Rekayasa Perangkat Lunak

Zatin Niqotaini, S.Tr.Kom., M.Kom., **Indah Purnamasari**, S.T., M.Kom., **Cholid Fauzi**, S.T.,M.T., Ir. **Yoga Sahria**, S.Kom., M.Kom., **Dartono**, S.Kom, M.Kom., **Dian Nursantika**, S.Kom., M.Cs., **Ida Afriliana**, ST, M.Kom., **Cahyo Prihantoro**, S.Kom., M.Eng., **Petrus Christo**, M.Kom., **Andi Wijaya**, M.Kom., **Anang Anggono Lutfi**, S.Pd., M.Eng., **Mohammad Robihul Mufid**, S.ST., M.Tr.Kom., **Arif Rizki Marsa**, S.Kom.,M.Kom., **Yuni Widiastiwi**, S.Kom, M.Si.

Editor: **Eri Mardiani**

Rekayasa Perangkat Lunak

Penamuda Media

Rekayasa Perangkat Lunak

Penulis:
Zatin Niqotaini, Indah Purnamasari, Cholid Fauzi, Yoga Sahria, Dartono, Dian Nursantika, Ida Afriliana, Cahyo Prihantoro, Petrus Christo, Andi Wijaya, Anang Anggono Lutfi, Mohammad Robihul Mufid, Arif Rizki Marsa, Yuni Widiastiwi

ISBN 978-623-09-4584-7

Editor	: Eri mardiani
Penyunting	: Tim PT Penamuda Media
Penata Letak	: Teguh Heri Purwanto
Desain Sampul	: Tim Desain PT Penamuda Media
Penerbit	: PT Penamuda Media

Redaksi	: Casa Sidoarum RT03 Ngentak, Sidoarum Godean Sleman Yogyakarta
Web	: www.penamuda.com
E-mail	: penamudamedia@gmail.com
Instagram	: @penamudamedia
WhatsApp	: 085700592256

Cetakan Pertama, Juni 2023
vi + 146 halaman; 15 x 23 cm

Kata Pengantar

Alhamdulillah dan Puji syukur kami panjatkan kepada Allah SWT yang telah melimpahkan rahmat dan karunianya serta memberikan petunjuk kepada penulis sehingga dapat menyelesaikan buku yang berjudul **Rekayasa Perangkat Lunak** hingga dapat dibaca oleh para pembaca.

Seiring dengan banyaknya perkembangan perangkat lunak saat ini, penulis mencoba memenuhi kebutuhan saat ini yang banyak dibutuhkan masyarakat dari berbagai kalangan mulai dari pelajar, mahasiswa, karyawan dan masyarakat pada umumnya

Dalam buku ini membahas ruang lingkup rekayasa perangkat lunak, prinsip-prinsip, konsep manajemen proyek, perencanaan proyek, analisis dan pengumpulan kebutuhan, perancangan antar muka, perancangan basis data, implementasi perangkat lunak, pendekatan menulis kode program,teknis menulis program,prinsip-prinsip pemrograman, pengujian perangkat lunak, metodologi pengembangan perangkat lunak, serta bagaimana melakukan pemeliharaan perangkat lunak

Kami berharap semoga buku ini bermanfaat walau mungkin masih banyak kekurangan yang perlu diperbaiki

Jakarta, 08 Juli 2023

Eri Mardiani

Daftar Isi

RUANG LINGKUP REKAYASA PERANGKAT LUNAK

1.1 Pengertian Perangkat Lunak

Ada beberapa definisi atau pengertian tentang perangkat lunak. Menurut Pressman perangkat lunak adalah:

"1) Instruksi (program komputer) yang bila dieksekusi dapat menjalankan fungsi tertentu; 2) Struktur data yang dapat membuat program memanipulasi informasi; dan 3) Dokumen yang menjelaskan operasi dan penggunaan program."

Sementara *The Institute of Electrical and Electronics Engineers* (IEEE) mendefinisikan perangkat lunak sebagai:

"Program komputer, prosedur, aturan, dan dokumentasi yang berkaitan dengannya serta data yang berkaitan dengan operasi suatu sistem komputer."

Dari dua definisi tentang perangkat lunak di atas dapat disimpulkan bahwa perangkat lunak adalah:

a. Program Komputer

   Kumpulan instruksi yang apabila dieksekusi oleh sistem komputerakan menjalankan proses atau fungsi tertentu.

b. Data

   Struktur data atau basis data yang memungkinkan data yang tersimpan padanya dimanipulasi oleh program.

c. Dokumen

   Dokumentasi yang menjelaskan bagaimana pengembangan, cara instalasi, penggunaan, dan pemeliharaan program.

1.2 Karakteristik Perangkat Lunak

Dalam suatu sistem berbasis komputer, perangkat lunak dikategorikan sebagai elemen logis. Oleh karena itu perangkat lunak memiliki karakteristik tertentu yang membedakannya dengan perangkat keras.

Menurut Pressman, ada tiga karakteristik yang menjadi ciri suatu perangkat lunak, yaitu:

1. Perangkat lunak merupakan produk pengembangan atau rekayasa,dan tidak diproduksi dalam pengertian klasik.
2. Perangkat lunak tidak akan pernah rusak atau aus.
3. Perangkat lunak pada umumnya dibangun sesuai keinginan, jadi tidakdibentuk dari komponen yang sudah ada.

Menurut Zoro, Selain itu tiga karakteristik di atas, suatu perangkat lunak:

1. Merupakan produk yang unik (tidak ada seri produksi).
2. Tidak terlihat (*invisible*).
3. Fleksibel, sehingga mudah dimodifikasi.
4. Terhubung dengan perangkat keras.

Cukup penting untuk mengetahui karakteristik perangkat lunak ini, karena proses rekayasa yang nanti akan dilaksanakan harus mampu menghasilkanproduk dengan karakteristik seperti itu.

1.3 Jenis-jenis Perangkat Lunak

Dilihat dari fungsinya, perangkat lunak dapat dikelompokkan menjadi:

1. Perangkat lunak sistem

   Perangkat lunak yang kegunaannya lebih banyak ditujukan untukoperasional komputer atau perangkat lunak lain. Perangkat lunak yang termasuk dalam kelompok ini adalah:

   a. sistem operasi
   b. penerjemah bahasa pemrograman (*compiler*/*interpreter*)
   c. utility software

2. Perangkat lunak aplikasi

Perangkat lunak yang kegunaannya lebih banyak ditujukan untukmembantu menyelesaikan masalah-masalah yang dihadapi olehpemakai. Perangkat lunak aplikasi dapat dibedakan menjadi:

a. program paket yang sudah jadi (*package program*)
b. program aplikasi buatan sendiri (*tailor made*)

Sedangkan jika dilihat dari bentuk aplikasinya, perangkat lunak dibedakan menjadi:

1. Perangkat lunak sistem (*system software*)

   Sekumpulan program yang ditulis untuk mendukung operasional komputer atau kepentingan program lain. Contoh:

   a. sistem operasi
   b. compiler/interpreter
   c. *utility software*

2. Perangkat lunak waktu nyata (*real-time software*)

   Perangkat lunak yang memonitor, menganalisis, atau mengendalikan kejadian-kejadian dunia nyata berdasarkan kejadian (*event*) yang diterimanya dengan kendala waktu yang ketat (teliti). Contoh:

   a. perangkat lunak untuk mengatur mekanisme kerja *lift*
   b. ATCS (*Automatic Traffic Control System*)

3. Perangkat lunak bisnis (*business software*)

   Perangkat lunak yang memberikan fasilitas operasi untuk bisnis atau fasilitas pengambilan keputusan manajemen. Contoh:

   a. aplikasi untuk sistem informasi, misalnya sistem persediaan,akuntansi, produksi, dan lain-lain.
   b. sistem pendukung keputusan
   c. Enterprise Resources Planning (ERP)

4. Perangkat lunak rekayasa dan ilmu pengetahuan (*engineering andscientific software*).

   Perangkat lunak jenis ini biasanya berhubungan dengan komputasi data numerik, CAD (*Computer Aided Design*), simulasi sistem, dan lain-lain. Contoh:

   a. aplikasi statistika (misalnya SPSS), matematika (Math Lab)
   b. aplikasi untuk merancang lengkung sayap pesawat terbang

5. Embedded Software

   Perangkat lunak yang menyatu (terintegrasi) dengan perangkat tertentu yang digunakan untuk mengontrol produk atau sistem, baik untuk konsumen atau pasar industri. Contoh:
   a. aplikasi *programmable logic circuit* (PLC)
   b. permainan (*game*) pada *mobile phone*
6. Perangkat lunak pribadi (*personal software*)

   Perangkat lunak yang banyak digunakan pada aplikasi-aplikasi yangbersifat perorangan. Contoh:
   a. pengolah kata (*word processor*), lembar kerja elektronik (*spreadsheet*), *organizer*
   b. aplikasi keuangan pribadi
7. Perangkat lunak intelegensia buatan (*artificial inteligent software*)
   Perangkat lunak yang dibuat dengan menggunakan teknik algoritmanon-numerik untuk memecahkan masalah yang kompleks. Contoh:
   a. sistem pakar (*expert system*)
   b. pemrosesan bahasa alami (*natural language*)
   c. permainan catur

Dari sekian banyak dan ragam jenis-jenis perangkat lunak yang ada, perangkat lunak bisnislah yang paling banyak dibuat sampai saat ini.

1.4 Pengertian Rekayasa Perangkat Lunak

Rekayasa perangkat lunak merupakan suatu proses rancang bangun perangkat lunak yang menggunakan kaidah-kaidah ilmu, seperti prinsip, konsep, dan metode sehingga dihasilkan perangkat lunak yang berkualitas (dapat digunakandan beroperasi dengan benar, bermanfaat, serta sesuai dengan kebutuhan yang diinginkan).

Beberapa definisi lain yang menjelaskan tentang rekayasa perangkat lunak:

1. Pembentukan dan penggunaan prinsip rekayasa (*engineering*) untuk mendapatkan perangkat lunak secara ekonomis namun andal dan dapat bekerja secara efisien pada komputer (Fritz Bauer, 1968, seperti dikutip oleh Pressman.

2. Suatu disiplin yang mengintegrasikan proses, metode, dan alat (*tools*)untuk pembangunan perangkat lunak komputer.
3. Suatu teknologi berlapis, yaitu proses, metode dan alat, dengan kualitas sebagai dasar utamanya.
4. Teori, metode, dan alat bantu yang dibutuhkan untuk mengembangkan perangkat lunak.
5. Penerapan pendekatan yang sistematis, disiplin dan terukur untuk pengembangan, operasi, dan pemeliharaan perangkat lunak.

1.5 Tujuan Rekayasa Perangkat Lunak

Seperti telah disinggung pada bagian sebelumnya, tujuan utama yang menjadi fokus dari rekayasa perangkat lunak adalah kualitas. Yang dimaksud dengan kualitas disini adalah:

1. perangkat lunak yang dihasilkan sesuai dengan kebutuhan yangdiinginkan.
2. dapat digunakan dan beroperasi dengan benar di lingkungan sebenarnya.
3. memberikan manfaat bagi pemakai yang menggunakannya.
4. biaya yang dikeluarkan untuk membuatnya rendah, dalam arti efektifdan sesuai dengan anggaran yang telah ditetapkan.
5. tepat waktu, baik saat pembuatan, penyerahan ke pemakai, maupuninstalasinya.
6. setiap tahap pekerjaan terjamin kualitasnya, terdokumentasi, dan dapat dipertanggungjawabkan kebenarannya (ada proses verifikasi dan validasi).

Lebih jauh lagi, rekayasa perangkat lunak harus mampu menghasilkan produk perangkat lunak dengan atribut kualitas:

1. *Maintainability*

   Perangkat lunak harus memungkinkan untuk dikembangkan sesuaidengan perubahan kebutuhan pemakai.
2. *Dependability*

   Berkaitan dengan rentang karakteristik yang mencakup keandalan, keamanan, dan keselamatan. Perangkat lunak harus tidak menyebabkan kerusakan fisis dan ekonomis saat sistem mengalami kegagalan.

3. *Efficiency*

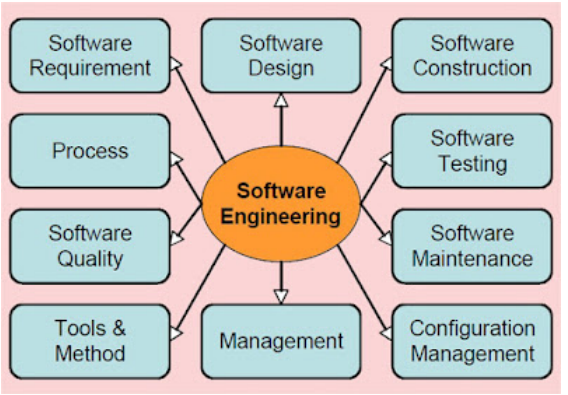
   Perangkat lunak harus efisien dalam penggunaan sumber daya seperti *memory* dan siklus pemroses.

4. *Usability*

   Perangkat lunak harus mempunyai antarmuka pengguna yang tepat dan dokumentasi yang memadai.

1.6 Ruang Lingkup Rekayasa Perangkat Lunak

Rekayasa Perangkat Lunak mempunyai ruang lingkup kerjanya Sebagai berikut:



Gambar 1. Ruang Lingkup Rekayasa Perangkat Lunak

a. *Software Requirements*

   berhubungan dengan spesifikasi kebutuhan dan persyaratan perangkat lunak.

b. *Software Design*

   meliputi proses penampilan arsitektur, komponen, antar muka, dan karakteristik lain dari perangkat lunak

c. *Software Construction*

   berhubungan dengan detil pengembangan perangkat lunak, termasuk algoritma, pengkodean, pencarian kesalahan dan pengujian.

d. *Software Testing*

   meliputi pengujian pada kinerja perangkat lunak secara keseluruhan

e. *Software Maintenance*

mencakup upaya-upaya perawatan ketika perangkat lunak telah dioperasikan.

f. *Software Configuration Management*

berhubungan dengan usaha perubahan konfigurasi perangkat lunak untuk memenuhi kebutuhan tertentu.

Software Engineering Management

berkaitan dengan pengelolaan dan pengukuran RPL, termasuk perencanaan proyek perangkat lunak.

g. *Software Engineering Tools and Methods*

mencakup kajian teoritis tentang alat bantu dan metode RPL.

h. *Software Quality*

menitikberatkan pada kualitas dan daur hidup perangkat lunak.

i. *Software Engineering Process*

berhubungan dengan implementasi, definisi, pengukuran, pengelolaan, perubahan dan perbaikan proses Rekayasa Perangkat Lunak.

1.7 Metode Rekayasa Perangkat Lunak

Pada rekayasa perangkat lunak, banyak model yang telah dikembangkan untuk membantu proses pengembangan perangkat lunak. Model-model ini pada umumnya mengacu pada model proses pengembangan sistem yang disebut *System Development Life Cycle* (SDLC) seperti terlihat pada Gambar berikut ini.

Gambar 2. *System Development Life Cycle*

1. Kebutuhan terhadap definisi masalah yang jelas. Input utama dari setiap model pengembangan perangkat lunak adalah pendefinisian masalah yang jelas. Semakin jelas akan semakin baik karena akan memudahkan dalam penyelesaian masalah.
2. Tahapan-tahapan pengembangan yang teratur. Meskipun model-model pengembangan perangkat lunak memiliki pola yang berbeda-beda, biasanya model-model tersebut mengikuti pola umum analysis - design - coding - testing - maintenance.
3. Stakeholder berperan sangat penting dalam keseluruhan tahapan pengembangan. Stakeholder dalam rekayasa perangkat lunak dapat berupa pengguna, pemilik, pengembang, pemrogram dan orang-orang yang terlibat dalam rekayasa perangkat lunak tersebut.

4. Dokumentasi merupakan bagian penting dari pengembangan perangkat lunak. Masing-masing tahapan dalam model biasanya menghasilkan sejumlah tulisan, diagram, gambar atau bentuk-bentuk lain yang harus didokumentasi dan merupakan bagian tak terpisahkan dari perangkat lunak yang dihasilkan**.**
5. Keluaran dari proses pengembangan perangkat lunak harus bernilai ekonomis. Nilai dari sebuah perangkat lunak sebenarnya agak susah dirupiahkan. Namun efek dari penggunaan perangkat lunak yang telah dikembangkan haruslah memberi nilai tambah bagi organisasi. Hal ini dapat berupa penurunan biaya operasi, efisiensi penggunaan sumber daya, peningkatan keuntungan organisasi, peningkatan image organisasi dan lain-lain

PRINSIP-PRINSIP REKAYASA PERANGKAT LUNAK

Berbagai kategori perangkat lunak dibangun untuk bermacam tujuan baik untuk bisnis, industri, digunakan pada insitusi pemerintahan, entertainment dsb. Banyak orang membuat program namun perangkat lunak yang dibangun secara professional bukan hanya sekedar program akan tetapi mencakup dokumentasi, library, struktur sistem, konfigurasi data, dsb (Sommerville, 2016)

Semakin tingginya kebutuhan dunia akan perangkat lunak maka akan semakin kompleks suatu sistem perangkat lunak tersebut sehingga kemungkinan kegagalan software akan semakin besar.

Perangkat lunak yang baik akan memenuhi kebutuhan, dan dapat digunakan dalam jangka waktu yang lama sehingga banyak memberikan manfaat bagi penggunanya.

Membangun perangkat lunak yang baik dapat ditinjau dari 3 faktor yaitu Biaya, Kinerja dan Waktu. Berikut ini tujuan dari Rekayasa Perangkat Lunak (Findawati, 2020) :

a. Mengeluarkan biaya produksi perangkat lunak yang rendah.
b. Menghasilkan perangkat lunak dengan kinerja tinggi, handal dan tepat waktu.
c. Menghasilkan perangkat lunak yang dapat bekerja pada berbagai jenis platform.
d. Menghasilkan perangkat lunak dengan biaya pemeliharaan rendah.

Untuk mencapai tujuan rekayasa perangkat lunak tersebut, maka berikut ini merupakan prinsip-prinsip dasar rekayasa perangkat lunak (Hooker, 2014)

2.1 Prinsip-Prinsip Dasar Rekayasa Perangkat Lunak

1. Alasan membangunnya

   Prinsip awal dalam membangun software yaitu dapat memberikan nilai manfaat bagi penggunanya. Sebelum dilakukan langkah lebih jauh dalam membangun perangkat lunak adalah memastikan terlebih dahulu apakah perangkat lunak ini memang diperlukan dan dapat memberikan nilai manfaat bagi penggunanya ? Apabila "tidak" maka jangan dilanjutkan. Sementara prinsip-prinsip lain akan mendukung prinsip awal ini.

2. Tetap sederhana

   Representasi perangkat lunak yang sederhana akan lebih disukai. Membangun perangkat lunak yang sederhana bukan berarti secara cepat dan sembarangan membuang fitur yang diperlukan namun mempertimbangkan suatu desain dan memikirkan bagaimana dapat menyederhanakannya tetapi tidak menjadikannya lebih rumit sehingga membuat sistem lebih mudah dipahami, mudah dalam pemeliharaan dan mengurangi kemungkinan error.

3. Pertahankan visi

   Prinsip ketiga dalam rekayasa perangkat lunak yaitu memiliki visi yang jelas, dapat dilakukan dengan cara memilih arsitek yang berintegritas dalam membangun perangkat lunak, yang fokus terhadap visi sehingga dapat menghasilkan desain perangkat lunak yang kompatibel dan sukses. Tanpa ini semua, desain akan terpecah menjadi beberapa pemikiran, tidak kompatibel dan menghancurkan sistem yang baik.

4. Perangkat lunak anda akan digunakan orang lain

   Perangkat lunak yang anda buat, akan digunakan oleh orang lain, baik untuk pemeliharaan, pendokumentasian maupun untuk memahami sistemnya. Oleh karena itu, saat membangun perangkat lunak, perhatikan siapa penggunanya, buatlah desain yang sesuai dengan pelaksananya, buatlah kode dengan memperhatikan orang yang melakukan pemeliharaan dan pengembangan sistem. Dengan mempermudah pekerjaan para pengguna maka perangkat lunak yang dibangun akan memiliki nilai tambah.

5. Terbuka terhadap masa depan

Perangkat lunak yang dapat beradaptasi dengan segala perubahan di masa depan merupakan perangkat lunak yang dapat bertahan lama. Perangkat lunak seperti inilah yang memiliki nilai lebih. Mengingat saat ini spesifikasi dan platform perangkat keras mengalami perkembangan dengan cepat dan apabila perangkat lunak tidak dirancang sejak awal untuk dapat beradaptasi maka masa pakai perangkat lunak pun hanya dalam hitungan bulan.

6. Rencanakan untuk ke depan sehingga perangkat lunak dapat digunakan kembali

Dalam pengembangan perangkat lunak, salah satu tujuannya adalah faktor anggaran biaya yang rendah. Dengan penggunaan kembali perangkat lunak tentunya dapat menghemat biaya dan tenaga. Pada pemrograman berorientasi objek, penggunaan kembali kode dan desain merupakan manfaat utama namun memerlukan pemikiran dan perencanaan. Ada banyak teknik yang dapat dilakukan untuk mewujudkan pengunaan kembali. Hal ini akan meningkatkan nilai dari perangkat lunak yang dibangun.

7. Berpikirlah

Berpikirlah sebelum bertindak, untuk mendapatkan hasil yang lebih baik. Dengan berpikir, maka dapat melakukan sesuatu dengan benar dan memperoleh pengetahuan, namun apabila melakukan kesalahan maka dapat menjadikannya sebagai pengalaman yang berharga.

Ketujuh prinsip-prinsip dasar diatas masih bersifat umum. Dalam penerapan rekayasa perangkat lunak, perlu adanya serangkaian prinsip detail sebagai panduan untuk memberikan cara menyelesaikan pekerjaan sehingga pendekatan yang kurang fokus menjadi lebih efektif dan mencapai hasil yang diinginkan(Pressman, 2010).

2.2 Prinsip-Prinsip Inti

Prinsip-prinsip inti dibagi menjadi dua yaitu prinsip-prinsip pemandu proses dan prinsip-prinsip pemandu praktek.

2.2.1 Prinsip-prinsip pemandu proses

Prinsip-prinsip inti pemandu proses berisi panduan dalam membuat kerangka kegiatan, mengarahkan aliran proses dan menghasilkan produk rekayasa perangkat lunak.

1. Menjadi cekatan
2. Berkualitas pada setiap tahap
3. Siap beradaptasi.
4. Membangun tim yang efektif
5. Membangun komunikasi dan kordinasi.
6. Mengelola perubahan.
7. Memperhitungkan tingkat resiko.
8. Menciptakan produk yang bermanfaat bagi orang lain

2.2.2 Prinsip-prinsip pemandu Praktik

Prinsip-prinsip inti pemandu praktik berisi panduan dalam menganalisis masalah, merancang solusi, menerapkan dan menguji solusi sampai menerapkan rekayasa perangkat lunak.

1. Berbagi tugas dan selesaikan.
2. Memahami penggunaan abstraksi.
3. Konsisten.
4. Fokus pada perpindahan informasi
5. Membangun perangkat lunak dengan modularitas efektif.
6. Mencari pola
7. Menunjukkan masalah dan solusinya menggunakan perspektif yang berbeda jika memungkinkan.
8. Ingatlah bahwa seseorang akan melakukan pemeliharaan terhadap perangkat lunak tersebut

2.3 Prinsip-Prinsip pemandu aktifitas framework

Prinsip-prinsip pemandu aktifitas framework merupakan pelengkap dan memiliki pengaruh terhadap keberhasilan kerangka kegiatan yang definisikan pada bagian proses rekayasa perangkat lunak.

2.3.1 Prinsip-prinsip Komunikasi

Komunikasi antara pelanggan, rekan teknis, pemanku kepentingan dan manajer proyek diperlukan untuk pemahaman bersama.

1. Dengarkan
2. Persiapkan sebelum berkomunikasi.
3. Seseorang harus memfasilitasi kegiatan komunikasi ini.
4. Komunikasi tatap muka adalah yang terbaik.
5. Catatlah dan dokumentasikan keputusan-keputusan yang dihasilkan.
6. Berusahalah untuk bisa berkolaborasi.
7. Tetap fokus; modulasikan diskusi Anda
8. Jika ada yang kurang jelas, gambarlah.
9. (a) Setelah Anda menyetujui sesuatu, lanjutkan. (b) Jika Anda tidak dapat menyetujui sesuatu, lanjutkan. (c) Jika suatu fitur atau fungsi tidak jelas dan tidak dapat diklarifikasi saat ini, lanjutkan
10. Negosiasi bukanlah kontes atau permainan. Yang terbaik bila kedua pihak menang

2.3.2 Prinsip-prinsip Perencanaan

Prinsip-prinsip perencanaan merupakan panduan untuk menentukan jalur menuju tujuan strategis dan taktisnya.

1. Pahami ruang lingkup proyek
2. Libatkan pemangku kepentingan dalam kegiatan perencanaan.
3. Kenali bahwa perencanaan bersifat iteratif
4. Perkirakan berdasarkan apa yang Anda ketahui.
5. Pertimbangkan risiko saat Anda menentukan rencana.
6. Jadilah Realistis
7. Sesuaikan perincian saat Anda menentukan rencana.
8. Tentukan bagaimana Anda ingin memastikan kualitas
9. Jelaskan bagaimana Anda mengakomodasi perubahan.
10. Lacak rencana secara teratur dan buat penyesuaian sesuai kebutuhan

2.3.3 Prinsip-prinsip Pemodelan

Prinsip-prinsip pemodelan merupkan panduan dalam membuat model yang mampu merepresentasikan informasi yang diubah oleh perangkat lunak, arsitektur dan fungsi yang memungkinkan transformasi, fitur yang diinginkan pengguna dan perilaku sistem

1. Tujuan utama tim perangkat lunak adalah membangun perangkat lunak, bukan membuat model.
2. Buatlah model sesuai yang Anda butuhkan.
3. Hasilkan model paling sederhana yang menggambarkan masalah atau perangkat lunak.
4. Bangun model dengan cara yang membuat mereka dapat menerima perubahan.
5. Mampu menyatakan tujuan yang jelas dari setiap model yang dibuat.
6. Sesuaikan model yang Anda kembangkan dengan sistem yang ada.
7. Cobalah untuk membuat model yang berguna, tetapi lupakan tentang membuat model yang sempurna.
8. Jangan menjadi dogmatis tentang sintaks model. Jika berhasil mengkomunikasikan konten, representasi adalah yang kedua.
9. Jika insting Anda memberi tahu Anda bahwa sebuah model tidak benar meskipun di atas kertas tampak baik-baik saja, Anda mungkin punya alasan untuk khawatir
10. Dapatkan umpan balik sesegera mungkin

2.3.4 Prinsip-prinsip Konstruksi

Prinsip-prinsip konstruksi merupakan paduan dalam melakukan serangkaian kegiatan kontruksi mencakup pengkodean dan pengujian rekayasa perangkat lunak.

1. Semua pengujian harus dapat dilacak ke persyaratan pelanggan
2. Tes harus direncanakan jauh sebelum tes dimulai
3. Prinsip Pareto berlaku untuk pengujian perangkat lunak

4. Pengujian harus dimulai “dalam hal yang kecil” dan berkembang menuju pengujian “dalam hal yang besar.
5. Pengujian menyeluruh tidak mungkin dilakukan

2.3.5 Prinsip-prinsip Deployment

1. Harapan pelanggan terhadap perangkat lunak harus dikelola.
2. Paket pengiriman lengkap harus dirakit dan diuji.
3. Rezim pendukung harus ditetapkan sebelum perangkat lunak dikirimkan.
4. Bahan instruksional yang sesuai harus disediakan untuk pengguna akhir.
5. Perangkat lunak yang memiliki *bug* harus diperbaiki dulu, dikirim kemudian

KONSEP MANAJEMEN PROYEK PERANGKAT LUNAK

Terdapat banyak teori dan panduan praktek tentang manajemen proyek yang berkembang di dunia, semua atau sebagian besar standar yang ada dan berkembang, telah dirangkum oleh Project Management Institute (PMI) ke dalam dokumen yang diberi judul Project Management Body of Knowledge (PMBOK). Oleh sebab itu isi panduan manajemen proyek yang ditulis dalam buku ini diarahkan atau difokuskan pada dasar teori dan praktek pada panduan PMBOK yang ditetapkan dan disusun oleh PMI, dengan didukung dengan referensi lainnya yang relevan.

Konsep proyek dalam PMBOK ini membahas konsep dasar pengelolaan proyek, yang akan disampaikan dalam tiga bagian yaitu:

1. Konsep dan elemen dasar proyek dan manajemen proyek.
2. Strategy pengelolaan proyek perangkat lunak
3. Proses Pelaksanaan pengelolaan proyek
4. Knowledge area manajemen proyek,
5. Process group manajemen proyek.
6. Process dalam pengelolaan proyek.

1. Konsep dan Elemen Dasar Proyek

Untuk memahami tentang manajemen proyek atau project management dapat dimulai dari memahami definisi dan pengertian tentang terminologi atau istilah proyek. Terdapat banyak definisi formal yang dikemukakan oleh para ahli dan organisasi profesi mengenai proyek, namun semua unsur intinya telah di rangkum kedalam definisi yang dibuat oleh Project Management Institue atau PMI yang dalam buku panduan Project Management Body Of Knowledge atau dikenal dengan PMBOK. Panduan ini akan digunakan secara luas sebagai standar dalam pengelolaan proyek. Penegertian dasar tentang proyek :

a. Menurut PMBOK suatu Proyek didefinisikan suatu kegiatan atau usaha yang bersifat sementara (*temporary*), yang dilakukan untuk menciptakan atau menghasilkan suatu keluaran dalam bentuk produk atau layanan atau hasil lainnya yang bersifat '*unique*'. Dari definisi tersebut suatu proyek adalah kegiatan yang bersifat sementara, yang artinya mempunyai durasi tertentu, dengan waktu mulai dan waktu selesai yang terdefinisi dengan pasti atau tertentu. suatu Proyek dibuat untuk menghasilkan produk atau layanan atau hasil tertentu yang bersifat unik, artinya berbeda dengan yang lainnya. suatu Proyek akan berhenti atau selesai pada saat tujuan yang ditetapkan telah tercapai.
b. Menurut Harvard Business School (HBS) suatu proyek merupakan Sekumpulan kegiatan unik untuk menghasilkan suatu hasil yang terdefinisi dengan waktu mulai dan waktu selesai, dengan alokasi sumber daya yang spesifik – ''*a unique set activities that are meant to produce a defined outcome, with a specific start and finish date, and a specific allocation of resources*'). Dari definisi ini juga dapat dijelaskan secara lebih rinci bahwa suatu proyek mempuyai kegiatan dan hasil yang bersifat unik, suatu proyek mempunyai durasi, waktu mulai dan waktu selesai, yang tertedefinisi. suatu Proyek mempunyai sumberdaya yang bersifat khusus dan terbatas.
c. Menurut Management Study Guide (MSG), Suatu proyek adalah kegiatan untuk mencapai penciptaan sebuah produk atau layanan yang unik – *A project is an activity to meet the creation of a unique product or service and thus activities that are undertaken to accomplish routine activities cannot be considered projects*. Menurut defiisi MSG proyek merupakan kegiatan khusus untuk menghasilkn produk atau layanan yang bersifat unik yang tidak dapat dicapai oleh kegiatan operasonal yang bersifat kegiatan rutin.

Agar dapat memahami tetang proyek secara komprehensif dapat dilakukan melalui pemahaman terhadap karakteristik yang dimiliki oleh suatu proyek. Dari beberapa definisi diatas dapat dirangkum bahwa suatu proyek dapat mempunyai pengertian yang beragam jika dilihat dari berbagai sudut pandang namun setiap definisi tersebut mempunyai esensi atau pengertian inti yang sama, sehingga proyek juga mempunyai karakteristik yang beragam namun serupa. PMI telah

merangkum karakteristik setiap Proyek menjadi beberapa sifat umum. Suatu proyek harus mempunyai sifat berikut:

A. Bersifat Temporer (temporary); Suatu Proyek adalah kegiatan yang bersifat temporer atau sementara, yang artinya suatu proyek bukan pekerjaan yang bersifat rutin dan berkelanjutan. Suatu proyek mempunyai durasi yang pasti, mempunyai waktu mulai dan selesai yang ditetapkan secara pasti, dan proyek bukan merupakan kegiatan rutin yang bersifat berulang-ulang dan terus menerus. Temporer mempunyai pengertian dilakukan dalam periode tertentu atau dikenal dengan durasi. Durasi proyek merupakan waktu total, sejak proyek dimulai sampai dengan proyek selesai , yang ditetapkan untuk suatu proyek. Durasi proyek tidak harus pendek atau panjang, tetapi yang pasti harus ditetapkan secara pasti. Durasi proyek secara umum dinyatakan secara implisit tetapi tidak tepat/ persis secara matematis dengan penetapan waktu mulai dan waktu berakhir, artinya durasi suatu proyek tidak selalu sama dengan waktu selesai dikurangi waktu mulai. Total Durasi proyek tidak harus sesuai dengan waktu kalender , karena ada waktu tertentu (waktu libur/ tenggat) yang bisa tidak dihitung dalam total waktu durasi proyek atau adanya waktu lembur (overtime). Suatu proyek harus berhenti pada saat tujuan yang ditetapkan telah dicapai. Suatu proyek dapat berakhir atau selesai dengan keadaan atau status sukses atau gagal. Proyek akan dinilai berakhir dengan sukses jika semua obyektip/ tujuan yang ditetapkan dalam setiap kegiatan yang ditetapka untuk proyek tersebut telah dicapai semuanya dengan baik, sedangkan proyek yang dinilai gagal jika secara jelas dalam keadaan normal , obyektip yang ditetapkan tidak dapat dicapai sesuai dengan rencana scope, schedule, dan budget, serta kualitas, atau proyek dapat dihentikan jika terjadi kondisi dimana obyektip/ tujuan yang ditetapkan dalam proyek tidak lagi bermanfaat sehingga proyek tidak dilanjutkan atau proyek dihentikan (terminated).

B. Bersifat Unik (Unique); Proyek bersifat Unik karena Suatu proyek harus menghasilkan suatu deliverables atau hasil yang bersifat unik dalam bentuk produk , layanan , atau hasil bentuk lainnya. Yang dimaksud dengan unik adalah tidak sama dan serupa antara

proyek satu dengan lainnya, harus ada pembeda antara proyek satu dan proyek lainnya. Salah satu bentuk pembeda yang membentuk keunikan suatu proyek diantaranya adalah adanya 'inovasi', teknik pelaksanaan, waktu pelaksanaan, spesifikasi produk yang dihasilkan, dalam kegiatan proyek tersebut. Suatu proyek dapat menghasilkan tiga bentuk deliverables yaitu:

1. Suatu produk atau artifak; Produk atau artifak yang dihasilkan oleh suatu proyek harus mempunyai sifat terukur (quantifiable).
2. Produk akhir / jadi (finished product); Hasil jenis ini merupakan produk atau barang yang siap digunakan oleh pemakai dalam kehidupan sehari-hari. Dari sudut pandang oganisasi atau prusahaan pemilik proyek produk yang dihasilkan olh proyek siap untuk digunakan atau dijual atau dipasarkan. Contoh produk jadi adalah antara lain rumah, mobil, motor, pesawat terbang, makanan jadi, dan produk sejenis lainnya.
3. Barang setengah jadi; Hasil jenis ini merupakan produk yang belum siap digunakan oleh pemakai, dan untuk memanfaatkannya diperlukan proses tambahan untuk menjadi produk jadi yang siap digunakan oleh pemakai. Dari susdut pandang proyek hasil ini merupakan produk final, tetapi dari sudut pandang oragnisasi pemilik proyek atau konsumen, produk ini merupakan bagian dari produk lainnya yang harus digabung untuk menjadi suatu produk final. Contoh produk setengah jadi antara lain makanan mentah (frozen food), meja belum di cat, mobil truk belum memakai bak,, dan produk lain sejenisnya.
4. Suatu kemampuan untuk menjalankan services ; Hasil jenis ini berupa produk yang memberikan nilai tambah bagi kegiatan lainnya, dokumen panduan, skill atau kemampuan menjalankan fungsi bisnis dalam mendukung pelaksanaan proses bisnis, produksi atau distribusi barang, atau kegiatan lainnya. Contoh hasil jenis ini antara lain panduan kerja, SOP, ketrampilan, dan produk sejeenis lainnya
5. Suatu hasil seperti outcomes atau dokumen mengenai trend atau proses baru, yang dapat digunakan untuk meningkatkan kinerja atau kesejahteraan masayarakat. Hasil jenis ini berupa impact

atau pengaruh yang timbul dari adanya hasil yang diperoleh dari suatu proyek. Contoh akibat dibangunnya jembatan diatas sungai pada lokasi terpencil maka dapat berdampak terhadap kelancaran transportasi yang dapat meningkatkan kegiatan ekonomi dan kesejahteraan masarakat, kemudahan menuju menuju sekolah di kampung seberang sehingga menigkatkan tingkat pendidikan masyarakat, dll.

C. Bersifat Bertahap (Progressive elaboration); Sifat ini merupakan karakteristik proyek yang mendukung karakteristik lainnya yaitu sifat temporary dan unik. Progressive elaboration artinya pengembangan atau pelaksanaan suatu proyek dilakukan secara tahap demi tahap, dan berkelanjutan selama durasi proyek. Pelaksanaan proyek dilaksanakan berdasarkan tahapan yang telah ditetapkan, sesuai dengan tahapan dalam fase pengembangan dan tahapan kelompok proses atau methodologi pengelolaan proyek yang digunakan.Suatu proyek tidak dapat dilakukan secara mendadak waktu dalam satu yang bersamaan atau dalam bahasa umum dikatakan sekaligus jadi. Hal ini disebabkan karena suatu proyek harus dikembangkan secara bertahap, sesuai tahapan pengembangan produk dan layanan yang dibuat dalam proyek, tahapan pengetahuan dan ketrampilan SDM yang dibutuhkan dalam proyek, dan penyerapan dana yang dianggarkan, serta pertimbangan lainnya. Waktu penyelesaian setiap kegiatan dalam proyek disusun sedemikian rupa, sesuai tahapan dan urutannya, ada beberapa pekerjaan dilaksanakan secara berurutan (serial) dan ada beberapa pekerjaan yang dilaksanakan secara paralel dalam waktu yang bersamaan. Setiap kegiatan atau tahap kegiatan diperlukan pengetahuan, ketrampilan, teknik yang berbeda, sehingga diperlukan banyak orang yang bekerja sesuai dengan tahap penugasannya, jenis keahlian, dan ketrampilannya. Suatu proyek memerlukan pembiayaan, yang disediakan secara bertahap karena tidak mungkin diserap sekaligus dalam satu waktu. Karena penyerapan biaya disesuaikan dengan rencana anggaran yang disusun sesuai dengan rencana progress dari setiap kegiatan tertentu yang ditetapkan.

D. Sumber daya terbatas; Karakteristik lain yang perlu ditambahkan adalah bahwa semua proyek dilaksanakan dengan sumberdaya yang terbatas, artinya baik waktu tersedia, dana yang dianggarkan, tenaga yang terlibat, dan scope yang dilaksanakan dalam suatu kegiatan proyek adalah terbatas/ tidak unlimited, maka dari itu pelaksanaan proyek membutuhkan tahapan kegiatan, metoda pelaksanaan, dan waktu yang tepat, serta ketrampilan yang sesuai kebutuhan. Dengan kata lain karena sumber daya yang terbatas, untuk menyelesaikan proyek dengan sukses diperlukan tool, teknik, kemampuan, pengalaman, dan manajemen pengelolahan proyek atau yang dikenal dengan project management. Ketersediaan sumber daya proyek yang terbatas, sangat krusial untuk diperhatikan jika terjadi ketidak susksesan penyelesaian proyek karena kelebihan waktu, kekurangan biaya, kekurangan peralatan, dan kekurangan SDM trampil, maka dapat dikatakan proyek tersebut gagal atau tidak berhasil.

1. Proyek vs pekerjaan rutin.

Untuk dapat memahami mengeenai pekerjaan rutin, dapat dilakukan pembandingan dengan pekerjaan jenis lainnya yaitu pekerjaan yang bersifat proyek. Terdapat perbedaan dan kesamaan antara kegiatan proyek dan kegiatan rutin.

Perbedaan pekerjaan rutin vs proyek;

1. Perbedaan cara kerja atau operasional; Kegiatan rutin dilakukan secara dan/atau bersifat berulang–ulang dan berkelanjutan, sementara kegiatan proyek bersifat sementara dan unik.
2. Perbedaan sasaran atau obejctives; Suatu proyek bertujuan untuk mencapai objective yang ditetapkan, dan bila obyektip telah tercapai maka proyek selesai. Dengan Demikian suatu Proyek akan berhenti jika objektip khusus yang ditetapkan telah dicapai, Sebaliknya kegiatan rutin bertujuan mempertahankan kelanjutan business. Pada Kegiatan rutin, jika objektive telah dicapai, akan menetapkan objektip atau tujuan baru, dan kegiatan yang sama akan berlanjut atau berulang.
3. Kegiatan rutin pada saat mencapai obyektip yang telah ditetapkan, akan dilakukan pengulangan untuk obyektip yang sama atau menciptakan obyektip baru pada frame waktu yang

berbeda. Sebagai contoh: kegiatan akademik mahasiswa yang dilakukan oleh dosen atau tata usaha, mereka mengerjakan kegiatan pengajaran pada semester genap tahun 2009, pada saat kegiatan selesai dilaksanakan maka mereka akan melaksanaan kegiatan yang sama atau sejenis pada semester berikutnya. Proses ini dilakukan berulang-ulang dari tahun ke tahun selama proses perkuliahan berlangsung. Sedangkan kegiatan proyek pada saat obyektip dicapai maka kegiatan berhenti. Sebagai Contoh kegiatan proyek adalah kegiatan penerimaan mahasiswa, kegiatan ini dilakukan dalam waktu tertentu dengan team tertentu. Pada saat kegiatan selesai maka semua kegiatan ditutup, team dibubarkan, termasuk biaya dihentikan.

Persamaan pekerjaan rutin vs proyek

Disamping perbedaan yang ada, terdapat kesamaan antara proyek dan kegiatan rutin diantaranya yaitu :

1. Memerlukan SDM; Baik proyek maupun kegiatan rutin keduanya dilaksanakan oleh Orang/ pekerja (SDM), kedua kegiatan tersebut harus dilaksanakan oleh tenaga yang kompeten dan berkualitas agar pencapaian obyektip dapat dilakukan dengan baik.
2. Resources terbatas; Baik proyek maupun kegiatan rutin keduanya terkendala oleh resources atau sumberdaya yang terbatas. Baik kegiatan rutin maupun proyek dilakukan dengan kendala tersedianya resource s baik berupa tenaga kerja, dana, waktu dan hasil yang ingin dicapai, yang terbatas, disesuaikan dengan kebutuhan di masing-masing kegiatan.
3. Terencana / Well planned; Baik proyek maupun kegiatan rutin keduanya direncanakan (planned). Baik proyek maupun kegiatan rutin harus direncanakan dengan baik jika ingin pencapaian masing-masing obyektip dapat dilakukan secara baik, jika perencanaannya lemah maka pencapaian obyektip tidak dapat dilakukan secara optimal.
4. Executed; Baik proyek maupun kegiatan rutin keduanya dilaksanakan. Kedua kegiatan harus dilaksanakan secara baik, jika tidak maka pencapain obyektip akan dapat dilakukan atau dengan kata lain gagal.

5. Controlled, Baik proyek maupun kegiatan rutin keduanya dikendalikan. Kedua kegiatan tersebut harus dilakukan pengontrolan agar proses pencapaian obyektip dapat dilakukan sesuai rencana dan koridor waktu, dan biaya yang ditetapkan.

2. Proses Pelaksanaan Pengelolaan Proyek

Project Life Cycle

3. Project Manager

Knowledge Area Manajemen Proyek

5. Process Group Manajemen Proyek.

PERENCANAAN PROYEK

PERENCANAAN PROYEK

Perencanaan proyek memiliki beberapa tahapan sebagai berikut. Fase proposal adalah fase perjanjian atau kontrak perjanjian untuk melakukan serangkaian prose pembangunan maupun pengembangan perangkat lunak. Kemudian fase permulaan yaitu mulai menentukan secara detail mengenai siapa saja yang akan menjadi anggota tim, pengalokasian sumber daya sehingga informasi sudah lebih lengkap dibandingkan ketika tahap proposal, sekaligus juga merupakan tahap untuk mempersiapkan perkiraan perkiraan sebagai awalan. Kemudian fase pemantauan secara berkala sepanjang proyek yaitu perencanaan dapat selalu dilakukan perubahan terbaru ketika ada informasi ban terkait perangkat lunak dan proses pengembangan atau pembangunannya. Hal-hal yang harus masuk dalam perencanaan proyek adalah pernyataan kerja yang merepresentasikan produk, daftar semua sumber daya (*resources*) yang dibutuhkan untuk pada rekayasa perangkat lunak beserta ketersediaannya, rincian struktur pekerjaan dan kumpulan perkiraan pekerjaan yang dilakukan selama proses rekayasa perangkat lunak, perencanaan jadwal dan biaya proyek, perencanaan risiko. Sebuat proyek dikatakan berhasil jika mampu mengirimkan produk yang berkualitas tinggi ada waktu dan biaya yang telah ditentukan sebelumnya.

Sinopsis

Perencanaan proyek memiliki beberapa tahapan sebagai berikut. Fase proposal adalah fase perjanjian atau kontrak perjanjian untuk melakukan serangkaian prose pembangunan maupun pengembangan perangkat lunak. Kemudian fase permulaan yaitu mulai menentukan secara detail mengenai siapa saja yang akan menjadi anggota tim, pengalokasian sumber daya sehingga informasi sudah lebih lengkap dibandingkan ketika tahap proposal, sekaligus juga merupakan tahap untuk mempersiapkan perkiraan perkiraan sebagai awalan. Kemudian fase pemantauan secara berkala sepanjang proyek yaitu perencanaan dapat selalu dilakukan perubahan terbaru ketika ada informasi ban terkait perangkat lunak dan proses pengembangan atau pembangunannya. Hal-hal yang harus masuk dalam perencanaan proyek adalah pernyataan kerja yang merepresentasikan produk, daftar semua sumber daya (*resources*) yang dibutuhkan untuk pada rekayasa perangkat lunak beserta ketersediaannya, rincian struktur pekerjaan dan kumpulan perkiraan pekerjaan yang dilakukan selama proses rekayasa perangkat lunak, perencanaan jadwal dan biaya proyek, perencanaan risiko. Sebuat proyek dikatakan berhasil jika mampu mengirimkan produk yang berkualitas tinggi ada waktu dan biaya yang telah ditentukan sebelumnya.

Definisi Perencanaan Proyek

Perencanaan Proyek adalah Perencanaan Proyek dalam konteks rekayasa perangkat lunak merujuk pada proses identifikasi dan pengaturan langkah-langkah yang diperlukan untuk merencanakan, mengelola, dan mengendalikan pengembangan perangkat lunak yang efektif dan efisien. Perencanaan proyek ini melibatkan pemahaman yang mendalam tentang tujuan proyek, ruang lingkup, sumber daya yang tersedia, jadwal, risiko, dan tugas-tugas yang harus dilakukan.

Perencanaan Proyek adalah proses menyusun rencana terperinci yang mencakup tujuan, jadwal, sumber daya, risiko, dan tugas-tugas yang akan dilakukan dalam suatu proyek. Ini melibatkan pemikiran sistematis dan analisis yang mendalam untuk mengidentifikasi langkah-langkah yang diperlukan, memperkirakan sumber daya yang diperlukan, dan merencanakan urutan kegiatan untuk mencapai tujuan proyek.

Perencanaan Proyek dapat juga diartikan sebagai proses sistematis untuk menentukan tujuan, mengidentifikasi langkah-langkah yang diperlukan, mengatur sumber daya, dan merencanakan jadwal dalam rangka mencapai tujuan proyek secara efektif. Ini melibatkan mengidentifikasi semua aktivitas yang perlu dilakukan, memperkirakan waktu dan sumber daya yang diperlukan untuk setiap aktivitas, serta menyusun rencana untuk mengelola risiko dan mengatasi hambatan yang mungkin muncul selama proyek berlangsung.

Berikut adalah beberapa langkah penting dalam perencanaan proyek perangkat lunak meliputi:

1. Identifikasi Kebutuhan: Memahami kebutuhan pengguna dan kebutuhan bisnis adalah langkah awal dalam perencanaan proyek. Ini melibatkan pengumpulan persyaratan, memahami lingkungan pengguna, dan mengidentifikasi masalah yang harus diselesaikan dengan perangkat lunak.
2. Penentuan Tujuan: Menentukan tujuan proyek yang jelas dan terukur sangat penting. Tujuan harus spesifik, terukur, dapat dicapai, relevan, dan berbatasan waktu (SMART). Misalnya, mengembangkan sistem manajemen inventaris yang dapat mengurangi kesalahan stok sebesar 20% dalam waktu 6 bulan.

3. Penyusunan Rencana Proyek: Rencana proyek mencakup jadwal kegiatan, alokasi sumber daya, dan estimasi biaya dan waktu. Ini melibatkan identifikasi tugas-tugas yang harus diselesaikan, pemetaan dependensi antar tugas, dan penentuan sumber daya yang dibutuhkan, seperti anggota tim, perangkat keras, dan perangkat lunak.
4. Estimasi Waktu dan Biaya: Membuat perkiraan yang realistis untuk waktu yang diperlukan dan biaya yang terkait dengan proyek. Ini melibatkan pemilihan metode estimasi yang sesuai, seperti estimasi berbasis pengalaman, analisis perbandingan, atau penggunaan teknik pemecahan masalah.
5. Penjadwalan Tugas: Menentukan urutan tugas, mengatur prioritas, dan membuat jadwal kegiatan yang rinci. Menggunakan alat bantu manajemen proyek seperti diagram Gantt atau jadwal PERT/CPM dapat membantu menggambarkan dan memvisualisasikan hubungan antar tugas serta mengelola jadwal proyek.
6. Pengelolaan Risiko: Mengidentifikasi dan mengevaluasi risiko yang mungkin terjadi selama proyek. Ini melibatkan mengidentifikasi risiko potensial, mengevaluasi dampaknya, dan mengembangkan strategi mitigasi untuk mengurangi atau menghindari risiko tersebut.
7. Pengawasan dan Pengendalian: Melakukan pemantauan terus-menerus terhadap proyek, termasuk pemantauan kemajuan, penyebaran tugas, pemantauan biaya, dan penanganan perubahan yang terjadi. Hal ini memungkinkan untuk mengidentifikasi masalah sejak dini dan mengambil tindakan korektif yang diperlukan.
8. Manajemen Tim: Merencanakan dan mengatur anggota tim proyek, termasuk menentukan peran dan tanggung jawab mereka. Ini melibatkan pemilihan tim yang sesuai dengan keahlian dan pengalaman yang diperlukan, serta memastikan komunikasi dan kolaborasi yang efektif di antara anggota tim.
9. Komunikasi Proyek: Mengembangkan strategi komunikasi yang efektif untuk memastikan informasi proyek yang tepat disampaikan kepada semua pemangku kepentingan. Ini meliputi pertemuan rutin, laporan kemajuan, dan komunikasi yang terbuka

dan transparan untuk memastikan pemahaman yang baik antara semua pihak terkait.

10. Pengendalian Perubahan: Mengelola perubahan yang terjadi selama siklus proyek. Ini melibatkan identifikasi, evaluasi, dan pengelolaan perubahan dalam persyaratan, jadwal, atau lingkup proyek yang mungkin terjadi. Proses ini juga mencakup pengambilan keputusan tentang penerimaan atau penolakan perubahan dan dampaknya pada proyek secara keseluruhan.
11. Evaluasi Proyek: Melakukan evaluasi berkala terhadap proyek untuk memantau kemajuan, kualitas, dan kepatuhan terhadap tujuan proyek. Evaluasi ini membantu mengidentifikasi masalah atau ketidaksesuaian yang mungkin timbul selama proyek, sehingga tindakan perbaikan dapat diambil sesuai kebutuhan.
12. Penyelesaian Proyek dan Penyerahan: Menyelesaikan semua tugas, menguji dan memverifikasi perangkat lunak, serta mempersiapkan dokumentasi dan pelatihan yang diperlukan. Pada tahap ini, proyek siap untuk diserahkan kepada pengguna atau pemangku kepentingan yang relevan.

Perencanaan proyek yang baik membantu mengatur dan mengarahkan jalannya proyek secara efisien, meminimalkan risiko, dan meningkatkan peluang keberhasilan. Ini juga membantu dalam mengelola sumber daya dengan lebih baik, mengoptimalkan jadwal, dan menghadapi perubahan yang mungkin terjadi selama proyek. Dengan perencanaan yang matang, pelaksanaan proyek perangkat lunak dapat dilakukan dengan lebih teratur, efisien, dan berhasil.

Perencanaan proyek biasanya terdiri atas:

a. organisasi proyek - pengorganisasian segala hal yang terkait dengan proyek sehingga saat berjalannya proyek akan lebih teratur dan terogarnisir dengan baik;
b. analisis risiko - melakukan perencanaan terhadap risiko-risiko yang mungkin terjadi dan dibuat rencana solusinya;
c. kebutuhan perangkat keras dan perangkat lunak pembagian kerja
d. pembagian- perencanaan kerja perlu dilakukan agar pengawasan dan peninjauan pekerjaan dapat secara jelas dilakukan dan terkendali pada setiap tahapan proyek;

e. penjadwalan proyek;
f. mekanisme pengawasan dan pelaporan.

Perencanaan dalam proyek perangkat lunak adalah salah satu hal yang sangat penting dan menjadi pekerjaan dari seorang manager proyek. Seorang manajer perencanaan proyek perangkat lunak harus memiliki sejumlah kemampuan dan kompetensi yang diperlukan untuk melaksanakan tugasnya dengan baik. Perencanaan proyek memiliki beberapa tahapan sebagai berikut:

A. Fase Proposal

Fase Proposal dalam perencanaan perangkat lunak adalah tahap awal dalam pengembangan proyek perangkat lunak di mana proposal proyek disusun untuk menggambarkan konsep, tujuan, dan ruang lingkup proyek kepada pihak-pihak terkait. Fase ini adalah fase perjanjian atau kontrak perjanjian untuk melakukan serangkaian proses pembangunan maupun pengembangan perangkat lunak. Pada tahapan ini maka ditentukan hal-hal terkait sumber daya, jadwal, biaya, dan spesif-kasi kebutuhan secara garis besar dari fungsi-fungsi yang akan dibuat di dalam perangkat lunak nantinya. Semua hal itu sebaiknya dibuat sejelas mungkin pada perjanjian yang dibuat sehingga tidak ada pihak yang akan dirugikan, misalkan mengenai waktu jangan sampai terjadi perpanjangan waktu karena spesifkasi kebutuhan yang terus bertambah tanpa diiringi dengan bertambahnya pembiayaan. Pada tahapan ini Fase ini bertujuan untuk mendapatkan persetujuan dan dukungan untuk melanjutkan pengembangan proyek.

B. Fase Permulaan

Fase Permulaan (Inception Phase) dalam perencanaan proyek dalam rekayasa perangkat lunak merupakan tahap awal dalam siklus pengembangan perangkat lunak di mana tujuan utama adalah untuk memahami dan mendefinisikan proyek dengan lebih rinci. Fase ini melibatkan analisis awal, penentuan kebutuhan, dan penyusunan rencana proyek yang lebih terperinci. Tahapan ini mulai menentukan secara detail mengenai siapa saja yang akan menjadi anggota tim, pengalokasian somber daya sehingga inforrnasi sudah lebih lengkap dibandingkan ketika tahap proposal, sekaligus juga merupakan tahap

untuk mempersiapkan perkiraan-perkiraan sebagai awalan. Tujuan dari fase Permulaan adalah untuk mengidentifikasi masalah, menyusun strategi, dan menentukan apakah proyek layak untuk dilanjutkan. Berikut adalah beberapa langkah yang umumnya dilakukan dalam fase Permulaan:

1. Analisis Awal: Tahap ini melibatkan analisis awal terhadap masalah yang ingin diselesaikan atau peluang yang ingin dimanfaatkan dengan pengembangan perangkat lunak. Tim proyek harus memahami latar belakang, tujuan, dan konteks bisnis proyek. Analisis awal ini membantu dalam menetapkan visi dan arah proyek.
2. Penentuan Kebutuhan: Tim proyek harus melakukan analisis kebutuhan dengan berkomunikasi dengan pemangku kepentingan dan pengguna potensial. Kebutuhan fungsional dan non-fungsional yang harus dipenuhi oleh perangkat lunak ditentukan. Ini melibatkan pengumpulan persyaratan, pemahaman terhadap proses bisnis yang ada, serta identifikasi fitur dan fungsi yang diperlukan.
3. Penyusunan Rencana Proyek: Rencana proyek yang lebih terperinci disusun dalam fase ini. Rencana ini mencakup jadwal kegiatan, alokasi sumber daya, estimasi biaya dan waktu, serta identifikasi risiko awal. Rencana proyek membantu dalam mengatur dan mengarahkan jalannya proyek secara efisien serta memberikan kerangka kerja yang jelas bagi tim proyek.
4. Penentuan Arsitektur: Tahap ini melibatkan penentuan arsitektur perangkat lunak yang akan digunakan dalam proyek. Arsitektur melibatkan pemilihan teknologi, desain sistem, dan pengorganisasian komponen perangkat lunak. Penentuan arsitektur yang baik memastikan bahwa proyek memiliki landasan yang kokoh untuk pengembangan perangkat lunak yang efektif.
5. Pembuatan Prototype: Dalam beberapa kasus, dalam fase Permulaan, tim proyek dapat membuat prototipe awal perangkat lunak. Prototipe ini membantu dalam memvalidasi kebutuhan dan memahami persyaratan lebih baik sebelum pengembangan perangkat lunak yang sebenarnya dimulai.

6. Penentuan Lingkup: Pada tahap ini, lingkup proyek yang lebih rinci ditentukan. Batasan dan prioritas fitur dan fungsi perangkat lunak ditetapkan. Hal ini membantu dalam mengelola harapan dan memastikan bahwa proyek tetap fokus pada tujuan yang telah ditentukan.
7. Penilaian Kelayakan: Evaluasi kelayakan proyek dilakukan untuk menentukan apakah proyek ini layak dilanjutkan.

C. Fase Pemantauan Secara Berkala

Pemantauan proyek secara berkala merupakan praktik yang penting untuk memastikan kelancaran proyek. Pemantauan proyek secara berkala melibatkan kegiatan terus-menerus untuk memeriksa kemajuan proyek, mengidentifikasi masalah atau hambatan yang muncul, dan mengambil tindakan yang diperlukan untuk mengatasi permasalahan tersebut. Perencanaan dapat selalu dilakukan perubahan terbaru ketika ada informasi baru terkait perangkat lunak dan proses pengembangan atau pembangunannya. Pengelola proyek akan melihat bagaimana perangkat lunak diimplementasikan dan kemampuan dart tim pengembang terkait pula dengan perubahan-perubahan kebutuhan perangkat. Tujuan utamanya adalah untuk memastikan bahwa proyek berjalan sesuai rencana, waktu, dan anggaran yang telah ditetapkan. Hal ini akan dapat terus memperbarui perencanaan jadwal penyelesaian proyek pada setiap tahapnya. Hal-hal yang harus masuk dalam perencanaan proyek adalah sebagai berikut.

1. Pernyataan kerja yang merepresentasikan produk seperti spesifikasi perangkat lunak (ruang lingkup perangkat lunak), rencana pengujian perangkat lunak, perencanaan kode program perangkat lunak, perencanaan lapor kesalahan kecacatan (defect), dan semua perencanaan pekerjaan selama proyek berjalan. Ruang lingkup perangkat lunak yang diperlukan diantaranya adalah:
   a. data dan kendali/kontrol yang perlu untuk diproses;
   b. fungsi-fungsi yang diperlukan ada di dalam perangkat lunak;
   c. performansi perangkat lunak, misalnya terkait waktu respon;
   d. aturan dan batasan yang perlu ada dalam perangkat lunak;
   e. antarrnuka atau tampilan seperti apa yang diperlukan dari perangkat lunak

f. kehandalan seperti apa yang dibutuhkan perangkat lunak, misalnya apakah akan dipakai di satu tempat atau banyak tempat, dapat digunakan oleh banyak pengguna secara bersama-sama, dan lainnya.

2. Daftar semua sumber daya (*resources*) yang dibutuhkan pada rekayasa perangkat lunak beserta ketersediaannya. Semua kemungkinan sumber daya harus diperhitungkan dengan detail, masalah kemampuan sumber daya dan ketersediaannya. Semakin banyak sumber daya yang dilibatkan dengan kemampuan yang kurang maka tidak akan membuat proyek berhasil, tapi kekurangan sumber daya juga dapat memperburuk kondisi proyek. Terkadang lebih baik melibatkan sumber daya secukupnya dengan kemampuan sesuai dengan yang dibutuhkan untuk menjalankan proyek. Sumber daya disini dapat berupa sumber daya manusia ataupun perangkat kerja.

   Berikut ini adalah pengelompokan sumber daya yang perlu direncanakan dalam pembuatan perangkat lunak.

   a. Sumber daya manusia - perencanaannya meliputi kemampuan (*skill*) yang dibutuhkan diberbagai posisi misalkan sebagai manajer, *software engineer*, ataupun posisi dengan kemampuan lain seperti telekomunikasi, basis data, jaringan, dan lainnya.
   b. Sumber daya komponen perangkat lunak pendukung yang dapat digunakan kembali (*reusable software components*) - sumber daya ini memiliki empat kategori. ***Off-the-shelf component***- merupakan perangkat lunak yang sudah ada dan dapat digunakan, dapat dihasilkan dari pengembangan proyek sebelumnya ataupun dibeli dari pihak ketiga dan sudah dipastikan bahwa perangkat lunak ini telah tervalidasi untuk digunakan. ***Full-experience components*** - merupakan perangkat lunak sejenis ataupun bagian perangkat lunak yang dapat dimanfaatkan yang telah dikembangkan sebelumnya. ***Partial-experience components*** - merupakan perangkat lunak sejenis ataupun bagian perangkat lunak yang dapat dimanfaatkan yang telah dikembangkan sebelumnya namun masih memerlukan modifikasi agar dapat digunakan pada proyek saat ini. ***New components*** - komponen perangkat lunak yang perlu untuk dibangun dari awal di proyek saat ini.

c. Sumber daya lingkungan - lingkungan yang mendukung proyek perangka lunak (*Software Engineering Environment* (SEE) termasuk didalamnya perangkat keras (*hardware*) dan perangkat lunak (*software*).

Tingkatan akan sumber daya dalam proyek perangkat lunak diilustrasikan dalam piramida sumber daya proyek dimana sumber daya manusia memiliki peran paling tinggi didalam piramida seperti gambar berikut

Gambar. 3.1 Piramida Sumber Daya Perangkat Lunak

3. Rincian struktur pekerjaan dan kumpulan perkiraan pekerjaan yang dilakukan selama proses rekayasa perangkat lunak. Rincian ini juga harus dilakukan secara detail, jangan sampai ada sebuah pekerjaan yang tidak diprediksi harus dikerjakan di tengah proyek, hal ini dapat menyebabkan kebingungan siapakah yang harus bertanggung jawab pada pekerjaan itu. Jika sumber daya manusia untuk pekerjaan lain dipekerjakan untuk mengerjakan pekerjaan yang tiba-tiba muncul maka pekerjaan lain akan menjadi terbengkalai dan proyek menjadi molor.
4. Perencanaan jadwal dan biaya proyek. Hal ini harus didasari dengan perencanaan yang sangat detail dan estimasi yang tepat. Jika perencanaan jadwal tidak tepat maka bisa jadi jadwal yang telah ditetapkan menjadi molor pada kenyataannya. Molornya jadwal dapat berimbas pada membengkaknya biaya (terutama jika mempergunakan pekerja *outsourcing*) dan rusaknya kredibilitas atau reputasi pekerja proyek perangkat lunak. Maka dari itu rincian penjadwalan atas pekerjaan yang harus dikerjakan perlu dilakukan sedetail mungkin. Begitu pula jika terjadi pembengkakan biaya

maka akan terjadi kebingungan siapakah yang harus menanggung biaya sedangkan pelanggan (*customer*) berpegang pada kontrak kerja di awal proyek yang biasanya berisi biaya yang harus dibayar.

5. Perencanaan risiko yang mengidentifikasi setiap risiko yang mungkin dihadapi dan merencanakan bagaimana risiko itu akan ditangani. Rincian daftar risiko harus diidentifikasi sedetail mungkin agar jika risiko terjadi tidak menjadi kebingungan menanganinya. Risiko biasanya ditulis dalam sebuah daftar risiko yang berhasil diidentifikasi, untuk setiap risiko diperkirakan kemungkinan terjadinya atau ditentukan sekala prioritas terjadinya, lalu dibuatlah penanganan untuk setiap risiko yang berhasil diindentif~kasi. Penanganan setiap risiko dapat direncanakan lebih dari satu penanganan untuk mengatasi jika terjadi kegagalan penanganan.

Sebuah proyek dikatakan berhasil jika mampu mengirimkan produk yang berkualitas tinggi ada waktu dan biaya yang telah ditentukan sebelumnya. Bagi seorang manajer proyek akan dianggap berhasil jika dapat memotivasi dan mengajak orangorang yang terkait proyek dapat berkerjasama sebagai tim yang efektif dan memfokuskan perhatian pada kebutuhan pelanggan (*costomer*) dan kualitas produk. Perencanaan proyek membutuhkan kemampuan menganalisis kondisi proyek perangkat lunak dari hal yang khusus hingga yang paling umum dan hal-hal ini biasanya didapatkan dari pengalaman berupa hal-hal yang bisa dipelajarai dari proyek-proyek sebelumnya yang pernah dilakukan (*lesson learned*) dan juga dapat dengan mempelajari banyak best practices yang telah dilakukan oleh para pakar yang berpengalaman di bidang proyek perangkat lunak.

Rangkuman

Perencanaan proyek memiliki beberapa tahapan sebagai berikut. Fase proposal adalah fase perjanjian atau kontrak perjanjian untuk melakukan serangkaian prose pembangunan maupun pengembangan perangkat lunak. Kemudian fase permulaan yaitu mulai menentukan secara detail mengenai siapa saja yang akan menjadi anggota tim, pengalokasian sumber daya sehingga informasi sudah lebih lengkap dibandingkan ketika tahap proposal, sekaligus juga merupakan tahap untuk mempersiapkan perkiraan perkiraan sebagai awalan. Kemudian fase pemantauan secara berkala sepanjang proyek yaitu perencanaan dapat selalu dilakukan perubahan terbaru ketika ada informasi ban terkait perangkat lunak dan proses pengembangan atau pembangunannya. Hal-hal yang harus masuk dalam perencanaan proyek adalah pernyataan kerja yang merepresentasikan produk, daftar semua sumber daya (*resources*) yang dibutuhkan untuk pada rekayasa perangkat lunak beserta ketersediaannya, rincian struktur pekerjaan dan kumpulan perkiraan pekerjaan yang dilakukan selama proses rekayasa perangkat lunak, perencanaan jadwal dan biaya proyek, perencanaan risiko. Sebuat proyek dikatakan berhasil jika mampu mengirimkan produk yang berkualitas tinggi ada waktu dan biaya yang telah ditentukan sebelumnya.

ANALISIS DAN PENGUMPULAN KEBUTUHAN

1. Pengantar Analisis dan Pengumpulan Kebutuhan

Analisis dan pengumpulan kebutuhan adalah proses mengidentifikasi dan memahami kebutuhan suatu sistem atau produk. proses ini merupakan langkah pertama dalam proses pengembangan perangkat lunak, dan tahapan ini sangat penting untuk memastikan bahwa produk akhir dapat memenuhi kebutuhan pengguna. proses pengumpulan informasi tentang kebutuhan suatu sistem atau produk dapat dikumpulkan dari berbagai sumber, termasuk pengguna, pemangku kepentingan, dan pakar.

Tujuan dari proses analisis dan pengumpulan kebutuhan adalah untuk membuat dokumen yang jelas dan ringkas yang menggambarkan kebutuhan sistem atau produk. Dokumen ini kemudian digunakan oleh tim pengembangan untuk membuat desain dan rencana implementasi.

Beberapa konsep kunci dari analisis dan pengumpulan kebutuhan kebutuhan:

a. **Analisis pemangku kepentingan:** Analisis pemangku kepentingan adalah proses mengidentifikasi dan memahami kebutuhan para pemangku kepentingan. Stakeholder adalah individu atau kelompok yang memiliki kepentingan terhadap sistem atau produk.
b. **Desain yang berpusat pada pengguna:** Desain yang berpusat pada pengguna adalah proses yang berfokus pada kebutuhan pengguna. Tujuan dari desain yang berpusat pada pengguna adalah untuk menciptakan sistem atau produk yang mudah digunakan dan memenuhi kebutuhan pengguna.

c. **Analisis ahli:** Analisis ahli adalah proses pengumpulan informasi dari para ahli di lapangan. Pakar dapat memberikan informasi berharga tentang kebutuhan sistem atau produk.

Analisis dan pengumpulan kebutuhan kebutuhan merupakan langkah penting dalam proses pengembangan perangkat lunak. Dengan meluangkan waktu untuk memahami kebutuhan pengguna dan pemangku kepentingan, Kita dapat memastikan bahwa produk akhir memenuhi kebutuhan orang yang akan menggunakannya.

Berikut adalah beberapa manfaat analisis dan pengumpulan kebutuhan kebutuhan:

a. **Peningkatan kualitas produk:** Dengan memahami kebutuhan pengguna dan pemangku kepentingan, Kita dapat membuat produk yang lebih sesuai dengan kebutuhan mereka. Hal ini dapat meningkatkan kualitas produk dan kepuasan pelanggan.
b. **Mengurangi biaya pengembangan:** Dengan memahami kebutuhan pengguna dan pemangku kepentingan, Kita dapat menghindari pembuatan fitur yang tidak diperlukan. Hal ini dapat menyebabkan berkurangnya biaya pengembangan.
c. **Peningkatan keberhasilan proyek:** Dengan memahami kebutuhan pengguna dan pemangku kepentingan, Kita dapat meningkatkan peluang keberhasilan proyek. Ini karena Kita lebih cenderung membuat produk yang memenuhi kebutuhan orang-orang yang akan menggunakannya.

2. Mengidentifikasi Pemangku Kepentingan

Langkah pertama dalam proyek pengembangan sistem informasi adalah mengidentifikasi pemangku kepentingan. Pemangku kepentingan adalah setiap individu atau kelompok yang memiliki kepentingan dalam proyek, baik secara langsung maupun tidak langsung. Mereka adalah pengguna sistem, manajer yang akan bertanggung jawab atas sistem, atau bahkan orang-orang yang tidak terlibat langsung dalam proyek tetapi akan terpengaruh olehnya.

Setelah pemangku kepentingan diidentifikasi, langkah selanjutnya adalah menganalisis kebutuhan dan persyaratan mereka. Hal ini dapat dilakukan melalui berbagai metode, seperti wawancara, survei, dan maupun FGD. Tujuan dari analisis ini adalah untuk memahami apa

yang diinginkan oleh para pemangku kepentingan dari sistem dan apa yang mereka butuhkan untuk sistem tersebut.

Setelah kebutuhan dan persyaratan dari para pemangku kepentingan dianalisis, langkah selanjutnya adalah mengumpulkan persyaratan. Ini adalah proses mendokumentasikan fitur dan fungsionalitas spesifik yang harus dimiliki sistem untuk memenuhi kebutuhan para pemangku kepentingan. Persyaratan dapat dikumpulkan melalui berbagai metode, seperti wawancara, survei, dan analisis dokumen.

Proses mengidentifikasi pemangku kepentingan, menganalisis kebutuhan dan persyaratan mereka, dan mengumpulkan persyaratan sangat penting untuk keberhasilan pengembangan sistem informasi apa pun. Dengan meluangkan waktu untuk memahami kebutuhan para pemangku kepentingan, pengembang dapat menciptakan sistem yang berguna dan efektif.

Berikut adalah beberapa tips untuk mengidentifikasi pemangku kepentingan:

a. Mulailah dengan mengidentifikasi pengguna sistem. Mereka adalah orang-orang yang akan berinteraksi langsung dengan sistem setiap hari.
b. Identifikasi manajer yang akan bertanggung jawab atas sistem. Mereka adalah orang-orang yang akan membuat keputusan tentang sistem, seperti anggaran dan cakupannya.
c. Identifikasi individu atau kelompok lain yang memiliki kepentingan dalam sistem. Hal ini dapat mencakup orang-orang yang tidak terlibat langsung dalam proyek namun akan terpengaruh oleh proyek tersebut, seperti pelanggan atau pemasok.

Setelah kita mengidentifikasi pemangku kepentingan, kita perlu memahami kebutuhan dan persyaratan mereka. Hal ini dapat dilakukan melalui berbagai metode, seperti wawancara, survei, dan FGD. Tujuan dari analisis ini adalah untuk memahami apa yang diinginkan oleh para pemangku kepentingan dari sistem dan apa yang mereka butuhkan untuk sistem tersebut.

Dengan meluangkan waktu untuk mengidentifikasi pemangku kepentingan, menganalisis kebutuhan dan persyaratan mereka, dan mengumpulkan persyaratan, kita dapat membuat sistem yang lebih berguna dan efektif.

3. Teknik dan Metode Analisis dan Pengumpulan Kebutuhan

Analisis kebutuhan adalah proses mengidentifikasi kesenjangan antara apa yang saat ini dilakukan dan apa yang perlu dilakukan untuk mencapai hasil yang diinginkan. Dalam konteks pelatihan, analisis kebutuhan digunakan untuk mengidentifikasi keterampilan dan pengetahuan yang dibutuhkan karyawan untuk melakukan pekerjaan mereka secara efektif.

Beberapa metode yang dapat digunakan untuk melakukan analisis kebutuhan yaitu:

a. **Observasi:** Hal ini melibatkan identifikasi tugas dan tanggung jawab yang diperlukan untuk pekerjaan tertentu. Pengamatan dapat digunakan untuk mengumpulkan data mengenai kinerja karyawan. Observasi dapat digunakan untuk mengidentifikasi area di mana sistem dapat ditingkatkan
b. **Survei karyawan:** Hal ini mencakup bertanya kepada karyawan tentang keterampilan, pengetahuan, dan kebutuhan pelatihan mereka. Survei adalah cara yang cepat dan mudah untuk mengumpulkan data dari banyak orang. Survei dapat digunakan untuk mengumpulkan informasi mengenai berbagai topik, termasuk keterampilan, pengetahuan, dan kebutuhan pelatihan karyawan. tetapi metode ini tidak dapat memberikan informasi detail yang sama seperti wawancara
c. **Forum Group Discussion:** Ini melibatkan pengumpulan sekelompok kecil karyawan untuk mendiskusikan kebutuhan pelatihan mereka. Kelompok fokus adalah cara yang baik untuk mendapatkan berbagai perspektif tentang topik tertentu. Mereka dapat digunakan untuk mengumpulkan informasi mengenai kebutuhan dan kekhawatiran karyawan, serta untuk mendapatkan umpan balik mengenai ide-ide pelatihan.
d. **Wawancara**: Wawancara dilakukan dengan karyawan, manajer, dan pemangku kepentingan lainnya untuk mengidentifikasi kebutuhan pelatihan. Wawancara adalah cara yang lebih mendalam untuk mengumpulkan data dari sejumlah kecil orang. Metode ini dapat digunakan untuk mendapatkan informasi yang lebih rinci mengenai kebutuhan dan kekhawatiran karyawan.

e. **Analisis dokumen**: Analisis dokumen dapat digunakan untuk meninjau dokumentasi yang ada, seperti manual pengguna, laporan, dan kebijakan. Ini dapat membantu mengidentifikasi kebutuhan pengguna dan pemangku kepentingan, serta kondisi sistem saat ini.

Teknik pengumpulan data yang paling baik untuk analisis kebutuhan tertentu akan tergantung pada kebutuhan spesifik organisasi.

4. Jenis Kebutuhan

Ada tiga jenis kebutuhan utama yang biasanya dipertimbangkan selama analisis kebutuhan dalam pengembangan sistem informasi:

a. **Kebutuhan fungsional:** Ini adalah persyaratan yang harus dipenuhi sistem agar berfungsi dengan baik. Misalnya, sistem manajemen hubungan pelanggan (CRM) harus dapat melacak interaksi pelanggan, menyimpan data pelanggan, dan menghasilkan laporan. Kebutuhan fungsional adalah jenis kebutuhan yang paling penting, karena mereka menentukan tujuan sistem. Kebutuhan ini biasanya diidentifikasi dengan mewawancarai pengguna dan pemangku kepentingan, dan dengan meninjau dokumentasi yang ada
b. **Kebutuhan non-fungsional:** Ini adalah persyaratan yang berhubungan dengan kinerja sistem, seperti kecepatan, keandalan, dan keamanannya. Misalnya, sistem CRM harus mampu menangani sejumlah besar pengguna secara bersamaan dan harus mampu melindungi data pelanggan dari akses yang tidak sah. Kebutuhan non-fungsional sering diabaikan, padahal bisa sama pentingnya dengan kebutuhan fungsional. Kebutuhan ini biasanya diidentifikasi dengan melakukan pengujian kinerja, pengujian keamanan, dan pengujian kegunaan
c. **Kebutuhan bisnis:** Ini adalah persyaratan yang terkait dengan dampak sistem pada organisasi, seperti kemampuannya untuk meningkatkan efisiensi, mengurangi biaya, atau meningkatkan penjualan. Misalnya, sistem CRM dapat membantu organisasi

meningkatkan layanan pelanggan dengan memberi karyawan satu tampilan dari setiap pelanggan. Kebutuhan bisnis adalah tujuan akhir dari sistem. Kebutuhan ini biasanya diidentifikasi dengan mewawancarai pemimpin bisnis dan meninjau rencana strategis.

Proses analisis kebutuhan harus mengidentifikasi semua kebutuhan yang relevan, baik fungsional maupun non-fungsional, dan kemudian memprioritaskannya berdasarkan kepentingannya bagi organisasi. Informasi ini kemudian dapat digunakan untuk merancang dan mengembangkan sistem yang memenuhi kebutuhan organisasi.

Dengan mempertimbangkan ketiga jenis kebutuhan tersebut, kita dapat memastikan bahwa sistem informasi yang akan kita kembangkan akan berhasil.

5. Tantangan dan Strategi dalam Analisis dan Pengumpulan Kebutuhan

Ada sejumlah tantangan yang dapat ditemui dalam analisis dan pengumpulan persyaratan, termasuk:

a. **Persyaratan yang tidak jelas atau tidak lengkap:** Pengguna mungkin tidak dapat mengartikulasikan kebutuhan mereka dengan jelas, atau mereka mungkin tidak mengetahui semua kebutuhan mereka. Hal ini dapat menyebabkan persyaratan yang tidak lengkap atau tidak jelas, yang dapat mempersulit pengembangan sistem yang memenuhi kebutuhan pengguna.
b. **Mengubah persyaratan:** Persyaratan dapat berubah dari waktu ke waktu, karena kebutuhan pengguna berubah atau saat informasi baru tersedia. Hal ini dapat menyulitkan untuk menjaga agar persyaratan selalu mutakhir, dan dapat menimbulkan masalah jika sistem tidak dikembangkan untuk memenuhi persyaratan saat ini.
c. **Masalah komunikasi:** Masalah komunikasi dapat terjadi antara analisis bisnis dan pengguna, atau antara analis bisnis dan tim pengembangan. Ini dapat menyebabkan kesalahpahaman tentang persyaratan, yang dapat menyebabkan masalah dengan sistem.

Ada beberapa strategi yang dapat digunakan untuk mengatasi tantangan analisis dan pengumpulan kebutuhan, antara lain:

a. **Mendengarkan secara aktif:** Analis bisnis harus secara aktif mendengarkan pengguna dan mencoba memahami kebutuhan mereka. Ini dapat dilakukan dengan mengajukan pertanyaan, mengklarifikasi poin, dan meringkas kebutuhan pengguna.
b. **Mendokumentasikan persyaratan:** Persyaratan harus didokumentasikan dengan cara yang jelas dan ringkas. Ini akan membantu untuk memastikan bahwa setiap orang yang terlibat dalam proyek memahami persyaratan.
c. **Menggunakan alat manajemen persyaratan:** Alat manajemen persyaratan dapat membantu melacak persyaratan, mengelola perubahan persyaratan, dan memastikan bahwa persyaratan terpenuhi.
d. **Melakukan tinjauan rutin:** Tinjauan berkala terhadap persyaratan dapat membantu mengidentifikasi masalah apa pun dengan persyaratan dan memastikan bahwa persyaratan tersebut masih akurat.

6. Kesimpulan

Analisis dan persyaratan kebutuhan merupakan langkah penting dalam pengembangan sistem informasi. Ini adalah proses memahami kebutuhan pengguna dan pemangku kepentingan, dan menerjemahkan kebutuhan tersebut ke dalam seperangkat persyaratan yang dapat digunakan untuk merancang dan mengembangkan sistem.

Ada sejumlah teknik berbeda yang dapat digunakan untuk analisis dan pengumpulan persyaratan, seperti wawancara, survei, dan observasi. Teknik yang paling efektif akan bervariasi tergantung pada proyek tertentu.

Setelah persyaratan telah dikumpulkan, mereka perlu dianalisis dan didokumentasikan. Proses ini melibatkan identifikasi berbagai jenis persyaratan, seperti persyaratan fungsional, persyaratan non-fungsional, dan persyaratan bisnis. Persyaratan juga harus diprioritaskan, sehingga persyaratan yang paling penting ditangani terlebih dahulu.

Dokumentasi persyaratan merupakan langkah penting, karena memberikan pemahaman yang jelas tentang apa yang seharusnya dilakukan oleh sistem. Dokumentasi ini dapat digunakan oleh pengembang untuk mendesain dan mengembangkan sistem, dan juga dapat digunakan oleh pengguna untuk menguji dan memverifikasi sistem.

Berikut adalah beberapa kesimpulan tentang analisis dan pengumpulan kebutuhan dalam pengembangan sistem informasi:

- Analisis kebutuhan merupakan langkah penting dalam pengembangan sistem informasi.
- Ada beberapa teknik yang dapat digunakan untuk analisis dan pengumpulan persyaratan.
- Persyaratan harus dianalisis dan didokumentasikan dengan hati-hati.
- Dokumentasi persyaratan merupakan langkah penting untuk keberhasilan setiap proyek sistem informasi.

Berikut adalah beberapa tip untuk analisis dan pengumpulan persyaratan yang efektif:

- Melibatkan semua pemangku kepentingan dalam proses tersebut.
- Menggunakan berbagai teknik untuk mengumpulkan persyaratan.
- Membuat dokumentasi dengan jelas dan ringkas.
- Memprioritaskan persyaratan.
- Selalu memperbaharui persyaratan.

Dengan mengikuti tips ini, kita dapat meningkatkan peluang keberhasilan untuk proyek sistem informasi berikutnya.

PERANCANGAN ANTAR MUKA

PENDAHULUAN

Rekayasa Perangkat Lunak (RPL) tidak terlepas dari pemahaman sistem arsitektur dan antarmuka, karena dengan mendalamnya pemahaman mengenai hal tersebut akan menghasilkan RPL sesuai dengan kebutuhan dan dapat digunakan oleh User sesuai dengan fungsinya. Oleh sebab itu pada bab ini akan dibahas menganai perancangan arsitektur antar muka. Kita awali pembahasan mengenai sistem. Berdasarkan [B1], sistem dapat diartikan sebagai kumpulan elemen yang memiliki kemampuan terbatas tapi dapat beroperasi secara terintegrasi, dan bekerja secara sinergi untuk meningkatkan pemrosesan, membantu kebutuhan user dalam menentukan pengoperasian dengan hasil yang spesifik dan dapat mengukur kemungkinan meraih goal dari sistem tersebut.

Maka akan terdapat banyak elemen/bagian yang mendukung terwujudnya sebuah sistem yang bekerja sesuai dengan kebutuhan baik dalam fungsi maupun User. Dengan adanya keterlibatan User sebagai pengguna sistem maka dibutuhkan pemahaman mengenai arsitektur sistem, yang dibagi menjadi dua bagian yaitu: 1) System of Interest (SOI), adalah fungsi sistem dan berbagai hal penunjangnya yang ditugaskan untuk mengatur fungsi secara spesifik dan menyelesaikannya berdasarkan kinerja fungsi tersebut dalam kurun waktu tertentu. Misalkan terdapat sistem transfer uang secara elektronik, maka fungsi sistem tersebut adalah untuk melakukan transfer uang secara elektronik, adapun yang menjadi penunjangnya adalah terdapat uang dalam rekening untuk ditransfer, terdapat user dengan nomor rekening penerima dan pengirim, terdapat jaringan internet. Berikutnya adalah berfungsinya tombol sebagai tanda melakukan transaksi, dan terdapat konfirmasi bahwa transaksi telah berhasil atau gagal. Dengan target setelah menekan tombol transaksi maka konfirmasi diberikan dalam waktu kurang dari 5 detik.

Pemahaman berikutnya mengenai arsitektur sistem adalah 2) Operating Environment (OE) yaitu ruang lingkup atau cangkupan pengoperasian sistem. Berdasarkan contoh kasus pada trnsaksi uang secara elektronik maka yang dimaksud dengan OE yaitu ruang lingkup tersebut dilakukan secara online, dilakukan dengan uang elektronik baik yang terdapat pada aplikasi bank maupun aplikasi lainnya, dan berbagai pihak terkait uang elektronik.

Setelah pemahaman mengenai sistem dan arsitektur sistem, berikutnya adalah mengenai antar muka sistem. Hasil akhir dari dibuatnya sistem yaitu adanya antar muka sistem. Hal tersebut dapat menjadi ukuran keberhasilan sebuah sistem, karena banyak sistem yang sudah jadi akan tetapi peminatnya tidak ada, sehingga sistem tersebut tidak digunakan lagi. Atau misalkan terdapat lima sistem A, B, C, D, dan E yang memiliki fungsi sama akan tetapi pengguna lebih memilih sistem B untuk digunakan, sedangkan sistem A, C, D, dan E cenderung sepi pengguna.

Kejadian seperti itu dapat dipengaruhi oleh user interface/ antar muka atau tampilan sistem, karena sistem digunakan oleh pengguna maka harus nyaman dan menarik jika digunakan pengguna. Sehingga fokus dalam pembuatan sistem tidak hanya pada fungsi atau pengoperasian sistem saja, akan tetapi diperlukan adanya antar muka yang interaktif dan menarik bagi pengguna.

Sebelum menuju pembahasan berikutnya, perlua adanya penyamaan persepsi mengenai sistem, arsitektur, dan antar muka. Ketiga pokok bahasan tersebut diasumsukan sebagai software yang dirancang dengan menggunakan arsitektur tertentu, yang memiliki tampilan atau antar muka, yang diimplementasikan pada perangkat tertentu sebagai sarana komunikasi antara pengguna dan software/ sistem, untuk memenuhi kebutuhan pengguna.

FUNDAMENTAL ANTAR MUKA

Setelah pendefinisian sistem pada bab ini, berikutnya akan dibahas mengenai pengguna sistem, yang merupakan inti sari dari sebuah sistem, karena tanpa pengguna maka sistem tidak akan pernah ada. Sehingga akan terlihat jelas kedudukan pengguna dan sistem yang saling berkaitan dan saling membutuhkan. Hal tersebut terjadi

karena adanya sebuah kebutuhan atau yang sering dikenal dengan istilah goal atau tujuan. Perlunya mengistimewakan pengguna dalam merncang atau membuat arsitektur, karena berdasarkan [B2] hal yang perlu diperhatikan dalam merancang antar muka yaitu "Know thy users, for they are not you", dengan kata lain kenali pengguna apa adanya mereka, karena pengguna itu bukan perancang. Sehingga dibutuhkan adanya transfer pengetahuan dari pengguna kepada perancang, sehingga perancang memahami betul apa yang dibutuhkan dan diinginkan pengguna. Sehingga software yang dihasilkan dapat mendukung pengguna meraih goal/tujuannya.

Pengetahuan yang perlu digali dari pengguna yaitu [B2]:

- Tujuan pengguna dalam menggunakan software
- Tugas spesifik yang dilakukan pengguna dalam mencapai tujuan
- Bahasa dan kata-kata yang digunakan untuk menggambarkan apa yang pengguna lakukan
- Kemampuan pengguna dalam menggunakan software yang mirip dengan software yang akan dirancang
- Sikap pengguna terhadap rancangan yang dibuat, dan seberapa jauh rancangan tersebut dapat mempengaruhi sikap pengguna

Akan cukup memakan waktu untuk menggali pengetahuan pengguna, karena perancang tidak mengenal kebiasaan pengguna, latar belakang pengguna, kinerja pengguna, maka diperlukan penjelasan yang konkret mengenai penggun, sehingga perancangan yang dibuat akan membumi sesuai dengan realita. Beberapa cara dan topik yang dapat dijadikan referensi saat melakukan penggalian pengetahuan yaitu [B2]:

- Direct observation, observasi dapat dilakukan dengan cara mewawancarai pengguna, alangkah lebih baik lagi jika dilakukan langsung dilokasi dimana pengguna akan menggunakan software tersebut, sehingga perancang mendapatkan gambaran bagaimana kondisi pengguna saat menggunakan software yang akan dirancang. Wawancara dilakukan untuk mempelajari hal yang perancang tidak ketahui mengenai pengguna.
- Case studies, perlu adanya penggalian mengenai studi kasus, misalkan perancang mencari software yang biasa digunakan pengguna sehingga dapat mengetahui tampilan/antar muka yang

sering digunakan pengguna, terutama jika dilakukan perancangan ulang terhadap software yang sudah ada. Perancang dapat menjelajahi software tersebut dalam hitungan bulan bahkan tahun, karena harus mempelajari terlebih dahulu kekurangan software tersebut kemudian merancang ulang untuk diimplementasikan terhadap software yang baru.

- Surveys, melakukan survei merupakan salah satu cara yang tepat untuk menggali pengetahuan user karena perancang akan mendapatkan nilai pasti berupa angka yang signifikan secara statistik dari responden. Perancangan survei harus dilakukan secara mendalam dari segi responden, pertanyaan, dan analisa jawaban responden secara tepat.
- Personas, dapat membantu perancang dalam menggambarkan data yang didapatkan, hal tersebut dapat memodelkan target pengguna. Sehingga dapat mengetahui tugas apa yang pengguna ingin selesaikan, tujuan akhir pengguna, tingkatan pengalaman pengguna dalam menggunakan komputer secara umum.

Pengenalan lain yang perlu dilakukan oleh perancang saat merancang antar muka yaitu perlu menyesuaikan tingkat pengetahuan pengguna, misalkan pengguna masih awam mengenai komputer, maka perancang harus membuat antar muka yang mudah digunakan, sehingga dapat meningkatkan minat pengguna untuk menggunakan sistem yang telah dirancang. Kasus lain misalkan pengguna sering menggunakan sistem operasi tertentu, maka tampilan pada sistem operasi tersebut dapat menjadi referensi bagi perancang dalam membuat rancangan antar muka. Dengan demikian akan memberikan kemudahan pengguna untuk mempelajari dan menggunakan software yang dibuat.

Pengenalan tersebut dapat berkembang sesuai dengan pengguna yang dihadapi perancang. Semakin tinggi tingkat pengetahuan pengguna tidak dapat menjamin semakin rumit perancangan antar muka. Karena pada dasarnya setiap pengguna dengan tingkatan pengetahuan apapun, tujuannya yaitu dapat mengoperasikan software untuk mencapai tujuannya. Sebagai perancang harus memerhatikan berbagai aspek, tidak hanya penataan antar muka saja, akan tetapi

perlu meperhatikan arsitektur/perancangan alur penggunaan software, pemilihan fitur (sistem operasi, tools pembuat perancangan, dan lainnya), dokumentasi, pemeliharaan software (update software), dan berbagai hal yang mendukung tingkat pengetahuan pengguna.

PENGORGANISASIAN KONTEN

Pembahasan bagian ini, perancang sudah memahami apa yang diinginkan oleh pengguna pada software yang akan dirancang, karena perancang telah mempelajari level pengetahuan pengguna, dan cara menggali pengetahuan pengguna. Target selanjutnya yaitu memilih platform apakah akan diterapkan pada website, dekstop, mobile, atau kombinasi diantara ketiganya. Jika sudah dapat menentukan, maka dapat dilanjutkan dengan menuliskan skenario yang menjelaskan bagaimana pengguna menggunakan aplikasi tersebut untuk mencapai tujuannya. Berikutnya, perancang harus membuat sketsa antar muka, hal tersebut dapat dilakukan dengan aplikasi pendukung, atau dapat menggunakan kertas. Perancang dituntut untuk dapat sekreatif mungkin dalam merancang software, sampai seluruh konten dapat terakomodir sesuai pengetahuan pengguna.

Didalam pengorganisasian konten, dikenal istilah Information architecture (IA), yaitu seni dalam mengorganisir informasi, yang terdiri dari penyajian, pencarian, penjelajahan, pelabelan, pengkategorian, pengurutan, menipulasi, dan mengatur strategi informasi apa saja yang dapat disajikan dan tidak dapat disajikan secara publik. Dengan adanya IA dapat menjadi acuan untuk perancang dalam membuat rancangan software. Pengimplementasian IA dapat dilihat pada laman software yang dapat disajikan dalam empat jenis, yaitu [B2]:

- Menampilkan satu konten, konten yang dimaksud dapat berupa image atau video, yang berada diberanda atau home laman aplikasi.
- Menampilkan daftar atau beberapa set konten, tampilan lain yang dapat disajikan yaitu menampilkan daftar atau beberapa set konten dalam satu laman pertama, sehingga pengguna dapat melihat secara bersamaan menu apa saja yang berada dalam aplikasi tersebut.
- Menyediakan alat untuk membuat konten, terdapat laman atau aplikasi yang berfungsi sebagai penyedia konten atau penyedia aplikasi, ada yang berbayar atau yang tidak berbayar. Misalkan aplikasi yang dapat membuat slide presentasi secara otomatis,

sehingga pengguna dapat langsung memasukkan materi yang akan dipresentasikan tanpa harus merancang tampilan presentasinya.

- Memfasilitasi tugas tertentu untuk menampilkan konten, fasilitas tersebut dapat berupa library yang dapat langsung digunakan tanpa harus membangun library tersebut dari awal. Misalkan aplikasi yang menyediakan library chatGPT yang menyediakan database kata atau kalimat, maka pengguna dapat menggunakan library tersebut tanpa harus mengumpulkan kata atau kealimat terlebih dahulu. Perlu di ingat bahwa konten dan tugas itu berbeda, jika konten merupakan isian apa saja yang ada dalam aplikasi, sedangkan tugas adalah library yang memiliki fungsi tertentu, tidak hanya tampilan saja seperti konten, akan tetapi lebih ke konten yang memiliki kemampuan atau tugas tertentu.

TAHAPAN PERANCANGAN ANTAR MUKA

Sebuah aplikasi utuh memerlukan beberapa tahap untuk menjadikannya sempurna, tahapan tersebut berfungsi untuk mengakomodasi kebutuhan pengguna, dan mendukung pengetahuan perancang. Tahapan tersebut yaitu:

- Planning atau perencanaan, fokus terhadap mengapa aplikasi akan dibangun, dan bagaimana struktur dari projek aplikasi tersebut, untuk mengetahui perencanaan project dan studi kelayakan aplikasi. Selain itu perlu adanya identifikasi peluang, untuk mengidentifikasi peluang apa saja yang akan didapatkan jika membuat aplikasi tersebut. Selain itu perlu adanya analisis kelayakan dari sudut teknis, ekonomis, dan organisasi. Berikutnya yaitu membuat rencana kerja, sehingga mengetahui ruang lingkup managemen, estimasi waktu pengerjaan, struktur pekerjaan dan berbagai bagan pendukung analisa sistem. Di tahap planning juga diperlukan adanya staf yang mendukung pembuatan aplikasi, serta kontrol dan arah aplikasi seperti adanya resiko manajemen, dokumentasi dan repository berbagai keperluan dan pendukung aplikasi.
- Analysis atau analisis, fokus pada tahap analysis adalah siapa, apa, dimana, dan kapan aplikasi akan dibangun, untuk mengetahui aplikasi yang akan dibuat. Perlunya mengembangkan strategi

analisis terutama dalam otomatisasi, peningkatan, dan rekayasa ulang aplikasi, sehingga dapat diketahui jangka panjang untuk keberlangsungan aplikasi. Tahap Analysis juga memperhatikan mengenai penentuan kebutuhan bisnis yang dapat dilakukan dengan cara memberikan interview, kuesioner, analisa dokumen, dan observasi. Berikutnya yang memerlukan analisis yaitu membuat use case, pemodelan proses melalui flow diagram, dan pemodelan data melalui entity relationship dan normalisasi data.

- Design atau desain, berfokus terhadap bagaimana kinerja dari aplikasi yang akan dibangun, untuk mengetahui spesifikasi aplikasi. Memerlukan perhatian pada desain fisik aplikasi sehingga dibutuhkan desain strategi aplikasi. Desain arsitektur yang meliputi arsitektur desain, hardware dan software. Desain interface atau antar muka yang perlu menyiapkan skenario aplikasi, struktur antar muka, stadar antar muka, prototipe atau mockup antar muka, dan evaluasi antar muka untuk mendukung antar muka aplikasi yang akan dibangun.Desain programs yang terdiri dari data flow diagram, spesifikasi dan struktur bagan aplikasi. Desain database dan file, perlu adanya pengorganisasian atau managemen data untuk malakukan pemodelan entity relationship, denormalisasi, dan estimasi ukuran database yang akan digunakan.
- Implementation atau implementasi, berfokus terhadap cara penyampaian dan penunjang untuk instalasi aplikasi untuk instalasi aplikasi. Pada tahap ini perlu adanya konstruksi aplikasi seperti programming, testing software dan testing kemampuan. Berikutnya adalah instalasi aplikasi perlu adanya konversi sistem untuk mendukung instalasi aplikasi. Maintain aplikasi untuk memelihara aplikasi, dimulai dari training, pemilihan penunjang aplikasi, pemeliharaan aplikasi, dan asesmen projek. Pasca implementasi, memerlukan adanya laporan audit untuk menjaga keberlangsungan aplikasi sehingga dapat digunakan secara berkelanjutan.

Desain

Sebuah perancangan merupakan dapat dijadikan sebagai pengambil keputusan akan seperti apa aplikasi yang dibangun berjalan, terdapat banyak aktifitas yang dibutuhkan untuk menyesuaikan kebutuhan aplikasi. Lima aktifitas pada fase desain:

- Desain kebutuhan aplikasi
- Desain arsitektur
- Desain antar muka
- Desain program
- Desain tempat penyimpanan data

Seluruh aktifitas tersebut harus berjalan secara berurutan, yang diawali dengan desain kebutuhan aplikasi yang merupakan tindak lanjut setelah tahap analisis. Didalamnya terdapat transisi pengetahuan yang dituangakan dalam bentuk desain kebutuhan, sehingga jelas dan akurat kebutuhan apa saja yang dibutuhkan sampai aplikasi tercipta.

Selanjutnya desain arsitektur yang mempelajari bagaimana menampilkan arsitektur untuk memenuhi kebutuhan aplikasi yang dapat dibagi kedalam kebutuhan:

- Operasional
- Performa
- Keamanan
- Kebijakan

Seperti terlihat pada gambar berikut, yang menunjukkan empat desain kebutuhan aplikasi terhadap jaringan yang mendukung aplikasi.

Requirements	Server-Based	Client-Based	Thin Client-Server	Thick Client-Server
Operational Requirements				
System Integration Requirements	✓		✓	✓
Portability Requirements			✓	
Maintainability Requirements	✓		✓	
Performance Requirements				
Speed Requirements			✓	✓
Capacity Requirements			✓	✓
Availability/Reliability Requirements	✓		✓	✓
Security Requirements				
High System Value	✓		✓	
Access Control Requirements	✓			
Encryption/Authentication Requirements			✓	✓
Virus Control Requirements	✓			
Cultural/Political Requirements				
Multilingual Requirements			✓	
Customization Requirements			✓	
Making Unstated Norms Explicit			✓	
Legal Requirements	✓		✓	✓

Desain Antar Muka

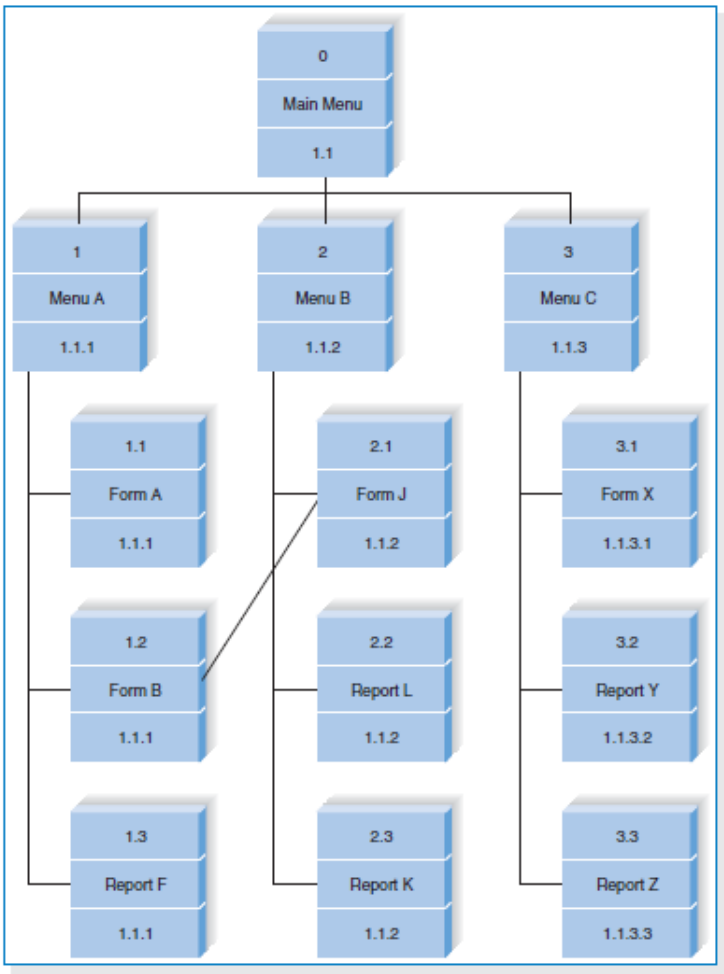
Aplikasi yang akan digunakan oleh pengguna atau entitas luar harus melalui desain antar muka, dengan harapan aplikasi dapat digunakan oleh pengguna dan sesuai dengan kebutuhan pengguna. Terdapat tiga mekanisme yang harus diperhatikan dalam membuat desain antar muka yaitu: 1). Navigasi adalah cara bagaimana pengguna memberikan perintah ke aplikasi dan memerintahkan aplikasi untuk melakukan sesuatu, contohnya adalah tombol atau menu pada aplikasi; 2). Input adalah bagaimana aplikasi menangkap informasi, yang dapat dibuktikan dengan sebuah form, dari form tersebut maka aplikasi mendapatkan informasi sesuai dengan isisn form; 3). Output adalah bagaimana aplikasi menyediakan informasi untuk pengguna atau aplikasi lain, seperti report atau laporan pada aplikasi.

PRINSIP DESAIN ANTAR MUKA

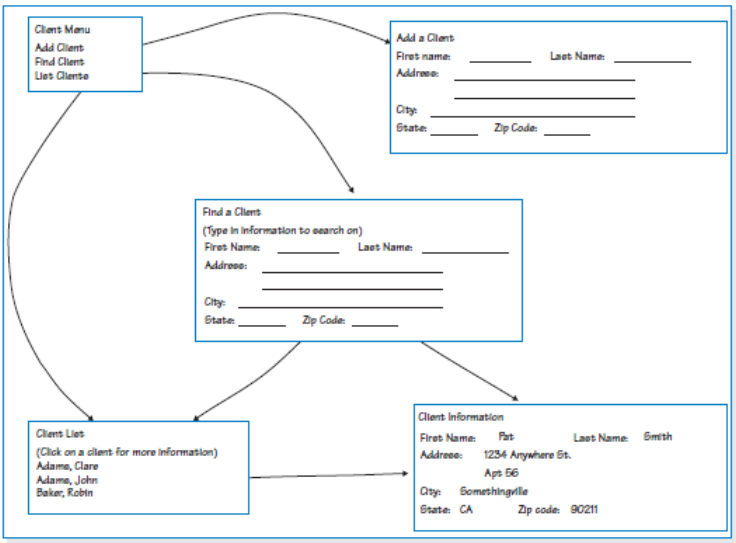
Sebuah antar muka merupakan karya seni, yang bertujuan untuk menampilkan antar muka yang enak dipandang dan simpel untuk digunakan, dengan cara meminimalkan upaya pengguna dalam menyelesaikan pekerjaannya melalui aplikasi yang akan dibangun. Terdapat enam prinsip yang harus dipahami dalam membuat antar muka yaitu:

1. Layout, dalam satu aplikasi terdapat beberapa layout yang merupakan area tertentu, seperti misalkan layout menu yaitu area yang menampilkan beberapa menu tertentu. Seperti terlihat pada layout di aplikasi perkantoran untuk mengetik open office yang menampilkan layout khusus untuk menu File, Home, Insert dan lainnya.
2. Content awareness, yaitu sebuah konten yang dapat menyadarkan pengguna mengenai lokasi keberadaan pengguna sehingga pengguna mengetahui pekerjaan apa yang harus dilakukan di laman tersebut.
3. Aesthetics, yaitu tampilan antar muka yang enak dipandang, antar muka tidak harus selalu bernilai seni tinggi, karena yang terpenting antar muka tersebut simpel dan sesuai dengan fungsinya, sehingga tidak menyulitkan pengguna dalam mengoperasikannya.
4. User experience, perancang harus menyeseuaikan desain antar muka apakah akan digunakan oleh pengguna yang berpengalaman dalam menggunakan aplikasi atau pengguna yang belum berpengalaman. Hal tersebut perlu dilakukan untuk mengetahui desain antar muka yang tepat sesuai dengan tingkatan pengguna aplikasi.
5. Consistency, hal yang perlu diperhatikan dalam merancang antar muka adalah adanya kekonsistenan dalam antar muka, mislkan warna atau ukuran teks pada antar muka selalu sama antara laman home dengan laman lainnya.
6. Minimize user effort, meminimalkan kegiatan pengguna untuk menuju goal aplikasi, sehingga pengguna tidak perlu melakukan banyak proses untuk mengetahui akhir aplikasi. Misalkan aplikasi untuk print, maka pengguna cukup klik menu *home* kemudian *print*. Ada istilah *three-clicks rule* dimana pengguna sudah dapat mencapai goal aplikasi hanya dengan tiga kali klik mouse.

Selain lima prinsip dalam merancang antar muka, diperlukan juga diagram struktur antar muka, sehingga perancang dapat mengetahui alur bagaimana aplikasi tersebut akan berkerja, yang diawali dengan adanya main menu kemudian menu lainnya hingga terlihat level menu yang ada pada aplikasi yang akan dibangun. Contoh diagram struktur antar muka terlihat pada gambar berikut.



Setelah mengetahui menu apa saja yang ada pada aplikasi, maka berikutnya diperlukan sebuah storyboard yang menunjukkan adanya fungsi atau data apa saja yang didapatkan dari setiap menu yang ada. Contoh storyboard dapat dilihat pada gambar berikut:

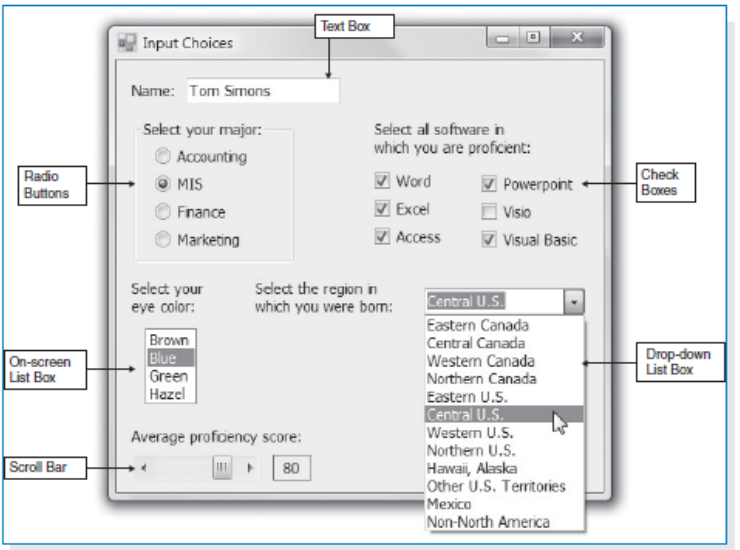


Sebuah desain perlu dievaluasi untuk mengetahui apakah desain tersebut telah sesuai dengan kebutuhan aplikasi dan kebutuhan pengguna, adapun evaluasi yang diperlukan yaitu:

1. Heuristic, Evaluasi heuristik dilakukan secar internal tim desain yang memeriksa kembali apakah sudah sesuai dengan prinsip antar muka yang dilihat dari navigasi, input, dan output perancangan.
2. Walk-through, yaitu evaluasi yang dilakukan dengan cara menunjukkan storyboard atau prototype kepada pengguna aplikasi, untuk mengetahui alur eari aplikasi tersebut.
3. Interactive, yaitu evaluasi perancangan yang dilakukan dengan pengguna yang akan menggunakan aplikasi tersebut, biasanya perancang akan memberikan prototype kepada pengguna, sehingga pengguna akan mengetahui kesesuaian dan ketidak sesuaian apa yang diminta oleh pengguna dengan prototype yang dirancang oleh perancang.
4. Formal usability testing, yaitu pengguna melakukan evaluasi dengan cara menggunakan aplikasi yang sudah menjadi software dan perancang merekan setiap kegiatan yang dilakukan pengguna pada aplikasi tersebut. Sehingga akan terekam seluruh kegiatan pengguna, yang dapat dievaluasi kembali jika terdapat kesalahan pada perancangan.

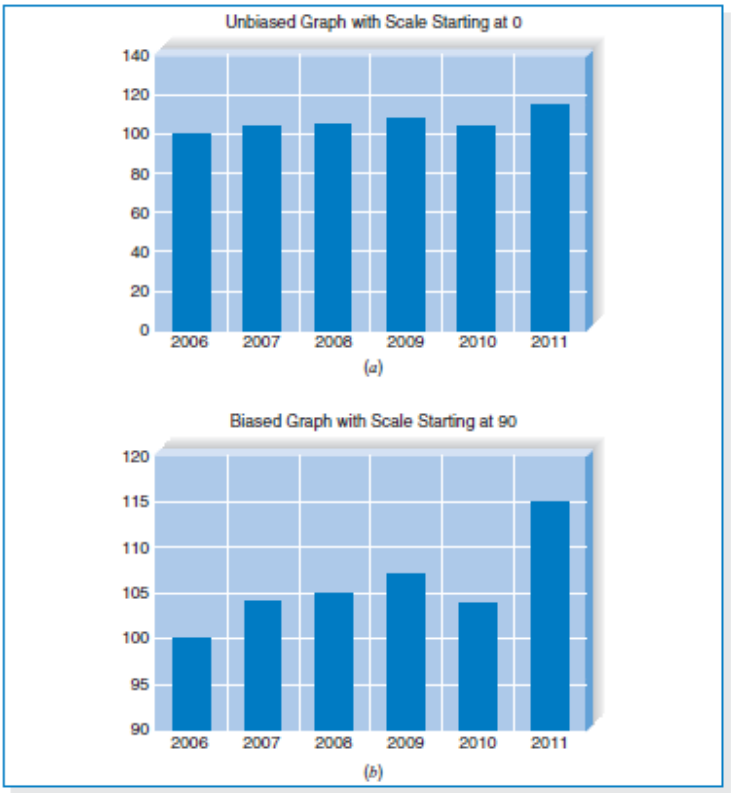
DESAIN INPUT

Input atau masukan untuk aplikasi dapat dilakukan dengan beragam konten, yang intinya memasukkan informasi atau data pada aplikasi yang akan dibangun. Berikut adalah ragam konten yang dapat menjadi input untuk aplikasi yaitu: text box, radio buttons, list box, scroll bar, check boxes, drop-down list box.



DESAIN OUTPUT

Setelah aplikasi mendapat input atau masukan dari pengguna, maka aplikasi tersebut akan memproses atau mengolah data input tersebut dan menampilkannya menjadi output. Output merupakan goal dari sebuah perancangan aplikasi. Terdapat beragam output yang dapat ditampilkan yaitu: report atau laporan yang dapat berupa grafik atau daftar informasi yang dibutuhkan pengguna.



PERANCANGAN BASIS DATA

Pendahuluan

Perkembangan teknologi informasi saat ini sangat pesat dimana hampir semua aspek kehidupan sudah berbasis sistem informasi. Sistem informasi dapat didefinisikan suatu sistem dalam organisasi yang terdiri dari beberapa elemen yakni orang-orang, fasilitas, teknologi, media, prosedur dan pengendalian untuk mendapatkan jalur komunikasi penting, memproses tipe transaksi rutin tertentu, memberi sinyal kepada manajemen dan yang lainnya terhadap kejadian-kejadian internal dan eksternal yang penting dalam penyediaan informasi guna pengambilan keputusan. Sistem informasi suatu organisasi berbasiskan komputer terdiri atas basis data, perangkat lunak basis data, perangkat lunak aplikasi, perangkat keras dan personal. Sistem informasi tak lepas dari Sistem Informasi Manajemen dimana sistem informasi akan dikelola guna pengolahan data lebih lanjut.

Sebelum membahas tentang peancangan basis data, penting kiranya mengenal terlebih dahulu tentang definisi data, pengertian basis data, kriteria basis data dan implementasinya serta *Database Management System* (DBMS). Data merupakan nilai atau *value* yang merepresentasikan deskripsi suatu objek atau kejadian atau *event,* akan dituliskan dalam simbol. Pengertian informasi yaitu hasil dari pengolahan data berbentuk yang lebih berguna dan lebih berarti bagi penerimanya dimana menggambarkan suatu kejadian-kejadian yang nyata (*fact*) untuk pengambilan keputusan. Maka perbedaan antara data dan informasi yakni jika data lebih bersifat historis, sedangkan informasi mempunyai tingkatan yang lebih tinggi, lebih dinamis, serta mempunyai nilai yang sangat penting(BasisData, 2017). Basis data didefinisikan sebagai intergrasi dari sekumpulan data dan diatur sedemikian rupa sehingga data tersebut dapat dimanipulasi, diambil, dan dicari secara cepat(Gat, 2015). Adapun kriteria basis data adalah:

1. bersifat *data oriented* dan bukan *program oriented*
2. dapat digunakan oleh beberapa program aplikasi tanpa mengubah basis datanya
3. dapat berkembang dengan mudah, baik volume maupun strukturnya
4. dapat digunakan dengan cara berbeda-beda
5. memiliki kerangkapan data minimal

Komponen penyusun suatu basis data antara lain sistem operasi, perangkat keras, sistem pengelola basis data (DBMS) dan level pemakai. Ada beberapa level pemakai yakni *Programmer*, User mahir, user umum dan user khusus.

Lingkungan Basis Data

Beberapa masalah yang terjadi dalam basis data antara lain redudansi data, inkonsistensi data, isolasi data untuk standarisasi, banyaknya pemakai, masalah keamanan dan integritas serta masalah kebebasan. Adapun gambaran permasalahan tersebut, antara lain:

a. redudansi data merupakan data yang muncul berulang-ulang pada beberapa file basis data yang semestinya tidak diperlukan. Tidak menutup kemungkinan kerangkapan data tersebut terjadi dalam satu file.
b. Inkonsistensi data,yakni adanya data yang tidak konsisten pada field yang sama,dan dapat juga pada file yang berbeda. Inkonsistensi data ini terjadi akibat kesalahan dalam pemasukan data atau update data yang tersebar dalam beberapa file, hal ini menyebabkan kesalahan pada hasil pengolahan basis data yang tidak sesuai dengan fakta.
c. Isolasi data untuk standarisasi disebabkan oileh pemakaian beberapa file basis data yang tersebar dalam beberapa file, halini mempersulit programmer untuk mengambil dan menyimpan data.
d. Penggunaan basis data yang diakses oleh banyak pemakai secara simultan, hal ini dapat dilakukan karena data yang diolah tidak bergantung dan menyatu program tetapi terlepas dalam satu kelompok data.

e. Keamanan basis data menjadi penting karena pada prinsipnya file basis data hanya dapat diakses oleh pemakai tertentu yang memiliki wewenang,dimana pembatasan dapat dilakukan melalu8i DBMS atau program aplikasi.
f. Integritas suatu basis data juga menjadi hal yang penting untuk menjaga agar kerja sistem tetap dalam pengendalian penuh,dan secara teknik akan ada kunci primer yang menghubungkan beberapa file yang saling berkaitan.
g. Basis data yang dirancang hendaknya tidak bergantung pada program aplikasi yang dibangun, sehingga jika ada perubahan terhadap file,programmer tidak perlu melakukan perubahan pada programnya.

Konsep Database Management System (DBMS)

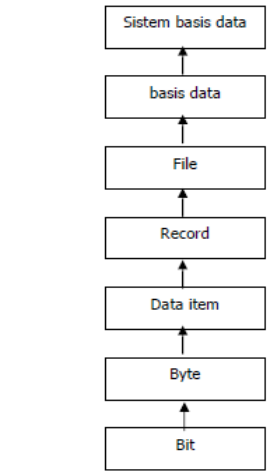
DBMS (*Data Base Management System*) adalah perangkat lunak yang memberikan berbagai fasilitas untuk melakukan fungsi pengaturan, pengawasan, pengendalian, pengolahan dan koordinasi terhadap semua proses yang terjadi pada sistem basis data. Adapun komponen-komponen utama DBMS:

1. *Query language*, bahasa yang digunakan oleh bagian lain dengan sedikit perintah sederhana, contohnya SQL (*Structure Query Language*), QBE (*Query By Example*)
2. *Report generator*, dirancang untuk membuat cetakan, yang memiliki berbagai perintah untuk membuat header, judul, kolom, simpulan, dan lain-lain
3. DML atau *Data Manipulation Language*, terdiri atas berbagai perintah yang disediakan dalam program aplikasi untuk melakukan manipulasi data seperti *append*, *list*, atau *update*.
4. DDL (*Data Definition Language*)
   - Dengan bahasa ini dapat membuat indeks, tabel baru, pengubahan tabel, penentuan struktur tabel, dan lainnya.
   - Kompilasi akan menghasilkan perintah DDL menjadi kamus data, dimana data yang menjelaskan data sesungguhnya

   Contoh: *Create*, *Modify report*, *Modify Structure*

5. *Recovery*, yakni kemampuan untuk mengembalikan data yang rusak atau hilang yang diakibatkan operasi data seperti *insert*, *update*, dan lain-lain.
6. *Data Dictionary*, digunakan untuk menjaga definsi-definisi standar seluruh data secara terinci dalam lingkup kecil pada sistem basis data
7. Basis data, merupakan bagian dari DBMS yang menyediakan data dalam berbagai tipe dan format untuk memenuhi kebutuhan pemakai
8. *Access routine*, suatu rutin yang dapat dipanggil dan dipergunakan oleh program lain untuk mengakses basis data

Penyusun Basis Data



Contoh : Data bilangan bulat (integer), Byte (1 byte), Small-Integer (2 byte), Long Integer (4 byte), Data bilangan nyata, Single (4 byte), Double (8 byte).

Keterangan :

- **Bit**, merupakan sistem angka biner yang terdiri atas angka 0 dan 1
- **Byte**, merupakan bagian terkecil, dapat berupa karakter numerik, huruf, ataupun karakter khusus yang membentuk suatu item data / field. 1 Byte digunakan untuk mengkodekan 1 karakter
- **Data item (field)**, merepresentasikan suatu atribut dari suatu record yang menunjukkan suatu item dari data, misalnya nama, alamat. Kumpulan dari field membentuk suatu record
- **Record**, menggambarkan suatu unit data individu yang tertentu. Kumpulan dari record membentuk suatu file.
- **File**, terdiri dari record-record yang menggambarkan satu kesatuan data yang sejenis
- **Basis data**, sekumpulan dari berbagai macam tipe record yang mempunyai hubungan terhadap suatu objek tertentu
- **Sistem basis data**, merupakan sekumpulan basis data, yang tersusun dari beberapa file

Sistem basis data merupakan lingkup terbesar dalam organisasi data. Sistem basis data mencakup semua bentuk komponen data yang ada dalam suatu sistem. Sedangkan basis data merupakan komponen utama yang menyusun sistem basis data. Komponen penyusun data seperti terlihat pada gambar 1.

Gambar 1. Penyusun Sistem Basis Data

Pada gambar diatas menunjukkan bahwa bagian terkecil dari basis data adalah Bit, dan pada tingkatan level diatasnya dimana hingga terbentuknya basis data.

Perancangan Basis Data

Perancangan basis data (*database design*) adalah proses merancang struktur dan hubungan antar tabel dalam sebuah basis data. Tujuan dari perancangan basis data sendiri adalah untuk memenuhi kebutuhan akan informasi dari pengguna dan aplikasi, dan menyediakan struktur informasi yang mudah dimengerti oleh pengguna, serta mendukung pemrosesan pada sistem basis data. Selain itu tujuan dari perancangan basis data juga untuk memastikan bahwa basis data dapat menyimpan data dengan efisien, mudah diakses, dan terstruktur dengan baik.

Proses perancangan basis data meliputi beberapa tahapan, yakni pengumpulan data dan analisis, perancangan basis data secara konseptual, pemilihan DBMS, perancangan basis data dengan logika (data model mapping), perancangan basis data fisik, dan implementasi sistem basis data dan pengujian basis data. Beberapa metode perancangan basis data yang umum digunakan adalah *Structured Analysis and Design* (SADT), *Data Flow Diagram* (DFD), dan *Database Life Cycle* (DBLC).

Tahap pertama untuk perancanngan basis data yakni pengumpulan data dan analisis. Pada tahap pertama yang perlu dilakukan adalah proses mengidentifikasi dan menganalisis kebutuhan dari pengguna aplikasi basis. Untuk itu, perlu mengenal bagian lain dari sistem informasi yang nantinya akan berhubungan dengan sistem basis data, termasuk user atau pengguna dan aplikasi. Data akan kebutuhan-kebutuhan itulah yang kemudian dikumpulkan dan dianalisis.

Pada tahap ini, ada 4 kegiatan yang dilakukan yaitu:

1. menentukan kelompok pengguna dan bidang aplikasinya, disini merupakan langkah penting dalam perancangan basis data dimana sebagai perancang basis data akan melihat pengguna basis data dalam beberapa sudut pandang. Langkah ini akan mempengaruhi penentuan entitas pada basis data yang akan dibuat. Misalnya saja, dalam basis data akademik maka pengguna akademik ada mahasiswa, dosen, program studi dan Bagian akademik institusi, nilai dan mata kuliah maka entitas dalam basis data akademik ada mahasiswa, dosen, mata kuliah, nilai, program studi dan BAA.

2. meninjau dokumentasi yang ada, untuk tahap ini akan memperhatikan formulir-formulir yang digunakan atau tepatnya sistem informasi yang berjalan pada basis data ini. Hal ini dilakukan untuk melihat alur informasi penting apa saja yang perlu dirancang dalam basis data ini. Misalnya saja dalam basis data akademik ada pengisian KRS oleh mahasiswa, pengimputan nilai oleh dosen, penginputan mata kuliah oleh BAA dan pengelolaan mahasiswa dan dosen oleh program studi.
3. menganalisis lingkungan operasi dan pemrosesan data, disini analisis sangat penting dilakukan karena akan menentukan relasi dalam basis data yang akan dirancang.
4. membuat daftar pertanyaan untuk wawancara tentang data apa saja yang diperlukan atau harus dianalisis.

Pada tahap ini memiliki tujuan utamanya adalah untuk memahami entitas, atribut, dan hubungan antara data dalam domain yang relevan.

Perancangan konseptual pada tahap ini melibatkan pembuatan model konseptual berdasarkan pemahaman tentang entitas, atribut, dan hubungan. Model ini biasanya menggunakan notasi seperti diagram *Entity-Relationship* (ER) atau notasi lainnya untuk mewakili entitas, atribut, dan hubungan antara mereka. Sebelum menuju Langkah berikutnya sedikit membahas notasi dalam *Entity-Relationship*. Entity Relationship Diagram (ERD) yang digunakan untuk mendesain dalam basis data yang menunjukkan secara detail hubungan atau relasi antara objek dari entitas dan atributnya sehingga terbentuk secara terstuktur dan jelas menggunakan beberapa notasi dan simbol(Mukhlis and Santoso, 2023).

Perancangan logical dimana pada tahap ini, model konseptual diterjemahkan ke dalam bentuk yang lebih terstruktur seperti skema relasional. Skema relasional menggunakan tabel untuk mewakili entitas dan atribut, dengan kunci primer dan kunci asing untuk menghubungkan tabel yang berbeda.

Tahap berikutnya adalah normalisasi. Normalisasi adalah proses mengorganisir skema relasional untuk mengurangi redundansi dan anomali data. Tujuannya adalah untuk memastikan bahwa basis data memenuhi aturan normalisasi, yaitu bentuk normal pertama atau 1NF, bentuk normal kedua atau 2NF, bentuk normal ketiga dan seterusnya.

Tahap perancangan fisik dimana pada tahap ini, struktur data yang telah dirancang diterjemahkan ke dalam implementasi fisik di lingkungan sistem basis data tertentu. Ini melibatkan pemilihan jenis kolom, indeks, konfigurasi penyimpanan, dan faktor-faktor lain yang relevan untuk kinerja dan keamanan sistem.

Tahap pengimplementasian basis data, yakni setelah perancangan fisik selesai, langkah berikutnya adalah mengimplementasikan basis data sesuai dengan perancangan yang telah dirancang sebelumnya. Ini melibatkan pembuatan tabel, indeks, dan objek basis data lainnya, serta memasukkan data awal ke dalam sistem basis data.

Setelah basis data diimplementasikan maka dilakukan pengujian dan penyesuaian basis dta. Tahap ini melibatkan pengujian sistem basis data untuk memastikan bahwa basis data berfungsi dengan baik dan memenuhi syarat-syarat bisnis yang sudah ditetapkan. Jika ada masalah atau kekurangan, penyesuaian dan perbaikan dilakukan pada basis data untuk efektifitas dan efisiensi basis data.

Yang tidak kalah penting dalam perancangan basis data adalah pemeliharaan dan pengembangan. Setelah basis data diimplementasikan, tugas pemeliharaan rutin seperti pemantauan kinerja, backup, pemulihan, dan peningkatan dapat dilakukan. Selain itu, jika ada kebutuhan baru atau perubahan bisnis, perancangan basis data dapat diperbarui dan dikembangkan lebih lanjut.

IMPLEMENTASI PERANGKAT LUNAK

1. Pengertian tahap implementasi

Implementasi sistem erat kaitannya dengan pengujian sistem, dimana jika kembali ke metode pengembangan sistem yang berkelanjutan pengujian bisa dilakukan berkali-kali. Bahkan setelah sistem diterapkan juga masih bisa dilakukan pengujian lanjutan, walaupun sejatinya fase awal pengujian dilakukan sebelum implementasi sistem dijalankan. Untuk fundamental bagian implementasi ini mari kita simak definisinya terlebih dahulu sebagai berikut:

a. Implementasi menurut para ahli

Romi Satia Wahono: Kekeliruan tentang keahlian pengkodean sangat penting bagi pengembang. "Hal ini kurang tepat karena coding berada pada tahap Implementasi dari siklus pengembangan perangkat lunak yang dibagi menjadi beberapa persentase kemampuan seperti Perencanaan (15%), Analisis (20%), Desain (35%) dan Implementasi (30%) ," jelas Romi.

Roger S. Pressman: Pengaruh perangkat lunak pada budaya dan masyarakat kita masih sangat kuat. Industri perangkat lunak terus bekerja untuk menciptakan teknologi yang akan membuatnya lebih sederhana, lebih cepat, dan lebih murah untuk membuat aplikasi komputer berkualitas tinggi. Beberapa dari teknologi ini diarahkan pada domain aplikasi tertentu, seperti desain dan implementasi situs web, sementara yang lain bersifat lebih umum, seperti sistem berorientasi objek dan sistem operasi seperti LINUX.

b. Definisi Umum

Tahap implementasi sistem adalah kondisi saat sistem dipersiapkan untuk digunakan. Diawali dan diakhiri dengan pengujian sistem dilakukan dalam implementasi ini. Atau dengan kata lain, itu adalah implementasi sistem berdasarkan desain. Pada tahap ini, sistem yang telah dirancang pada tahap sebelumnya mulai digunakan melalui penggunaan perangkat lunak dan perangkat keras. Hasilnya dapat dioperasikan dan digunakan sesuai kebutuhan dengan sebaik-baiknya dengan penerapan sistem yang telah dirancang.

Implementasi sistem adalah tahap mempraktikkan desain sistem yang telah dibuat sebelumnya dalam bentuk program. Menerapkan perangkat keras, perangkat lunak, dan antarmuka adalah semua aspek implementasi sistem.

2. Pemrograman Fungsional

Suatu jenis bahasa pemrograman yang dikenal sebagai "pemrograman berbasis fungsi" memanfaatkan gagasan fungsi matematika. Dalam matematika, suatu fungsi harus selalu menghasilkan hasil yang sama ketika pembenaran yang sama digunakan. Kontrol program diserahkan ke prosedur yang terdaftar dalam bahasa prosedural saat aliran program melewati prosedur. Status program berubah saat kontrol mengalir dari satu prosedur ke prosedur lainnya. Python dapat digunakan untuk ini karena pendekatan pemrograman fungsional lebih menekankan pada matematika, khususnya bagian percabangan dan perulangan.

Apa yang terjadi jika kita ingin menjelajahi masalah pemrograman dan pengkodean teknis yang lebih kompleks? Dua disiplin ilmu lagi, yaitu Algoritma dan Struktur Data serta Bahasa Pemrograman, yang mempelajari lebih detail tentang topik ini.

Dari mana asalnya dan bagaimana itu mungkin? Dengan kata lain, Ilmu Komputer memiliki definisi, ruang lingkup, klasifikasi, dan kategorisasi seperti halnya disiplin ilmu lainnya. Satuan Tugas, diarahkan oleh Peter J. Denning dan kemudian dikenal sebagai Denning Matrix, didirikan oleh IEEE (Institute of Electrical and Electronics

Engineers) dan ACM (Association for Computing Machinery (http://acm.org)) dan diproduksi klasifikasi yang paling terkenal. Matriks Denning dengan jelas membedakan antara bidang rekayasa perangkat lunak dan algoritme, struktur data, dan bahasa pemrograman. SMK, perguruan tinggi, atau universitas sebaiknya menggunakan nama jurusan (atau program studi): Pemrograman Komputer, Algoritma dan Struktur Data, atau Bahasa Pemrograman, jika kontennya memang hanya mencakup masalah teknis terkait bahasa pemrograman. Inilah yang saya sebutkan di paragraf pertama.

Pemrograman Berbasis Aspek, Paradigma berorientasi objek untuk rekayasa perangkat lunak cukup berhasil. upaya peningkatan readability, reusability, dan structure decomposition (modularity) guna meningkatkan kualitas rancangan program. Kekuatan pendorong utama di balik transisi dari pemrograman prosedural ke pemrograman berbasis objek adalah ini. Lokalisasi komponen dan objek dalam desain berbasis objek didasarkan pada unit fungsional (seperti buku, akun, dan log). desain unit fungsional sederhana yang dapat diubah untuk memasukkan fitur yang melibatkan beberapa unit fungsional yang berbeda. Misalnya, unit fungsi Log ditambahkan ke unit fungsi akun untuk membuat unit fungsi pelacakan, pemrofilan, atau audit. Sebagai akibat alami dari adanya banyak fitur dalam satu unit fungsi, hamburan (satu kekhawatiran muncul di mana-mana) dan kekusutan (satu objek/komponen memiliki berbagai jenis kekhawatiran) cenderung terjadi secara bersamaan. Desain berbasis objek tidak lagi memiliki modularitas yang tajam sebagai akibatnya. Sebuah konsep untuk menciptakan paradigma baru yang berorientasi pada aspek telah berkembang. Sebuah fitur dalam satu unit fungsi dapat diwakili oleh satu aspek, yang mewakili satu unit fungsi (kepedulian).

3. Tujuan implementasi sistem

Tahap implementasi sistem dibagi menjadi dua tahapan yaitu implementasi sistem dan batasan implementasi. Tujuan membangun sistem adalah untuk dapat diimplementasikan dan dioperasikan, dengan kata lain sistem yang sudah dirancang dapat digunakan sesuai kebutuhan organisasi.

4. Tahapan proses implementasi

Tindakan yang dilakukan untuk menyelesaikan prosedur penginstalan program berbasis web agar beroperasi secara offline adalah sebagai berikut:

4.1) Pastikan server menjalankan perangkat lunak XAMPP (Anda dapat menggunakan salah satunya).

4.2) Aktifkan Apache dan MySQL di program Konfigurasi Perangkat Lunak XAMPP (xampp-control.exe).

4.3) Buka situs berikut untuk menyiapkan database saat ini di PhpMyAdmin: http://localhost/phpmyadmin/

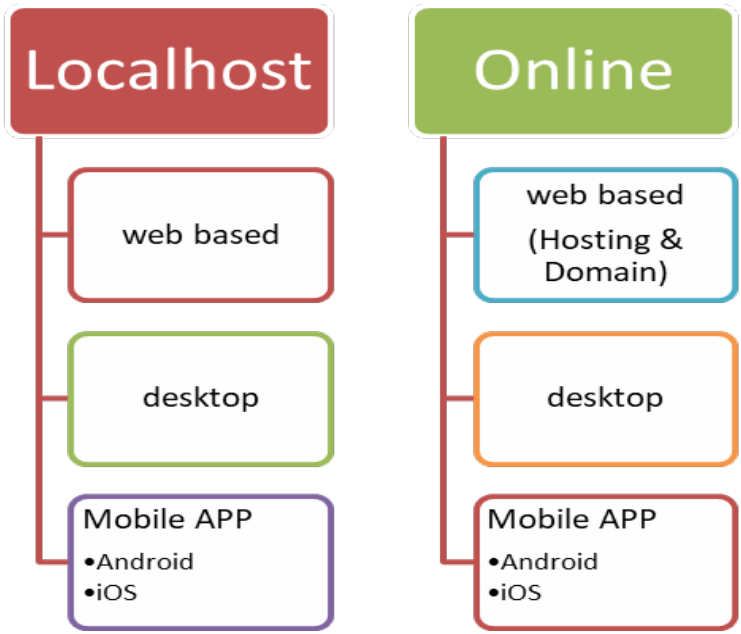
4.4) Simpan file web_siakad ke folder htdocs untuk mengatur penyimpanan file data offline.

4.5) Memasukkan database yang disediakan

4.6) Setelah prosedur instalasi selesai, gunakan alamat berikut untuk mengakses menu utama Sistem Informasi Akademik: http://localhost/web_siakad/ dan http://localhost/web_siakad/admin.manage/ untuk administrator.

Tubuh rekayasa perangkat lunak dari pengetahuan yang diakui secara umum dijelaskan dalam Panduan untuk Badan Pengetahuan Rekayasa Perangkat Lunak (SWEBOK) Software Engineering Body of Knowledge. 15 bidang pengetahuannya (KA) mencakup bagian referensi dengan daftar sumber untuk informasi yang lebih mendalam.

5. Jenis-jenis Implementasi



Gambar 1. Implementasi aplikasi lokalhost dan online

6. Client server Implementasi

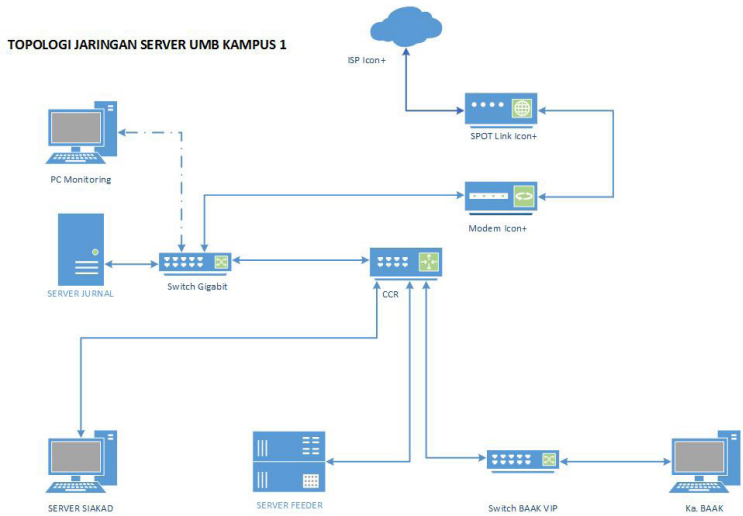
Pada bagian analisis sistem atau yang sering kita kenal dengan nama SKPL (Spesifikasi Kebutuhan Perangkat Lunak) harusnya sudah dideskripsikan kebutuhan ini. Jika cukup akses lokal maka banyak masalah teknik, namun jika akan dionlinekan maka butuh banyak kajian dan pertimbangan yang matang. Diantaranya adalah:

6.1 Sewa domain dan hosting

6.2 Mambangun data center sendiri

6.3 Sewa cloud hosting

7. Topologi System



Gambar 2. Contoh topologi implementasi sistem jaringan

8. Komposisi Tim

Alangkah baiknya dalam tahap implementasi ini melibatkan orang yang sama dalam fase perencanaan, sehingga nyambung dalam komunikasi dan pandangan. Pun begitu juga sebaliknya, dalam fase awal peracangan harusnya melibatkan orang teknis lapangan bukan hanya level manajaerial semata agar hasilnya baik dan maksimal.

9. Dokumentasi Implementasi

Jangan lupa dokumentasikan setiap proses implementasi, baik yang gagal maupun sukses, sehingga ketika suatu saat dibutuhkan dapat terdata dengan baik. Ini sangat penting apalagi organisasi yang perputaran SDMnya sangat kencang.

Rekayasa Perangkat Lunak tidak hanya mempelajari pengkodean teknis seperti cabang ilmu komputer lainnya. Hal ini sering disalah pahami, sehingga ketika dihadapkan dengan buku teks Rekayasa Perangkat Lunak yang selalu padat dengan penjelasan yang sangat luas tentang bagaimana perangkat lunak diproduksi, dari aspek menangkap persyaratan, desain, arsitektur, pengujian, kualitas perangkat lunak, hingga manajemen orang/biaya, sehingga para siswa, mahasiswa, atau bahkan calon dosen kaget. Dan ini adalah sudut pandang yang diterima

secara umum tentang rekayasa perangkat lunak, sejak publikasi Roger S. Pressman tentang "Rekayasa Perangkat Lunak: Pendekatan Praktisi" dan berlanjut melalui Ian Sommerville, serta penulis yang lebih baru seperti Hans Van Vliet, Shari Lawrence Pfleeger, dan James F. Peters.

10. Strategi Implementasi

Strategi yang paling ampuh selama ini yang penulis alami adalah *top down*, yaitu program yang memang sudah direstui pimpinan dan level manajemen tinggal diimplementasikan saja dilapangan tanpa adanya kendala yang berarti. Namun tidak menutup kemungkinan strategi *buttom up* juga bisa dilakukan.

Ungkapan "semua aspek produksi perangkat lunak" menyiratkan bahwa rekayasa perangkat lunak tidak hanya berkaitan dengan aspek teknis pengembangan perangkat lunak tetapi juga dengan tugas-tugas strategis seperti manajemen proyek perangkat lunak, memilih metodologi dan proses pengembangan, serta pertimbangan teoretis yang semuanya berfungsi untuk mendukung produksi perangkat lunak. Fase implementasi adalah salah satu area yang sama pentingnya dalam tahapan ini.

11. Tantangan Implementasi

Dari berbagai tulisan dapat kita simpulkan bahwa rekayasa perangkat lunak adalah:

suatu bidang keilmuan yang membahas semua aspek pengembangan perangkat lunak, mulai dari tahap awal requirements capture (menganalisis kebutuhan pengguna), spesifikasi (menentukan spesifikasi kebutuhan pengguna), desain, pengkodean, pengujian, dan pemeliharaan sistem setelah digunakan.

Dokumentasi pendukung dan informasi konfigurasi yang diperlukan untuk memastikan pengoperasian program yang benar juga disertakan dalam perangkat lunak, yang didefinisikan lebih dari sekadar program komputer. Menurut definisi ini, keluaran otomatis dari pengembangan perangkat lunak juga mencakup semua dokumentasi terkait selain program komputer. Pengembang terkadang gagal memahami hal ini, oleh karena itu mereka percaya bahwa memberikan perangkat lunak yang sedang berjalan kepada pengguna (klien) sudah cukup.

12. Contoh Implementasi

PT. Rajawali Nusindo yang bergerak di bidang perdagangan dan distribusi baik food maupun non-food, Iskak mengklaim sejak tahun 2022, bisnis yang tergabung dalam ID FOOD ini mengedepankan inovasi kapabilitas teknologi informasi dengan membuat aplikasi yang dibuat secara mandiri oleh tim Teknologi Informasi Korporat.

Aplikasi yang terintegrasi dengan ERP perusahaan ini menyajikan data penjualan dan laba secara edukatif yang dikumpulkan perusahaan melalui 43 cabangnya di seluruh Indonesia. Pembuatan aplikasi ini dimaksudkan untuk memungkinkan pemantauan pergerakan inventaris dan persyaratan lainnya di masa mendatang. Selain itu, aplikasi yang dibuat secara mandiri oleh tim Teknologi Informasi Perusahaan dikonsentrasikan untuk membantu tim operasional dalam mendokumentasikan dan melacak rencana kerja yang telah ditetapkan.

Dengan menggunakan software MySQL, database dibuat dengan menggunakan bahasa SQL sebagai gambaran implementasi database. Berikut ini menjelaskan bagaimana database diimplementasikan dalam SQL:

```
-- phpMyAdmin SQL Dump
-- version 2.11.9.2
-- http://www.phpmyadmin.net
--
-- Host: localhost
-- Waktu pembuatan: 02. Juni 2023 jam 19:55
-- Versi Server: 5.0.67
-- Versi PHP: 5.2.6
SET SQL_MODE=”NO_AUTO_VALUE_ON_ZERO”;
--
-- Struktur dari tabel `tbl_absensi`
--
CREATE TABLE IF NOT EXISTS `tbl_absensi` (
`idlokal` varchar(9) NOT NULL,
`nim` varchar(9) NOT NULL,
`tanggal` date NOT NULL,
`kehadiran` varchar(50) NOT NULL,
`idtahunajaran` varchar(6) NOT NULL
```

) ENGINE=MyISAM DEFAULT CHARSET=latin1;

--

-- Dumping data untuk tabel `tbl_absensi`

-- INSERT INTO `tbl_absensi` (`idlokal`, `nim`, `tanggal`, `kehadiran`, `idtahunajaran`) VALUES

('IoT-101', '2114100001', '2022-12-31', 'Izin', '2022-1'),

('IoT-101', '2114100002', '2022-12-31', 'Hadir', '2022-1'),

('IoT-101', '2114100001', '2022-12-30', 'Hadir', '2022-1'),

('IoT-101', '2114100002', '2022-12-30', 'Hadir', '2022-1'),

('IoT-102', '2114100003', '2022-12-31', 'Hadir', '2022-1'),

('IoT-102', '2114100004', '2022-12-31', 'Hadir', '2022-1'),

('IoT-101', '2114100001', '2023-01-01', 'Sakit', '2022-1'),

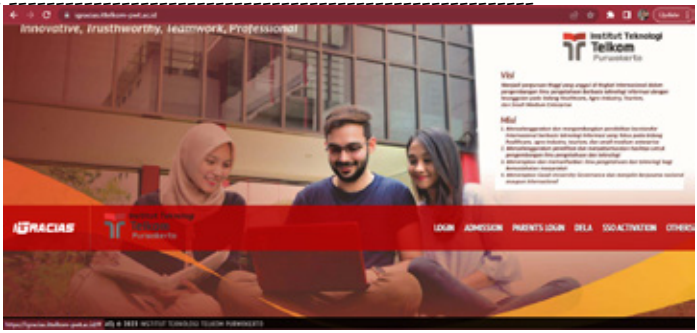
('IoT-101', '2114100002', '2023-01-01', 'Hadir', '2022-1'),

('REK-03', '2114100002', '2023-01-01', 'Hadir', '2022-1'), ('REK-03', '2114100006', '2023-01-01', 'Sakit', '2022-1'), ('REK-01', '2114100003', '2023-01-01', 'Hadir', '2022-1'), ('REK-02', '2114100001', '2023-01-01', 'Izin', '2022-1'),

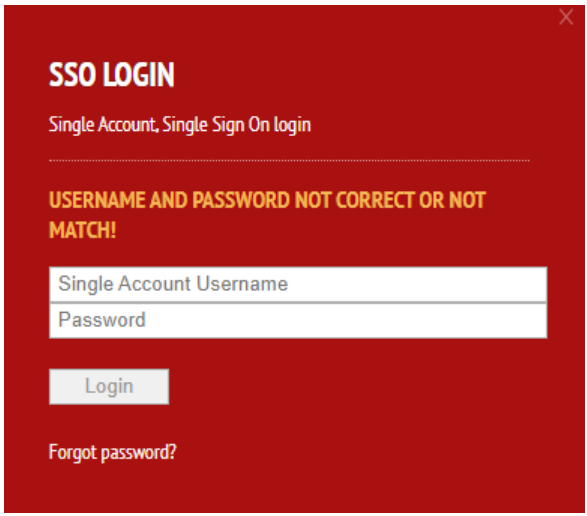
('REK-02', '2114100001', '2023-01-06', 'Alpa', '2022-1'), ('REK-03', '2114100005', '2023-01-06', 'Izin', '2022-1'),

('REK-03', '2114100005', '2023-01-01', 'Hadir', '2022-1'), ('REK-01', '2114100004', '2023-01-07', 'Sakit', '2022-1');

--



Gambar 3. Contoh Tampilan home Sistem Informasi Akademik



Gambar 5. Contoh hasil implementasi pengujian whitebox

PENDEKATAN MENULIS KODE PROGRAM

Pendekatan menulis program merupakan cara atau metode yang perlu dicari, ditemukan dan dijalankan sebagai bagian dari keseluruhan aktivitas rekayasa perangkat lunak. Hal ini dapat dimulai dengan melakukan pertimbangan-pertimbangan pencarian model seperti bagaimana memulai menulis program , bagaimana melakukan penulisan kode program berdasarkan persepsi model yang dipilih sehingga pada akhirnya dapat membantu dan mendukung keseluruhan kebutuhan dalam rekayasa perangkat lunak.

Kode Program dalam kaitannya dengan pemrograman mempunyai ruang lingkup yang luas. Ruang lingkup yang dimaksud bukan hanya bertumpu pada dua sudut pandang dari sisi perangkat lunak maupun perangkat keras saja namun bila dikaji lebih mendalam, hal ini juga termasuk bagian aktivitas dan bagaimana menyusunnya ke dalam konsep pemrograman.

Dengan menggunakan pendekatan dan menerapkannya ke dalam aktivitas rekayasa perangkat lunak pada akhirnya bertujuan untuk memperkecil ruang jarak persepsi antara penggunaan dan kebutuhan dalam koridor rekayasa perangkat lunak.

11.1 Menulis Kode Program

Menulis kode program seperti menulis sebuah cerita. Kita harus memahami alur cerita yang ingin kita buat dan memilih kata-kata yang tepat untuk menyampaikan pesan yang ingin disampaikan. Seperti halnya menulis cerita, menulis kode program juga memerlukan kreativitas dan pemikiran yang jernih.

Kode program adalah bahasa yang digunakan oleh komputer untuk memahami instruksi yang kita berikan. Seperti bahasa manusia, bahasa pemrograman juga memiliki aturan dan sintaksis yang harus diikuti agar komputer dapat memahami instruksi yang kita berikan.

Dalam menulis kode program, kita harus memperhatikan beberapa hal seperti struktur kode, variabel yang digunakan, dan algoritma yang dibuat. Struktur kode harus mudah dipahami dan diikuti oleh orang lain yang membaca kode tersebut. Variabel yang digunakan harus jelas dan mudah dimengerti. Algoritma yang dibuat harus efektif dan efisien agar program dapat berjalan dengan baik.

Selain itu, menulis kode program juga memerlukan ketelitian dan kesabaran. Kita harus memeriksa setiap baris kode yang kita tulis untuk memastikan tidak ada kesalahan yang terjadi. Kesabaran juga diperlukan ketika kita menghadapi masalah dalam menulis kode program. Kita harus bersabar dan terus mencari solusi agar program dapat berjalan dengan baik.

Dalam kesimpulannya, menulis kode program adalah sebuah seni yang memerlukan kreativitas, pemikiran jernih, ketelitian, dan kesabaran. Dengan menulis kode program yang baik, kita dapat membuat program yang efektif dan efisien untuk memenuhi kebutuhan pengguna.

11.2 Pendekatan Pada Perangkat lunak

Definisi dari pendekatan atau metafora pada perangkat lunak adalah seperti sebuah "pencarian cahaya" atau sebuah proses "penantian" daripada dilihat sebagai sebuah "peta perjalanan". Metafora tidak memberitahu dimana menemukan jawaban tapi metafora akan memberi tahu bagaimana cara untuk melihat. Metafora lebih berperan secara heuristik daripada sebagai suatu algoritma. (Kuhn, 2005)

Algoritma adalah kumpulan dari instruksi yang telah didefinisikan dengan baik, dapat diprediksi, derministik dan tidak mungkin ada karena faktor kebetulan. Sedangkan heuristik adalah sebuah teknik yang dapat membantu melihat dari sebuah jawaban.

Dari penggambaran ini dapat disimpulkan bahwa untuk dapat menggali pengetahuan mengenai perangkat lunak itu sendiri maka diperlukan model pendekatan dan pengalaman yang berbeda pada setiap orang dimana model dan bentuk pengalaman itu sendiri perlu melewati proses panjang sebagai proses "penantian" menuju "pencarian cahaya".

11.3. Pendekatan Menulis Program

Pendekatan penulisan program adalah cara atau metode yang digunakan dalam menulis kode program. Ada beberapa pendekatan yang umum digunakan dalam penulisan program, antara lain:

11.3.1 Metafora Menulis Kode Program

Berangkat dari pemahaman mengenai metafora perangkat lunak, metafora menulis kode program dapat dianggap sebagai sebuah konsep heuristik / penemuan yang dapat dicapai melalui berbagai cara yang berbeda pada setiap orang. Adapun metafora-metafora yang dijalankan dalam hal ini seperti:

a. Metafora menulis dengan pensil / pen

Metafora ini lebih melihat bahwa menulis kode program itu seperti menulis lepas di atas sebuah kertas orat-oret / kertas buram dan ketika tulisan itu menemukan jalan buntu selanjutnya kertas tersebut dibuang ke tong sampah.

Namun ketika usai melakukan penemuan melalui explorasi, jatuh bangun, pengungkapan gagasan dsb tulisan tersebut akan ditulis lebih rapi dan terstruktur dan kemudian disimpan sebagai "acuan". Tujuannya agar pencapaian "penemuan" ini setidaknya dapat dipakai sebagai sebuah "milestone" atau batu pijakan untuk beralih ke proses berikutnya.

b. Metafora bertani menumbuhkan bibit

Metafora ini lebih melihat bahwa menulis kode program itu dianggap sebagai kegiatan bertani. Di awal aktfitas, kode program ditulis seperti menyemai bibit secara serial (berurut) atau pararel (simultan).

Seri atau pararel ini dianggap sebagai suatu bagian dari rangkaian aktivitas yang dapat dijalankan secara berdiri sendiri atau salng melengkapi. Dalam persepsi penulisan secara seri sebuah kode program dapat ditulis setelah kode program sebelumnya usai dikerjakan. Dalam persepsi pararel serangkaian kode program dapat ditulis secara bersamaan tanpa harus menunggu kode program sebelum atau sesudahnya selesai dilakukan.

11.3.2 Pendekatan Berdasarkan Teknik Pemrograman

a. Procedural Programming:

Pendekatan ini fokus pada prosedur atau langkah-langkah yang harus dilakukan untuk menyelesaikan suatu tugas. Kode program ditulis dalam bentuk fungsi atau prosedur yang dapat dipanggil secara berulang-ulang.

Karakterisitik utama dari Pemrograman Prosedural adalah

- Menekankan pada penggunaan prosedur atau subrutin untuk membangun struktur program.
- Menggunakan urutan perintah untuk menjalankan program dan memanipulasi data.
- Fokus pada tugas-tugas yang harus dilakukan oleh program.
- Memiliki kemampuan untuk memecah program menjadi bagian-bagian yang lebih kecil dan terorganisir dengan baik.
- Bahasa pemrograman yang menggunakan konsep ini antara lain Pascal, C, dan BASIC.

(Yusnitasari and Umniati, 2009)

b. Object-Oriented Programming:

Pendekatan ini fokus pada objek sebagai unit dasar dalam penulisan program. Objek memiliki atribut dan metode yang dapat digunakan untuk memanipulasi data. Karakteristik utama dari pemrograman berorientasi pada objek adalah sebagai berikut:

- Menggunakan konsep objek sebagai dasar dari program.
- Objek memiliki atribut dan metode yang dapat digunakan untuk memanipulasi data.
- Fokus pada pengelompokan data dan fungsi-fungsi yang berhubungan ke dalam objek-objek terpisah.
- Memiliki kemampuan untuk memecah program menjadi bagian-bagian yang lebih kecil dan terorganisir dengan baik.

- Bahasa pemrograman yang menggunakan paradigma ini antara lain Java, C++, dan Python.
- Memungkinkan untuk membuat program yang lebih mudah dipahami dan di-maintain.
- Dapat digunakan untuk membuat program yang lebih modular dan reusable.
- Dapat digunakan untuk membuat program yang lebih aman dan terstruktur.

  (N, 2021)

c. Functional Programming:

Pendekatan ini fokus pada fungsi sebagai unit dasar dalam penulisan program. Fungsi dianggap sebagai nilai yang dapat dipindahkan dan diproses oleh fungsi lainnya.

Karakteristik Pemrograman Fungsional:

- Menghindari status dan data yang dapat berubah, dan sebaliknya menekankan penerapan fungsi.
- Fokus pada informasi apa yang diinginkan dan transformasi apa yang diperlukan, bukan cara melakukan tugas (algoritma) dan cara melacak perubahan status.
- Tidak memperhatikan urutan eksekusi, sehingga kepentingan urutan eksekusi rendah.
- Menggunakan panggilan fungsi, termasuk rekursi, sebagai kontrol aliran primer.
- Pemrograman fungsional membolehkan kita untuk menulis kode dengan potensi bug yang rendah karena setiap komponen telah diisolasi

Selain itu, pemrograman fungsional juga memungkinkan program untuk ditulis dalam gaya deklaratif, yang dapat membuat kode lebih mudah dipahami dan di-maintain

d. Event-Driven Programming:

Pendekatan ini fokus pada peristiwa atau event yang terjadi dalam program. Kode program ditulis untuk menangani peristiwa tertentu, seperti klik tombol atau input pengguna.

Pemrograman *even driven* memiliki beberapa karakteristik, yaitu:

- Pemrograman even driven didasarkan pada kejadian atau *event* tertentu, seperti klik tombol atau input dari pengguna.
- Program yang dibuat menggunakan pemrograman even driven tidak akan mengeksekusi instruksi secara berurutan, melainkan menunggu kejadian atau *event* tertentu terjadi.
- Pemrograman even driven memungkinkan program untuk merespons kejadian atau *event* secara cepat dan efisien.
- Program yang dibuat menggunakan pemrograman even driven biasanya lebih modular dan mudah di-maintain karena setiap bagian program hanya menangani satu jenis *event* atau kejadian tertentu.
- Pemrograman even driven memungkinkan program untuk berjalan secara asinkronus, artinya program dapat menjalankan beberapa tugas secara bersamaan tanpa harus menunggu tugas sebelumnya selesai terlebih dahulu.

11.3.3 Pendekatan Berdasarkan Visual

Pendekatan visual dapat membantu pemula dalam belajar pemrograman dengan cara sebagai berikut:

- Menyediakan visualisasi yang jelas dan mudah dipahami tentang konsep pemrograman yang kompleks, sehingga memudahkan pemula untuk memahami konsep tersebut. (Oktarina and Mulyani, 2022)
- Membantu pemula untuk memahami sintaks dan struktur kode dengan lebih baik, karena visualisasi dapat membantu pemula untuk melihat bagaimana kode bekerja dan bagaimana setiap bagian kode saling berhubungan.(Suhartini, 2021)
- Meningkatkan daya tarik dan minat pemula dalam belajar pemrograman, karena visualisasi dapat membuat pembelajaran lebih menarik dan menyenangkan.

- Membantu pemula untuk mengembangkan keterampilan pemecahan masalah, karena visualisasi dapat membantu pemula untuk memecahkan masalah dengan cara yang lebih sistematis dan terstruktur.
- Meningkatkan kemampuan pemula dalam mengingat dan memahami konsep pemrograman, karena visualisasi dapat membantu pemula untuk mengaitkan konsep dengan gambar atau visualisasi yang mudah diingat. (Pendidikan and Biasa, 2023)

Dalam menggunakan pendekatan visual, media visual seperti video tutorial, poster, dan animasi dapat digunakan untuk membantu pemula memahami konsep pemrograman yang abstrak. Namun, perlu diingat bahwa visualisasi yang tidak akurat atau tidak lengkap dapat menyebabkan pemahaman yang tidak akurat atau tidak lengkap tentang konsep pemrograman. Oleh karena itu, penting untuk memilih visualisasi yang akurat dan lengkap, serta menyeimbangkan penggunaan visualisasi dengan pemahaman yang lebih dalam tentang konsep pemrograman yang mendasar. Selain itu, pemilihan visualisasi yang menarik dan sesuai dengan minat pemula dapat membuat pembelajaran menjadi lebih menarik dan efektif.

Akhir kata pilihan pendekatan penulisan kode program tergantung pada kebutuhan dan tujuan dari program yang akan dibuat. Setiap pendekatan memiliki kelebihan dan kekurangan masing-masing, sehingga penting untuk memilih pendekatan yang paling sesuai dengan kebutuhan program yang akan dibuat dalam aktivitas rekayasa perangkat lunak

TEKNIS MENULIS PROGRAM: PANDUAN PRAKTIS UNTUK PEMULA

Chapter ini bertujuan untuk memberikan panduan praktis kepada pemula dalam menulis program. Kami akan membahas langkah-langkah dasar yang perlu diperhatikan dalam proses penulisan program, termasuk pemahaman dasar tentang sintaksis, alur kontrol, dan penggunaan fungsi. Kami juga akan membahas praktik terbaik dalam menulis program yang efisien dan mudah dibaca. Dalam dunia yang semakin tergantung pada teknologi, pengetahuan dan keterampilan dalam menulis program menjadi sangat penting. Dengan memahami dasar-dasar penulisan program, seseorang dapat mengembangkan aplikasi, memecahkan masalah, dan berkontribusi dalam dunia digital yang terus berkembang.

Pada bab ini, akan memebahas tentang pengenalan bahasa pemrograman dan pemahaman dasar tentang sintaksis, penjelasan konsep dasar seperti variabel, tipe data, dan operasi dasar yang digunakan dalam hampir semua bahasa pemrograman. Selanjutnya, akan membahas alur kontrol, yang merupakan struktur dasar dalam programan. Anda akan belajar tentang urutan, percabangan, dan perulangan, serta penggunaan pernyataan kondisional seperti if-else dan switch-case. Kami juga akan membahas penggunaan loop dengan menggunakan for, while, dan do-while untuk mengulang kode dengan efisien.

Selain itu, dalam bab ini kami akan menjelaskan pentingnya penggunaan fungsi dalam penulisan program. Anda akan belajar tentang pengenalan dasar fungsi, membuat fungsi, dan menggunakan parameter serta pengembalian nilai. Memahami konsep fungsi akan membantu Anda dalam mengorganisir kode menjadi bagian-bagian yang lebih terpisah dan dapat digunakan kembali. Pembahasan penanganan error dan debugging. Kami akan menjelaskan bagaimana mengidentifikasi dan menangani kesalahan umum dalam penulisan

program. Selain itu membahas penggunaan mekanisme debugging yang dapat membantu menemukan dan memperbaiki kesalahan dalam penulisan kode dengan cepat dan efisien.

Selama memepelajari penulian kode program ini, diharapkan dapat mengerti dan menginternalisasi konsep-konsep dasar yang dibahas. Namun, penting untuk diingat bahwa menulis program bukan hanya tentang memahami sintaksis dan algoritma, tetapi juga melibatkan kreativitas dan pemecahan masalah. Dalam perjalanan menjadi seorang programmer, selalu berusaha untuk terus belajar dan mengembangkan diri, mengikuti perkembangan teknologi dan tren industri.

Kami harap chapter ini dapat menjadi panduan yang berguna bagi Anda sebagai pemula dalam menulis program. Selamat belajar dan semoga sukses dalam perjalanan menjadi seorang programmer yang terampil dan terampil!

1. Pemahaman Dasar

Pemahaman dasar merupakan fondasi yang sangat penting dalam menulis program. Dalam bab ini, kita akan melihat secara lebih mendalam konsep-konsep yang mendasari bahasa pemrograman dan sintaksis. Pertama-tama, mari kita mempelajari bahasa pemrograman itu sendiri. Bahasa pemrograman adalah sistem yang digunakan untuk mengomunikasikan instruksi kepada komputer. Ada banyak bahasa pemrograman yang tersedia, seperti Python, Java, C++, dan lainnya, masing-masing dengan sintaksis dan fitur unik. Pemilihan bahasa pemrograman yang tepat tergantung pada kebutuhan dan tujuan proyek yang sedang Anda kerjakan.

Selanjutnya, kita akan mempelajari konsep dasar dalam bahasa pemrograman. Salah satu konsep utama adalah variabel. Variabel digunakan untuk menyimpan data dalam program, seperti angka, teks, atau nilai logika. Saat menggunakan variabel, Anda perlu memperhatikan tipe data yang sesuai, misalnya integer, float, string, atau boolean. Memahami tipe data dan bagaimana menggunakannya secara efektif sangat penting untuk menghindari kesalahan dan memastikan konsistensi data dalam program.

Setelah mempelajari variabel dan tipe data, kita akan menjelajahi

operasi dasar yang dapat dilakukan pada data. Operasi dasar termasuk operasi matematika seperti penjumlahan, pengurangan, perkalian, dan pembagian, serta operasi logika seperti AND, OR, dan NOT. Pemahaman yang kuat tentang operasi dasar akan membantu Anda dalam memanipulasi dan memanfaatkan data dalam program.

Selain itu, kita juga akan membahas konsep struktur data dasar seperti array dan string. Array adalah koleksi elemen-elemen yang serupa, sementara string adalah kumpulan karakter. Pemahaman yang baik tentang bagaimana menggunakan array dan string akan memungkinkan Anda untuk mengelola dan memanipulasi data dengan lebih efisien.

Dalam bab ini, kami akan menggali lebih dalam tentang konsep pemrograman seperti ekspresi, operator, pernyataan, dan blok kode. Ekspresi adalah kombinasi dari variabel, operasi, dan nilai yang menghasilkan hasil tertentu. Operator digunakan untuk melakukan operasi pada ekspresi. Pernyataan adalah instruksi dalam program yang mengendalikan alur eksekusi, sedangkan blok kode adalah kumpulan pernyataan yang dikelompokkan bersama.

Selain itu, kami akan membahas konsep pemrograman yang lebih lanjut seperti struktur pengendalian alur, termasuk percabangan (if-else, switch-case) dan perulangan (for, while, do-while). Memahami cara mengendalikan alur program menggunakan struktur pengendalian alur ini sangat penting dalam membuat program yang adaptif dan responsif.

Dalam bab ini, kami akan memberikan contoh nyata dan latihan untuk menguji pemahaman Anda tentang konsep-konsep dasar ini. Pemahaman yang kuat tentang pemrograman dasar akan memberikan landasan yang kokoh untuk memahami konsep-konsep yang lebih kompleks di bab-bab berikutnya. Selain itu, pemahaman yang baik tentang pemrograman dasar akan membantu Anda mengatasi tantangan dalam menulis program, menganalisis dan memecahkan masalah dengan lebih efisien.

Dalam bab ini, kami juga akan membahas beberapa prinsip desain dan gaya penulisan yang baik dalam pemrograman. Kami akan menyoroti pentingnya menggunakan nama variabel yang deskriptif,

mengikuti konvensi penamaan yang umum, dan mengatur kode dengan cara yang mudah dibaca dan dimengerti. Selain itu, kami akan menjelaskan pentingnya komentar dalam kode, yang dapat membantu Anda dan orang lain memahami logika program dengan lebih baik.

Sebagai pemula dalam menulis program, memahami pemrograman dasar adalah langkah awal yang penting untuk membangun keterampilan yang lebih tinggi. Dengan dasar yang kuat, Anda akan dapat memahami dan menerapkan konsep-konsep yang lebih canggih dalam pemrograman seperti pemrograman berorientasi objek, manipulasi data, dan pemecahan masalah kompleks.

Kami berharap bahwa setelah membaca bab ini, Anda akan memiliki pemahaman yang lebih baik tentang pemrograman dasar dan mampu mengaplikasikan konsep-konsep tersebut dalam penulisan program. Melalui latihan dan eksperimen yang berkelanjutan, Anda akan dapat memperkuat pemahaman Anda dan terus berkembang sebagai seorang pemrogram yang terampil.

Ingatlah bahwa belajar pemrograman adalah proses yang berkelanjutan dan memerlukan dedikasi serta latihan yang konsisten. Selalu mencari kesempatan untuk menerapkan konsep-konsep yang telah Anda pelajari dalam proyek nyata dan terus mencari sumber belajar yang relevan. Dengan kesabaran dan tekad, Anda akan menjadi seorang pemrogram yang mahir dan dapat menghasilkan solusi kreatif dalam dunia teknologi yang terus berkembang.

2. Alur Kontrol

Alur kontrol adalah konsep penting dalam pemrograman yang memungkinkan kita mengendalikan bagaimana program berperilaku dalam berbagai situasi. Dalam bagian ini, kita akan menjelajahi lebih dalam tentang struktur pengendalian alur, termasuk percabangan (if-else, switch-case) dan perulangan (for, while, do-while).

Pertama-tama, mari kita bahas struktur percabangan. Percabangan memungkinkan program untuk memilih jalur eksekusi yang berbeda berdasarkan kondisi tertentu. Salah satu bentuk percabangan yang umum adalah pernyataan if-else. Dalam pernyataan if-else, program akan mengevaluasi kondisi tertentu. Jika kondisi tersebut benar (true),

maka blok kode yang terkait dengan if akan dieksekusi. Jika kondisi tersebut salah (false), maka blok kode yang terkait dengan else akan dieksekusi, jika ada.

Selain pernyataan if-else, kita juga dapat menggunakan pernyataan switch-case. Pernyataan switch-case digunakan ketika ada beberapa pilihan yang mungkin dalam alur eksekusi program. Program akan mengevaluasi ekspresi tertentu dan mencocokkannya dengan pilihan yang sesuai. Jika pilihan cocok, maka blok kode yang terkait dengan kasus (case) tersebut akan dieksekusi.

Selanjutnya, mari kita bahas perulangan. Perulangan memungkinkan program untuk mengulang bagian kode tertentu berulang kali sampai kondisi tertentu terpenuhi. Salah satu bentuk perulangan yang umum adalah perulangan for. Perulangan for memungkinkan kita untuk menentukan awal, kondisi, dan perubahan iterasi. Dengan menggunakan perulangan for, kita dapat mengulang kode dengan jumlah iterasi yang telah ditentukan dengan jelas.

Selain perulangan for, kita juga memiliki perulangan while dan do-while. Perulangan while akan mengulang kode selama kondisi tertentu terpenuhi. Sedangkan perulangan do-while akan menjalankan kode setidaknya satu kali sebelum memeriksa kondisi untuk menentukan apakah harus melanjutkan pengulangan atau tidak. Memahami dan menggunakan dengan benar struktur pengendalian alur ini sangat penting untuk mengendalikan aliran eksekusi program. Dengan penggunaan yang tepat, kita dapat membuat program yang lebih adaptif, fleksibel, dan efisien.

Dalam bab ini, kami akan memberikan contoh nyata dan latihan untuk melatih pemahaman Anda tentang struktur pengendalian alur. Anda akan diberikan kesempatan untuk menerapkan percabangan dan perulangan dalam situasi yang berbeda, dan belajar bagaimana menggunakan alur kontrol untuk mengatasi tantangan pemrograman yang lebih kompleks. Selain itu, kami juga akan membahas beberapa praktik terbaik dalam menggunakan alur kontrol, seperti menghindari nested if-else yang berlebihan dan menggunakan struktur yang lebih bersih dan mudah dipahami. Kami akan membahas penggunaan komentar untuk menjelaskan logika di balik alur kontrol yang kompleks

dan memberikan tip dan trik dalam mengoptimalkan penggunaan struktur pengendalian alur.

Selain itu, kita juga akan menjelajahi konsep lain yang terkait dengan alur kontrol, seperti penggunaan pernyataan break dan continue. Pernyataan break digunakan untuk menghentikan perulangan atau keluar dari blok switch-case secara paksa. Pernyataan continue, di sisi lain, digunakan untuk melompati iterasi saat ini dalam perulangan dan melanjutkan ke iterasi berikutnya.

Selama mempelajari alur kontrol, penting untuk menghindari penggunaan yang berlebihan atau terlalu rumit. Terlalu banyak percabangan dan perulangan yang bersarang dapat mengakibatkan kode yang sulit dipahami dan dipelihara. Kami akan membahas strategi untuk menghindari penggunaan yang berlebihan dan mengoptimalkan alur kontrol, termasuk penggunaan logika boolean, pengelompokan kondisi, dan penulisan kode yang bersih dan terstruktur.

Dalam bab ini, kami akan memberikan contoh-contoh praktis yang melibatkan alur kontrol yang kompleks, seperti pengolahan data, pengulangan dengan kondisi yang bervariasi, dan penanganan kasus-kasus khusus. Anda akan diajak untuk menerapkan konsep-konsep yang telah dipelajari dalam situasi yang realistis, sehingga Anda dapat mengasah kemampuan pemrograman Anda dalam mengendalikan alur eksekusi program dengan tepat.

Selama melanjutkan pembelajaran Anda tentang alur kontrol, penting untuk berlatih secara konsisten. Praktik membuat kode dengan alur kontrol yang berbeda dalam skala yang semakin kompleks akan membantu Anda memperkuat pemahaman dan keterampilan pemrograman Anda. Selalu mencoba untuk mencari solusi yang efisien dan elegan, serta melibatkan diri dalam proyek-proyek nyata yang membutuhkan penggunaan yang kuat dari alur kontrol.

Dengan pemahaman yang mendalam tentang alur kontrol, Anda akan memiliki kekuatan untuk mengontrol perilaku program Anda dengan cermat, menghadapi tantangan pemrograman yang kompleks, dan menciptakan solusi yang efektif dalam pengembangan perangkat lunak. Teruslah belajar, berlatih, dan eksperimen, dan Anda akan melihat kemajuan yang pesat dalam perjalanan Anda sebagai seorang pemrogram yang terampil.

Misalkan Anda sedang membuat program sederhana untuk menghitung nilai rata-rata dari sejumlah angka yang diinput oleh pengguna. Program ini akan terus meminta pengguna memasukkan angka-angka baru hingga pengguna memasukkan angka negatif sebagai penanda akhir. Setelah itu, program akan menghitung nilai rata-rata dari angka-angka yang dimasukkan.

Berikut adalah contoh algoritma yang menggambarkan alur kontrol untuk kasus tersebut:

Inisialisasi variabel total dengan nilai 0 dan variabel count dengan nilai 0.

Mulai perulangan yang berulang selama kondisi tertentu terpenuhi (misalnya, menggunakan perulangan while dengan kondisi "selama angka yang dimasukkan bukan negatif").

Dalam setiap iterasi perulangan:

Minta pengguna memasukkan angka.

Jika angka yang dimasukkan negatif, keluar dari perulangan menggunakan pernyataan break.

Tambahkan angka ke variabel total.

Tambahkan 1 ke variabel count.

Setelah keluar dari perulangan, hitung nilai rata-rata dengan membagi variabel total dengan variabel count.

Tampilkan nilai rata-rata kepada pengguna.

Berikut adalah contoh implementasi dalam bahasa pemrograman Python:

Dalam contoh ini, perulangan while digunakan dengan kondisi "True", sehingga perulangan akan berlanjut sampai pengguna memasukkan angka negatif dan pernyataan break dieksekusi untuk keluar dari perulangan.

```
total = 0
count = 0

while True:
    angka = int(input("Masukkan angka (masukkan angka negatif untuk selesai)

    if angka < 0:
        break

    total += angka
    count += 1

if count > 0:
    rata_rata = total / count
    print("Nilai rata-rata:", rata_rata)
else:
    print("Tidak ada angka yang dimasukkan.")
```

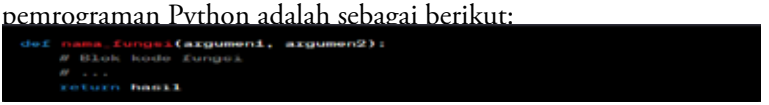
Semoga contoh kasus di atas membantu Anda memahami cara penggunaan alur kontrol dalam sebuah program. Anda dapat melakukan variasi pada kasus tersebut dengan menambahkan fitur atau logika tambahan sesuai kebutuhan.

3. Fungsi

Fungsi merupakan konsep penting dalam pemrograman yang memungkinkan kita untuk mengorganisasi dan mengelompokkan bagian-bagian kode yang melakukan tugas tertentu. Dengan menggunakan fungsi, kita dapat memecah program menjadi bagian-bagian yang lebih kecil dan lebih terkelola, sehingga memudahkan dalam pengembangan, pemeliharaan, dan penggunaan kembali kode. Sebuah fungsi biasanya menerima input *(argumen)* tertentu, melakukan operasi tertentu, dan menghasilkan *output* (nilai kembalian). Fungsi dapat digunakan dengan memanggilnya dari bagian lain dalam program, sehingga menghindari duplikasi kode dan mempromosikan reusabilitas.

Berikut adalah beberapa konsep yang perlu dipahami tentang fungsi:

1. Deklarasi Fungsi: Fungsi dideklarasikan dengan menentukan nama fungsi, argumen (jika ada), dan blok kode yang akan dieksekusi ketika fungsi dipanggil. Contoh deklarasi fungsi dalam bahasa pemrograman Python adalah sebagai berikut:



2. Argumen: Argumen adalah nilai atau variabel yang diteruskan ke fungsi saat pemanggilan. Fungsi dapat memiliki argumen yang bersifat wajib atau opsional. Argumen wajib harus diberikan saat pemanggilan fungsi, sedangkan argumen opsional memiliki nilai default yang digunakan jika tidak diberikan argumen saat pemanggilan.
3. Nilai Kembalian: Fungsi dapat mengembalikan nilai *(output)* setelah menjalankan operasi tertentu. Nilai kembalian ini berguna jika kita ingin menggunakan hasil operasi fungsi di bagian lain dalam program. Penggunaan pernyataan return digunakan untuk mengembalikan nilai dari fungsi.

Berikut adalah contoh sederhana fungsi dalam bahasa pemrograman Python yang menghitung luas segitiga:

Pemanggilan fungsi tersebut dapat dilakukan dengan memeberikan nilai untuk argumen “alas” dan “tinggi”, seperti ini :

Dalam contoh ini, **fungsi hitung_luas_segitiga** menerima dua argumen, yaitu alas dan tinggi. Fungsi tersebut menghitung luas segitiga berdasarkan rumus yang diberikan, kemudian mengembalikan nilai luas tersebut dengan menggunakan pernyataan return. Hasil kembalian dari fungsi dipindahkan ke variabel luas_segitiga, dan kemudian ditampilkan ke layar.

Dengan menggunakan fungsi, kita dapat mengelompokkan kode yang terkait dengan tugas tertentu, sehingga mempermudah pemahaman, pemeliharaan, dan pengembangan program. Selain itu, fungsi juga mempromosikan reusabilitas kode

4. Penanganan Error dan Debugging

Kesalahan dalam penulisan program adalah hal yang umum terjadi saat kita melakukan pengembangan perangkat lunak. Namun, dengan penanganan yang tepat, kita dapat mengidentifikasi dan memperbaiki kesalahan tersebut. Dalam bab ini, kita akan membahas beberapa kesalahan umum dalam penulisan program serta mekanisme debugging yang dapat digunakan untuk menemukan dan memperbaiki kesalahan tersebut. Kesalahan-kesalaha umum dalam penulisan program:

1) **Kesalahan sintaksis:** Kesalahan ini terjadi ketika kode tidak mengikuti aturan sintaksis yang ditentukan oleh bahasa pemrograman. Contohnya, kurangnya tanda titik koma (;) di akhir pernyataan, atau penggunaan tanda kurung yang tidak seimbang.
2) **Kesalahan logika:** Kesalahan ini terjadi ketika logika di dalam kode tidak sesuai dengan yang diinginkan. Contohnya, penggunaan kondisi yang salah dalam pernyataan if, atau perhitungan matematika yang salah.
3) **Kesalahan penulisan nama variabel atau fungsi:** Kesalahan ini terjadi ketika kita salah mengetik nama variabel atau fungsi saat penggunaan di dalam kode. Contohnya, penulisan print sebagai prinnt.
4) **Kesalahan dalam pemrosesan input:** Kesalahan ini terjadi ketika program tidak memproses input dengan benar, seperti tidak melakukan validasi input atau tidak menangani kasus-kasus khusus.

Mekanisme Debugging:

Pernyataan print: Salah satu cara paling sederhana untuk melakukan debugging adalah dengan menambahkan pernyataan print di dalam kode untuk mencetak nilai-nilai variabel pada titik-titik tertentu. Ini membantu kita melacak bagaimana nilai berubah selama eksekusi program.

Debugger: Debugger adalah alat yang digunakan untuk menganalisis dan memeriksa kode secara interaktif. Dengan debugger, kita dapat menghentikan eksekusi program pada titik-titik tertentu, memeriksa nilai variabel, melangkah ke baris kode berikutnya, dan banyak lagi.

Logging: Logging adalah teknik yang melibatkan pencatatan pesan atau kejadian penting dalam program. Pesan-pesan log ini membantu dalam memahami alur eksekusi dan menemukan kesalahan saat program berjalan.

Mencoba-coba (trial and error): Kadang-kadang, untuk menemukan kesalahan dalam kode, kita perlu melakukan percobaan berulang dengan mengubah atau mengomentari beberapa bagian kode. Ini membantu dalam mengisolasi dan mengidentifikasi penyebab kesalahan.

Selama proses debugging, penting untuk memiliki pemahaman yang kuat tentang kode yang ditulis, dan melibatkan diri secara aktif dalam memeriksa dan mencari kesalahan. Juga, menggunakan teknik debugging yang tepat dan mengambil langkah-langkah yang sistematis dalam mengidentifikasi dan memperbaiki kesalahan dapat mempercepat proses debugging.

Ingatlah bahwa kesalahan adalah bagian dari proses belajar dan mengembangkan perangkat lunak. Dengan pengalaman dan latihan yang terus berkembang, Anda akan menjadi lebih terampil dalam mengenali dan menangani kesalahan dalam penulisan program. Berikut adalah beberapa tips tambahan untuk membantu Anda dalam penanganan error dan debugging:

1) Membaca Pesan Error: Ketika program menghasilkan pesan error, sangat penting untuk membacanya dengan cermat.
2) Pesan error sering kali memberikan petunjuk tentang jenis kesalahan yang terjadi dan lokasi di mana kesalahan tersebut terjadi.
3) Perhatikan pesan error secara seksama untuk mengidentifikasi masalah potensial.
4) Penggunaan Try-Except: Dalam beberapa kasus, kita dapat menggunakan blok try-except untuk menangkap dan menangani kesalahan yang mungkin terjadi selama eksekusi program. Dengan mekanisme ini, kita dapat mengantisipasi kesalahan tertentu dan melakukan tindakan yang sesuai.

   Contoh penggunaan try-except dalam bahasa pemrograman Python:

```
try:
    # Blok kode yang mungkin menyebabkan kesalahan
    # ...
except JenisKesalahan:
    # Tindakan yang akan diambil jika kesalahan terjadi
    # ...
```

JenisKesalahan dapat berupa jenis kesalahan spesifik yang ingin ditangani, seperti ValueError, TypeError, atau FileNotFoundError. Dengan menggunakan try-except, kita dapat menghindari program langsung berhenti ketika kesalahan terjadi dan memberikan penanganan yang lebih terkendali.

5) Penggunaan Logging Level: Ketika menggunakan mekanisme logging untuk debugging, kita dapat menggunakan tingkatan logging yang berbeda untuk mengontrol tingkat detail pesan yang dicatat. Misalnya, kita dapat menggunakan tingkatan DEBUG untuk mencatat pesan detail selama proses debugging, dan kemudian mengubahnya menjadi tingkatan INFO atau ERROR di lingkungan produksi. Hal ini membantu dalam memfokuskan pesan log yang relevan dengan tahap pengembangan dan penggunaan aplikasi.
6) Uji Kasus (Test Cases): Pembuatan uji kasus yang komprehensif dapat membantu dalam mengidentifikasi kesalahan dalam kode. Dengan menguji berbagai skenario dan situasi yang mungkin terjadi, kita dapat mengidentifikasi kemungkinan bug dan menguji apakah kode berperilaku sesuai dengan yang diharapkan.
7) Berkolaborasi dengan Komunitas: Jika Anda mengalami kesulitan dalam menemukan dan memperbaiki kesalahan, jangan ragu untuk berkolaborasi dengan komunitas pemrograman. Forum online, grup diskusi, atau komunitas pengembang dapat menjadi sumber informasi dan bantuan yang berharga. Berbagi masalah yang Anda hadapi dengan komunitas dapat membantu Anda mendapatkan sudut pandang baru dan solusi yang mungkin tidak Anda pertimbangkan sebelumnya.

Ingatlah bahwa debugging adalah proses iteratif dan kadang-kadang membutuhkan waktu dan kesabaran. Dengan ketekunan dan pendekatan yang sistematis, Anda dapat meningkatkan kemampuan Anda dalam menemukan dan memperbaiki kesalahan dalam kode program.

5. Praktik Terbaik dalam Menulis Program

Menulis program dengan praktik terbaik adalah penting untuk menghasilkan kode yang mudah dipahami, terkelola, dan dapat dipelihara. Berikut adalah beberapa praktik terbaik dalam menulis program:

1) Penamaan Variabel yang Deskriptif: Memberikan nama yang deskriptif dan bermakna pada variabel sangat penting untuk memahami tujuan dan konteks penggunaan variabel tersebut. Hindari menggunakan nama variabel yang terlalu umum seperti "a", "x", atau "temp". Pilih nama yang menggambarkan informasi yang disimpan dalam variabel, misalnya "jumlah_mahasiswa" atau "nama_pengguna".
2) Konvensi Penulisan yang Baik: Mengikuti konvensi penulisan yang baik membantu meningkatkan keterbacaan kode dan memudahkan kolaborasi dengan pengembang lain. Setiap bahasa pemrograman memiliki konvensi penulisan yang disarankan, seperti gaya penulisan CamelCase atau snake_case. Penting untuk mengikuti konvensi penulisan yang umum digunakan dalam bahasa pemrograman yang Anda gunakan.
3) Penggunaan Komentar: Menyertakan komentar yang jelas dan informatif dalam kode membantu menjelaskan logika program dan mempermudah pemahaman kode oleh orang lain atau oleh Anda sendiri di masa depan. Komentar sebaiknya digunakan untuk menjelaskan algoritma, keputusan desain, atau penjelasan singkat tentang bagian penting dalam kode.
4) Mengorganisir Kode dengan Metode yang Terstruktur: Mengorganisir kode dengan metode yang terstruktur membantu meningkatkan keterbacaan dan pemeliharaan kode. Beberapa praktik yang dapat digunakan termasuk.
5) Memecah kode menjadi fungsi atau metode yang memiliki tanggung jawab tunggal. Hal ini membantu mengisolasi dan mengelompokkan logika terkait sehingga lebih mudah dipahami dan diuji.
6) Menggunakan kelas dan objek untuk mengorganisir kode yang terkait. Ini membantu dalam memodelkan konsep yang lebih kompleks dan mengelompokkan fungsi yang berhubungan dalam satu unit yang terpisah.

7) Menggunakan modul dan paket untuk mengorganisir kode dalam struktur yang hierarkis. Modul dan paket memungkinkan pemisahan dan pengelompokan fungsi dan kelas berdasarkan fungsionalitas mereka.
8) Meminimalkan Duplikasi Kode: Duplikasi kode dapat menyebabkan masalah dalam pemeliharaan kode. Upayakan untuk menghindari duplikasi kode dengan menerapkan prinsip DRY (Don't Repeat Yourself). Jika ada blok kode yang sering digunakan, pertimbangkan untuk memindahkannya ke fungsi atau metode yang dapat digunakan kembali.
9) Pengujian Kode (Testing): Praktik terbaik dalam menulis program mencakup pengujian kode yang baik. Buatlah unit tes yang komprehensif untuk memastikan bahwa setiap bagian kode berperilaku sesuai dengan yang diharapkan. Pengujian membantu menemukan bug atau kesalahan dalam kode sejak awal dan memastikan bahwa perubahan pada kode tidak mempengaruhi fungsionalitas yang telah ada sebelumnya. Dalam pengujian kode, pertimbangkan untuk mencakup skenario normal dan ekstrem, serta menguji batasan dan kondisi yang mungkin mempengaruhi kinerja atau keandalan program.
10) Menggunakan Version Control: Menggunakan sistem kontrol versi seperti Git membantu dalam mengelola perubahan kode dan memungkinkan kolaborasi dengan pengembang lain secara efisien. Dengan sistem kontrol versi, Anda dapat melacak perubahan, mengembalikan ke versi sebelumnya jika diperlukan, dan memudahkan proses kerja tim.
11) Membuat Dokumentasi yang Jelas: Menyertakan dokumentasi yang jelas untuk kode program sangat penting, terutama ketika kode tersebut akan digunakan oleh orang lain. Dokumentasi yang baik menjelaskan tujuan, fungsi, dan penggunaan kode, serta memberikan contoh penggunaan yang berguna. Ini membantu pengguna lain untuk dengan mudah memahami dan menggunakan kode yang telah Anda tulis.
12) Mengoptimalkan Kode: Saat menulis program, pertimbangkan untuk mengoptimalkan kode agar berjalan dengan efisien. Hindari penggunaan loop yang tidak perlu, periksa kompleksitas waktu

dan ruang, dan pertimbangkan penggunaan struktur data yang tepat. Optimasi kode dapat meningkatkan kinerja dan efisiensi program Anda.

13) Mempraktikkan Pemrograman Defensif: Pemrograman defensif adalah pendekatan di mana Anda mengantisipasi dan menangani kasus-kasus yang tidak diharapkan atau input yang tidak valid. Validasi input, penanganan kesalahan yang baik, dan penggunaan pernyataan pengaman seperti assertions adalah contoh praktik pemrograman defensif. Ini membantu mencegah kesalahan dan memperbaiki masalah sebelum mereka menyebabkan kerusakan yang lebih besar.

Dengan menerapkan praktik terbaik ini, Anda dapat meningkatkan kualitas dan keterbacaan kode program Anda, mempermudah pemeliharaan dan pengembangan lebih lanjut, serta meningkatkan kolaborasi dengan pengembang lain.

6. Studi Kasus: Program Sederhana

Dalam bab ini, kita akan melihat contoh pengembangan program sederhana dari awal hingga akhir. Kami akan menerapkan langkah-langkah yang telah dipelajari sebelumnya untuk menggambarkan proses pengembangan yang sistematis. Mari kita lihat contoh berikut: Tujuan Program: Membuat program sederhana untuk menghitung luas dan keliling persegi panjang.

Langkah-langkah Pengembangan:

1) Pemahaman Masalah: Pertama, kita harus memahami masalah yang ingin kita selesaikan. Dalam kasus ini, kita ingin membuat program untuk menghitung luas dan keliling persegi panjang.
2) Perencanaan dan Desain: Setelah memahami masalah, kita perlu merencanakan dan merancang solusi. Ini melibatkan pemikiran tentang algoritma yang akan digunakan dan struktur kode yang akan dibangun. Dalam kasus ini, algoritma untuk menghitung luas dan keliling persegi panjang cukup sederhana: Luas = panjang x lebar dan Keliling = 2 x (panjang + lebar).
3) Implementasi Kode: Selanjutnya, kita dapat mulai mengimplementasikan kode berdasarkan perencanaan dan desain yang telah kita buat. Dalam bahasa pemrograman Python, kode program mungkin tampak seperti ini:

```python
# Program untuk menghitung luas dan keliling persegi panjang

def hitung_luas(panjang, lebar):
    luas = panjang * lebar
    return luas

def hitung_keliling(panjang, lebar):
    keliling = 2 * (panjang + lebar)
    return keliling

# Input dari pengguna
panjang = float(input("Masukkan panjang persegi panjang: "))
lebar = float(input("Masukkan lebar persegi panjang: "))

# Hitung luas dan keliling
luas = hitung_luas(panjang, lebar)
keliling = hitung_keliling(panjang, lebar)

# Output hasil
print("Luas persegi panjang:", luas)
print("Keliling persegi panjang:", keliling)
```

4) Uji dan Debugging: Setelah implementasi kode, penting untuk menguji program dan menemukan serta memperbaiki kesalahan yang mungkin terjadi. Jalankan program dengan memberikan beberapa input dan periksa apakah keluaran sesuai dengan harapan.
5) Dokumentasi: Terakhir, penting untuk menyertakan dokumentasi yang jelas untuk program kita. Kita dapat menambahkan komentar di dalam kode untuk menjelaskan logika program, input yang diharapkan, dan output yang dihasilkan. Dokumentasi yang baik membantu pengguna lain untuk memahami dan menggunakan program dengan mudah.

Dalam contoh ini, kita mengikuti langkah-langkah yang telah dipelajari sebelumnya. Mulai dari pemahaman masalah, perencanaan dan desain, implementasi kode, uji dan debugging, hingga dokumentasi. Dengan pendekatan yang sistematis seperti ini, kita dapat mengembangkan program dengan lebih efektif dan terorganisir.

Kesimpulan

Ringkasan dari apa yang telah dipelajari dalam chapter ini dan pentingnya menguasai teknis menulis program. Mendorong pembaca untuk terus berlatih dan mengembangkan keterampilan mereka dalam penulisan program.

PRINSIP-PRINSIP PEMROGRAMAN

Apakah Anda seorang *software developer*? Apakah Anda pernah mengembangkan perangkat lunak tanpa menemukan hambatan? Tentu tidak. *Software developer* sering menemukan hambatan dalam proses pengembangan. Hal ini sering menyulitkan dalam proses pengembangan perangkat lunak. Kendala ditemukan dari berbagai faktor seperti logika proses bisnis yang rumit, penulisan stuktur kode yang kurang baik, sampai permasalah pribadi yang mempengaruhi ke pekerjaan.

Pengembangan perangkat lunak membutuhkan serangkaian proses untuk mendapatkan perangkat lunak yang handal. Proses ini harus memperhatikan prinsip-prinsip pemrograman. Prinsip-prinsip pemrograman dapat dijadikan sebagai acuan bagi para *software developer* agar menghasilkan perangkat lunak yang handal.

Pada dasarnya pengembangan perangkat lunak melalui proses analisis kebutuhan pengguna, perancangan, pengembangan dan pengujian. Proses tersebut harus dijalankan dengan baik sesuai dengan prinsip agar dapat menghasilkan perangkat lunak yang stabil, handal dan dapat memudahkan pengguna untuk menyelesaikan pekerjaannya. Disamping itu pengembangan perangkat lunak sering melibatkan banyak orang. Prinsip-prinsip pengembangan perangkat lunak akan menjadi panduan bagi mereka untuk mengembangkan perangkat lunak sesuai dengan tujuan yang ingin dicapai. Salah satu prinsip yang harus ditekankan dalam pengembangan perangkat lunak yaitu prinsip-prinsip pemrograman. Prinsip pemrograman ini dapat menjadi pedoman untuk mengembangkan perangkat lunak. Pedoman ini jika diterapkan secara konsisten dan berkesinambungan akan mempermudah dalam proses pengembangan perangkat lunak.

***Keep It Simple, Stupid* (KISS)**

Keep it simple, stupid, menekankan pada konsep kesederhanaan. Penulisan kode program semakin sederhana akan semakin mudah untuk dipahami dan mudah dalam pemeliharaan. Sederhana dalam hal ini tidak boleh mengorbankan fungsi. Sederhana, tapi fungsi harus tetap sesuai dengan kebutuhan pengguna. Berikut beberapa pedoman *keep it simple, stupid*, yang dapat dijalankan.

Dalam menulis kode program, tentu akan banyak *method* yang dibuat. Dalam sebuah *method* usahakan maksimal memiliki 50 baris kode. Baris kode yang penting, usahakan diberi komentar untuk mempermudah membaca kode program. Pecah kode yang berhubungan dengan tampilan, logika dan *database* menggunakan berbagai metode yang ada seperti *model-view-controller*(MVC), *hierarki-model-view-controller*(HMVC), *model view-view model* (MVVM), dan *model-view-presenter*(MVP).

Don't repeat yourself

Jangan mengulang penulisan kode program! Tanpa disadari *software developer* sering menulis kode secara berulang. *Software developer* tidak bisa menjamin kode akan selalu benar akan tetapi dapat meminimalisir pengulangan kode. Jika kode digunakan berulang kali, solusi yang dapat digunakan yaitu membuat *method* tersendiri agar penulisan kode tidak berulang-ulang.

***You Aren't Gonna Need It* (YAGNI)**

Prinsip ini menekankan bahwa *software developer* sebaiknya tidak membuat kode program atau fitur yang yang tidak memiliki kegunaan pada saat itu. *Software developer* sebaiknya tidak berandai-andai bahwa sebuah fitur akan berguna di masa depan. *Software developer* sebaiknya lebih berfokus untuk membuat kode program sesuai dengan fitur yang dibutuhkan saat itu. Hal ini dapat menghemat waktu, tenaga dan biaya.

***Big Design Upfront* (DBUF)**

Big design upfront memiliki prinsip bahwa dalam pengembangan perangkat lunak yang paling utama diperhatikan yaitu proses perancangan. Perancangan pengembangan perangkat lunak akan sangat mempengaruhi proses selanjutnya sehingga pada tahap ini harus benar-benar diperhatikan.

SOLID

***S**ingle responsibility principle* (SRP). Prinsip ini menekankan bahwa saat menulis program satu kelas atau *method* diperuntukkan untuk satu fungsi tertentu. Hal ini akan membuat kode program lebih teratur dan mudah dibaca. ***O**pen closed principle* (OCP), gagasan yang memberikan kemampuan untuk mengembangkan kode yang dirancang sedemikian rupa agar memenuhi kaidah terbuka untuk menambahkan fungsi baru akan tetapi menutup kemungkinan untuk memodifikasi fungsi yang telah dibuat. Implementasi dalam pemrograman berorientasi objek dapat menggunakan konsep pewarisan.

***L**iskov subtitution principle* (LSP) merupakan sebuah konsep yang terkait dengan pewarisan. Objek yang ada di *subclass* harus memiliki perilaku mirip dengan objek yang ada di *superclass*. ***I**nterface segregation principle* (ISP), prinsip ini berpedoman bahwa pengguna tidak perlu bergantung pada antarmuka yang sebenarnya tidak dibutuhkan. Buat antarmuka sesederhana mungkin dengan prinsip fungsi kebutuhan pengguna terpenuhi. ***D**ependency inversion principle* (DIP), modul dengan tingkatan tinggi tidak boleh bergantung dengan modul yang lebih rendah akan tetapi diarahkan pada abstraksinya.

Occam's Razor

Prinsip dari *Occam's Razor* ini sudah sering diterapkan oleh banyak *software developer*. Garis besar dari prinsip ini berpedoman bahwa tidak perlu untuk membuat entitas tambahan jika memang tidak diperlukan. Efisiensi adalah kunci dari prinsip ini.

Law of Demeter

Law of Demeter menekankan bahwa suatu objek tidak boleh mengetahui detail dari objek lain. Hal ini akan mengurangi ketergantungan satu sama lain sehingga dapat mempermudah pemeliharaan dan meningkatkan flesibilitas kode nantinnya.

Avoid Premature Optimization

Software developer kadang berfikir terlalu lama untuk menentukan metode manakah yang harus mereka gunakan untuk mengoptimalkan

performa sistem. Hal ini malah akan menjadi penghambat dalam proses pengembangan perangkat lunak. Lebih baik jika berfokus untuk menggunakan metode yang sederhana terlebih dahulu. Ketika sudah berjalan baru kemudian nanti difikirkan kembali apakah metode yang digunakan tadi sudah optimal atau belum.

Measure Twice and Cut Once

Prinsip ini berpedoman bahwa kita harus berfikir dua kali untuk mengambil satu tindakan. Setiap akan membuat fungsi kode baru, pastikan bahwa kode yang akan dibuat itu berguna dan sesuai dengan kebutuhan pengguna. Pastikan juga bahwa kode dapat berfungsi dengan benar, mudah diakses dan kredibel.

Principle of Least Astonishment

Software developer harus menulis kode yang dapat dipahami dengan mudah. Sebagai contoh ketika membuat kelas yang berhubungan dengan mobil maka sebaiknya diberi nama yang berhubungan dengan mobil tersebut. Jangan memberikan nama yang terlalu jauh dari konteks.

Jika prinsip-prinsip di atas dapat dijalankan dengan baik dan berkesinambungan, maka akan dapat mempermudah pengembangan perangkat lunak. Disamping itu prinsip-prinsip pemrograman di atas dapat menjadi acuan bagi semua anggota tim *software developer* agar memiliki pandangan terkait pengembangan perangkat lunak yang sama.

PENGUJIAN PERANGKAT LUNAK

Pendahuluan

Proses didalam pengembangan perangkat lunak tidak luput dari adanya proses pengujian yang merupakan salah satu hal terpenting agar dapat menjamin kualitas perangkat lunak dan sebagai verifikasi akhir dari proses desain dan pengkodean. Pengujian merupakan anomali yang terjadi didalam proses pengembangan perangkat lunak. Karena pada dasarnya didalam proses pengujian perangkat lunak merupakan hal-hal yang bersifat destruktif atau merusak daripada hal-hal yang bersifat konstruktif atau membangun. Pada definisi awal fase pengembangan, pengembangan bertujuan untuk membuat perangkat lunak dari konsep abstrak hingga implementasi. Pentingnya pengujian perangkat lunak dan pengaruhnya terhadap kualitas perangkat lunak tidak dapat dilebih-lebihkan karena melibatkan berbagai operasi produksi di mana potensi kesalahan manusia tinggi. Selain itu karena manusia tidak dapat mempunyai sifat sempurna, maka pengembangan perangkat lunak perlu melibatkan tindakan-tindakan yang bersifat kualitatif (Sulistyanto, 2017).

Pengujian perangkat lunak adalah proses yang dilakukan untuk menguji kualitas dan fungsionalitas perangkat lunak sebelum diluncurkan atau digunakan secara luas. Tujuan dari pengujian perangkat lunak adalah untuk menemukan bug, kesalahan, dan masalah potensial lainnya sehingga dapat diperbaiki sebelum perangkat lunak tersebut digunakan oleh pengguna.

Prinsip Pengujian

Didalam proses pengujian perangkat lunak, terdapat beberapa prinsip yang harus kita pegang agar proses pengujian dapat dilakukan secara efektif dan efisien. Berikut ini adalah beberapa prinsip pengujian yang harus kita ketahui:

1) Tidak perlu melakukan pemeriksaan secara menyeluruh

Pemeriksaan lengkap tidak memungkinkan untuk dilakukan didalam pengujian perangkat lunak. Oleh sebab itu diperlukan pemilihan jumlah pengujian yang membuatnya bisa optimal, dimana salah satunya bisa didasarkan dari penilaian resiko aplikasi (Putri, 2022). Untuk melakukan penilaian resiko dari sebuah aplikasi, bisa dilakukan dengan cara memilih fitur mana yang mempunyai kemungkinan tertinggi bisa menyebabkan sistem gagal. Pemilihan fitur ini bisa dilakukan dengan cara mencoba membuka beberapa program berbeda sekaligus, lalu temukan bugnya, kemudian lakukan pengujian menyeluruh pada klasifikasi bug tersebut.

2) Lakukan pengelompokan kesalahan

Melakukan pengelompokan kesalahan dapat dijadikan sebagai indikasi berada dilokasi mana kesalahan itu berada. Karena sejumlah kecil modul yang terdeteksi kesalahan dapat menjadi indikator sebagian besar kesalahan yang ada pada sistem. Hal ini merupakan penerapan dari prinsip Pareto yang mana 80% masalah itu bisa dideteksi dari 20% modul yang ada. Meskipun hal ini bisa diterapkan didalam proses pengujian, tetapi pendekatan ini juga terdapat kelemahan tersendiri. Karena jika pengujian seperti itu dilakukan pada kasus yang sama, maka tidak akan ditemukan bug baru.

3) Gunakan pedoman dari paradoks pestisida

Sama halnya dengan penggunaan pestisida untuk serangga yang dilakukan berulang kali, lama-kelamaan serangga akan mulai resistan. Begitu juga didalam pengujian perangkat lunak, jika metode yang sama dilakukan untuk melakukan pengujian berulang kali, maka metode tersebut tidak efektif untuk mendeteksi kesalahan baru. Sehingga dibutuhkan metode pengujian yang terus dilakukan pembaharuan.

4) Pengujian harus bertujuan untuk menemukan cacat

Itu sebabnya prinsip pengujian mengatakan: Pengujian adalah tentang adanya kesalahan, bukan tidak adanya kesalahan. Dengan kata lain, pengujian perangkat lunak mengurangi kemungkinan bug yang tidak terdeteksi dalam perangkat lunak, meskipun didalam pengujian tidak ditemukan bug sama sekali,

maka hal ini tidak membuktikan bahwa itu bebas bug. Sehingga sebagai penguji harus selalu berusaha untuk bisa menemukan cacat atau bug bagaimanapun caranya.

5) Pengujian perangkat lunak harus bersifat kontekstual

Pengujian perangkat lunak tidak boleh disamaratakan antara sistem yang satu dengan sistem yang lain. Pengujian harus peka terhadap konteks yang dihadapi, karena tidak semua program yang sedang diuji itu identik. Bergantung pada jenis aplikasinya, Anda dapat menggunakan berbagai pendekatan, metode, teknik, dan jenis pengujian. Misalnya, uji sistem POS setiap took ataupun metode yang lain.

6) Tim pengujian harus bersifat kemandirian

Prinsip ini maksudnya adalah bahwa tim pengujian itu harus bekerja secara independen atau mandiri dari tim pengembangan perangkat lunak. Sehingga hal ini dapat mencegah adanya bias atau konflik kepentingan yang dapat mengurangi efektivitas pengujian.

7) Pengujian harus sudah direncanakan sejak sistem mulai dirancang

Prinsip ini menyatakan bahwa pengujian perangkat lunak harus terlibat sejak awal dalam siklus pengembangan perangkat lunak. Ini berarti tim pengujian harus terlibat dalam perencanaan, desain, dan implementasi perangkat lunak untuk mengidentifikasi cacat dan risiko sejak dini.

Tahap Pengujian

Tahapan pengujian perangkat lunak mengacu pada serangkaian langkah yang dilakukan untuk memastikan bahwa perangkat lunak berfungsi sebagaimana yang diharapkan dan memenuhi persyaratan yang telah ditetapkan sebelum dirilis kepada pengguna (Sulistyanto, 2017). Berikut adalah tahapan umum dalam pengujian perangkat lunak:

1. Perencanaan Pengujian (Test Planning): Tahap ini melibatkan perumusan strategi pengujian dan perencanaan aktivitas pengujian. Tujuannya adalah untuk mengidentifikasi lingkup pengujian, mengatur sumber daya, menentukan jadwal, dan mengembangkan rencana pengujian secara keseluruhan.

2. Analisis Persyaratan (Requirement Analysis): Pada tahap ini, tim pengujian menganalisis persyaratan perangkat lunak untuk memahami fungsi, kinerja, dan batasan yang diharapkan. Hal ini membantu dalam menyusun rencana pengujian yang efektif.
3. Perancangan Pengujian (Test Design): Tahap ini melibatkan perancangan rinci dari skenario pengujian, pengembangan data pengujian, dan pemilihan teknik pengujian yang sesuai. Tujuannya adalah untuk mencakup semua aspek yang relevan dalam pengujian perangkat lunak.
4. Implementasi Pengujian (Test Implementation): Pada tahap ini, skenario pengujian yang telah dirancang diimplementasikan dengan menjalankan tes perangkat lunak sesuai dengan rencana pengujian yang telah disusun. Data pengujian dikumpulkan, dan hasil pengujian dicatat.
5. Eksekusi Pengujian (Test Execution): Ini adalah tahap di mana tes perangkat lunak sebenarnya dijalankan dengan menggunakan data pengujian yang telah dipersiapkan. Hasil pengujian dicatat dan dibandingkan dengan hasil yang diharapkan.
6. Pelaporan Hasil (Test Reporting): Setelah selesai menjalankan tes perangkat lunak, hasil pengujian dianalisis dan dilaporkan. Laporan pengujian biasanya berisi ringkasan hasil pengujian, temuan, masalah yang diidentifikasi, dan rekomendasi untuk perbaikan.
7. Pelacakan dan Pemantauan (Defect Tracking and Monitoring): Selama pengujian, masalah dan bug yang ditemukan akan didokumentasikan, dilacak, dan dipantau hingga penyelesaiannya. Pelacakan dan pemantauan ini penting untuk memastikan bahwa semua masalah telah diperbaiki sebelum perangkat lunak dirilis.
8. Pengujian Ulang (Retesting): Jika ada masalah atau bug yang ditemukan selama pengujian, mereka akan diperbaiki oleh tim pengembang. Setelah perbaikan dilakukan, pengujian ulang dilakukan untuk memastikan bahwa perubahan yang dilakukan tidak mempengaruhi fungsi yang ada dan tidak menyebabkan masalah baru.
9. Validasi dan Verifikasi (Validation and Verification): Tahap ini melibatkan evaluasi keseluruhan perangkat lunak untuk

memastikan bahwa perangkat lunak memenuhi persyaratan dan berfungsi dengan benar. Validasi dan verifikasi juga dapat melibatkan pengujian

Black Box Testing

Blackbox testing adalah teknik pengujian perangkat lunak yang berfokus pada pengujian fungsionalitas sistem atau aplikasi tanpa memiliki pengetahuan tentang struktur internal, detail implementasi, atau kodenya. Dalam pengujian kotak hitam, penguji memperlakukan sistem sebagai "kotak hitam" dan mengujinya berdasarkan masukan dan keluaran yang diharapkan, tanpa mempertimbangkan bagaimana sistem memproses atau menangani masukan tersebut (Rouf & Riyanto, 2012).

Tujuan dari Blackbox testing adalah untuk mengevaluasi perilaku eksternal sistem dan memastikannya memenuhi persyaratan yang ditentukan dan bekerja seperti yang diharapkan. Penguji biasanya bergantung pada dokumentasi persyaratan sistem, antarmuka pengguna, dan spesifikasi lainnya untuk merancang kasus uji.

Blackbox testing dapat diterapkan pada berbagai tingkat pengujian, termasuk pengujian unit, pengujian integrasi, pengujian sistem, dan pengujian penerimaan. Ini dapat melibatkan berbagai teknik seperti partisi kesetaraan, analisis nilai batas, pengujian tabel keputusan, dan pengujian use case.

Keuntungan dari Blackbox testing meliputi:

1. *Independence*: Penguji dapat melakukan pengujian kotak hitam tanpa mengetahui implementasi internal, memungkinkan evaluasi sistem yang tidak bias.
2. *Simplicity*: Penguji tidak perlu memiliki pengetahuan pemrograman atau teknis, karena mereka berfokus pada perilaku eksternal sistem.
3. *Focus on user perspective*: Pengujian kotak hitam membantu memastikan bahwa sistem memenuhi persyaratan pengguna dan berperilaku seperti yang diharapkan dari perspektif pengguna akhir.
4. *Effective in finding integration issues*: Pengujian kotak hitam berguna untuk mengidentifikasi masalah yang mungkin timbul akibat interaksi antar komponen atau sistem yang berbeda.

Namun, ada juga batasan untuk Blackbox testing:

1. *Limited coverage*: Karena penguji tidak memiliki pengetahuan tentang cara kerja internal sistem, mereka mungkin melewatkan skenario atau jalur tertentu yang mungkin penting untuk pengujian.
2. *Inefficient test cases*: Kasus uji mungkin tumpang tindih atau berlebihan, yang menyebabkan pengujian tidak efisien.
3. *Lack of detailed error identification*: Blackbox testing dapat mengidentifikasi masalah, tetapi mungkin tidak memberikan informasi terperinci tentang akar penyebab masalah tersebut.
4. *Incomplete testing*: Ada kemungkinan kehilangan kasus atau skenario tepi tertentu yang dapat diungkap melalui kotak putih atau teknik pengujian lainnya.

Untuk mencapai pengujian menyeluruh, kombinasi Blackbox testing dengan teknik lain seperti Whitebox testing (yang berfokus pada struktur internal dan kode) sering digunakan. Hal ini memungkinkan untuk penilaian kualitas sistem yang lebih komprehensif dan membantu mengungkap berbagai cacat yang lebih luas.

White Box Testing

Pengujian whitebox adalah teknik pengujian perangkat lunak yang berfokus pada pemeriksaan struktur internal dan detail implementasi dari suatu aplikasi perangkat lunak. Dalam pengujian whitebox, pengujian memiliki akses ke kode internal, arsitektur, dan desain dari sistem perangkat lunak yang sedang diuji (Mustaqbal, 2015).

Tujuan utama dari pengujian whitebox adalah untuk memastikan bahwa semua komponen, termasuk fungsi-fungsi, pernyataan-pernyataan, percabangan, dan jalur-jalur dalam kode, diuji secara menyeluruh. Pengujian menganalisis logika internal, aliran data, dan aliran kontrol dari aplikasi untuk merancang kasus uji dan memvalidasi perilaku perangkat lunak tersebut.

Teknik-teknik pengujian whitebox meliputi:

1. Coverage Pernyataan (Statement Coverage): Teknik ini bertujuan untuk menjalankan setiap pernyataan dalam kode sumber setidaknya satu kali selama pengujian. Hal ini membantu mengidentifikasi kode yang tidak terpakai atau tidak terjangkau.

2. Coverage Percabangan (Branch Coverage): Teknik ini fokus pada pengujian semua percabangan (keputusan) yang mungkin dalam kode. Setiap titik keputusan diuji baik untuk kondisi benar maupun salah.
3. Coverage Jalur (Path Coverage): Teknik ini melibatkan pengujian semua jalur yang mungkin melalui kode, termasuk loop dan kondisi bersarang. Hal ini memastikan bahwa setiap kombinasi pernyataan dan kondisi yang mungkin dieksekusi.
4. Coverage Kondisi (Condition Coverage): Teknik ini memeriksa kondisi Boolean dalam kode dan memastikan bahwa mereka dievaluasi baik menjadi benar maupun salah selama pengujian.
5. Coverage Loop (Loop Coverage): Teknik ini bertujuan untuk menguji kode di dalam loop, memastikan bahwa kode tersebut dieksekusi nol kali, satu kali, dan beberapa kali.
6. Data Flow Testing: Teknik ini memeriksa aliran data dalam program, mengidentifikasi variabel, definisi, dan penggunaannya. Kasus uji dirancang untuk menguji berbagai skenario aliran data.

Keuntungan dari pengujian whitebox meliputi:

1. Mendeteksi cacat secara dini karena pengujian dilakukan selama tahap pengembangan.
2. Memungkinkan cakupan kode yang mendalam dan mengidentifikasi kesalahan potensial dalam implementasi.
3. Membantu meningkatkan kualitas, keterbacaan, dan keterpeliharaan kode.
4. Membantu pengembang dalam debugging dan memahami perilaku internal sistem.

Namun, pengujian whitebox juga memiliki beberapa keterbatasan:

1. Memerlukan pengetahuan tentang kode internal dan detail implementasi, yang mungkin tidak tersedia bagi semua pengujian.
2. Mungkin fokus pada aspek teknis sistem, yang berpotensi mengabaikan masalah fungsionalitas tingkat tinggi atau pengalaman pengguna.
3. Dapat memakan waktu dan memerlukan upaya yang signifikan untuk mencapai cakupan yang lengkap.

4. Mungkin menantang untuk mensimulasikan semua masukan yang mungkin, yang mengarah ke potensi kesenjangan dalam cakupan pengujian.

Pengujian whitebox sering dilengkapi dengan teknik pengujian lain seperti pengujian blackbox, yang berfokus pada perilaku eksternal dan input/output sistem, dan pengujian kotak abu-abu, yang menggabungkan elemen pengujian kotak putih dan kotak hitam. Pemilihan teknik pengujian tergantung pada persyaratan dan tujuan proyek tertentu.

METODOLOGI PENGEMBANGAN PERANGKAT LUNAK

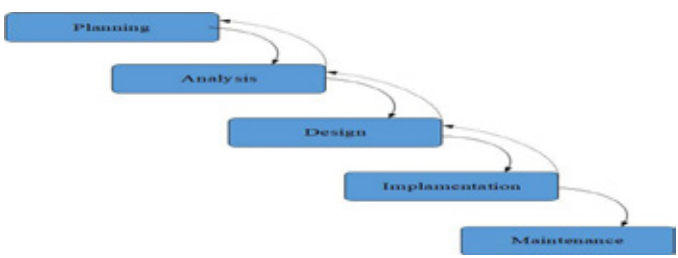
Pendahuluan

Pengembangan perangkat lunak telah mengalami perkembangan yang cukup pesat, pengembangan ini menuntut untuk dapat memberikan respon dan solusi terhadap kesempatan baru selera pasar, perubahan perekonomian, dan kebutuhan akan persaingan produk dan layanan. Perangkat lunak dapat menjangkau semua bidang yang memiliki proses bisnis seiring dengan perkembangan teknologi informasi yang cepat diperlukan sebuah metodologi pengembangan perangkat lunak.

Pengembangan metodologi pengembangan perangkat lunak yang semakin berkembang saat ini, ada beberapa metodologi yang dapat dipilih oleh pengembang (*Software Developers). Menurut Azhar Susanto SDLC (System Development Life Cycle)* adalah salah satu metode pengembangan sistem informasi yang popular pada saat sistem informasi pertama kali dibuat, dalam sdlc ada 4 tahapan untuk membangun sebuah perangkat lunak yaitu: planning, analysis, design, dan implementation .(Widiyana et al., 2021)

Ada juga beberapa macam - macam klasifikasi metodologi pengembangan perangkat lunak sebagai berikut

16.1. *Linear Sequential Model*



Gambar 1 Metodologi *Linear Sequential Model* (Prasetijo et al.,

2016)

Model ini sering disebut juga dengan "classic life cycle" atau "waterfall model". Model ini termasuk ke dalam model generic pada rekayasa perangkat lunak dan pertama kali diperkenalkan oleh Winston Royce sekitar tahun 1970. Linear sequential model adalah metode pengembangan perangkat lunak dengan pendekatan sekuensial dengan cakupan aktivitas yang tersusun secara terprogram dengan urutan kegiatan yang sistematis, terdiri dari lima tahap yaitu planing, analysis, desain, implementasi, Maintenance

Planing : Perencanaan Sistem adalah proses membuat sebuah Laporan Perencanaan Sistem yang menggunakan sumber sistem informasi yang berhubungan dan mendukung tujuan bisnis dan operasi organisasi.

Analysis : Penguraian dari suatu Sistem Informasi yang utuh ke dalam bagian-bagian komponennya dengan maksud untuk mengidentifikasikan dan mengevaluasi permasalahan, kesempatan, hambatan yang terjadi dan kebutuhan yang diharapkan sehingga dapat diusulkan perbaikannya.

Desain : Desain perangkat lunak adalah proses multi langkah yang fokus pada desain pembuatan program perangkat lunak termasuk struktur data, arsitektur perangkat lunak, representasi antar muka dan prosedur pengkodean.

Implementasi : Tahapan ini adalah tahapan pengimplementasian dari design yang telah dibuat sebelumnya menjadi sebuah bentuk nyata berupa aplikasi baik berbentuk desktop, website maupun mobile yang diikuti dengan pengujian unit. "Pengujian adalah suatu proses pelaksanaan suatu program dengan tujuan menemukan suatu kesalahan.

Maintenance : Pada tahap ini meliputi penyesuaian atau perubahan yang berkembang seiring dengan adaptasi perangkat lunak dengan kondisi atau situasi sebenarnya setelah disampaikan kepada konsumen atau pelanggan.

Keuntungan dan kerugian metodologi Linear Sequential Model Sebagai berikut :

Keuntungan :

1. Tahapan proses pengembangannya tetap (pasti), mudah diaplikasikan, dan prosesnya teratur
2. Cocok digunakan untuk produk software/program yang sudah jelas kebutuhan nya di awal, sehingga minim kesalahannya.
3. Software yang dikembangkan dengan metode ini biasanya menghasilkan kualitas yang baik
4. Dokumen pengembangan sistem sangat terorganisir, karena setiap fase harus terselesaikan dengan lengkap sebelum melangkah ke fase berikutnya

Kerugian :

1. Proyek yang sebenarnya jarang mengikuti alur sekuensial seperti diusulkan, sehingga perubahan yang terjadi dapat menyebabkan hasil yang sudah didapatkan tim pengembang harus diubah kembali/iterasi sering menyebabkan masalah baru
2. Terjadinya pembagian proyek menjadi tahap-tahap yang tidak fleksibel, karena komitmen harus dilakukan pada tahap awal proses.
3. Sulit untuk mengalami perubahan kebutuhan yang diinginkan oleh klien (pelanggan).
4. Perubahan ditengahtengah pengerjaan produk akan membuat bingung tim pengembang yang sedang membuat produk
5. Adanya waktu kosong (menganggur) bagi pengembang, karena harusmenunggu anggota tim proyek lainnya menuntaskan pekerjaannya (Deni Murdiani, 2022)

16.2. *Rapid Application Development* (RAD)

Putri dan Hendra Effendi (2018:131) mengatakan bahwa: RAD merupakan model proses perangkat lunak yang menekankan pada daur pengembangan hidup yang singkat. RAD merupakan versi adaptasi cepat dari *Linear Sequential Model*, dengan menggunakan pendekatan konstruksi komponen. RAD merupakan gabungan dari bermacam-macam teknik terstruktur dengan teknik prototyping dan teknik pengembangan joint application untuk mempercepat pengembangan sistem/aplikasi. (Puteri & Effendi, 2018)

Metode Rapid Application Development (RAD) tahapan-tahapannya terstruktur ,pengembangan perangkat lunak dapat dilakukan dalam waktu yang cepat dengan menekankan pada siklus

yang pendek, software yang dikembangkan dapat diketahui hasilnya tanpa menunggu waktu yang lama karena pengerjaannya di bagi ke dalam modul-modul dan alasan utama menggunakan metode pengembangan Rapid Application Development (RAD) adalah metode pengembangan ini akan bekerja dengan baik jika diterapkan pada aplikasi yang berskala kecil. (Jijon Raphita Sagala, 2021)

Metodologi RAD memiliki beberapa fase seperti gambar 2 berikiut ini:

Gambar 2 Metodologi RAD (Wahyuningrum & Januarita, 2014)

Rencana Kebutuhan (Requirement Planning) : User dan analyst melakukan pertemuan untuk mengidentifikasi tujuan dari sistem dan kebutuhan informasi untuk mencapai tujuan. Pada tahap ini merupakan hal terpenting yaitu adanya keterlibatan dari kedua belah pihak.

Proses Desain Sistem (Design System) : Pada tahap ini keaktifan user yang terlibat menentukan untuk mencapai tujuan karena pada proses ini melakukan proses desain dan melakukan perbaikan-perbaikan apabila masih terdapat ketidak sesuaian desain antara user dan analyst.

Implementasi (Implementation) : Tahapan ini adalah tahapan programmer yang mengembangkan desain suatu program yang telah disetujui oleh user dan analys. Sebelum diaplikasikan pada suatu organisasi terlebih dahulu dilakukan proses pengujian terhadap program tersebut apakah ada kesalahan atau tidak.

Metodologi RAD memilik keuntungan dan kerugian dalam mengembangakan perangkat lunak diantaranya (Deni Murdiani, 2022):

Keuntungan :

1. Lebih efektif dari Pengembangan Model waterfall/seque ntia llinear dalam menghasilkan sistem yang memenuhi kebutuhan langsung dari klien
2. Cocok untuk proyek yang memerlukan waktu yang singkat.

3. Model RAD mengikuti tahap pengembangan sistem seperti pada umumnya, tetapi mempunyai kemampuan untuk menggunakan kembali komponen yang ada sehingga pengembang tidak perlu membuatnya dari awal lagi sehingga waktu pengembangan menjadi lebih singkat dan efisien

Kerugian :

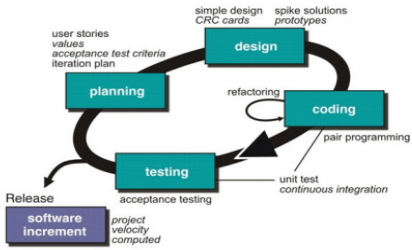
1. Model RAD menuntut pengembangan dan klien memiliki komitmen di dalamaktivitas rapid-fire yang diperlukan untuk melengkapi sebuah sistem, di dalam kerangka waktu yang sangat diperpendek Jika komitmen tersebut tidak ada, proyek RAD akan gagal.
2. Tidak semua aplikasi sesuai untuk RAD, bila system tidak dapat dimodulkan dengan teratur, pembangunan komponen penting pada RAD akan menjadi sangat bermasalah
3. RAD tidak cocok digunakan untuk sistem yang mempunyai resiko teknik yang tinggi.
4. Membutuhkan tenaga kerja yang banyak untuk menyelesaikan sebuah proyek dalam skala besar
5. Jika ada perubahan di tengahtengah pengerjaan maka harus membuat kontrak baru antara pengembang dan pelanggan

16.3 *Agile Development*

Model agile adalah pendekatan evolusioner yang menghasilkan produk perangkat lunak berkualitas tinggi dengan biaya yang efektif dan dalam waktu singkat (Balaji & Murugaiyan , 2012)

Motodologi agile merupakan motode yang cepat, beberapa metodologi agile yang berkembang saat sekarang ini antaranya Extreme Programming

Berdasakan gambar 3 diatas, maka kerangka kerja dalam metodologi pengembangan Extreme Programming adalah sebagai berikut :



Gambar 3 Extreme Programming (Suryantara,2017)

1) Planning (Metodologi Extreme Programming Perencanaan): Tahap perencanaan dimulai dengan pemahaman konteks bisnis dari aplikasi, mendefinisikan output, fitur yang ada pada aplikasi, fungsi dari aplikasi yang dibuat, serta alur pengembangan aplikasi. Dapat dikatakan bahwa tahapan ini menentukan fungsionalitas keseluruhan yang akan dikembangkandalam sistem.
2) Design (Perancangan): Pada tahap fokus pada design aplikasi secara sederhana, alat untuk mendesign pada tahap ini dapat menggunakan CRC (Class Responsibility Collaborator). CRC memetakan kelas-kelas yang akan dibangun dalam use case diagram, class diagram dan activity diagram.
3) Coding : Coding atau pengkodean merupakan penerjemahan dari perancangan dalam bahasapemrograman yang dikenali oleh komputer. Pada penelitian ini aplikasi dibagi menjadi dua, yaitu untuk front-end dan back-end. Pengkodean dengan menggunakan baahasa pemrograman PHP dengan compailer Sublime Text 3 dan database MySQL.
4) Testing (Pengujian): Sistem yang telah dibangun harus diuji terlebih dahulu agar dapat menemukan kesalahan dan dapat mempelajari dan menggunakan produk (program) untuk mencapai tujuannya dan seberapa kepuasaan pengguna terhadap penggunaan aplikasi serta kegunaanya.

Programming (XP) dapat diterapkan bila :

- Adanya perubahan-perubahan yang sangat cepat
- Memiliki resiko yang tinggi pada aplikasi yang dibuat
- Adanya tantangan baru pada pembuatan aplikasi

- Dalam tim pengembangan aplikasi dengan sedikit progremer 20-10 orang.
- Mampu mengotomatisasikan uji system/testing
- Keterlibatan/perang serta klain (pelanggan) secara lansung.

Metode XP juga memiliki keuntungan dan kerugian sebagai berikut:

Keuntungan :

1. Dengan metode ini pengembang dapat menjalin komunikasi/ interaksi yang baik dengan pelanggan (klain) pada pengembangan aplikasi.
2. Saling menghargai antara developer dan meningkatkan komunikasi.
3. Dapat menjadi pembelajaran bagi orang lain (klain)
4. Klain dapat umpan balik yang akurat mengena aplikasi yang dibuat.
5. Dapat membuat keputusan-keputusan yang bersifat teknikal.

Kerugian :

1. perubahan selalu diterima sehingga developer harus selalu siap dengan perubahan-perubahandari klien.
2. awal-awal pengembagan sistem tidak bisa membuat code (program) yang detail(prinsip simplicity)
3. XP hanya mengerjakan satu proyek dan dijalankan oleh satu tim, tidak efektif mengerjakan proyek yang berbeda secara bersamaan (paralel)
4. xp tidak dapat dikerjakan bila developer saling terpisah jauh dengan klien, karena mengakibatkan terhambatnya komunikasi antara developer dengan klien.

PEMELIHARAAN PERANGKAT LUNAK

A. Pendahuluan

Pada tahun 2005, seorang peneliti bernama Humphrey mengumpulkan data bertahun-tahun dari ribuan programmer dan menunjukkan bahwa mereka secara tidak sengaja memperkenalkan 100 bug per seribu baris kode. Humphrey juga menunjukkan bahwa perangkat lunak komersial biasanya berisi antara satu hingga sepuluh kesalahan per seribu baris kode. Masalah ini seperti bom waktu tersembunyi yang meledak ketika kondisi tertentu terpenuhi. Oleh karena itu, diperlukan penerapan kebijakan untuk mengidentifikasi dan memperbaiki kekurangan tersebut pada setiap tahap siklus pengembangan dan pemeliharaan.

Ketika proses perancangan ataupun pengembangan perangkat lunak telah selesai dilaksanakan atau dikerjakan dan masuk dalam tahap penggunaan, bukan berarti selesai juga seluruh kegiatan pada tahap rekayasa perangkat lunak, karena setelah itu perangkat lunak akan masuk ke dalam tahap selanjutnya yaitu tahap pengujian dan implementasi Pada tahap pengujian memastikan bahwa perangkat lunak yang dirancang sudah bebas dari bug dan error, tahap implementasi melakukan proses konversi sistem dan pelatihan user. Setelah keseluruh tahapan tersebut telah selesai dikerjakan maka sebagai pengembang sistem akan tetap melaksanakan proses daur hidup sistem secara berkelanjutan, dimana pada tahap ini akan dilakukan proses pemeliharaan sistem untuk memastikan sistem dapat berjalan sesuai dengan *software requirement* yang telah ditentukan pada tahap inisiasi perangkat lunak.

Pemeliharaan software merupakan suatu proses yang meliputi perubahan, modifikasi, dan peningkatan software agar tetap memenuhi kebutuhan pengguna. Tujuan dari proses ini adalah untuk meningkatkan performa keseluruhan software serta mendeteksi dan memperbaiki kesalahan sistem atau bug yang ditemukan oleh pengguna.

Pemanfaatan perangkat lunak yang digunakan dalam waktu yang relatif lama dapat menimbulkan masalah jika dalam pengembangannya kerap diabaikan atau tidak mendapatkan prioritas khusus dalam penanganan keberlanjutan penerapan sistem. Beberapa masalah yang timbul dikarenakan oleh penggunaan jangka panjang dari perangkat lunak meliputi:

Tabel 1. Masalah Penggunaan Jangka Panjang Perangkat Lunak

Deskripsi Masalah	Contoh
Kebutuhan perbaikan atau penyesuaian yang ditimbulkan karena adanya perkembangan dalam proses, fungsi, organisasi dan teknologi terbaru.	Perangkat lunak berbasis Android saat ini sedang dikembangkan
Terjadinya keruwetan, anggaran dan periode penjualan/ pemasaran	Perkembangan dalam bahasa pemprograman yang lebih efektif dan efisien
Terjadinya kelemahan dari perangkat lunak yang dibuat	Dibuatnya ketentuan baku pada saat pembuatan pengkodean

B. Kegiatan, Target Dan Manfaat Pemeliharaan

Secara umum kegiatan pada tahap pemeliharaan dapat dibagi menjadi dua pendekatan, yaitu :

Gambar 1. Kegiatan Pemeliharaan

1. **Pre-delivery**

Merupakan aktivitas pemeliharaan di tahap perencanaan yang lebih berfokus pada kegiatan operasi post-delivery, dukungan serta penentuan kebutuhan logistik dalam pemeliharaan sistem

2. **Post-delivery**

Merupakan aktivitas pemeliharaan yang dikerjakan setelah tahapan pre delivery selesai dikerjakan, atau pada saat sistem perangkat lunak telah dimanfaatkan oleh user.

Pada pelaksanaan kegiatan pemeliharaan sistem memiliki target yang harus dicapai agar perangkat lunak dapat berjalan dengan baik dan lancar, adapun targetnya tercermin sebagai berikut :

1. Mengoreksi kesalahan untuk perbaikan sistem
2. Beradaptasi dengan lingkungan baru
3. Menangkal terjadinya kesalahan
4. Menambahkan fitur untuk optimalisasi fungsional sistem

Kegiatan pemeliharaan perangkat lunak sangat perlu dilakukan karena pada tahap pemeliharaan dapat memberikan banyak manfaat baik untuk organisasi maupun individual, adapun beberapa manfaat pemeliharaan sebagai berikut ;

1. Peningkatan Efektivitas Perangkat Lunak

Selama dalam proses pemeliharaan, user dapat berbagi pengalaman dengan pengembang perangkat lunak, termasuk fitur mana yang paling sering digunakan, fitur mana yang jarang digunakan, dan fitur mana yang tidak pernah digunakan. Dengan cara ini, pengembang dapat mengurangi fitur yang tidak terpakai dan membuat sistem operasi perangkat lunak lebih ramping dan tampak lebih bersih, sehingga user dapat lebih fokus untuk menyelesaikan pekerjaan dengan fitur yang masih ada di dalam perangkat lunak.

2. Mempertahankan Optimalisasi Kinerja Perangkat Lunak

Secara umum, perangkat lunak dirancang untuk bekerja dengan baik pada perangkat sistem yang bersesuaian dengan versi sistem dari perangkat tersebut. Jika versi perangkat sistem mengalami perubahan maka kinerja perangkat lunak dapat saja tidak berjalan secara optimal. Pada saat user menggunakan perangkat lunak pada perangkat sistem pada saat pertama kali sistem di jalankan, maka semua fungsinya dapat bekerja dengan sempurna. Namun, setelah dilakukan pembaharuan atau update terhadap perangkat sistem yang diakibatkan adanya penyesuaian terhadap perkembangan teknologi baik perangkat keras maupun perangkat lunak mengakibatkan terjadinya masalah terhadap

perangkat lunak. Tentu hal ini mengakibatkan user menjadi kurang nyaman dalam menggunakan perangkat lunak tersebut. Untuk mengatasi masalah ini, maka pengembang sistem melakukan proses pemeliharaan sistem untuk memastikan perangkat lunak dapat berjalan secara optimal.

3. Perbaikan Terhadap Kesalahan Sistem

Pemanfaatan atau penggunaan perangkat lunak yang berkesinambungan dapat memunculkan kesalahan pada perangkat, dimana apabila hal in terjadi maka akan akan sangat mengganggu keseluruhan proses pekerjaan. Untuk mengatasinya, maka diperlukan proses pemeliharaan sistem, dimana selama pemeliharaan, pengembang sistem akan terlebih dahulu mencari kode kesalahan berdasarkan laporan masalah yang telah disampaikan oleh user. Setelah itu proses akan terus mencari solusi dan mulai memperbaikinya.

4. Peningkatan Keamanan Data

Pemanfaatan komputer yang meluas di segala bidang dalam era revolusi industri 4.0 menyebabkan munculnya ragam kejahatan yang terjadi pada sistem komputer. Hal ini juga menjadi prioritas pengembang sistem untuk meningkatkan keamanan perangkat lunak, hal Ini dilakukan dengan memperbarui metode keamanan perangkat lunak dengan metode terbaru yang dianggap paling aman oleh pengembang sistem pada setiap pelaksanaan kegiatan pemeliharaan dilakukan, hal ini bertujuannya untuk memastikan bahwa keamanan data terjaga dengan baik.

C. Kategorisasi Pemeliharaan

Pemeliharaan sistem dapat dikategorikan ke dalam empat jenis sebagai berikut :

Gambar 2. Kategorisasi Pemeliharaan

1. **Kategori korektif**: Tipe pemeliharaan ini muncul disebabkan ditemukan adanya kesalahan pada sistem.

   Tipe ini merupakan tipe maintenance umum yang sering digunakan oleh pengembang sistem. Seperti namanya, proses pemeliharaan korektif dilakukan hanya ketika sistem mengalami masalah seperti munculnya bug atau error pada perangkat lunak. Ketika masalah ini muncul, pengembang sistem harus segera memperbaikinya agar tidak menyebar ke fungsi lain. User juga dapat membantu peran pengembang sistem, dimana user memiliki tugas untuk menemukan kesalahan pada sistem dan mengirim laporan kesalahan ke pengembang. Namun terkadang pengembang juga dapat menemukan bug sistem sebelum user menemukannya. Untuk pemeliharaan korektif yang tidak terjadwal untuk menjaga sistem tetap berjalan, masuk dalam kategori pemeliharaan emergensi.

   Contoh:

   § Terdapat bugs yang ditemukan pada perangkat lunak
   § Kesalahan dalam melakukan interpretasi dan representasi proses bisnis kedalam logika program sehingga menyebabkan munculnya kesalahan yang terjadi baik pada prosedur/ maupun fungsi dari perangkat lunak.

2. **Kategori adaptif**: tipe pemeliharaan ini merupakan penyesuaian terhadap adanya perubahan yang terjadi pada lingkungan sistem.

   Pemeliharaan ini dilakukan oleh pengembang untuk memastikan kompatibilitas perangkat lunak dengan sistem operasinya. Hal ini karena kebutuhan teknologi dan bisnis terus berkembang. Jika perawatan seperti ini tidak dilakukan, perangkat lunak yang digunakan akan menghadapi berbagai masalah.

   Ada tiga kondisi dalam mengoperasikan layanan ini.

   Pertama, ketika versi sistem operasi perangkat lunak berubah.

   Kedua, jika ingin menggunakan perangkat lunak tersebut di perangkat kerja baru.

Ketiga, saat pengembang sistem sampai meyakini berdasarkan analisis mendalam bahwa sistem dapat membahayakan data di masa mendatang.

Contoh:

§ Berubahnya aturan atau regulasi yang diacu, misalnya:
  - Adanya penghapusan/ pemutihan terhadap pajak terhutang
  - Adanya aturan/ kebijakan baru oleh pimpinan menyebabkan terjadinya perubahan terhadap regulasi yang lama

§ Berubahnya versi web service atau API

§ Berubahnya format template Surat, dsb

3. **Kategori preventif** : tipe pemeliharaan ini merupakan tipe pencegahan terjadinya keberlanjutan operasional sistem

   Tipe ini merupakan tindakan pencegahan sebelum terjadinya masalah yang muncul pada sistem, pada kategori ini pemeliharaan terjadi secara berkala ketika perangkat lunak berfungsi dengan baik maupun apabila terdapat permasalahan pada perangkat lunak. Oleh karena itu, dibuatlah rencana pemeliharaan secara terjadwal agar prosesnya dalam berjalan secara sistematis.

   Dibandingkan dengan perawatan korektif, dapat dikatakan bahwa masalah yang diselesaikan dalam perawatan preventif kurang penting atau disebut cacat laten. Namun, jika masalah kecil ini diabaikan begitu saja, sangat mungkin masalah besar atau kegagalan kinerja dapat tiba-tiba terjadi. Tindakan preventif berupaya meningkatkan unjuk kerja perangkat lunak untuk mengantisipasi atau mencegah masalah di masa mendatang.

   Contoh:

   § Melakukan antisipasi meningkatnya jumlah user yang mengakses sistem atau basis data dapat menyebabkan sistem mengalami failure ataupun secara masif dapat menyebabkan sistem menjadi down

§ Mencegah terjadinya terhadap request terhadap layanan dengan waktu proses eksekusi yang lama, misalkan dilakukan proses join terhadap beberapa tabel dengan ukuran file yang besar

4. **Kategori perfektif** : tipe pemeliharaan ini menangkap terjadinya perubahan kebutuhan yang diajukan oleh user sebagai antisipasi untuk meningkatkan keunggulan kompetitif.

   Jenis pemeliharaan ini berfokus pada peningkatan fungsionalitas perangkat lunak yang digunakan. Perbaikan lengkap dapat dilakukan dengan memperbaiki, menghapus, dan menambahkan fitur baru sesuai keinginan user. Keinginan ini dapat berkembang seiring waktu dan beban kerja yang semakin meningkat. Tujuan dari pemeliharaan ini adalah untuk meningkatkan penerimaan user terhadap perangkat lunak dan dengan demikian membuat pekerjaan menjadi lebih mudah.
   Contoh:
   § Kemudahan penggunaan perangkat lunak untuk dapat digunakan dalam beberapa proses bisnis yang berbeda seperti pemanfaatan e-money untuk semua transaksi
   § Pemanfaatan User interface berbasis grafis yang lebih interaktif dan memberikan user experience yang menyenangkan
   § Peningkatan fitur perangkat lunak dengan menambah fasilitas Help desk atau Tutorial

   Pemeliharaan perangkat lunak merupakan kegiatan yang membutuhkan biaya tinggi karena beberapa faktor antara lain stabilitas sistem, tanggung jawab kontraktual, keahlian staf, serta usia dan struktur program.

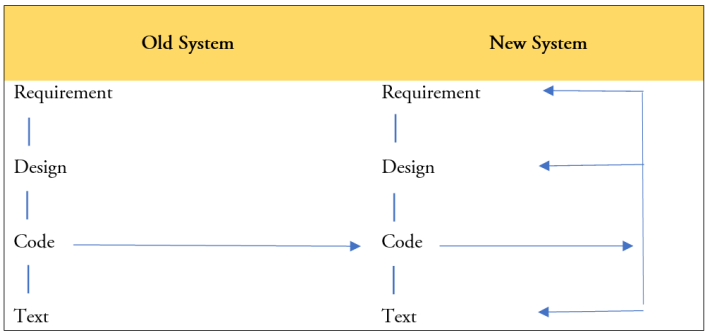
D. Model Pemeliharaan Perangkat Lunak

Tiga model pemeliharaan perangkat lunak yaitu :

1. Quick-Fix Model : dengan mengubah kode program terlebih dahulu, kemudian membuat perubahan yang diperlukan pada dokumentasi program.

   Model pendekatan jenis ini dilakukan dengan membuat perubahan terhadap kode program sebagai langkah awal, setelah itu akan masuk ke tahapan melakukan perubahan

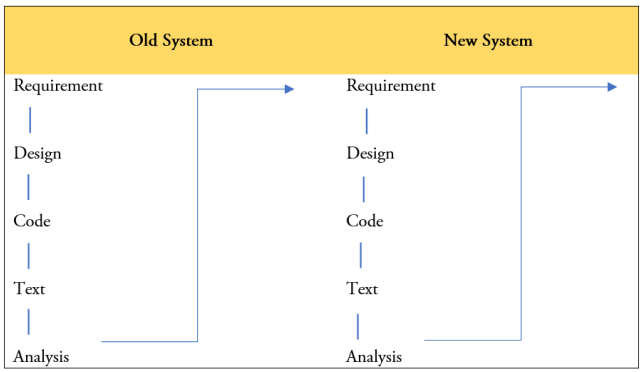
terhadap dokumentasi program. Seharusnya ketika terjadi perubahan pada kode program otomatis proses dokumentasi terhadap sistem pasti mengikuti perubahan yang terjadi baik terhadap persyaratan, analisis, desain, pengujian, dan topik terkait perangkat lunak yang relevan juga harus dikonversi sesuai dengan perubahan yang terjadi pada kode program. Namun kenyataan di lapangan menunjukkan bahwa perubahan kode program kerap kali tidak selaras dengan perubahan yang ada pada tahap dokumentasi, disebabkan pengaruh dari tekanan waktu dan biaya yang dihadapi oleh tim pemeliharaan.



Gambar 3. Quick Fix Model

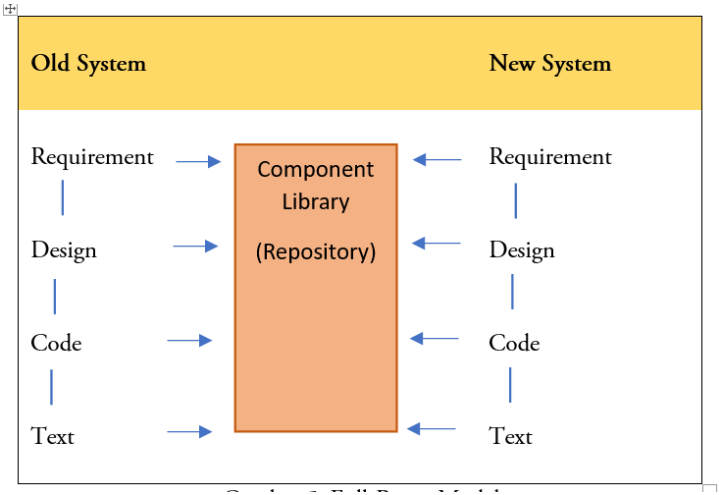
2. Iterative Enhancement Model : analisis terhadap sistem yang digunakan pada sistem yang memiliki umur yang panjang dan berevolusi seiring dengan waktu. Model ini mendukung evolusi sistem untuk memudahkan modifikasi ke depannya.

Model ini menyatakan bahwa persyaratan sistem tidak dapat ditangkap dan dipahami pada tahap awal. Oleh karena itu, sistem dibuat dengan meningkatkan persyaratan dari sistem yang dibangun sebelumnya berdasarkan umpan balik pengguna. Keuntungan dari model ini adalah dokumentasi sistem terus berubah seiring dengan perubahan kode program.



Gambar 4. Iterative Enhancement Model

3. Full-Reuse Model : dimulai dengan analisis kebutuhan dan perancangan dari sistem yang baru dan menggunakan ulang kebutuhan, rancangan, kode dan pengujian dari sistem versi sebelumnya yang telah ada



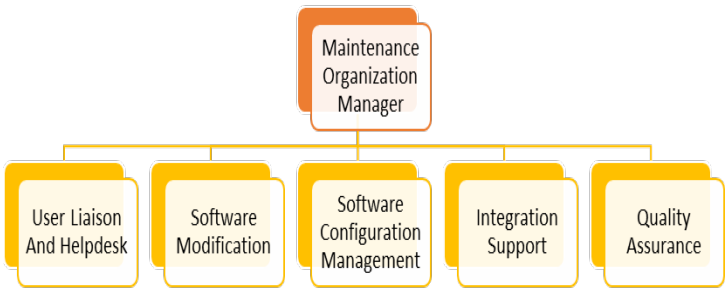
Gambar 5. Full-Reuse Model

E. Struktur Organisasi Pemeliharaan

Tiga tipe struktur organisasi pemeliharaan perangkat lunak yaitu :

1. Model fungsional

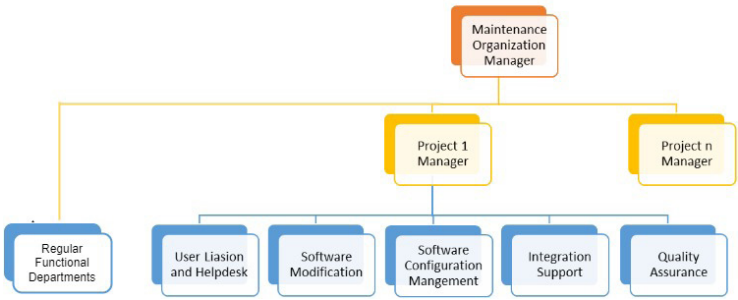
   Model fungsional : Organisasi dibagi menjadi unit-unit fungsional yang berbeda-beda, seperti modifikasi perangkat lunak, pengujian, dokumentasi dan penjaminan kualitas.



Gambar 6. Model Organisasi Fungsional

2. Model proyek

Model Proyek : Pada tipe ini, seorang manajer diberikan tanggung jawab dan hak penuh untuk mengatur orang, semua sumber daya yang dibutuhkan untuk pengerjaan proyek dipisahkan dari struktur fungsional regulernya.



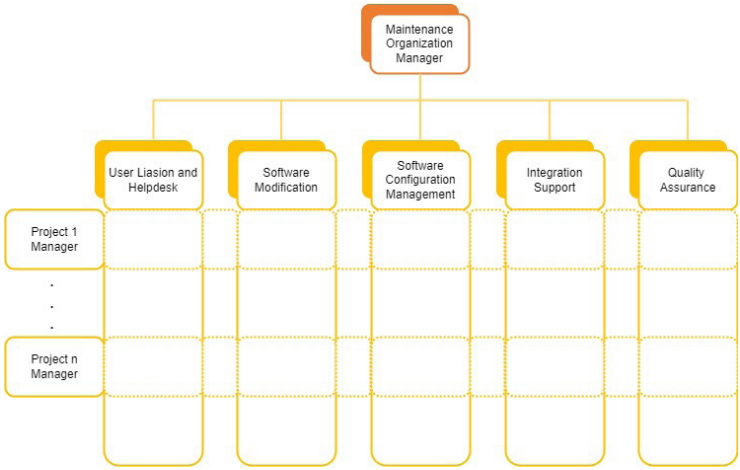
Gambar 7. Model Organisasi Proyek

3. Model matriks

Model Proyek : Pada tipe ini, seorang manajer diberikan tanggung jawab dan hak penuh untuk mengatur orang, semua sumber daya yang dibutuhkan untuk pengerjaan proyek dipisahkan dari struktur fungsional

Model Matriks merupakan gabungan dari model fungsional dan model proyek dengan tujuan untuk memaksimalkan kelebihan dan meminimalkan kekurangan dari kedua model struktur organisasi pemeliharaan tersebut.

Kelebihan dari tipe ini adalah adanya keseimbangan antara sasaran departemen fungsional dengan sasaran proyek itu sendiri



Gambar 8. Model Organisasi Matrik

F. Proses Pemeliharaan Versi IEEE-1219 dan Versi ISO-1220

IEEE adalah organisasi profesional nirlaba dari para profesional teknik yang mempromosikan pengembangan standar dan bertindak sebagai akselerator teknologi baru di semua bidang industri dan teknologi.

International Organization for Standardization atau lebih dikenal dengan ISO merupakan salah satu standar internasional dalam sistem manajemen untuk mengukur kualitas suatu organisasi, yang berperan penting dalam mengukur kredibilitas suatu perusahaan yang ingin bersaing di pasar global.

Kedua organisasi tersebut memiliki proses pemeliharaan dengan menggunakan beberapa pendekatan yang disesuaikan dengan kebijakan organisasi, seperti berikut ini :

Proses Pemeliharaan Versi IEEE-1219 dibagi menjadi tujuh fase, yakni:

1. Identifikasi, klasifikasi, dan penentuan prioritas modifikasi
2. Analisis
3. Perancangan
4. Implementasi
5. Regression/system testing
6. Acceptance testing
7. Delivery

Proses Pemeliharaan Versi ISO-1220 membagi aktivitas pemeliharaan ini menjadi beberapa aktivitas yakni:

1. Implementasi Proses
2. Analisis Masalah dan Modifikasi
3.Implementasi Modifikasi
4. Penerimaan/Pengkajian Pemeliharaan
5. Migrasi
6. Pemberhentian Operasi Perangkat Lunak

G. Evaluasi

Berikut merupakan evaluasi dari pembahasan mengenai pemeliharaan perangkat lunak, evaluasi ini untuk melihat ketercapaian materi yang telah diberikan.

1. Jelaskan menurut pendapat anda, mengapa proses pemeliharaan dibutuhkan pada kegiatan pembuatan perangkat lunak
2. Jelaskan manfaat dari pemeliharaan yang anda ketahui
3. Jelaksan kategorisasi pemeliharaaan perangkat lunak yang anda ketahui dan berikan contohnya
4. Jelaskan model pemeliharaan perangkat lunak yang anda ketahui
5. Jelaskan struktur organisasi pemeliharaan dan berikan contohnya

Profil Penulis

Indah Purnamasari, S.T., M.Kom. Penulis merupakan dosen di Fakultas Teknologi Informasi Universitas Nusa Mandiri dan dosen luar biasa di Universitas Bina Sarana Informatika. Penulis menyelesaikan program Sarjana di Sekolah Tinggi Teknologi Telkom Bandung dan Master Ilmu Komputer di Universitas Nusa Mandiri Jakarta. Melengkapi keahlian di bidang Ilmu Komputer dengan memperoleh sertifikasi CPDSA (Certified Python for Data Science Associate), Web Programming dan Network Administrator.

Yoga Sahria adalah seorang penulis buku dan profesional di bidang pengembangan perangkat lunak. Lahir pada 11 Juli 1995 di Grobogan,Kradenan, Jawa Tengah, Indonesia. Yoga menunjukkan minat yang besar terhadap dunia teknologi sejak usia muda. Yoga menyelesaikan pendidikannya dengan gelar Sarjana Teknik Informatika dari Universitas Teknologi Yogyakarta Lulusan *Cumlaude* Terbaik pada tahun 2020. Kemudian S2 Jurusan Informatika di Universitas Islam Indonesia lulusan terbaik termuda *Cumlaude* pada tahun 2021. Kemudian Lulus Program Profesi Insinyur di Universitas Muhammadiyah Yogyakarta lulus tahun 2023 . Setelah lulus S2 beliau langsung Melanjutkan S3 *on Going* jurusan Pendidikan Teknologi & Kejuruan. Selama masa kuliah, dia aktif dalam penelitian-penelitian dan berbagai organisasi dan mengikuti program-program pengembangan perangkat lunak yang memberikan pemahaman mendalam tentang industri ini. Profesi beliau menjadi tenaga pengajar professional DOSEN tetap di Universitas AMIKOM Yogyakarta. Selain itu sebagai tenaga pengajar di Universitas Teknologi Yogyakarta dan Universitas Terbuka. Bidang Minat Penelitian Augmented Reality, Rekayasa Perangkat Lunak, Text Mining, Health Informatics, Pendidikan Teknologi.

Email; yogasahria@amikom.ac.id

Dartono adalah seorang praktisi IT. Saat ini, ia menjabat sebagai Kepala Seksi Perancangan Sistem Informasi di Direktorat Jenderal Bea dan Cukai. Selain itu, dia juga aktif sebagai dosen di ITB Swadharma dan beberapa kampus swasta di Jakarta, di mana ia berbagi pengetahuan dan pengalaman praktisnya kepada generasi muda yang tertarik dengan bidang teknologi informasi. Dartono telah mengabdikan dirinya untuk mengembangkan dan menyempurnakan sistem informasi di Direktorat Jenderal Bea dan Cukai. Mulai dari Programmer, System Analyst, Business Analyst hingga menjadi Project Manager. Selain itu dia juga menjadi Project Analyst untuk Program National Logistics Ecosystem. Dedikasinya terhadap pekerjaan dan keahliannya dalam bidang sistem informasi telah membuatnya mendapat penghargaan sebagai Pegawai Berprestasi Direktorat Jenderal Bea dan Cukai tahun 2019 dan Penghargaan World Customs Organization Certificate of Merit tahun 2022.

Ida Afriliana, ST, M.Kom. Saya lahir 24 April 1977 di Tegal. Saya anak bungsu dari 7 bersaudara dan ibu dengan 4 anak . Saya Mengajar di Prodi D3 Teknik Komputer, Politeknik Harapan Bersama Tegal. Hobi saya menyanyi dan menulis. Dari kecil saya sudah terbiasa menulis diary,. Tahun 2012 saya pernah menulis buku tentang orang tua saya, tetapi buku ini belum pernah diterbitkan oleh penerbit manapun. Hanya saya cetakversi hard cover saja. Saya selalu menulis bahan ajar dari setiap mata kuliah yang saya ampu, tetapi bahan ajar yang telah ber-ISBN ada 2 yakni Basis Data (th.2017) dan Bahan Ajar Metodologi Riset (2019). Buku lain yang sudah pernah diterbitkan adalah book chapter dengan judul Jejak Digital Motivator Andal, Book chapter Mendadak PJJ di masa pandemi, Untaian Kasih untuk Bunda, book chapter Tetap Kreatif di Tengah Pandemi Covid 19, Kreatif Menulis menuju Eksistensi di Dunia Literasi, Perjalanan menuju Dunia Pendidikan, Big data Pemanfaatan Anfis untuk Klasifikasi Engagement tenaga Pendidik,dan Implementasi Mikrokontroler dan Sensor MQ2 Pada Sistem Proteksi Kebocoran Gas LPG Rumah Tangga.

Penulis adalah dosen di Institut Teknologi Telkom Purwokerto (ITTP). Dalam melaksanakan Tri Dharma Perguruan Tinggi, baik Pengajaran, Penelitian, maupun Pengabdian Masyarakat penulis tidak terlepas dari bahasan Rekayasa Perangkat Lunak. Bahkan salah satu mata kuliah yang diajar adalah Rekayasa Perangkat Lunak. Bentuk Implementasi Perangkat Lunak yang pernah penulis bersamai yaitu salah satunya di RSUD Wates Yogyakarta, PT. Sari Husada (SGM) Yogyakarta, dan Universitas Muhammadiyah Bengkulu. Implementasi Perangkat Lunak adalah salah satu materi utama dari sebuah Rekayasa Perangkat Lunak. Untuk Edisi perdana ini, penulis menyajikan khusus bersama para penulis lainnya dari berbagai institusi dalam wadah kolaborasi. Semoga bermanfaat bagi kita semua dan menambah wawasan bagi pembaca. Terimakasih.

Petrus Christo, M.Kom. Lahir di Jakarta 05 Mei 1974, menyeleaikan pendidikan formal terakhirr dalam bidang ilmu Teknik Informatika dan Teknologi Informasi, Saat ini selain selain sudah menggeluti bidang pengajaran lebih dari 20 tahun dalam bidang llmu Komputer , penulis saat ini bekerja sebagai praktisi Staff Informasi Teknologi dan Jaringan Komputer

Anang Anggono Lutfi merupakan salah satu akademisi di bidang pendidikan, khususnya teknik informatika. Saat ini beliau menjadi seorang pengajar di salah satu instituti pendidikan di Jawa Tengah. Disamping itu beliau juga aktif sebagai tenaga pengajar lepas di salah satu institusi pendidikan di Jakarta. Bidang yang menjadi perhatian beliau yaitu pemrograman web dan *natural language processing*.

Mohammad Robihul Mufid atau biasa dipanggil dengan Mufid. Lahir di Gresik, 22 Agustus 1994. Dengan berlatar belakang Pendidikan mulai dari D4 Teknik Informatika dan S2 Teknik Informatika dan Komputer di Politeknik Elektronika Negeri Surabaya.

Merupakan salah satu penulis dari buku ini yang berfokus untuk memberikan pemahaman tentang bagaimana proses dari rekayasa perangkat lunak. Khususnya bagaimana melakukan suatu pengujian terhadap perangkat lunak. Apa saja yang perlu diperhatikan dan bagaimana metode yang tepat untuk melakukan pengujian terhadap perangkat lunak.

Arif Rizki Marsa, S.Kom., M.Kom. Penulis menyelesaikan Pendidikan Strata 1 Program Studi sistem Informasi Konsentrasi sistem Analis dan Database pada Universitas Putra Indonesia "YPTK" Padang tahun 2010, menyelesaikan Pendidikan Strata 2 Program Studi Magister Teknik Informatika Konsentrasi Sistem Informasi pada Universitas Universitas Putra Indonesia "YPTK" Padang tahun 2013. Penulis adalah dosen tetap Program Studi S1 Informatika Sekolah Tinggi Teknologi Payakumbuh di payakumbuh. Pengajaran dan penelitian penulis berfokus pada bidang rekayasa perangkat lunak.yang membahas seputar pemrograman komputer.

Yuni Widiastiwi, lahir di Kota Jakarta, Pendidikan tingkat dasar hingga menengah dan atas ditempuh di daerah Jakarta Selatan. Melanjutkan pendidikan S1 Manajemen Informatika di Universitas Pembangunan Nasional Veteran Jakarta (UPNVJ), Melanjutkan Pendidikan S2 Ilmu Komputer di Institut Pertanian Bogor (IPB) Saat ini sedang menginisiasi situs youtube widiastiwi channel sebagai media sharing dan dokumentasi beberapa kegiatan yang pernah dikerjakan.

Zatin Niqotaini, S.Tr.Kom., M.Kom, lahir pada 10 Februari 1993 di Brebes, Provinsi Jawa Tengah. Penulis menyelesaikan pendidikan D4 Teknik Informatika di Politeknik Pos Indonesia Bandung dan S2 Sistem Informasi di STMIK Likmi Bandung. Penulis merupakan Dosen Pegawai Negeri Sipil di Fakultas Ilmu Komputer Universitas Pembangunan Nasional Veteran Jakarta dan homebasenya di Program Studi S1 Sistem Informasi. Bidang peminatan penulis berkaitan dengan Sistem Informasi Bisnis.

Andi Wijaya, S. Kom., M.Kom, Lahir pada 3 Mei 1987 di bondowoso, provinsi jawa timur. Penulis menyelesaikan pendidikan S1 di Sekolah tinggi teknologi Nurul Jadid dan S2 Teknik Informatika di Universitas Dian Nuswantoro Semarang. Penulis merupakan dosen Prodi Rekayasa Prangkat Lunak Fakutas Teknik Universitas Nurul Jadid. Bidan Keilmuan Teknologi Informasi dan Komunikasi, Teknik Sistem Informasi, Pengembangan Perangkat Lunak.

Dian Nursantika, S.Kom., M.Cs. Merupakan Dosen di Program Studi Sistem Informasi – Universitas Terbuka. Telah menyelesaikan pendidikan pasca sarjana di Ilmu Komputer Universitas Gadjah Mada. Bidang Minat keilmuan yaitu Computer Vision, Artificial Intelligence, Machine Learning.

Cholid Fauzi, S.T.,M.T. Penulis adalah dosen di Politeknik Negeri Bandung (Polban). Dalam melaksanakan Tri Dharma Perguruan Tinggi, baik Pengajaran, Penelitian, maupun Pengabdian Masyarakat penulis tidak terlepas dari bahasan Rekayasa Perangkat Lunak. Bahkan salah satu mata kuliah yang diajar adalah Manajemen Proyek Perangkat Lunak. Penulis juga mendapatkan sertifikasi di bidang project management Project Management Institute yaitu Certified Associate Project Management. Semoga bermanfaat bagi kita semua dan menambah wawasan bagi pembaca. Terimakasih.

Daftar Pustaka

Findawati, Y. (2020) Buku Ajar Rekayasa Perangkat Lunak, Buku Ajar Rekayasa Perangkat Lunak. doi: 10.21070/2018/978-602-5914-09-6.

Hooker, D. (2014) 'Seven Principles Of Software Development', in. Available at: http://wiki.c2.com/?SevenPrinciplesOfSoftwareDevelopment.

Pressman, R. S. (2010) Software Quality Engineering: A Practitioner's Approach, Software Quality Engineering: A Practitioner's Approach. America: The McGraw-Hill. doi: 10.1002/9781118830208.

Sommerville, I. (2016) Software Engineering. 10th edn. America: Pearson. Available at: https://www.academia.edu/50882590/Sommerville_Software_Engineering_10ed.

Software Engineering, Ian Sommerville

Rosa A.S dan M. Shalahuddin, Rekayasa perangkat lunak : Terstruktur dan berorientasi objek, Informatika Bandung Indonesia, 2018

Gat (2015) 'Perancangan Basis Data Perputakaan Sekolah dengan Menerapkan Model Data Relasional', Citec Journal, 2(4), pp. 305–3015.

Ida Afriliana (2017) 'Basis Data',PHB Press,Cetakan Pertama.

Mukhlis, I. R. and Santoso, R. (2023) 'Perancangan Basis Data Perpustakaan Universitas Menggunakan MySQL dengan Physical Data Model dan Entity Relationship Diagram', 4(2), pp. 2–8. doi: 10.37802/joti.v4i2.330.

Roger S. Pressman, Software Engineering: A Practitioner's Approach Fifth Edition, McGraw-Hill, 2004.

Ian Sommerville, Software Engineering 7th Edition, Addison-Wesley, 2004.

Romi Satria Wahono, Meluruskan Salah Kaprah Rekayasa Perangkat Lunak, 2006.

Friska Yolandha, Rajawali Nusindo Targetkan Pendapatan Rp 5,2 Triliun pada 2023, Republika, 2023.

Even Driven Program Available at https://www.keinsinyuran.com/kamus/event-driven-program/

(Accessed: June 15, 2023).

Functional vs Imperative Programming Available at: https://learn.microsoft.com/id-id/dotnet/standard/linq/functional-vs-imperative-programming (Accessed: June 15, 2023).

Kuhn, A.M. (2005) Code Complete, Technometrics. Available at: https://doi.org/10.1198/tech.2005.s332.

N, R. (2021) 'Secure Code Design Pada Pemrograman Berorientasi Objek Dengan Penanganan Pengecualian', Cyber Security dan Forensik Digital, 4(1), pp. 13–17. Available at: https://doi.org/10.14421/csecurity.2021.4.1.2341.

Oktarina, Y. and Mulyani, S. (2022) 'Pengembangan Media Pembelajaran Berbasis Android pada Mata Kuliah Keperawatan Medikal Bedah III', Jurnal Ilmiah Universitas Batanghari Jambi, 22(1), p. 651. Available at: https://doi.org/10.33087/jiubj.v22i1.1976.

Pendidikan, D. and Biasa, L. (2023) 'Meningkatkan Kemampuan Terjemahan Al-Quran Bagi Anak Berkesulitan Belajar Melalui Pola Belajar Visual Auditori Kinestetik (VAK)', (1), pp. 124–129. Available at: https://doi.org/10.24036/pedagogi.v23i1.1534.

Penerapan Media Audio Visual dalam Meningkatkan Prestasi Belajar Siswa terhadap Pembelajaran IPS di Kelas V MIN Montasik Aceh Besar, https://repository.ar-raniry.ac.id/id/eprint/1506/ (Accessed: June 15, 2023).

Suhartini (2021) 'Peningkatan Kompetensi Menulis Deskripsi Teks Eksplanasi Melalui Pendekatan Kontekstual Berbasis Audio Visual Siswa Kelas XI IPA 5 pada Semester 2 Tahun Pelajaran 2018 / 2019 SMA Negeri 1 Tunjungan Kabupaten Blora', Educatif Journal of Education Research, 3(3), pp. 126–135. Available at: https://doi.org/10.36654/educatif.v3i3.137.

Yusnitasari, T. and Umniati, N. (no date) 'ANALYZING PROGRAMMING LANGUAGE GENTEE WITH C LANGUAGE (Case Study APPLICATION PROGRAM N FAKTORIAL)', p. 8.

Putri Mentari Endraswari, S., & Kom, M. (2022). Buku ajar: rekayasa perangkat lunak.

Sulistyanto, H. (2017). Urgensi Pengujian pada Kemajemukan Perangkat Lunak dalam Multi Perspektif. Komuniti: Jurnal Komunikasi dan Teknologi Informasi, 6(1), 65-74.

Rouf, A., & Riyanto, E. (2012). Pengujian perangkat lunak dengan menggunakan metode white box dan black box. HIMSYATECH, 8(1).

Mustaqbal, M. S., Firdaus, R. F., & Rahmadi, H. (2015). Pengujian aplikasi menggunakan black box testing boundary value analysis (studi kasus: Aplikasi prediksi kelulusan smnptn). Jurnal Ilmiah Teknologi Infomasi Terapan, 1(3).

Deni Murdiani, M. S. (2022). Jurnal teknik informatika. 10(2).

Jijon Raphita Sagala. (2021). Model Rapid Application Development (Rad)Dalam Pengembangan Sistem Informasi Penjadwalanbelajar Mengajar. Jurnal Mantik Penusa, 2(1), 88.

Prasetijo, L. H., Syah, F., Wibowo, S. H. S., Ardanu, F., Suyadi, & Utami, E. (2016). Penerapan Pendekatan Model Waterfall dalam Pengembangan Sistem E-Rapor. Jurnal Teknologi Technoscientia, 9(1), 39–47.

Puteri, M. P., & Effendi, H. (2018). Implementasi Metode RAD Pada Website Service Guide "Tour Waterfall South Sumatera." Jurnal Sisfokom (Sistem Informasi Dan Komputer), 7(2), 130–136. https://doi.org/10.32736/sisfokom.v7i2.570

Wahyuningrum, T., & Januarita, D. (2014). Perancangan Web e-Commerce dengan Metode Rapid Application Development (RAD) untuk Produk Unggulan Desa. 2014(November), 81–88.

Widiyana, N., Pratama, T. W. Y., & Prasetyo, A. A. (2021). Rancang Bangun Sistem Informasi Pendaftaran Pasien Berbasis Web Di Klinik Dander Medical Center Design and Build a Web-Based Patient Registration Information System At D. Indonesian Journal of Health Information Management, 1(2), 1–7. https://ijhim.stikesmhk.ac.id/index.php/ojsdata/article/view/9

Suryantara, I. G. N., Kom, S., & Kom, M. (2017). Merancang Applikasi dengan Metodologi Extreme Programming. Elex Media Komputindo.

Pressman, R.S. (2015). Software Engineering : A Practioner's Approach. 8th ed. McGraww-Hill Companies. Inc, Americas, New York. Sommerville, I. (2018). Software Engineering. Hallbergmoos/ Germany: Pearson.

Hasanah dan Untari. 2020. Rekayasa Perangkat Lunak. Umsida Press. https://press.umsida.ac.id/index.php/umsidapress/article/view/978-623-6833-89-6/728

Subakti, Widiastiwi, et all. 2022. Rekayasa Perangkat Lunak. Media Sains Indonesia.

https://www.arvis.id/insight/pemeliharaan-perangkat-lunak-pengertian-manfaat-dan-jenisnya/

http://elearning.unla.ac.id/pluginfile.php/30890/mod_resource/content/1/8_Deployment%20dan%20Pemeliharaan%20PL%20-%20Pemeliharaan%20PL.pdf

https://lmsspada.kemdikbud.go.id/pluginfile.php/663310/mod_resource/content/2/Materi%2010%20-%20Pemeliharaan%20Perangkat%20Lunak.pdf

https://se.ittelkom-pwt.ac.id/2019/04/14/software-maintenance-bagian-penting-yang-sering-diabaikan/

Rekayasa Perangkat Lunak

Saat ini pemrograman sangat berkembang, dan pengembangan pemrograman membutuhkan perangkat lunak yang dapat mendukung perencanaan, analisis, desain, implementasi, pengujian, dan pemeliharaan dari perangkat lunak.

Proses rekayasa perangkat lunak melibatkan berbagai tahapan dan aktivitas, tujuan dari rekayasa perangkat lunak menghasilkan perangkat lunak yang berkualitas, dapat diandalkan, efisien, dan memenuhi kebutuhan pengguna

Rekayasa perangkat lunak mencakup aspek manajemen proyek, di mana perencanaan, pengorganisasian, pengendalian, dan pemantauan proyek pengembangan perangkat lunak dilakukan. Manajemen proyek ini melibatkan alokasi sumber daya, perencanaan jadwal, manajemen risiko, dan komunikasi yang efektif antara anggota tim dan pemangku kepentingan proyek.

Rekayasa perangkat lunak memiliki peran penting dalam menghasilkan perangkat lunak yang berkualitas. Dengan pendekatan yang sistematis dan disiplin membantu mengurangi risiko kesalahan, meningkatkan efisiensi pengembangan, dan memastikan bahwa perangkat lunak yang dihasilkan dapat memberikan nilai tambah kepada penggunanya.

ISBN 978-623-09-4584-7

PT Penerbit Penamuda Media
Godean, Yogyakarta
085700592256
@penamuda_media
penamuda.com