknik simulasi dependensi, seperti penggunaan objek palsu (*dummy object*), objek pengganti (*mock object*), atau stub untuk mengisolasi unit yang diuji dari dependensi luar. Hal ini memungkinkan pengujian unit dilakukan secara terisolasi tanpa ketergantungan pada komponen lain dalam sistem.
6. Pemeriksaan Kode: Selama unit testing, pemeriksaan kode dilakukan untuk memastikan bahwa kode unit terstruktur dengan baik, mematuhi standar penulisan kode, dan tidak ada kesalahan sintaksis atau logika yang tersembunyi. Alat-alat pemeriksaan kode seperti static code analysis atau code review juga dapat digunakan untuk memeriksa kualitas kode.
7. Coverage Testing: Unit testing sering melibatkan pengujian cakupan (*coverage testing*) untuk memastikan bahwa seluruh kode unit telah diuji secara memadai. Cakupan pengujian dapat diukur menggunakan alat pengujian seperti JaCoCo (untuk Java) atau Cobertura. Tujuan cakupan pengujian adalah mencapai tingkat cakupan yang tinggi, sehingga sebagian besar atau seluruh kode unit telah diuji.
8. Automasi Unit Testing: Unit testing umumnya diotomatiskan untuk memungkinkan pengujian yang cepat, berulang, dan konsisten. Penggunaan framework pengujian dan alat pengujian otomatis membantu dalam menjalankan kasus uji secara terprogram dan menghasilkan laporan hasil pengujian yang akurat.

Unit testing sangat penting dalam pengembangan perangkat lunak karena membantu dalam memvalidasi kebenaran dan fungsionalitas komponen individu sebelum mereka digabungkan menjadi sistem yang lebih besar. Dengan melakukan unit testing secara teratur dan menyeluruh, kesalahan dan masalah dapat terdeteksi lebih awal, mempercepat proses debugging, dan meningkatkan kualitas perangkat lunak secara keseluruhan.

10.7 Integration Testing

Integration testing adalah jenis pengujian yang dilakukan untuk menguji interaksi antara komponen-komponen yang berbeda dalam sebuah sistem. Tujuan dari integration testing adalah memastikan bahwa komponen-komponen tersebut berfungsi dengan baik ketika diintegrasikan dan berinteraksi satu sama lain. (Jorgensen, 2014)

Berikut ini adalah beberapa aspek yang perlu diperhatikan dalam integration testing:

1. Unit yang Diuji: Pada level integration testing, komponen yang diuji bukanlah unit-unit terkecil seperti pada unit testing, melainkan komponen yang lebih besar, seperti modul, kelas, atau bagian dari sistem yang terdiri dari beberapa unit.
2. Strategi Pengujian: Ada beberapa strategi yang dapat digunakan dalam integration testing, seperti top-down, bottom-up, sandwich, atau big bang. Strategi yang dipilih tergantung pada kompleksitas sistem, ketergantungan antara komponen-komponen, dan ketersediaan komponen-komponen yang diperlukan untuk pengujian.
3. Simulasi Dependensi: Dalam integration testing, seringkali komponen yang terkait dengan komponen lain belum sepenuhnya tersedia atau belum siap untuk diuji. Oleh karena itu, teknik simulasi dependensi seperti menggunakan objek palsu (*dummy object*), objek pengganti (*mock object*), atau stub

digunakan untuk mengisolasi komponen yang diuji dari komponen yang belum siap.

4. Uji Fungsionalitas dan Antarmuka: Integration testing melibatkan pengujian fungsionalitas komponen-komponen yang diintegrasikan, serta pengujian interaksi antara komponen-komponen tersebut. Hal ini mencakup pengujian antarmuka komponen dan memastikan bahwa komunikasi dan pertukaran data antara komponen berjalan dengan baik.
5. Pengujian Aliran Data: Pengujian aliran data dilakukan untuk memastikan bahwa data dikirim dan diterima dengan benar antara komponen-komponen yang diuji. Pengujian ini mencakup validasi data input dan output, pengujian transformasi data, serta pengujian pengiriman data secara asinkron atau sinkron.
6. Pengujian Kasus Uji Terpadu: Dalam integration testing, kasus uji terpadu (*end-to-end test cases*) juga dapat digunakan untuk menguji aliran proses bisnis yang melibatkan beberapa komponen. Kasus uji ini mencoba mensimulasikan situasi penggunaan yang lebih nyata dan menguji keseluruhan sistem dari awal hingga akhir.
7. Kesalahan Integrasi: Integration testing bertujuan untuk menemukan kesalahan integrasi yang mungkin terjadi antara komponen-komponen. Kesalahan integrasi dapat termasuk masalah komunikasi, kesalahan pemanggilan antara komponen, inkompatibilitas antarmuka, atau ketidakcocokan data.
8. Debugging dan Perbaikan: Jika kesalahan atau masalah ditemukan selama integration testing, langkah-langkah debugging dilakukan untuk mengidentifikasi dan memperbaiki kesalahan. Perbaikan yang dilakukan dapat melibatkan perubahan pada kode, konfigurasi, atau alur kerja komponen yang terlibat.

9. Verifikasi Performa: Dalam beberapa kasus, integration testing juga dapat melibatkan verifikasi kinerja komponen-komponen yang diintegrasikan. Pengujian ini mencakup pengujian respons waktu, throughput, dan penggunaan sumber daya untuk memastikan bahwa sistem dapat berkinerja dengan baik saat diintegrasikan.
10. Regression Testing: Setelah integration testing selesai, regression testing dapat dilakukan untuk memastikan bahwa perubahan atau integrasi yang dilakukan tidak mempengaruhi fungsi yang sudah ada. Kasus uji yang relevan dijalankan kembali untuk memverifikasi bahwa tidak ada efek samping atau kesalahan yang muncul akibat integrasi.

Integration testing penting dalam menguji dan memastikan bahwa komponen-komponen yang berbeda dalam sistem dapat berinteraksi dengan baik dan berfungsi sesuai yang diharapkan. Dengan melakukan integration testing secara efektif, kesalahan integrasi dapat diidentifikasi lebih awal dalam siklus pengembangan dan meminimalkan risiko kegagalan sistem saat diimplementasikan.

10.8 Usability Testing

Usability testing adalah jenis pengujian yang dilakukan untuk mengevaluasi sejauh mana pengguna dapat dengan mudah menggunakan sebuah produk atau sistem. Tujuan dari usability testing adalah untuk mengidentifikasi masalah pengalaman pengguna dan menyediakan umpan balik yang dapat digunakan untuk meningkatkan kemudahan penggunaan, efisiensi, dan kepuasan pengguna. (Romeo, 2003)

Berikut ini adalah beberapa hal yang dilakukan dalam usability testing:

1. Perencanaan Pengujian: Tahap perencanaan penting dalam usability testing. Tim pengujian perlu menentukan tujuan

pengujian, pengguna target yang akan terlibat, skenario pengujian yang akan dilakukan, dan metrik pengukuran yang relevan. Perencanaan juga melibatkan pengembangan skrip pengujian dan pengumpulan peralatan atau lingkungan pengujian yang diperlukan.

2. Identifikasi Pengguna Target: Pengujian usability melibatkan pengguna target yang mewakili kelompok pengguna yang sebenarnya. Tim pengujian perlu mengidentifikasi karakteristik pengguna target seperti latar belakang, pengalaman, dan tujuan penggunaan. Pemilihan pengguna yang tepat akan memastikan bahwa hasil pengujian merefleksikan pengalaman nyata pengguna.
3. Membuat Skenario Pengujian: Skenario pengujian atau tugas yang diberikan kepada pengguna diatur untuk menciptakan situasi nyata yang menunjukkan penggunaan sistem atau produk dalam konteks yang relevan. Skenario pengujian dapat mencakup serangkaian tugas atau aktivitas yang harus dilakukan oleh pengguna untuk menguji berbagai aspek kemudahan penggunaan dan kinerja sistem.
4. Melakukan Pengujian dengan Pengguna: Dalam usability testing, pengguna diundang untuk menguji produk atau sistem dengan menggunakan skenario pengujian yang telah ditentukan. Tim pengujian mengamati dan mencatat interaksi pengguna dengan sistem, kesulitan yang dihadapi, dan tanggapan pengguna terhadap pengalaman penggunaan. Pengujian dapat dilakukan dalam lingkungan laboratorium, observasi langsung, atau melalui alat perangkat lunak yang memfasilitasi pengujian jarak jauh.
5. Pengumpulan Umpan Balik: Selama pengujian, pengguna diwawancarai atau diminta untuk memberikan umpan balik mengenai pengalaman penggunaan, kesulitan yang mereka hadapi, dan saran perbaikan. Umpan balik ini sangat berharga

untuk memahami perspektif pengguna dan mengidentifikasi area yang perlu diperbaiki dalam hal kemudahan penggunaan.

6. Analisis Hasil dan Identifikasi Masalah: Setelah pengujian selesai, tim pengujian menganalisis data yang telah dikumpulkan dan mengidentifikasi masalah usability yang ditemukan. Masalah dapat mencakup kesalahan dalam navigasi, kebingungan dengan antarmuka pengguna, keterbatasan fungsionalitas, atau waktu respon yang lambat. Hasil analisis digunakan untuk membuat laporan yang merangkum temuan dan rekomendasi perbaikan.
7. Implementasi Perbaikan: Rekomendasi perbaikan yang dihasilkan dari usability testing dapat digunakan oleh tim pengembangan untuk memperbaiki masalah yang ditemukan. Ini dapat melibatkan perubahan desain antarmuka, penyesuaian alur kerja, perbaikan pesan kesalahan, atau perbaikan lainnya yang meningkatkan kemudahan penggunaan produk atau sistem.
8. Iterasi Pengujian: Usability testing sering melibatkan iterasi, di mana perbaikan yang diimplementasikan diperiksa kembali melalui pengujian berikutnya untuk memastikan bahwa masalah telah diatasi dan pengalaman pengguna telah meningkat.
9. Pelaporan dan Komunikasi: Hasil pengujian usability, termasuk temuan, rekomendasi perbaikan, dan langkah-langkah yang diambil, didokumentasikan dalam laporan pengujian. Laporan ini dapat digunakan untuk berbagi informasi dengan tim pengembangan, manajemen, dan pemangku kepentingan terkait lainnya.

Usability testing penting untuk memastikan bahwa produk atau sistem yang dikembangkan memenuhi kebutuhan dan ekspektasi pengguna. Dengan menguji produk dengan pengguna asli, tim pengembangan dapat memperoleh wawasan berharga

tentang kesulitan yang dihadapi pengguna dan menerapkan perbaikan yang dibutuhkan untuk meningkatkan kemudahan penggunaan dan kepuasan pengguna.

10.9 Smoke Testing

Smoke testing (juga dikenal sebagai *build verification testing*) adalah jenis pengujian perangkat lunak yang dilakukan untuk memverifikasi bahwa versi terbaru dari sebuah aplikasi atau sistem telah berhasil diinstal dan dapat berjalan dengan baik setelah perubahan atau pembaruan dilakukan. (Romeo, 2003)

Berikut ini adalah beberapa hal yang perlu diperhatikan dalam smoke testing:

1. Tujuan: Smoke testing bertujuan untuk melakukan pengujian yang cepat dan ringkas pada aplikasi atau sistem setelah perubahan signifikan atau pembaruan. Tujuan utamanya adalah untuk memverifikasi apakah sistem telah terinstal dengan baik dan mampu berjalan tanpa kesalahan serius yang dapat menghambat fungsi dasar.
2. Cakupan Pengujian: Smoke testing melibatkan pengujian pada fungsionalitas inti dan fitur utama dari sistem. Fokus utama adalah pada aspek yang kritis dan penting untuk penggunaan awal. Pengujian ini tidak mendalam dan tidak mencakup semua detail pengujian yang lebih mendalam.
3. Kasus Uji: Smoke testing melibatkan pengujian dengan menggunakan kasus uji yang sederhana dan terbatas. Kasus uji ini dirancang untuk mencakup skenario dasar yang menguji fungsionalitas utama dan mengidentifikasi kesalahan besar atau masalah yang mungkin ada.
4. Pengujian Stabilitas: Selain menguji fungsionalitas inti, smoke testing juga mencoba memastikan stabilitas sistem secara keseluruhan. Hal ini meliputi pengujian apakah sistem dapat diinstal dengan benar, dapat memulai dan menghentikan dengan baik, serta tidak mengalami kegagalan yang signifikan.

5. Waktu Pelaksanaan: Smoke testing dilakukan setelah perubahan atau pembaruan sistem, biasanya setelah proses build atau rilis baru. Tujuannya adalah untuk segera mengetahui apakah ada masalah besar yang menghalangi penggunaan sistem secara umum, sehingga masalah tersebut dapat segera diidentifikasi dan ditangani.
6. Hasil Pengujian: Jika smoke testing berhasil, artinya sistem melewati pengujian dengan baik dan siap untuk pengujian yang lebih mendalam. Jika ada kesalahan atau masalah yang muncul selama pengujian, sistem mungkin perlu diperbaiki sebelum melanjutkan pengujian lebih lanjut.
7. Automasi: Smoke testing dapat diotomatiskan menggunakan alat pengujian otomatis. Ini memungkinkan pengujian dilakukan secara otomatis setiap kali terjadi perubahan atau pembaruan dalam sistem. Automasi dapat mempercepat proses pengujian dan memastikan pengujian yang konsisten dan terulang.

Smoke testing adalah langkah awal yang penting dalam pengujian perangkat lunak yang membantu memverifikasi bahwa versi terbaru dari aplikasi atau sistem dapat berjalan dengan baik setelah perubahan atau pembaruan. Dengan melakukan smoke testing secara rutin, masalah serius yang dapat menghambat fungsi dasar sistem dapat segera diidentifikasi dan ditangani sejak dini.

10.10 Stress Testing

Stress testing adalah jenis pengujian perangkat lunak yang dilakukan untuk menguji batas kinerja sistem di bawah beban atau tekanan yang sangat tinggi. Tujuan utama dari stress testing adalah untuk mengidentifikasi batasan-batasan sistem, melihat bagaimana sistem merespons dalam situasi yang ekstrem, dan memastikan bahwa sistem tetap stabil dan berkinerja baik di bawah tekanan yang tinggi. (Romeo, 2003)

Berikut ini adalah beberapa hal yang perlu diperhatikan dalam stress testing:

1. Tujuan dan Skenario Pengujian: Tujuan dari stress testing adalah untuk menguji batas-batas kinerja sistem dan melihat bagaimana sistem berperilaku dalam situasi beban tinggi. Skenario pengujian ditentukan untuk mensimulasikan beban ekstrem pada sistem, misalnya dengan meningkatkan jumlah pengguna secara signifikan, meningkatkan lalu lintas transaksi, atau memperpanjang durasi pengujian.
2. Beban Pengujian: Dalam stress testing, beban atau tekanan yang diberikan pada sistem meningkat secara bertahap atau dalam pola yang ditentukan untuk menguji batas kinerja sistem. Beban ini dapat mencakup lalu lintas data yang tinggi, jumlah pengguna yang besar, permintaan yang intensif, atau kondisi sumber daya yang terbatas.
3. Pengujian Durasi: Stress testing melibatkan menjalankan pengujian dalam jangka waktu yang cukup lama untuk melihat bagaimana sistem menangani tekanan yang berkelanjutan. Pengujian durasi ini membantu mengidentifikasi apakah ada masalah seperti kebocoran memori atau penumpukan sumber daya yang terjadi seiring berjalannya waktu.
4. Mengukur Kinerja: Selama stress testing, berbagai metrik kinerja diukur untuk melihat bagaimana sistem merespons beban yang tinggi. Metrik ini dapat mencakup waktu respon, throughput, penggunaan CPU dan memori, kecepatan pemrosesan transaksi, dan lain-lain. Dengan mengukur kinerja, dapat diidentifikasi batas-batas dan masalah yang mungkin muncul saat sistem berada di bawah tekanan.
5. Monitoring dan Analisis: Selama stress testing, pemantauan sistem dilakukan untuk mengumpulkan data performa dan penggunaan sumber daya secara real-time. Data ini dianalisis untuk melihat bagaimana sistem berperilaku dan mengidentifikasi bottleneck atau titik kelemahan dalam

sistem. Analisis ini membantu dalam menemukan masalah dan memperbaikinya.

6. Debugging dan Perbaikan: Jika masalah atau kegagalan ditemukan selama stress testing, langkah-langkah debugging dilakukan untuk mengidentifikasi penyebab masalah. Setelah penyebabnya diketahui, langkah-langkah perbaikan diambil untuk mengatasi masalah dan meningkatkan kinerja sistem di bawah tekanan.
7. Pengujian Skala: Stress testing seringkali melibatkan pengujian skala, di mana sistem diberikan beban yang melebihi batas kapasitas normalnya. Tujuannya adalah untuk melihat bagaimana sistem merespons dan berperilaku saat mencapai batas kapasitas maksimumnya. Pengujian skala membantu mengidentifikasi batas-batas dan kelemahan dalam sistem.
8. Dokumentasi dan Pelaporan: Hasil pengujian, temuan, dan rekomendasi perbaikan dicatat dan didokumentasikan dalam laporan pengujian. Laporan ini berisi detail pengujian, hasil kinerja, masalah yang ditemukan, langkah-langkah perbaikan, dan rekomendasi untuk meningkatkan kinerja sistem di bawah tekanan.

Stress testing penting dalam memastikan bahwa sistem dapat beroperasi secara stabil dan efektif bahkan dalam situasi beban yang tinggi. Dengan melakukan stress testing secara menyeluruh, dapat diidentifikasi batas-batas kinerja sistem, mengidentifikasi masalah terkait kinerja, dan mengambil langkah-langkah perbaikan yang diperlukan untuk meningkatkan stabilitas dan responsivitas sistem.

10.11 User Acceptence Test

User Acceptance Testing (UAT) adalah jenis pengujian perangkat lunak yang dilakukan oleh pengguna atau pihak yang memiliki kepentingan langsung dalam sistem. Tujuan dari UAT

adalah untuk memverifikasi apakah sistem atau aplikasi yang dikembangkan memenuhi persyaratan bisnis, kebutuhan pengguna, dan kriteria penerimaan yang telah ditetapkan sebelumnya. (Romeo, 2003)

Berikut ini adalah beberapa hal yang perlu diperhatikan dalam *User Acceptance Testing*:

1. Pengguna yang Terlibat: UAT melibatkan pengguna akhir atau pemangku kepentingan yang memiliki pengetahuan mendalam tentang kebutuhan bisnis dan proses yang terlibat. Pengguna ini adalah orang-orang yang akan menggunakan sistem secara aktif dan memiliki pemahaman yang baik tentang tujuan, fungsi, dan harapan dari sistem yang akan diuji.
2. Persiapan Kasus Uji: Kasus uji atau test cases dikembangkan berdasarkan persyaratan bisnis dan skenario penggunaan yang telah ditentukan sebelumnya. Kasus uji ini mencakup serangkaian tugas atau skenario yang mencerminkan situasi nyata yang akan dihadapi oleh pengguna saat menggunakan sistem.
3. Lingkungan Pengujian: Lingkungan pengujian yang mencerminkan lingkungan produksi atau lingkungan operasional yang nyata disiapkan untuk menjalankan UAT. Ini mencakup perangkat keras, perangkat lunak, database, dan infrastruktur yang dibutuhkan untuk menjalankan sistem dengan benar.
4. Proses Pengujian: UAT dilakukan dengan mengikuti skenario kasus uji yang telah ditetapkan. Pengguna atau pemangku kepentingan menjalankan sistem dan mengikuti langkah-langkah yang telah ditentukan dalam kasus uji. Mereka mencatat hasil dan pengamatan mereka selama pengujian.
5. Penerimaan atau Penolakan: Setelah pengujian selesai, hasil UAT dievaluasi. Jika sistem memenuhi persyaratan dan ekspektasi pengguna, sistem diterima dan siap untuk digunakan. Namun, jika masalah atau kegagalan ditemukan,

sistem ditolak dan tim pengembangan diminta untuk melakukan perbaikan sebelum pengujian ulang.

6. Pemberian Umpan Balik: Pengguna atau pemangku kepentingan memberikan umpan balik terperinci tentang pengalaman mereka dalam menggunakan sistem. Umpan balik ini mencakup kepuasan mereka terhadap sistem, kegunaan, performa, dan kepatuhan terhadap persyaratan bisnis. Umpan balik ini berfungsi sebagai masukan berharga untuk perbaikan dan pengembangan sistem di masa depan.
7. Batas Waktu dan Anggaran: UAT dilakukan dalam batas waktu dan anggaran yang ditetapkan sebelumnya. Tujuannya adalah untuk memastikan bahwa pengujian dilakukan secara efisien dan efektif, dengan memperhatikan kebutuhan dan batasan yang ada.
8. Koordinasi dengan Tim Pengembangan: Tim pengujian berkoordinasi dengan tim pengembangan selama UAT untuk memastikan pemahaman yang jelas tentang persyaratan bisnis, perbaikan yang diperlukan, dan rencana tindak lanjut.
9. Pelaporan Hasil: Hasil UAT, termasuk temuan, kesimpulan, dan rekomendasi, dicatat dan dilaporkan. Laporan ini mencakup detail hasil pengujian, masalah yang ditemukan, dan rekomendasi perbaikan untuk memastikan bahwa sistem memenuhi kebutuhan pengguna dan persyaratan bisnis.

User Acceptance Testing penting untuk memastikan bahwa sistem atau aplikasi memenuhi harapan pengguna dan persyaratan bisnis yang telah ditetapkan sebelumnya. Dengan melibatkan pengguna atau pemangku kepentingan secara langsung, UAT membantu mengidentifikasi masalah atau ketidaksesuaian yang mungkin tidak terdeteksi dalam pengujian lainnya. Hasil dari UAT digunakan untuk mengevaluasi kelayakan sistem sebelum diluncurkan secara penuh dan memberikan keyakinan bahwa sistem siap digunakan oleh pengguna akhir

DAFTAR PUSTAKA

Jorgensen, P.C. 2014. 'System Testing', *Software Testing*, pp. 251–280. Available at: https://doi.org/10.1201/b16592-23.

Romeo. 2003. 'T e s t i n g d a n i m p l e m e n t a s i s i s t e m', *Testing dan Implementasi Sistem*, p. 52.

BAB 11
PEMELIHARAAN SISTEM

Oleh Melissa Indah Fianty

11.1 Pendahuluan

Dalam era pasar yang kompetitif dan ekonomi yang selalu berubah, organisasi di sektor manufaktur dan jasa menghadapi tekanan intens untuk merespons permintaan pelanggan dan memberikan produk berkualitas tinggi secara tepat waktu. Hal ini mendorong organisasi untuk mengarahkan strategi mereka pada kecepatan, kualitas, otomatisasi, dan kinerja tinggi. Hal ini menghasilkan investasi yang sangat besar dalam peralatan dan tenaga kerja. Untuk mencapai tingkat pengembalian investasi yang ditargetkan dan bertahan dalam ekonomi yang dinamis ini, peralatan harus dapat diandalkan dan aman digunakan tanpa adanya henti kerja yang mahal dan perbaikan. Banyak perusahaan manufaktur telah mengadopsi program lean just-in-time (JIT) dan beroperasi dengan tingkat persediaan dalam proses yang sangat rendah sehingga tidak ada cadangan persediaan yang dapat digunakan jika terjadi ketidaktersediaan peralatan. Perkembangan di atas menyoroti peran pemeliharaan sebagai kegiatan utama dalam organisasi manufaktur dan jasa.

Pendekatan sistem holistik untuk pemeliharaan dan kaitannya dengan strategi dan tujuan organisasi. Pendekatan ini memungkinkan para pengambil keputusan pemeliharaan untuk memandang pemeliharaan sebagai penyedia keunggulan kompetitif bukan sebagai kejahatan yang diperlukan. Konsekuensi adalah bahwa pemeliharaan dipandang sebagai sistem terintegrasi dengan tujuan, strategi, proses, dan aktivitas yang perlu diselaraskan dengan tujuan organisasi, direncanakan, dirancang,

dikelola, dan dikendalikan menggunakan teknik statistik dan optimisasi. Bab ini memberikan gambaran umum dan menyoroti fitur-fitur kunci dari buku ini. Selain itu, bab ini mempresentasikan komponen-komponen dari sistem pemeliharaan yang perlu direncanakan, diorganisir, dan dioptimalkan, untuk memaksimalkan output dari sistem pemeliharaan dan mencapai tujuan organisasi dengan penggunaan sumber daya yang terbaik.

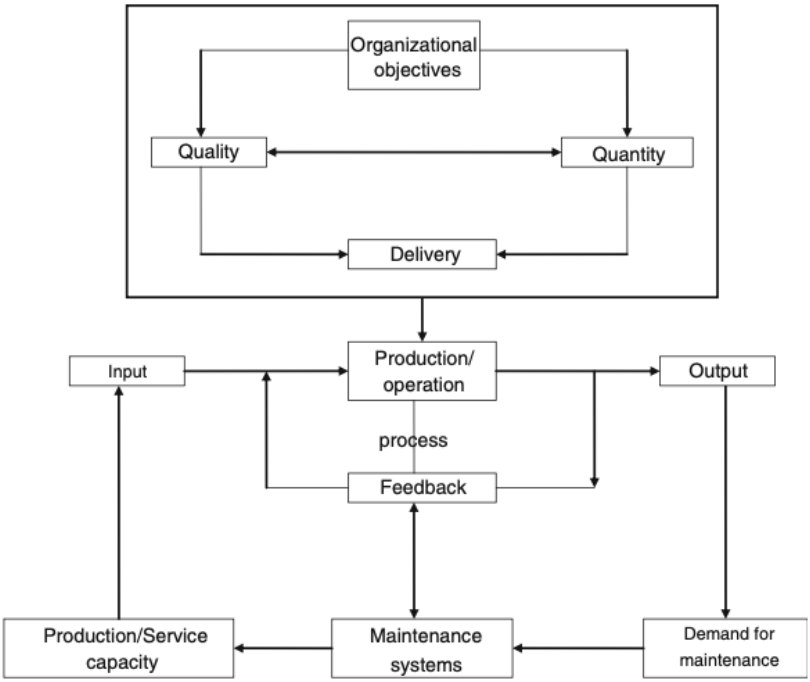
11.2 Pendekatan Pemeliharaan Sistem

Pemeliharaan didefinisikan sebagai kombinasi kegiatan yang dilakukan untuk menjaga atau mengembalikan peralatan atau sistem ke kondisi di mana mereka dapat menjalankan fungsi yang ditentukan. Pemeliharaan merupakan faktor penting dalam kualitas produk dan layanan, dan dapat digunakan sebagai strategi untuk bersaing dengan sukses. Ketidaksesuaian dalam operasi peralatan produksi menghasilkan variabilitas yang berlebihan dalam produk dan dengan demikian menyebabkan output yang cacat. Untuk dapat menghasilkan tingkat kualitas yang tinggi, peralatan produksi atau layanan harus beroperasi sesuai dengan spesifikasi yang dapat dicapai melalui tindakan pemeliharaan tepat waktu.

Sebuah sistem adalah kumpulan komponen yang bekerja bersama untuk mencapai tujuan bersama. Pemeliharaan dapat dianggap sebagai sebuah sistem dengan serangkaian proses dan kegiatan yang dilakukan sejalan dengan sistem produksi atau layanan. Hubungan diagramatis antara tujuan organisasi, proses produksi/operasi, dan pemeliharaan ditunjukkan dalam Gambar 11.1. Sistem produksi/operasi biasanya berfokus pada mengubah input seperti bahan baku, tenaga kerja, dan proses menjadi produk/layanan yang memenuhi kebutuhan pelanggan. Output utama dari sistem produksi/operasi adalah produk/layanan jadi, dan output sekundernya adalah peralatan yang rusak atau gagal. Output sekunder ini menciptakan permintaan untuk pemeliharaan.

Sistem pemeliharaan mengambil hal ini sebagai input dan menambahkannya dengan pengetahuan, tenaga kerja, dan suku cadang, dan menghasilkan peralatan/fasilitas dalam kondisi operasi yang baik, yang menyediakan kapasitas untuk produksi atau layanan.

Tujuan utama secara keseluruhan dari sistem produksi atau sistem layanan adalah memaksimalkan keuntungan dari peluang pasar yang tersedia, dan tujuan sekundernya berkaitan dengan aspek ekonomi dan teknis dari proses konversi. Sistem pemeliharaan membantu mencapai tujuan ini juga, dengan meningkatkan keuntungan dan kepuasan pelanggan. Hal ini dicapai dengan meminimalkan waktu henti pabrik, meningkatkan kualitas, meningkatkan produktivitas, dan dengan pengiriman pesanan yang andal dan tepat waktu kepada pelanggan. Sistem produksi dan layanan telah dioptimalkan sebagai sistem terintegrasi dan telah diteliti cukup ekstensif dibandingkan dengan sistem pemeliharaan. Beberapa alasan mungkin termasuk: (1) Secara tradisional, pemeliharaan dianggap sebagai kejahatan yang diperlukan dan pada dasarnya merupakan sistem yang didorong oleh produksi, (2) pemeliharaan dalam sebuah organisasi memiliki hubungan yang kompleks dengan fungsi lain, dan (3) output dari pemeliharaan sulit diukur dan dikuantifikasi.

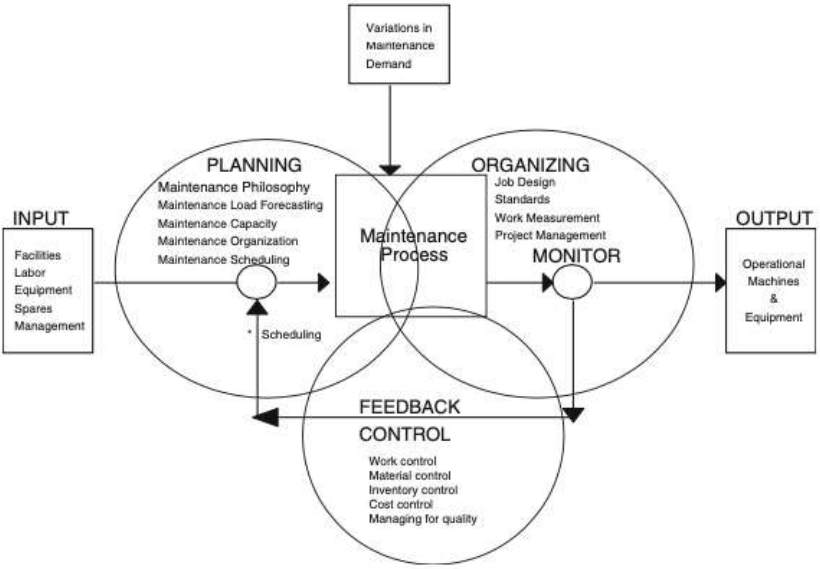


Gambar 11.1. Hubungan antara sistem pemeliharaan dan tujuan organisasi
(sumber: Salih O. Duffuaa, 2015)

Peran sistem pemeliharaan telah lama diakui dalam organisasi manufaktur; namun, telah menjadi jelas bahwa fungsi-fungsi pemeliharaan juga penting untuk organisasi layanan seperti rumah sakit.

Sistem pemeliharaan dapat dipandang sebagai model input-output terintegrasi. Input dalam model ini meliputi tenaga kerja, manajemen, alat, suku cadang, peralatan, dan sebagainya, sedangkan outputnya adalah peralatan yang berfungsi, handal, dan dikonfigurasi dengan baik untuk mencapai operasi yang direncanakan dalam pabrik. Hal ini memungkinkan kita untuk mengoptimalkan sumber daya guna memaksimalkan output dari

sistem pemeliharaan. Sebuah sistem pemeliharaan yang khas ditunjukkan dalam Gambar 11.2. Kegiatan-kegiatan yang diperlukan untuk menjadikan sistem ini efisien dan efektif termasuk perencanaan, perancangan, pengorganisasian, pengendalian, dan peningkatan yang ditunjukkan dalam gambar tersebut.



Gambar 11.2. Pemeliharaan Sistem
(Sumber: Salih O. Duffuaa, 2015)

11.3 Kegiatan Perencanaan Pemeliharaan Sistem

Kegiatan perencanaan umumnya meliputi hal-hal berikut:

1. Strategic systems alliances
2. Maintenance strategies
3. Maintenance load forecasting
4. Maintenance capacity
5. Maintenance organization
6. Maintenance scheduling.

Deskripsi dari masing-masing kegiatan ini diberikan pada subbagian berikutnya:

11.3.1 Strategic systems alliances

Departemen pemeliharaan harus memiliki rencana strategis sendiri yang selaras dengan tujuan strategis organisasi mereka dan dikelola dengan maksimal efisiensi dan efektivitas dengan memperhatikan pendekatan sistem holistik. Strategi untuk operasi pemeliharaan harus dipilih di antara alternatif-alternatif untuk mencapai tujuan organisasi. Masalah terkait perencanaan strategis pemeliharaan dan mengidentifikasi beberapa isu penting yang menjadi faktor penentu dalam memutuskan rencana strategis pemeliharaan. Isu-isu ini meliputi pemilihan pihak ketiga untuk pemeliharaan, organisasi, metodologi, dan dukungan. Pada tingkat strategis, manajemen menentukan tujuan strategis organisasi, dan dari tujuan-tujuan ini, tujuan unit pemeliharaan diturunkan untuk mendukung tujuan dan misi organisasi. Kemudian, organisasi yang tepat, metodologi pemeliharaan, dan dukungan dipilih untuk mencapai tujuan tersebut. Metodologi pemeliharaan adalah strategi yang akan digunakan pada tingkat pemeliharaan peralatan, dan dukungan meliputi proses utama pemeliharaan, tenaga kerja, sistem informasi, pelatihan, dan sistem manajemen kinerja dan penghargaan.

11.3.2 Maintenance strategies

Perencanaan pemeliharaan pada tingkat strategis yang termasuk dalam perencanaan jangka panjang. Di masa lalu, pemeliharaan tidak dianggap sebagai unit strategis dalam organisasi, tetapi belakangan ini hal ini telah berubah dan para peneliti serta pengambil keputusan pemeliharaan telah mengemukakan peran strategis pemeliharaan. Perencanaan strategis pemeliharaan adalah proses memastikan keselarasan misi, tujuan, dan program pemeliharaan dengan misi dan tujuan organisasi. Beberapa isu penting yang diperlukan dalam

menentukan rencana strategis pemeliharaan. Isu-isu ini meliputi mode penyampaian pemeliharaan, organisasi, metodologi, dan dukungan (Murthy et al., 2002). Berikut ini adalah sepuluh langkah yang disarankan untuk menerapkan kerangka yang diusulkan dalam literatur terbaru:

1. Manajemen puncak mengembangkan misi organisasi, tujuan strategis, dan objektif.
2. Manajemen pemeliharaan meninjau misi dan objektif organisasi dan mengidentifikasi peran pemeliharaan dalam pencapaian mereka.
3. Manajemen pemeliharaan menganalisis situasi internal dan eksternal terkini yang terkait dengan fungsi pemeliharaan menggunakan metodologi untuk menentukan kekuatan, kelemahan, peluang, dan ancaman yang dikenal sebagai analisis SWOT. SWOT mengarah pada identifikasi isu strategis yang perlu diatasi. Isu-isu strategis ini diharapkan mencakup mode penyampaian, organisasi pemeliharaan, metodologi, pengaturan tenaga kerja, dan pelatihan.
4. Manajemen pemeliharaan merumuskan peran yang diidentifikasi sebagai pernyataan misi untuk pemeliharaan dan mengembangkan tujuan strategis dan objektif untuk mengatasi isu-isu strategis tersebut.
5. Menyelaraskan dan memberikan prioritas pada tujuan strategis, objektif, dan program pemeliharaan dengan misi pemeliharaan.
6. Manajemen pemeliharaan memeriksa keselarasan tujuan dan objektif strategis pemeliharaan dengan misi, tujuan, dan objektif strategis organisasi yang ditetapkan.
7. Manajemen pemeliharaan mengembangkan program pemeliharaan untuk mencapai tujuan dan objektif strategis yang ditetapkan.
8. Mengembangkan seperangkat ukuran kinerja kuantitatif untuk tujuan dan objektif strategis pemeliharaan.

9. Melakukan evaluasi secara berkala terhadap kemajuan dalam mencapai tujuan dan objektif menggunakan ukuran kinerja, serta mengidentifikasi kesenjangan antara situasi aktual dan yang diinginkan.
10. Mengidentifikasi penyebab akar dari kesenjangan dan melaksanakan tindakan korektif yang efektif.

Metodologi pemeliharaan berasal dari tujuan sistem pemeliharaan yang selaras dengan misi dan tujuan strategis organisasi. Untuk menerapkan metodologi tersebut, strategi-strategi berikut dapat memainkan peran yang efektif jika diterapkan dalam campuran dan cara yang tepat. Strategi-strategi tersebut adalah sebagai berikut:

1. Pemeliharaan perbaikan/perbaikan ketika terjadi kerusakan
   Jenis pemeliharaan ini hanya dilakukan ketika peralatan tidak mampu beroperasi lebih lanjut. Tidak ada unsur perencanaan untuk jenis pemeliharaan ini. Ini terjadi ketika biaya tambahan dari jenis pemeliharaan lain tidak dapat dibenarkan. Strategi jenis ini kadang-kadang disebut strategi run to failure. Hal ini umumnya berlaku untuk komponen elektronik.
2. Pemeliharaan preventif (Pemeliharaan preventif berdasarkan waktu atau penggunaan dan Pemeliharaan preventif berdasarkan kondisi)
   Pemeliharaan preventif (PM) adalah segala bentuk pemeliharaan yang direncanakan untuk mengatasi potensi kegagalan. Pemeliharaan ini dapat dilakukan berdasarkan penggunaan atau kondisi peralatan. Pemeliharaan preventif berbasis waktu atau penggunaan dilakukan berdasarkan jam operasi atau kalender. Ini membutuhkan tingkat perencanaan yang tinggi. Rutinitas khusus yang akan dilakukan sudah diketahui beserta frekuensinya. Dalam menentukan frekuensi, biasanya diperlukan pengetahuan tentang distribusi kegagalan atau keandalan peralatan.

Pemeliharaan preventif berbasis kondisi dilakukan berdasarkan kondisi diketahui dari peralatan. Kondisi peralatan ditentukan dengan memantau parameter-parameter kunci peralatan yang nilainya dipengaruhi oleh kondisi peralatan. Strategi ini juga dikenal sebagai pemeliharaan prediktif.

3. Pemeliharaan peluang
   Pemeliharaan peluang dilakukan sesuai dengan namanya, yaitu ketika kesempatan muncul. Kesempatan semacam itu bisa muncul selama periode shutdown sistem tertentu dan dapat dimanfaatkan untuk melakukan tugas pemeliharaan yang sudah diketahui.
4. Pencarian kesalahan
   Fault finding adalah tindakan atau pemeriksaan yang dilakukan untuk menilai tingkat kegagalan yang terjadi. Contoh dari fault finding adalah memeriksa ban cadangan mobil sebelum melakukan perjalanan jauh.
5. Modifikasi desain
   Modifikasi desain dilakukan untuk membawa peralatan ke kondisi yang diterima saat ini. Ini melibatkan perbaikan dan kadang-kadang perluasan kapasitas dan manufaktur. Biasanya membutuhkan koordinasi dengan departemen teknik dan departemen lain dalam organisasi untuk melakukan jenis pekerjaan ini.
6. Overhaul (perbaikan besar)
   Overhaul adalah pemeriksaan menyeluruh dan pemulihan peralatan atau perbaikan besar peralatan ke kondisi yang dapat diterima. Ini biasanya merupakan tugas besar.
7. Penggantian
   Setiap strategi pemeliharaan yang telah dijelaskan di atas memiliki peran dalam operasi pabrik. Kombinasi optimal dari strategi pemeliharaan ini menghasilkan filosofi pemeliharaan yang paling efektif. Ukuran pabrik dan tingkat operasi yang

direncanakan bersama dengan strategi pemeliharaan yang diterapkan dapat membantu dalam memperkirakan beban pemeliharaan atau output yang diinginkan dari sistem pemeliharaan.

Beberapa isu strategis dalam pemeliharaan sistem, yaitu: Isu pertama adalah mode pengiriman. Isu ini berkaitan dengan apakah pemeliharaan dioutsourcing, dilakukan di dalam perusahaan, atau kombinasi keduanya. Setiap pilihan memiliki kelebihan dan kekurangan. Pemilihan opsi yang tepat harus dilakukan dengan mempertimbangkan tujuan dan target strategis pemeliharaan yang telah ditetapkan sekaligus untuk meminimalkan risiko dan ancaman terhadap efisiensi dan efektivitas sistem pemeliharaan. Isu strategis pemeliharaan kedua adalah organisasi pemeliharaan. Beberapa opsi tersedia dalam isu ini, termasuk fungsional, proses, atau jaringan. Setiap opsi organisasi sebelumnya dapat dikombinasikan dengan opsi terpusat, terdesentralisasi, atau opsi kaskade. Isu ketiga adalah struktur dan pengendalian kerja. Banyak opsi yang tersedia untuk struktur kerja tergantung pada struktur pelaporan dan pengawasan. Isu strategis pemeliharaan keempat adalah pemilihan sistem pendukung yang mencakup sistem informasi, pelatihan, dan manajemen kinerja serta sistem penghargaan. Setiap elemen harus dipilih dengan hati-hati untuk mendukung tujuan keseluruhan organisasi.

11.3.3 Maintenance load forecasting

Peramalan beban pemeliharaan adalah proses untuk memprediksi beban pemeliharaan. Beban pemeliharaan di suatu pabrik tertentu bervariasi secara acak dan, di antara faktor lain, dapat dipengaruhi oleh usia peralatan, tingkat penggunaannya, kualitas pemeliharaan, faktor iklim, dan keterampilan para pekerja pemeliharaan. Peramalan beban pemeliharaan sangat penting

untuk mencapai tingkat efektivitas dan pemanfaatan sumber daya yang diinginkan, dan tanpa itu, banyak fungsi pemeliharaan tidak dapat dilakukan dengan baik (Lee et al., 2006).

Sebelum melakukan perencanaan kapasitas atau merancang organisasi pemeliharaan baru, penting untuk memiliki beberapa perkiraan tentang beban pemeliharaan yang diharapkan. Beban tersebut terdiri dari hal-hal berikut:

1. Beban pemeliharaan darurat. Ini dapat diprediksi menggunakan beban kerja historis aktual dan teknik prediksi yang tepat dan/atau pengalaman manajemen. Komponen beban ini bersifat acak dan dapat diminimalkan dengan memiliki pemeliharaan terencana yang dirancang dengan baik.
2. Beban pemeliharaan preventif. Ini dapat diprediksi menggunakan catatan historis aktual yang dikombinasikan dengan program pemeliharaan preventif yang baru dikembangkan. Ini harus mencakup inspeksi rutin dan pelumasan.
3. Pemeliharaan korektif tertunda. Ini dapat diprediksi berdasarkan catatan historis dan rencana masa depan.
4. Perkiraan untuk perbaikan dan fabrikasi suku cadang yang dihapus dan direvisi. Ini dapat diestimasi dari catatan historis yang dikombinasikan dengan rencana masa depan untuk perbaikan.
5. Penghentian operasi, perawatan besar, dan modifikasi desain. Ini dapat diprediksi berdasarkan catatan historis aktual dan jadwal pemeliharaan masa depan.

11.3.4 Maintenance capacity

Perencanaan kapasitas pemeliharaan menentukan sumber daya yang diperlukan untuk memenuhi permintaan pekerjaan pemeliharaan. Sumber daya tersebut meliputi tenaga kerja, bahan, suku cadang, peralatan, dan alat. Aspek penting dari kapasitas pemeliharaan adalah jumlah dan keterampilan para pekerja, alat

pemeliharaan yang diperlukan, dan lain sebagainya. Karena beban pemeliharaan merupakan variabel acak, jumlah tepat dari berbagai jenis pekerja tidak dapat ditentukan. Oleh karena itu, tanpa perkiraan yang cukup akurat untuk permintaan pekerjaan pemeliharaan di masa depan, tidak mungkin melakukan perencanaan kapasitas jangka panjang yang tepat. Untuk memanfaatkan tenaga kerja secara lebih baik, organisasi cenderung mengurangi jumlah pekerja yang tersedia dibandingkan dengan kebutuhan yang diharapkan. Hal ini kemungkinan akan mengakibatkan tertundanya pekerjaan pemeliharaan yang belum selesai. Hal ini dapat diselesaikan dengan membiarkan para pekerja yang ada bekerja lembur atau dengan mencari bantuan kontraktor dari luar. Tertundanya pekerjaan pemeliharaan ini juga dapat diselesaikan ketika beban pemeliharaan lebih rendah dari kapasitas. Sebenarnya ini adalah alasan utama untuk mempertahankan pekerjaan yang tertunda. Membuat estimasi jangka panjang adalah salah satu area dalam perencanaan kapasitas pemeliharaan yang kritis dan belum terlalu berkembang.

Perencanaan kapasitas pemeliharaan menentukan tingkat optimal sumber daya (pekerja, keterampilan, suku cadang, peralatan persediaan, dan alat) untuk memenuhi beban pemeliharaan yang diproyeksikan yang terdiri dari perkiraan beban masa depan ditambah backlog pemeliharaan. Elemen penting dari perencanaan kapasitas adalah penentuan keterampilan tukang, jumlah yang tepat dari berbagai jenis tukang, tingkat backlog yang sehat, kapasitas lembur, dan pemeliharaan kontrak. Pengalokasian optimal sumber daya pemeliharaan untuk memenuhi beban kerja yang acak dan bervariasi merupakan masalah yang kompleks dan menantang. Teknik perencanaan kapasitas memainkan peran sangat penting dalam menangani masalah kompleks ini (Al-Turki, 2011). Langkah-langkah yang terlibat dalam perencanaan kapasitas dapat dirangkum sebagai berikut:

1. Tentukan total beban pemeliharaan.

2. Perkirakan suku cadang dan material yang diperlukan untuk memenuhi beban tersebut.
3. Tentukan peralatan dan alat yang diperlukan untuk semua jenis pekerjaan pemeliharaan.
4. Tentukan keterampilan dan jumlah tukang dari setiap keterampilan. Perhatian khusus harus diberikan pada tukang dengan keterampilan ganda.
5. Terkadang pada peralatan yang sangat terkomputerisasi, mungkin perlu menyediakan keahlian khusus yang memerlukan rencana khusus.

Dalam perencanaan kapasitas pemeliharaan, isu utama adalah menentukan campuran optimal keterampilan tukang dari sumber yang tersedia bagi organisasi. Sumber yang biasa tersedia adalah kru internal reguler dan lembur serta pemeliharaan kontrak. Campuran terbaik menggunakan sumber-sumber ini ditentukan dengan menggunakan biaya dan ukuran ketersediaan.

Multi-skill memberikan opsi dan alternatif kepada perencana kapasitas yang memperkaya dan menghasilkan berbagai pilihan untuk menentukan kapasitas optimal. Tukang dengan multi-skill dapat digunakan dalam lebih dari satu jenis pekerjaan pemeliharaan. Hal ini biasanya meningkatkan pemanfaatan tenaga kerja.

Teknik perencanaan kapasitas dapat dibagi menjadi dua kategori utama: deterministik dan stokastik. Pendekatan deterministik mengasumsikan bahwa beban pemeliharaan yang diproyeksikan, waktu standar, dan variabel acak lainnya adalah konstanta tetap. Kemudian, sebuah rencana yang meminimalkan biaya atau memaksimalkan ketersediaan tergantung pada tujuan organisasi yang mungkin mencakup keandalan, ketersediaan, dan biaya ditentukan. Dalam bab ini, dua teknik deterministik akan disajikan, yaitu:

1. Metode tabel heuristik, dan
2. Pemrograman linier.

Pendekatan stokastik memodelkan beban pemeliharaan, waktu standar, waktu kedatangan pekerjaan, dan variabel lainnya sebagai variabel acak dengan distribusi probabilitas tertentu dan menggunakan teknik statistik standar untuk mengidentifikasi distribusi-distribusi ini. Kemudian, model stokastik digunakan untuk menentukan kapasitas optimal yang memenuhi beban pemeliharaan. Dua teknik stokastik akan disajikan dalam bab ini, yaitu:

1. Model antrian, dan
2. simulasi peristiwa diskret stokastik.

11.3.5 Maintenance organization

Banyak faktor yang memengaruhi organisasi pemeliharaan. Faktor-faktor tersebut meliputi ukuran pabrik, beban pemeliharaan, jenis organisasi, dan keterampilan para pekerja. Berdasarkan faktor-faktor ini dan lainnya, pemeliharaan dapat diorganisir secara departemental, area, atau pusat. Setiap jenis organisasi memiliki kelebihan dan kekurangannya masing-masing. Pada organisasi besar, desentralisasi pemeliharaan dapat menghasilkan waktu respons yang lebih cepat dan memungkinkan para pekerja menjadi lebih berpengalaman dalam menghadapi masalah di bagian tertentu dari pabrik. Namun, pembentukan sejumlah unit kecil cenderung mengurangi fleksibilitas sistem pemeliharaan secara keseluruhan. Rentang keterampilan yang tersedia menjadi terbatas, dan pemanfaatan tenaga kerja biasanya kurang daripada unit pemeliharaan yang terpusat. Dalam beberapa kasus, solusi kompromi yang disebut sistem cascading mungkin memungkinkan. Sistem ini memungkinkan unit pemeliharaan area produksi terhubung dengan unit pemeliharaan pusat (Duffuaa & Raouf, 2015).

11.3.6 Maintenance scheduling.

Penjadwalan pemeliharaan adalah proses penugasan sumber daya dan tenaga kerja untuk pekerjaan yang harus diselesaikan pada waktu tertentu. Hal ini diperlukan untuk memastikan bahwa para pekerja terampil, suku cadang, dan bahan yang diperlukan tersedia sebelum tugas pemeliharaan dapat dijadwalkan. Peralatan kritis dalam sebuah pabrik adalah peralatan yang kegagalannya akan menghentikan proses produksi atau membahayakan kehidupan dan keselamatan manusia. Pekerjaan pemeliharaan yang berkaitan dengan peralatan tersebut diperlakukan dengan prioritas dan ditangani sebelum pekerjaan lain dilakukan. Kejadian pekerjaan semacam itu tidak dapat diprediksi dengan pasti, sehingga jadwal untuk pemeliharaan terencana dalam kasus-kasus ini harus direvisi. Efektivitas sistem pemeliharaan sangat dipengaruhi oleh jadwal pemeliharaan yang dikembangkan dan kemampuannya untuk menyesuaikan perubahan. Tingkat efektivitas jadwal pemeliharaan yang tinggi mengindikasikan tingkat efektivitas pemeliharaan yang tinggi.

Sebuah jadwal pemeliharaan harus dikembangkan untuk setiap peralatan dalam program tersebut. Jadwal tersebut adalah daftar komprehensif dari tugas-tugas pemeliharaan yang akan dilakukan pada peralatan. Jadwal tersebut mencakup nama dan nomor identifikasi peralatan, lokasi, nomor referensi jadwal, daftar rinci tugas-tugas yang akan dilakukan (pemeriksaan, pemeliharaan pencegahan, penggantian), frekuensi setiap tugas, tukang yang dibutuhkan untuk melaksanakan tugas, waktu untuk setiap tugas, alat khusus yang dibutuhkan, material yang dibutuhkan, dan rincian tentang pengaturan pemeliharaan kontrak apapun.

Penjadwalan pemeliharaan adalah proses di mana pekerjaan disesuaikan dengan sumber daya dan diurutkan untuk dilaksanakan pada titik waktu tertentu. Sebuah jadwal yang dapat diandalkan harus mempertimbangkan hal-hal berikut:

1. Peringkat prioritas pekerjaan yang mencerminkan urgensi dan kritikalitas pekerjaan.
2. Apakah semua material yang dibutuhkan untuk work order tersedia di pabrik. (Jika tidak, work order tidak boleh dijadwalkan.)
3. Jadwal induk produksi dan koordinasi yang erat dengan operasi.
4. Estimasi yang realistis dan apa yang kemungkinan akan terjadi daripada apa yang diinginkan oleh penjadwal.
5. Fleksibilitas harus dibangun dalam jadwal. Penjadwal harus menyadari bahwa fleksibilitas diperlukan terutama dalam pemeliharaan. Jadwal sering direvisi dan diperbarui.

Jadwal pemeliharaan dapat disusun dalam tiga tingkat tergantung pada jangka waktu jadwal. Tingkat-tingkat tersebut adalah (1) jadwal jangka panjang atau jadwal utama yang mencakup periode 3 bulan hingga satu tahun; (2) jadwal mingguan; ini adalah pekerjaan pemeliharaan yang mencakup satu minggu; dan (3) jadwal harian yang mencakup pekerjaan yang harus diselesaikan setiap hari.

Jadwal jangka panjang didasarkan pada work order pemeliharaan yang ada termasuk work order berkala, backlog, pemeliharaan pencegahan, dan pemeliharaan darurat yang diantisipasi. Jadwal ini harus menyeimbangkan permintaan jangka panjang untuk pekerjaan pemeliharaan dengan ketersediaan tenaga kerja. Berdasarkan jadwal jangka panjang, kebutuhan suku cadang dan material dapat diidentifikasi dan dipesan sebelumnya. Jadwal jangka panjang biasanya direvisi dan diperbarui untuk mencerminkan perubahan rencana dan pekerjaan pemeliharaan yang terealisasi.

Jadwal mingguan pemeliharaan dihasilkan dari jadwal jangka panjang dan mempertimbangkan jadwal operasi saat ini dan pertimbangan ekonomi.

Jadwal mingguan harus memperhitungkan sekitar 10-15% dari tenaga kerja yang tersedia untuk pekerjaan darurat. Perencana harus menyediakan jadwal untuk minggu ini dan minggu berikutnya, dengan mempertimbangkan backlog yang tersedia. Work order yang dijadwalkan untuk minggu ini diurutkan berdasarkan prioritas. Analisis jalur kritis dan pemrograman bilangan bulat adalah teknik yang dapat digunakan untuk menghasilkan jadwal. Di sebagian besar perusahaan kecil dan menengah, penjadwalan dilakukan berdasarkan aturan heuristik dan pengalaman.

Jadwal harian dihasilkan dari jadwal mingguan dan biasanya disiapkan sehari sebelumnya. Jadwal ini sering kali terganggu oleh pemeliharaan darurat. Prioritas yang telah ditetapkan digunakan untuk menjadwalkan pekerjaan. Di beberapa organisasi, jadwal diserahkan kepada kepala bagian area dan ia diberikan kebebasan untuk menugaskan pekerjaan kepada tukang-tukangnya dengan syarat bahwa ia harus menyelesaikan pekerjaan sesuai dengan prioritas yang telah ditetapkan.

11.4 Mengorganisir dan Merancang Kegiatan Pemeliharaan Sistem

Mengorganisir dan merancang sistem pemeliharaan mencakup hal-hal berikut:

1. Desain pekerjaan
2. Standar
3. Manajemen proyek.

Sistem pemeliharaan diketahui didorong oleh beban kerja yang dikeluarkan oleh departemen produksi atau operasi sebagai permintaan pekerjaan. Permintaan pekerjaan direncanakan, dieksekusi, dan dikendalikan oleh sistem pesanan kerja. Pesanan kerja menggambarkan pekerjaan, lokasinya, keahlian yang dibutuhkan, dan prioritas pekerjaan.

11.4.1 Desain pekerjaan

Desain pekerjaan, yang berkaitan dengan pekerjaan pemeliharaan, mencakup konten pekerjaan setiap pekerjaan dan menentukan metode yang akan digunakan, alat khusus yang dibutuhkan, dan tenaga terampil yang diperlukan. Jika desainnya sudah terstandarisasi, maka menjadi pekerjaan standar.

11.4.2 Standar

Setelah tugas pemeliharaan melewati tahap desain pekerjaan, penting untuk memperkirakan waktu yang dibutuhkan untuk menyelesaikan pekerjaan tersebut. Standar waktu yang realistis sangat membantu dalam memantau dan meningkatkan efektivitas tenaga ahli, sehingga meminimalkan waktu henti pabrik. Tidak semua pekerjaan pemeliharaan perlu memiliki standar waktu. Diketahui bahwa sekitar dua puluh persen pekerjaan pemeliharaan memakan sekitar delapan puluh persen waktu yang tersedia bagi tenaga kerja pemeliharaan. Upaya harus dilakukan untuk mengembangkan standar waktu untuk pekerjaan yang membutuhkan waktu lama tersebut. Jelaslah bahwa standar waktu pekerjaan diperlukan untuk peramalan beban pemeliharaan, perencanaan kapasitas, dan pengembangan jadwal pemeliharaan.

11.4.3 Manajemen proyek.

Perawatan besar yang direncanakan dan layanan pemeliharaan preventif dilakukan secara berkala di sebagian besar pabrik besar. Selama periode ini, seluruh pabrik atau bagian dari pabrik ditutup. Merencanakan dan mengatur pekerjaan menjadi keuntungan untuk meminimalkan waktu henti dan memanfaatkan sumber daya dengan sebaik-baiknya. Manajemen proyek melibatkan pengembangan jaringan kegiatan dan kemudian menggunakan teknik seperti metode jalur kritis (CPM) atau *program evaluation and review technique* (PERT). Setelah jaringan telah dikembangkan yang mencakup pembagian pekerjaan, urutan

pekerjaan, dan perkiraan waktu untuk setiap kegiatan, perangkat lunak komputer seperti Primavera dapat digunakan untuk menjadwalkan kegiatan dan menentukan pemanfaatan sumber daya yang terbaik. Tahap pengendalian proyek tersebut melibatkan pengukuran kemajuan secara teratur, membandingkannya dengan jadwal, dan menganalisis varian sebagai persentase dari total pekerjaan. Tindakan perbaikan dapat dilakukan untuk mengatasi kekurangan-kekurangan tersebut.

11.5 Mengontrol Kegiatan Pemeliharaan Sistem

Kontrol adalah bagian penting dari manajemen ilmiah. Kontrol yang diterapkan pada sistem pemeliharaan meliputi aktivitas-aktivitas berikut:

1. Pengendalian pekerjaan
2. Pengendalian persediaan
3. Pengendalian biaya
4. Pengendalian kualitas.

11.5.1 Pengendalian pekerjaan

Sistem pemeliharaan didorong oleh permintaan untuk pekerjaan pemeliharaan. Beban kerja pemeliharaan sangat dipengaruhi oleh strategi pemeliharaan. Manajemen dan pengendalian pekerjaan pemeliharaan adalah hal yang penting untuk mencapai rencana yang telah ditetapkan. Sistem perintah kerja adalah alat yang digunakan untuk mengendalikan pekerjaan pemeliharaan. Perintah kerja yang dirancang dengan baik dengan sistem pelaporan yang baik adalah inti dari sistem pemeliharaan.

11.5.2 Pengendalian persediaan

Telah dikatakan sebelumnya bahwa untuk menjadwalkan pekerjaan pemeliharaan, penting untuk memastikan bahwa suku cadang dan material yang diperlukan tersedia. Secara fisik tidak mungkin dan tidak praktis secara ekonomi jika setiap suku cadang

tiba tepat waktu dan di tempat yang dibutuhkan. Oleh karena itu, persediaan dipelihara. Pengendalian persediaan adalah teknik untuk menjaga suku cadang dan material pada tingkat yang diinginkan. Penting untuk menjaga tingkat suku cadang yang optimal, yang meminimalkan biaya penyimpanan barang dan biaya yang dikeluarkan jika suku cadang tidak tersedia. Ini juga memberikan informasi yang diperlukan untuk memastikan ketersediaan suku cadang yang diperlukan untuk pekerjaan pemeliharaan. Jika suku cadang tidak tersedia, maka tindakan harus diambil untuk mengadaan suku cadang dan menginformasikan departemen penjadwalan ketika suku cadang yang dibutuhkan menjadi tersedia.

11.5.3 Pengendalian biaya

Biaya pemeliharaan memiliki banyak komponen, antara lain kehilangan produksi langsung karena pemeliharaan, penurunan kualitas peralatan, biaya backup, dan biaya pemeliharaan berlebih. Pengendalian biaya pemeliharaan merupakan fungsi dari filosofi pemeliharaan, pola operasi, jenis sistem, serta prosedur dan standar yang diadopsi oleh organisasi. Hal ini merupakan komponen utama dalam siklus hidup peralatan.

Pengendalian biaya pemeliharaan mengoptimalkan semua biaya dalam pemeliharaan, sambil mencapai tujuan organisasi yang ditetapkan seperti ketersediaan, "tingkat kualitas," dan ukuran efisiensi dan efektivitas lainnya. Pengurangan dan pengendalian biaya dapat digunakan sebagai keunggulan dalam persaingan dalam menyediakan produk dan layanan.

11.5.4 Pengendalian kualitas.

Dalam proses produksi, kualitas output dapat dianggap sebagai kesesuaian penggunaan dan "kualitas adalah melakukan dengan benar pada kali pertama." Pengendalian kualitas dilakukan dengan mengukur atribut produk atau layanan dan

membandingkannya dengan spesifikasi produk atau layanan yang sama. Pemeliharaan juga dapat dipandang sebagai suatu proses, dan kualitas outputnya dapat dikendalikan.

Dalam kasus pekerjaan pemeliharaan, "melakukan dengan benar pada kali pertama" sangat penting. Kualitas dapat dinilai sebagai persentase pekerjaan pemeliharaan yang diterima sesuai dengan standar yang diadopsi oleh organisasi. Kualitas tinggi biasanya dijamin dengan memeriksa pekerjaan pemeliharaan kritis atau melalui pengawasan pemeliharaan.

11.6 Manajemen Kualitas dan Pelatihan

Manajemen kualitas merupakan tanggung jawab manajerial. Biasanya, manajer/perekayasa pemeliharaan tidak sepenuhnya menyadari pentingnya meningkatkan kualitas produksi pemeliharaan (J. Davidson Frame, 2003). Kunci untuk manajemen kualitas terletak pada kesadaran akan perlunya perbaikan dan pemilihan teknik perbaikan yang tepat. *Crafsmen* yang melakukan pekerjaan pemeliharaan yang kurang memadai harus diidentifikasi. Hal ini dapat dicapai dengan melacak pekerjaan yang diulang oleh seorang tukang tertentu. Analisis lebih lanjut dapat dilakukan untuk menemukan penyebab dari pekerjaan yang kurang memadai tersebut. Kemungkinan penyebabnya adalah ketidaktersediaan alat khusus, tukang tidak memiliki tingkat keterampilan yang dibutuhkan, atau pengawasan yang buruk, dan sebagainya. Menghilangkan penyebab-penyebab tersebut dan memantau pekerjaan pemeliharaan yang telah selesai dapat menghasilkan peningkatan kualitas produksi pemeliharaan. Perkiraan teknologi/proses baru yang akan diadopsi oleh perusahaan harus dibuat, dan *Crafsmen* harus dilatih sebelum kedatangan peralatan yang dimaksud.

11.7 Pemeliharaan Berbasis Keandalan

Pemeliharaan berbasis keandalan adalah strategi tingkat tinggi yang dapat menghasilkan pemeliharaan preventif yang dioptimalkan yang berfokus pada fungsi sistem dan peralatan. Pemeliharaan berbasis mengikuti pendekatan sistem yang terstruktur dengan baik untuk mengembangkan PM. Pemeliharaan berbasis telah diterapkan dalam beberapa industri, termasuk nuklir, energi, dan petrokimia.

Keberhasilan Pemeliharaan berbasis keandalan berasal dari fakta bahwa memiliki tujuan yang jelas, prinsip yang kokoh, dan strategi yang efektif. Tujuan utamanya adalah menentukan program pemeliharaan yang optimal menggunakan berbagai strategi pemeliharaan. Pemeliharaan berbasis keandalan memiliki prinsip-prinsip yang membedakannya dari semua metodologi lain untuk mengembangkan program pemeliharaan.

Pemeliharaan berbasis keandalan memiliki tujuan-tujuan berikut:

1. Menentukan tugas pemeliharaan yang paling efektif secara biaya dan dapat diterapkan untuk meminimalkan risiko dan dampak kegagalan pada fungsi sistem/peralatan.
2. Memastikan kinerja keselamatan dan keandalan yang tinggi.
3. Menjaga fungsionalitas sistem dan peralatan dengan cara yang paling ekonomis.

Objektif Pemeliharaan berbasis keandalan yang spesifik seperti yang adalah sebagai berikut:

1. Memastikan tercapainya tingkat keselamatan dan keandalan bawaan peralatan.
2. Mengembalikan peralatan ke tingkat bawaan ini ketika terjadi penurunan kualitas.
3. Mendapatkan informasi yang diperlukan untuk perbaikan desain pada item-item yang reliabilitas bawaannya terbukti tidak memadai.

4. Mencapai tujuan-tujuan ini dengan biaya total minimum, termasuk biaya pemeliharaan, biaya dukungan, dan konsekuensi ekonomi dari kegagalan operasional.

Tujuan dan objektif spesifik di atas merupakan dasar yang jelas untuk program pemeliharaan yang efektif yang biasanya dihasilkan dari penerapan metodologi Pemeliharaan berbasis keandalan.

Pemeliharaan berbasis keandalan memiliki prinsip-prinsip utama berikut yang membedakannya dari metodologi pemeliharaan lainnya:

1. Preservasi fungsi sistem peralatan: Fokus di sini adalah menjaga sistem agar tetap menjalankan fungsinya, bukan menjaga agar sistem beroperasi seolah-olah baru. Hal ini mengindikasikan bahwa dalam hal sistem menjalankan fungsinya, tidak perlu melakukan pemeliharaan berlebihan yang dapat menyebabkan kegagalan dalam beberapa kasus. Prinsip ini telah mengarah pada pengurangan pemeliharaan preventif berbasis waktu dalam industri penerbangan yang mengurangi biaya dan meningkatkan keandalan sistem. Redundansi fungsi melalui penggunaan peralatan ganda meningkatkan keandalan fungsional, tetapi juga meningkatkan biaya siklus hidup dalam hal pengadaan dan biaya operasional.
2. Fokus pada sistem: Pemeliharaan berbasis keandalan berfokus pada sistem daripada komponen, karena fungsi-fungsi biasanya didorong oleh sistem.
3. Pemeliharaan berbasis keandalan berpusat pada keandalan: Ia memperlakukan statistik kegagalan sehubungan dengan usia. Pemeliharaan berbasis keandalan berusaha untuk mengetahui probabilitas kegagalan yang bersyarat pada usia-usia tertentu (probabilitas kegagalan yang akan terjadi dalam setiap periode usia operasional yang diberikan).

4. Keselamatan dan ekonomi adalah kriteria utama: Keselamatan harus dijamin pertama-tama dengan segala cara, diikuti oleh biaya yang dihasilkan dari dampak terhadap produksi dan operasi.
5. Ada batasan-batasan desain: Tujuan Pemeliharaan berbasis keandalan adalah menjaga keandalan bawaan dari desain peralatan, dengan mengakui bahwa perubahan dalam keandalan bawaan muncul dari desain itu sendiri, bukan dari pemeliharaan. Pada dasarnya, pemeliharaan hanya dapat mencapai dan menjaga tingkat keandalan suatu sistem.
6. Umpan balik diperlukan untuk perbaikan: Pemeliharaan berbasis keandalan mengakui bahwa umpan balik pemeliharaan dapat meningkatkan desain asli. Selain itu, Pemeliharaan berbasis keandalan menyadari bahwa seringkali terdapat perbedaan antara masa pakai desain yang dirasakan dan masa pakai intrinsik atau sebenarnya.
7. Kegagalan adalah kondisi yang tidak memuaskan: kegagalan dapat berarti kehilangan fungsi (operasi berhenti) atau kehilangan kualitas yang dapat diterima (operasi tetap berlanjut).
8. Tugas pemeliharaan harus didasarkan pada logika: Pemeliharaan berbasis keandalan menggunakan pohon logika untuk mengembangkan dan memilih tugas pemeliharaan.

11.8 Pemeliharaan Produktif Total

Pemeliharaan Produktif Total adalah pendekatan untuk mengelola pemeliharaan dengan menggabungkan pendekatan dan teknik manajemen kualitas total Pemeliharaan Produktif Total ke dalam pemeliharaan. Pemeliharaan Produktif Total bergantung pada manajemen peralatan dan pemberdayaan karyawan. Tulang punggung Pemeliharaan Produktif Total adalah program pemeliharaan preventif yang efektif. Pemeliharaan Produktif Total mengidentifikasi enam kerugian akibat pemeliharaan dan

bertujuan untuk mengeliminasinya. Pemeliharaan Produktif Total berupaya memaksimalkan efektivitas keseluruhan peralatan.
Pemeliharaan Produktif Total terdapat lima tujuan kunci berikut:

1. Maksimalkan OEE (*Overall Equipment Effectiveness*), yang mencakup ketersediaan, efisiensi proses, dan kualitas produk;
2. Mengambil pendekatan sistematis terhadap keandalan, keterpeliharaan, dan biaya siklus hidup (LCC);
3. Melibatkan operasi, manajemen material, pemeliharaan, rekayasa, dan administrasi dalam manajemen peralatan;
4. Melibatkan semua tingkatan manajemen dan pekerja;
5. Meningkatkan kinerja peralatan melalui kegiatan kelompok kecil dan kinerja tim.

Elemen-elemen kunci dari Pemeliharaan Produktif Total mencakup hal-hal berikut:

1. Pemeliharaan mandiri: Operator peralatan menjadi titik fokus dari aktivitas Pemeliharaan Produktif Total. Meskipun sebagian besar operator memahami apa yang dilakukan peralatan mereka, sedikit yang memahami mekanisme dasar bagaimana peralatan tersebut bekerja. Istilah "pemeliharaan mandiri" digunakan untuk menggambarkan aktivitas operator yang terkait dengan pemeliharaan peralatan, serta aktivitas perbaikan peralatan lainnya yang bersifat mandiri. Operator akan melakukan pembersihan, inspeksi, pelumasan, penyesuaian, dan penggantian komponen kecil serta tugas pemeliharaan ringan lainnya yang memerlukan pelatihan dan instruksi tertentu, namun tidak memerlukan keterampilan tukang yang komprehensif. Secara bertahap, operator belajar bagaimana mendiagnosis masalah peralatan sebelum masalah tersebut menjadi serius.
2. Manajemen Peralatan: Dalam Pemeliharaan Produktif Total, ketika peralatan tidak berperforma sesuai yang dibutuhkan, kerugian performa dicatat dan dipantau. Kerugian ini dapat

dikelompokkan menjadi enam kategori: kerusakan, persiapan dan penyesuaian, waktu henti dan berhenti kecil, kecepatan berkurang, cacat, dan kerugian hasil. Kerusakan dan persiapan menyebabkan waktu henti dan mempengaruhi ketersediaan, kecepatan berkurang mempengaruhi waktu siklus, dan cacat serta kerugian hasil mempengaruhi kualitas. OEE adalah ukuran kinerja Pemeliharaan Produktif Total yang utama dan merupakan hasil dari ketersediaan, waktu siklus, dan tingkat kualitas. Operator dan pemelihara dilatih untuk mengidentifikasi masalah terkait OEE dan melakukan analisis akar penyebab dalam tim untuk menyelidiki kerugian tersebut.

3. Perencanaan Sistematis dan Peningkatan Berkelanjutan: Dalam departemen pemeliharaan, metodologi Pemeliharaan Produktif Total mendorong pengembangan perencanaan sistematis dan pengendalian pemeliharaan preventif dan korektif, serta mendukung sepenuhnya aktivitas mandiri yang dilakukan oleh operator. Di pabrik-pabrik di mana lingkungan operasi dan pemeliharaan dasar telah ditingkatkan hingga titik pengembalian yang menurun, dilakukan kegiatan pencegahan pemeliharaan aktif, seperti yang dijelaskan sebelumnya dalam bagian tentang desain untuk keperawatan. Selama proses ini, perlu ditekankan peningkatan keterampilan operator dan pemelihara. Pengeluaran untuk pelatihan biasanya sekitar 5-8% dari anggaran tenaga kerja.

11.9 Pemeliharaan Intelligent

Perkembangan teknologi informasi dan komunikasi memberikan fasilitas bagi pemeliharaan yang efektif dan cerdas. Perkembangan ini memungkinkan pemantauan peralatan, transfer data, berbagi informasi, menganalisis data dengan sistem terintegrasi, dan mengkoordinasikan aktivitas melalui Web. Kemampuan ini memungkinkan untuk melampaui pemeliharaan prediktif menjadi prognostik cerdas yang didefinisikan sebagai

pendekatan sistematis yang dapat terus-menerus melacak penurunan kesehatan peralatan dan menggambarkan perilaku temporal indikator kesehatan untuk memprediksi risiko perilaku yang tidak dapat diterima dari waktu ke waktu serta menentukan komponen mana yang mungkin gagal. Hal ini mengarah pada pengembangan konsep e-pemeliharaan yang merupakan penggunaan luas dan penerapan maju pemeliharaan berbasis kondisi.

11.10 Perilaku Manusia pada Pemeliharaan Sistem

Dalam menjalankan fungsi perencanaan, pengorganisasian, dan pengendalian, manajer pemeliharaan prihatin dengan bagaimana tindakan mereka mempengaruhi perilaku manusia. Mereka harus mencoba mengetahui bagaimana perilaku bawahan dapat mempengaruhi perencanaan, pengorganisasian, dan pengendalian manajemen. Dalam pengambilan keputusan pemeliharaan, perilaku bawahan harus menjadi perhatian bagi manajemen. Harus dipastikan bahwa tingkat kepuasan tukang yang diinginkan tercapai dan dipertahankan.

DAFTAR PUSTAKA

Al-Turki, U. 2011. A framework for strategic planning in maintenance. *Journal of Quality in Maintenance Engineering*, *17*, 150–162. https://doi.org/10.1108/13552511111134583

Duffuaa, S. O., & Raouf, A. 2015. Maintenance Systems. In S. O. Duffuaa & A. Raouf (Eds.), *Planning and Control of Maintenance Systems: Modelling and Analysis* (pp. 1–17). Springer International Publishing. https://doi.org/10.1007/978-3-319-19803-3_1

J. Davidson Frame. 2003. *Managing Risk in Organizations*.

Lee, J., Ni, J., Djurdjanovic, D., Qiu, H., & Liao, H. 2006. Intelligent Prognostics Tools and E-maintenance. *Haitao Liao*, *57*. https://doi.org/10.1016/j.compind.2006.02.014

Murthy, D., Atrens, A., & Eccleston, J. 2002. Strategic maintenance management. *Journal of Quality in Maintenance Engineering*, *8*. https://doi.org/10.1108/13552510210448504

Salih O. Duffuaa, A. R. 2015. *Planning and Control of Maintenance Systems: Modelling and Analysis* (Vol. 2).

BIODATA PENULIS

Muhammad Fairuzabadi
Dosen pada Program Sarjana Informatika Fakultas Sains dan Teknologi Universitas PGRI Yogyakarta

Lahir di Bulukumba, Sulawesi Selatan tahun 1974. Alumni S1 Program Sarjana Ilmu Komputer FakultaS MIPA Universitas Kristen Immanuel (UKRIM) Yogyakarta Tahun 1998 dan S2 Program Magister Ilmu Komputer Universitas Gadjah Mada Tahun 2006. Bekerja sebagai dosen pada Program Sarjana Informatika Fakultas Sains dan Teknologi Universitas PGRI Yogyakarta. Selain itu, sebagai tenaga ahli dalam berbagai proyek pengembangan sistem informasi, *master plan* dan *blue print* TIK diintansi Pemerintah Daerah, Kementerian dan Lembaga. Diantaranya: Masterplan Jogja Smart Province, Cetak Biru Sistem Informasi Terintegrasi PDLKWS Kementerian Kehutanan dan Lingkungan Hidup, dan Rencana Induk Sistem Pemerintahan Berbasis Elektronik (SPBE) Pemerintah Kota Metro.

BIODATA PENULIS

Irani Hoeronis, S.Si., M.T., CRP., CIISA.
Dosen Program Studi Informatika
Fakultas Teknik Universitas Siliwangi Tasikmalaya

Penulis merupakan dosen di Program Studi Informatika Fakultas Teknik Universitas Siliwangi sejak 2017. Penulis menyelesaikan pendidikan S1 pada tahun 2007 di jurusan Matematika bidang Minat Ilmu Komputer Universitas Padjadjaran Bandung kemudian melanjutkan ke jenjang S2 pada tahun 2010 di Informatics Engineering Telkom University Bandung. Ketertarikan penulis dalam bidang pendidikan dimulai sejak menjadi mahasiswa ketika mendapat tugas sebagai asisten dosen pada laboratorium komputer di Universitas Padjadjaran. Selain itu, penulis pernah mengajar di beberapa Universitas/Politeknik di wilayah Bandung.
Email Penulis: iranihoeronis@unsil.ac.id

BIODATA PENULIS

Zen Munawar.
Dosen Program Studi Manajemen Informatika
Politeknik LP3I Bandung

Penulis memiliki jabatan akademik lektor, dan alumni dari TI UNPAS, Pasca Sarjana AdPen UNINUS, Sistem Informasi dari STMIK IM dan Pasca Sarjana Sistem Informasi STMIK LIKMI Bandung serta mengikuti Program Doktoral Teknologi Informasi dan Komunikasi di UTeM Melaka Malaysia. Penulis aktif menjadi editor board dan reviewer jurnal nasional dan internasional terakreditasi Sinta serta aktif dalam menulis artikel di jurnal nasional maupun internasional serta menulis buku. Selain itu pula penulis aktif sebagai pengurus bidang kerjasama industri pada organisasi APTIKOM Prov. Jabar, sebagai sekretaris DPW Jawa Barat pada Ikatan Ahli Informatika Indonesia (IAII) sebagai Dewan Pengawas pada organisasi Perkumpulan Auditor Teknologi Informasi dan Komunikasi Indonesia (PASTIKINDO). Penulis memiliki kompetensi Information Technology Expert Auditor, ICT Project Manager, System Analyst dan Assesor Of Competency dari BNSP.

BIODATA PENULIS

Ir. Johni S. Pasaribu, MT.
Dosen Program Studi Sistem Informasi
Fakultas IT Politeknik Piksi Ganesha

Penulis lahir di Medan tanggal 1 Juni 1966 dari pasangan HA Pasaribu (alm) dan Taruli Tambunan (alm). Penulis adalah anak keempat dari enam bersaudara. Pendidikan yang ditempuh yaitu SD Methodist Medan lulus tahun 1979, SMPN 88 Jakarta lulus tahun 1982 dan SMAN 3 Jakarta lulus tahun 1985. Kemudian melanjutkan studi di ITB jurusan Teknik Fisika spesialisasi Instrumentasi dan Kontrol. Penulis melakukan penelitian untuk tugas akhir di PT IPTN dengan judul Perancangan Sistem Kendali Fly By Wire Gerak Longitudinal Pada Engineering Flight Simulator Pesawat N250-100 dan menyelesaikan gelar sarjana teknik Insinyur (Ir) tahun 1992.

Kemudian penulis bekerja di PT. IPTN (sekarang PT Dirgantara Indonesia) di Departemen Iron Bird kemudian berpindah di Departemen Engineering Flight Simulator (EFS). Penulis terlibat dalam Operation and Maintenance untuk EFS sehingga simulator N250-100 ini dapat digunakan saat dilakukan pengujian-pengujian pesawat melalui simulator sebelum pengujian langsung pesawat di udara. Tahun 1998 penulis melanjutkan studi

S2 di ITB pada jurusan Teknik Informatika untuk spesialisasi Software Engineering, dimana ilmu yang diperoleh dapat diaplikasikan dalam pekerjaan di lapangan. Tahun 2001 penulis menyelesaikan pendidikan S2 dengan gelar MT mengambil penelitian yang berjudul Pengembangan Perangkat Lunak Simulasi Sistem Perasa Kendali Elevator Pesawat N250-100 dimana penelitian ini juga dilakukan di Engineering Flight Simulator (EFS) N250-100. Tahun 2004 penulis mulai aktif mengajar di Perguruan Tinggi di Bandung diantaranya pernah mengajar di Unikom, ITHB, STT Telkom (kini TelU). Kini penulis penuh mengajar dan menjadi dosen tetap di Politeknik Piksi Ganesha. Bidang keahlian mengajar dan penelitian penulis adalah Artificial Intelligence (AI), Decision Support System (DSS), Software Engineering dan Software Testing. Penulis juga aktif melakukan bimbingan mahasiswa S1 dan publikasi di jurnal-jurnal nasional maupun internasional.

BIODATA PENULIS

Sarji, M.Kom.
Dosen Program Studi Sistem Informasi
Institut Teknologi dan Bisnis Riyadlul Ulum

Penulis lahir di Ciamis tanggal 05 Oktober 1985. Penulis adalah dosen tetap pada Program Studi Sistem Informasi, Institut Teknologi dan Bisnis Riyadlul Ulum. Menyelesaikan pendidikan S1 pada Jurusan Sistem Informasi dan melanjutkan S2 pada Jurusan Teknik Informatika. Penulis tertatik pada bidang pemrograman dan menguasai beberapa bahasa pemrograman diantaranya PHP, Java, Visual Basic 6, dan Visual Basic .Net.

Diluar aktifitas dikampus Penulis juga melayani jasa pengembangan sistem informasi baik dekstop, web dan android. Selain itu Penulis juga mengelola kelas online terkait dengan pemrograman dan database. Penulis dapat dihubungi melalui wa di nomor 082328183708.

BIODATA PENULIS

Dr. Irmawati, S.Kom., MMSI.
Dosen Program Studi Sistem Informasi
Fakultas Teknik dan Informatika
Universitas Multimedia Nusantara

Penulis adalah dosen tetap pada Program Studi Sistem Informasi Fakultas Teknik dan Informatika Universitas Multimedia Nusantara. Menyelesaikan pendidikan S1 dan S2 pada Program Studi Sistem Informasi Universitas Gunadarma dan S3 pada departemen Teknik Elektro Universitas Indonesia. Penulis menekuni bidang keilmuan meliputi *Artificial Intelligence*, *Image Processing*, *Machine Learning*, dan *Deep Learning*.

BIODATA PENULIS

Rita Komalasari, S.Si., M.Kom.
Dosen tetap di perguruan tinggi vokasi swasta di Bandung

Penulis merupakan dosen tetap di perguruan tinggi vokasi swasta di Bandung. Saat ini penulis sudah memiliki jabatan fungsional akademik lektor, alumni dari Ilmu Komputer Universitas Padjadjaran dan Sistem Informasi STMIK LIKMI. Penulis aktif menjadi editor dan reviewer jurnal nasional serta aktif dalam menulis artikel di jurnal nasional maupun internasional serta menjadi editor dan penulis buku. Selain itu pula penulis aktif sebagai pengurus bidang publikasi dan HAKI pada organisasi Asosiasi Pendidikan Tinggi Informatika dan Komputer (APTIKOM) Propinsi Jawa Barat, sebagai Bendahara Umum pada organisasi Perkumpulan Auditor Teknologi Informasi dan Komunikasi Indonesia (PASTIKINDO). Penulis memiliki kompetensi Information Technology Expert Auditor, ICT Project Manager dan System Analyst.

BIODATA PENULIS

Nurul Aini, S.Kom., M.T.
Dosen Program Studi Rekayasa Perangkat Lunak
Universitas Dipa Makassar

Nurul Aini lahir di Barru, Sulawesi Selatan, 06 Desember 1986. Ketertarikan penulis terhadap ilmu komputer dimulai pada tahun 2005 silam. Hal tersebut membuat penulis lanjut ke perguruan tinggi di UNIVERSITAS DIPA MAKASSAR dengan memilih Jurusan SISTEM INFORMASI dan berhasil lulus pada tahun 2009. Penulis kemudian melanjutkan pendidikan ke studi S2 di prodi TEKNIK INFORMATIKA PROGRAM PASCA SARJANA UNIVERSITAS HASANUDDIN dan menyelesaikan studi di tahun 2013. Penulis memiliki ketertarikan dan aktif menulis karya ilmiah Data Mining, Decision Support System (DSS) dan Data Science. Dan untuk mewujudkan karir sebagai dosen profesional, penulis pun aktif sebagai peneliti dibidang kepakarannya tersebut. Selain meneliti, penulis juga aktif menulis buku dengan harapan dapat memberikan kontribusi positif bagi bangsa dan negara yang sangat tercinta ini. Email Penulis: nurulaini.m11@undipa.ac.id

BIODATA PENULIS

Dr. Santo Fernandi Wijaya., S.Kom., M.M.
Dosen Program Studi Sistem Informasi,
Fakultas Teknik dan Informatika,
Universitas Multimedia Nusantara, Tangerang.

Penulis lahir di Jakarta dan sampai saat ini sebagai dosen tetap pada Program Studi Sistem Informasi Fakultas Teknik dan Informatika, Universitas Multimedia Nusantara, Tangerang. Penulis menyelesaikan pendidikan studi Strata-1 Jurusan Komputerisasi Akuntansi, Universitas Bina Nusantara, Jakarta (1994). Strata-2 Jurusan Sistem Informasi, Universitas Bina Nusantara, Jakarta (2000). Strata-3 Jurusan Doctor Computer Science, Universitas Bina Nusantara, Jakarta (2022). Penulis telah menulis beberapa buku mengenai ERP dan solusi bisnis. Disamping itu, penulis telah memiliki pengalaman pada tingkatan manajerial sebagai praktisi selama 34 tahun (1988 sd 2022) dalam bidang Finance, Accounting, dan berpengalaman mengimplementasikan sistem ERP pada beberapa perusahaan industri manufaktur. Penulis aktif sebagai dosen praktisi pada prodi Sistem Informasi di Binus University sejak tahun 2001. Penulis aktif melakukan riset dalam bidang Enterprise System, Knowledge Management, Business

Process Management, Change Management, Analisis dan pengembangan sistem informasi. Penulis dapat dihubungi dengan email: santofwijaya@gmail.com, Google scholar ID: wbwFg.AAAAAJ, https://orcid.org/0000-0002-0650-2066, Sinta ID: 6009307, Scopus ID: 57194540113.

BIODATA PENULIS

Angga Aditya Permana
Dosen Informatika
Universitas Multimedia Nusantara

Angga Aditya Permana (lahir pada Desember 1989 di Jakarta) seorang dosen full time di bidang Computer Science pada Universitas Multimedia Nusantara sejak tahun 2021, fokus penelitian pada kriptografi, steganografi serta keamanan computer, namun pada tahun 2021 sedang mendalami topik bioinformatika dan network science. Angga juga memiliki hobi yaitu touring dan juga bermain bulutangkis, saat ini sedang menjadi mahasiswa program doctoral pada IPB University. Memulai karir pertama kali sebagai Network Enginner tahun 2011 dan memulai profesinya sebagai dosen pada 2013 di kampus swasta yang ada di Jakarta, pernah juga mengajar di kampus negri yang berada di Jakarta dan Tangerang, juga pernah di undang menjadi dosen tamu untuk mengenalkan Bioinformatika di kampus negeri di Jakarta. Terimakasih..

BIODATA PENULIS

Melissa Indah Fianty, S.Kom., MMSI
Dosen Program Studi Sistem Informasi Fakultas Teknik dan Informatika Universitas Multimedia Nusantara

Penulis lahir di Bengkulu tanggal 13 Januari 1992. Penulis adalah dosen tetap pada Program Studi Sistem Informasi Fakultas Teknik dan Informatika, Universitas Multimedia Nusantara. Menyelesaikan pendidikan S1 pada Teknik Informatika dan melanjutkan S2 pada Manajemen Sistem Informasi di Bina Nusantara University.