**RANCANG BANGUN APLIKASI**

**LAYANAN PDAM TIRTA RAHARJA BERBASIS ANDROID**

**TUGAS METODOLOGI PENELITIAN**

Diajukan Sebagai

Tugas Metodologi Penelitian

Disusun Oleh :

**RIZKY DARMAWAN**

**17111035**



**PROGRAM STUDI TEKNIK INFORMATIKA**

**FAKULTAS TEKNOLOGI DAN INFORMATIKA**

**UNIVERSITAS INFORMATIKA DAN BISNIS INDONESIA**

**2019**

**RANCANG BANGUN APLIKASI**

**LAYANAN PDAM TIRTA RAHARJA BERBASIS ANDROID**

**TUGAS METODOLOGI PENELITIAN**

Diajukan Sebagai

Tugas Metodologi Penelitian

Disusun Oleh :

**RIZKY DARMAWAN**

**17111035**



**PROGRAM STUDI TEKNIK INFORMATIKA**

**FAKULTAS TEKNOLOGI DAN INFORMATIKA**

**UNIVERSITAS INFORMATIKA DAN BISNIS INDONESIA**

**2019**

# BAB 1

# PENDAHULUAN

## Latar Belakang Penelitian

Penerapan Penerapan Teknologi Informasi dalam membantu layanan dan operasional sehari-hari sudah bukan hal baru pada saat ini. Hampir semua perusahaan dan instansi pemerintahan sudah menggunakan teknologi informasi ini untuk mempermudah dan meningkatkan pelayanan. Disamping perkembangan pesat dari perangkat komputer yang sudah memasuki era laptop/notebook, perkembangan handphone pun tidak kalah pesatnya. Pada saat ini handphone bukan sebuah alat komukasi saja tetapi perkembangan handphone saat ini juga telah merambah ke segala aspek kehidupan sehingga saat ini seolah masyarakat telah dimanjakan oleh adanya alat yang dapat memberikan kemudahan dalam aktifitas sehari-hari, yaitu ditandai dengan lahirnya teknologi smartphone.

Perkembangan smartphone saat ini berkembang dengan pesat dan teknologinya tidak hanya digunakan oleh penggunanya sebagai media komunikasi, tetapi untuk berkoneksi dengan dunia luar seperti internet. Salah satunya smartphone yang memiliki sistem operasi berbasis Android. Android merupakan sistem perangkat mobile yang berkembang dengan pesat pada saat ini.

Android sebagai sistem operasi telepon pintar yang paling banyak digunakan di dunia. Android juga menjadi pilihan bagi perusahaan teknologi yang menginginkan sistem operasi berbiaya rendah dan ringan untuk perangkat berteknologi tinggi tanpa harus mengembangkannya dari awal. Sifat Android yang terbuka juga telah mendorong munculnya komunitas pengembang aplikasi dengan menambahkan fitur-fitur baru. Salah satunya adalah aplikasi pelayanan publik.

Seiring dengan perkembangan teknologi dan komunikasi, sistem informasi memberikan peran yang sangat penting dalam bidang pelayanan publik, sehingga pelayanan publik menggunakan keunggulan sistem informasi yang akan di gunakan sebagai kunci strategi penggelolaan pelayanan. Manajemen pelayanan publik dapat berjalan dengan baik untuk mencapai tujuan memerlukan adanya Sistem Informasi Manajemen yang membantu atau menjadi pertimbangan dalam mengambil keputusan bagi pimpinan dalam pengelolaan pelayanan publik. Dengan perkembangan teknologi semua pelayanan publik dapat dilakukan menggunakan teknologi yang memudahkan dalam pelayanan. Pelayanan publik yang menggunakan media elektronik dapat diterapkan dalam berbagai bidang seperti keluhan mengenai lalulintas, pendidikan, kesehatan, administrasi, dan masih banyak lagi.

Pada saat ini PDAM Tirta Raharja hanya memiliki website resmi dimana didalam website itu terdapat profile, berita, galeri foto perusahaan dan beberapa fitur pelayanan pelanggan, antara lain:

* Simulasi perhitungan tariff

Didalam fitur ini pelanggan bisa mengetahui detail perhitungan yang berisi Beban Tetap dan Nilai Air, dan Biaya Pemeliharaan Meter berdasarkan Golongan Pelanggan, Water Meter, Dan Tagihan (m3).

* Informasi tagihan pelanggan

Pelanggan dapat mengetahui informasi tagihan yang sedang berjalan dengan memasukkan nomor SL pelanggan.

* Halaman Prosedur pemasangan baru

Dalam halaman prosedur pemasangan baru berisi informasi tentang prosedur pemasangan baru, yaitu :

1. Calon pelanggan melakukan permohonan sambung baru dengan mengisi form permohonan sambung baru di Bagian Pelayanan.
2. Calon Pelanggan membawa berkas-berkas yang terdiri dari: Copy Kartu Identitas Diri yang Berlaku, Bukti Kepemilikan Rumah/Izin Pemilik dan Rekening SR Terdekat
3. Pelaksana Pelayanan melakukan identifikasi jaringan eksisting\* atau non-eksisting\* pada permohonan sambung baru.
4. Apabila jaringan berada di lokasi eksisting, Pelaksana Pelayanan melakukan input data pemohon dan memberikan dokumen bukti pendaftaran
5. Bagian Pelayanan mengatur jadwal untuk melakukan proses survey lapangan dan kemudian melakukan evaluasi hasil survey.
6. Apabila memungkinkan untuk dipasang sambungan baru, maka Bagian Langganan memberikan informasi kepada calon pelanggan bahwa permohonan disetujui beserta besaran biaya pasang.
7. Apabila tidak memungkinkan untuk pemasangan sambungan baru, maka calon pelanggan akan masuk ke dalam daftar tunggu potensi.
8. untuk memperoleh informasi apakah lokasi calon pelanggan berada pada jaringan eksisting atau non-eksisting, bisa menghubungi terlebih dulu Kantor Perumda Air Minum Tirta Raharja Wilayah Pelayanan masing-masing. Daftar alamat dan nomor telepon seluruh Kantor WIlayah Pelayanan Perumda Air Minum Tirta Raharja bisa dilihat di: <http://tirtaraharja.co.id/kontak>

* Dan halaman informasi kontak yang berisi Alamat, No Telepon, No Fax, alamat halaman facebook PDAM Tirta Raharja, Alamat dan No Telepon tiap kantor wilayan pelayanan PDAM Tirta Raharja beserta form yang berisi Nama Lengkap, Alamat Email, Subjek dan pesan untuk mengirim pesan terhadap PDAM Tirtja raharja.

PDAM Tirta Raharja pada saat ini tidak memiliki aplikasi resmi pelayanan terhadap pelanggan, dimana hanya terdapat aplikasi pengecheckan tagihan pelanggan di Playstore bernama PDAM BANDUNG KAB, BARAT, CIMAHI yang kemungkinan besar di buat oleh developer independen karena terdapat iklan di aplikasinya dan tidak ada pernyataan resmi dari pihak PDAM Tirta Raharja bahwa PDAM Tirta Raharja telah membuat aplikasi pelayanan pelanggan, fitur yang terdapat di aplikasi itu hanya membuat pelanggan PDAM Tirta Raharja mengetahui tagihan PDAM yang sedang berjalan dan tidak dapat melihat tagihan pada bulan sebelumnya.

Pada saat ini kemudahan pelanggan dalam mendapatkan pelayanan adalah hal yang wajib diberikan oleh perusahaan, dimana pada saat ini sebagian besar pelanggan telah memiliki smartphone maka dari itu PDAM Tirta Raharja seharusnya memiliki aplikasi smartphone resmi yang dapat memberikan kemudahan kepada pelanggan.

Berdasarkan hasil pengamatan saya, PDAM Tirta Raharja perlu membuat aplikasi untuk pelayanan pelanggan dengan menambahkan fitur baru dan memperbaiki ftur yang telah ada di website, yaitu :

* Pelaporan kerusakan PDAM

Pelaporan kerusakan PDAM masih dengan cara manual yaitu menelpon atau fax dan memerlukan biaya yang besar

* Proses pendaftaran pemasangan layanan baru yang mudah

Prosedur pemasangan yang ada sekarang di PDAM Tirta Raharja termasuk rumit dan memakan waktu dan biaya.

* Informasi tagihan bulan sebelumnya

Pelanggan hanya dapat mengetahui tagihan yang sedang berjalan

* Tidak tersedianya informasi jalur cakupan area PDAM

Informasi ini terkait dengan memudahkan calon pelanggan dalam mendaftar layanan PDAM, dimana petugas PDAM tidak perlu datang ke lokasi calon pengguna,

* Pelayanan pelanggan responsif dengan fitur chatbot dan chat realtime dengan customer service PDAM Tirta Raharja

## Rumusan Masalah

Berdasarkan Latar Belakang Penelitian maka dapat dirumuskan masalah sebagai berikut:

1. Bagaimana cara merancang sistem informasi yang dapat memudahkan calon pelanggan untuk mengetahui tercakup atau tidaknya area pelanggan oleh PDAM Tirta Raharja, mengetahui tarif layanan pdam dan mendaftar layanan PDAM Tirta Raharja ?
2. Bagaimana cara merancang sistem informasi yang dapat memenuhi kebutuhan pelanggan untuk melakukan pengecheckan tagihan layanan dan pelaporan kerusakan/keluhan kepada Tirta Raharja?
3. Bagaimana cara mengimplementasikan rancangan menjadi sebuah aplikasi yang dapat membuat calon pelanggan dan pelanggan merasa puas?

## Ruang Lingkup Penelitian

Hal yang akan di bahas dalam penelitian ini adalah cara merancang dan membangun aplikasi yang dapat memudahkan calon pengguna dan pengguna untuk mendapat pelayanan yang baik dari PDAM Tirta Raharja.

## Tujuan Penelitian

1. Untuk memberi solusi rancangan aplikasi yang dapat memenuhi kebutuhan dari calon pelanggan
2. Untuk memberi solusi rancangan aplikasi yang dapat memenuhi kebutuhan dari pelanggan.
3. Untuk membangun aplikasi yang dapat membuat calon pelanggan dan pelanggan puas terhadap pelayan PDAM Tirta Raharja

## Sistematika Penulisan

### Bahan dan Ukuran

1. Buku Tugas Akhir dibuat dengan menggunakan **HVS** **80** **gram/m2** , dengan ukuran A4 (21,5 cm x 29,7 cm) warna putih dan tidak boleh diketik bolak-balik;
2. Untuk sampul luar (kulit luar) ditetapkan **sampul** **kertas** (hard cover). Bahan yang digunakan adalah karton **buffalo** atau **linen**, dengan warna dasar biru tua (dark blue) dengan tulisan tinta berwarna emas untuk Fakultas Teknologi dan Informatika;
3. Tiap bab diberi pembatas dengan kertas **dorslag** (doorslag), dengan warna sesuai fakultas bergambar logo Universitas Informatika dan Bisnis Indonesia dengan warna sesuai warna sampul luar.

### Pengetikan

#### Batas Tepi

Tata letak halaman untuk pengetikan naskah Tugas Akhir dengan mesin tik manual, mesin tik elektronik atau komputer adalah sebagai berikut:

* + - 1. Tepi Kertas atas : 4 cm dari tepi kertas
      2. Tepi Kertas kiri : 4 cm dari tepi kertas
      3. Tepi Kertas bawah : 3 cm dari tepi kertas
      4. Tepi Kertas kanan : 3 cm dari tepi kertas

#### Cara Pengetikan

Pengetikan hanya dilakukan pada satu muka kertas, tidak bolak-balik.

1. Pengetikan dapat dilakukan dengan mesin tik manual, mesin tik elektronik, atau komputer.
2. Ukuran huruf yang digunakan harus standar, yaitu Pica untuk mesin tik, atau 12 points untuk komputer.
3. Pita atau tinta pada komputer, yang digunakan berwarna hitam.
4. Apabila menggunakan komputer, pencetakannya pada dot - matrix printer harus menggunakan NLQ (Near Letter Quality). Kalau menggunakan Deskjet atau laserjet tidak boleh menggunakan mode draft.
5. Perbanyakan hasil ketikan, atau print-out komputer, dilakukan dengan photocopy sejumlah yang ditetapkan masing-masing program studi. Bahan yang digunakan adalah kertas putih ukuran A4.

#### Spasi

1. Jarak antara baris yang satu dengan baris berikutnya adalah 1,5 spasi, perataan kiri kanan full justifikasi.
2. Jarak antara penunjuk bab (BAB I) dengan tajuk bab (PENDAHULUAN) adalah dua spasi.
3. Jarak antara tajuk bab (judul bab) dengan teks pertama yang ditulis, atau antara tajuk bab dengan tajuk anak bab adalah 4 (empat) spasi.
4. Jarak antara tajuk anak bab dengan baris pertama teks adalah 2 (dua) spasi.
5. Jarak antara baris akhir teks dengan tajuk anak bab berikutnya adalah 4 (empat) spasi.
6. Jarak antara teks dengan tabel, gambar, grafik, diagram, atau judulnya adalah 3 (tiga) spasi.
7. Alinea baru diketik menjorok ke dalam dan dimulai pada ketikan yang ke-6 dari batas tepi kiri ketikan. Jarak antara alinea yang satu denga alinea yang lain adalah 1,5 spasi.
8. Penunjuk bab dan tajuk bab (bab baru) selalu mulai pada halaman baru.

#### Judul dan Standar Huruf

1. Judul bab harus ditulis seluruhnya dengan huruf besar (capital) dengan jenis huruf yang digunakan adalah Times New Roman, ukuran 12 point, dipertebal (Bold), dan menggunakan perataan paragraf tengah (center).
2. Judul subbab, anak subbab, dan anak anak subbab diketik dengan jenis huruf Times New Roman, ukuran 12 point dipertebal (Bold), dan menggunakan paragraf rata kiri (Align left). Semua kata diawali dengan huruf besar (kapital), kecuali kata penghubung dan kata depan, dan semua diketik tebal tanpa diakhiri dengan titik. Kalimat pertama sesudah judul subbab dimulai dengan alinea baru.
3. Teks diketik dengan jenis huruf yang digunakan adalah Times New Roman ukuran 12 point.
4. Judul Tabel, gambar, diagram diketik jenis huruf yang dipakai Times New Roman, ukuran 10 point, 1 spasi, rata tengah terhadap tabel, gambar, dan diagram.
5. Huruf miring atau huruf khusus lain dapat dipakai untuk tujuan tertentu sesuai dengan ketentuan, misalnya untuk menandai istilah asing.
6. Tanda-tanda yang tidak dapat diketik, harus ditulis dengan rapi memakai tinda hitam.

#### Pengisian Ruangan

Ruangan yang terdapat pada halaman naskah harus diisi penuh, artinya pengetikan harus dimulai dari batas tepi kiri sampai batas kanan dan jangan sampai ada ruangan yang terbuang kecuali alinea baru, persamaan, daftar, gambar, judul, atau hal-hal yang khusus.

#### Permulaan Kalimat

Bilangan, lambang atau rumus yang mengawali suatu kalimat harus dieja, misalnya: seratus lima puluh juta orang pendudukan Indonesia berada dalam tarap kehidupan yang layak.

#### Rincian ke Bawah

Jika pada penulisan naskah ada rincian yang harus disusun ke bawah, dapat digunakan urutan dengan angka atau huruf dengan derajat rincian. Penggunaan tanda hubung (-), simbol, atau gambar yang ditempatkan di depan rincian tidak dibenarkan.

#### Catatan Kaki

Catatan kaki dapat dicantumkan bila diperlukan penjelasan/keterangan yang singkat terhadap yang ditulis. Tanda catatan kaki dibuat dalam angka latin dan berupa superscript (terletak lebih tinggi dan lebih kecil dari tulisan lainnya dalam naskah). Adapun penulisan keterangan catatan kaki dibuat di bawah nasakah.

#### Bilangan dan Satuan

1. Bilangan harus diketik dengan angka, misalnya: 10 g bahan baku, kecuali pada permulaan kalimat, angka harus dieja: Sepuluh gram bahan baku.
2. Bilangan desimal ditandai dengan koma, bukan titik, misalnya: Kapasitas disket 1,4 MB.
3. Satuan yang dipakai sedapat-dapatnya satuan SI (Standar Internasional) dan dinyatakan dengan singkatan resminya tanpa titik dibelakangnya, misalnya: MB (Mega Byte).

#### Kutipan

Kutipan merupakan bagian dari pernyataan, pendapat, buah pikiran, definisi, rumusan atau hasil penelitian dari penulis lain, atau penulis sendiri yang telah terdokumentasi, yang digunakan pada materi Tugas Akhir.

1. Kutipan langsung diketik terpisah dari teks, dengan jarak satu spasi dan menjorok masuk lima ketukan dari margin kiri teks, diikuti nama penulis, tahun, dan halaman. Jika nama penulis telah disebutkan di awal, maka tidak dituliskan lagi pada akhir kutipan.
2. Jarak antar baris teks dengan kutipan langsung adalah dua spasi.
3. Jika teks kutipan kurang dari 4 baris, maka diketik membentuk paragraf dengan spasi 1,5 spasi.
4. Penggunaan gagasan atau pemikiran seseorang penulis buku, artikel, dsb., walaupun disusun dengan menggunakan kata-kata sendiri, harus pula dicantumkan namanya (apabila perlu dapat pula dicantumkan judul karya 23 tulisnya) dan tahun buku/artikel itu ditulis, sesuai dengan kebiasaan penulis pada masing-masing disiplin ilmu.

Kutipan dapat ditulis dengan format berikut

1. Kutipan yang dicantumkan langsung dalam teks, dituliskan dalam dua tanda kutip. Bila penulis dicantumkan sebelum kutipan, maka setelah kalimat kutipan cantumkan tahun terbit dan nomor halaman dalam tanda kurung.

Contoh:

Fritz Bauer yang dikutip Roger S. Pressman mengemukakan: “Rekayasa perangkat lunak adalah pengembangan dan penggunaan prinsip pengembangan suara untuk memperoleh perangkat lunak secara ekonomis yang reliable dan bekerja secara efisien pada mesin nyata” (2002:28).

atau

IEEE yang dikutip Roger S. Pressman mengemukakan definisi yang lebih komprehensif sebagai berikut: “Rekayasa perangkat lunak : (1) Aplikasi dari sebuah pendekatan kuantifiabel, disiplin, dan sistematis kepada pengembangan, operasi, dan pemeliharaan perangkat lunak; yaitu aplikasi dari rekayasa perangkat lunak. (2) Studi tentang pendekatan-pendekatan seperti pada (1).” (2002:28).

1. Bila penulis dicantumkan setelah kutipan, maka sumber kutipan dicantumkan nama penulis, tahun penerbitan dan nomor halaman dalam tanda kurung.

Contoh:

Fritz Bauer mengemukakan: “[Rekayasa perangkat lunak adalah] pengembangan dan penggunaan prinsip pengembangan suara untuk memperoleh perangkat lunak secara ekonomis yang reliable dan bekerja secara efisien pada mesin nyata” (Roger S. Pressman, 2002:28).

#### Penomoran

1. **Halaman Bagian Awal**
2. Penomoran halaman bagian awal Tugas Akhir, mulai dari halaman ‘judul bagian dalam / cover dalam’ sampai dengan halaman ‘daftar lampiran’, menggunakan angka Romawi kecil.
3. Halaman ‘judul bagian dalam / cover dalam’ sampai dengan halaman ‘pernyataan keaslian tugas akhir’ tidak diberi nomor urut halaman, tetapi diperhitungkan sebagai halaman i, ii dan seterusnya (nomor halaman tersebut tidak diketik).
4. ‘Halaman abstrak’ sampai dengan ‘halaman daftar lampiran’ diberi nomor dengan angka Romawi kecil, yang merupakan kelanjutan dari halaman sebelumnya (nomor halaman diketik).
5. Pada tiap halaman yang bertajuk, mulai dari abstrak sampai dengan Daftar Lampiran, nomor halaman diletakkan pada pias bawah persis di tengahtengah, berjarak 1,5 cm dari tepi kertas (baris akhir teks pada halaman itu).
6. **Halaman Bagian Inti**

Pemberian nomor pada bagian inti Tugas Akhir ditetapkan seperti di bawah ini.

1. Penomoran bagian inti Tugas Akhir, mulai dari BAB PENDAHULUAN sampai dengan BAB PENUTUP, menggunakan angka Arab.
2. Nomor halaman diletakkan pada pias atas sebelah kanan atas halaman, berjarak 1,5 cm dari tepi kertas bagian atas dan kanan (baris pertama teks pada halaman itu) dan angka terakhir nomor halaman itu lurus dengan margin kanan.
3. Pada tiap halaman yang berjudul, mulai dari BAB PENDAHULUAN sampai dengan BAB PENUTUP, nomor halaman diletakkan pada pias bawah persis di tengah-tengah, berjarak 1,5 cm dari tepi kertas bagian bawah (baris akhir teks pada halaman itu).
4. Penomoran halaman berurutan tanpa mencantumkan nomor bab (1,2,3 ...)
5. **Halaman Bagian Akhir**

Pemberian nomor pada bagian akhir Tugas Akhir dilakukan sebagai berikut:

1. Penomoran bagian akhir Tugas Akhir, mulai dari DAFTAR PUSTAKA sampai dengan RIWAYAT HIDUP (kalau ada), diberi nomor halaman tetapi tidak ditampilkan
2. Penomoran halaman berurutan dan merupakan kelanjutan dari bagian inti.
3. **Persamaan**
4. Bila persamaan dijelaskan dalam tulisan, maka huruf p pada persamaan ditulis dengan huruf besar, misal: Persamaan (L=PxL).
5. Kata sambung penjelasan persamaan dapat ditulis dengan kata: dengan atau keterangan sebagai kata ganti where dalam bahasa Inggris.
6. Nomor urut persamaan atau rumus matematika, reaksi kimia dan lainlainnya ditulis dengan angka latin di dalam tanda kurung () dan ditempatkan didekat batas tepi kanan. Penomoran persamaan diberikan angka depan sesuai nomor bab, kemudian diberi titik dan dilanjutkan dengan nomor berupa angka latin.

#### Tabel, Gambar, Grafika dan Diagram

Sebelum table atau gambar ditampilkan, maka tabel atau gambar tersebut sudah disebutkan pada uraian sebelumnya namun paling jauh satu halaman sebelumnya.

1. **Tabel**
2. Tabel dimuat kira-kira di tengah-tengah halaman.
3. Judulnya ditik di atas tabel, mengikuti lebar tabel memperhitungkan keseimbangan halaman.
4. Penulisan Sumber berada pada kiri bawah Tabel dan ditebalkan (bold)
5. Nomor tabel terdiri atas dua bagian yaitu:
6. Bagian pertama menunjukkan nomor bab dimana tabel itu dimuat;
7. Bagian kedua menunjukkan nomor urut tabel pada bab itu.
8. Misalnya, Tabel 2.4, menunjukkan bahwa tabel itu ada pada BAB II dan merupakan tabel urutan keempat pada bab itu.
9. Kalimat pertama judul tabel ditulis sesudah nomor tabel, dengan jarak dua ketukan.
10. Awal baris kedua judul tabel berada di bawah awal judul tabel (bukan di bawah nomor tabel).
11. **Gambar**
12. Gambar dimuat kira-kira di tengah-tengah halaman.
13. Judulnya diketik di bawah gambar, mengikuti lebar gambar, dengan memperhitungkan keseimbangan halaman.
14. Nomor gambar terdiri atas dua bagian, yaitu:
15. Bagain pertama menunjukkan nomor bab dimana gambar itu dimuat.
16. Bagian kedua menunjukkan nomor urut gambar pada bab itu. Misalnya, Gambar 3.8, menunjukkan bahwa gambar itu ada pada BAB III dan merupakan gambar urutan kedelapan pada bab itu.
17. Kalimat pertama judul gambar ditulis sesudah nomor gambar, dengan jarak dua ketukan.
18. Awal baris kedua judul gambar berada di bawah awal judul gambar (bukan di bawah nomor gambar).
19. Penulisan Sumber berada pada tengah bawah gambar
20. **Grafik**
21. Grafik dimuat kira-kira di tengah-tengah halaman.
22. Judulnya diketik di atas grafik, mengikuti lebar grafik, dengan memperhitungkan keseimbangan halaman.
23. Nomor grafik terdiri atas dua bagian, yaitu:
24. Bagian pertama menunjukkan nomor bab dimana grafik itu dimuat;
25. Bagian kedua menunjukkan nomor urut grafik pada bab itu.

Misalnya, Grafik 4.5, menunjukkan bahwa grafik itu ada pada BAB IV dan merupakan grafik urutan kelima pada bab itu.

1. Kalimat pertama judul grafik ditulis sesudah nomor grafik, dengan jarak dua ketukan.
2. Awal baris kedua judul grafik berada di bawah awal judul grafik (bukan di bawah nomor grafik)
3. Penulisan Sumber berada pada tengah bawah grafik
4. **Diagram**
5. Diagram dimuat kira-kira di tengah-tengah halaman.
6. Judul ditik di bawah diagram, mengikuti lebar diagram, dengan memperhitungkan keseimbangan halaman.
7. Nomor diagram terdiri atas dua bagian yaitu:
8. Bagian pertama menunjukkan nomor bab ini dimana diagram itu dimuat;
9. Bagian kedua menunjukkan nomor urut diagram pada bab itu.

Misalnya, Diagram 2.1, menunjukkan bahwa diagram itu ada pada BAB II dan merupakan diagram urutan pertama pada bab itu.

1. Kalimat pertama judul diagram ditulis sesudah nomor diagram, dengan jarak dua ketukan.
2. Awal baris kedua judul diagram berada di bawah awal judul diagram (bukan di bawah nomor diagram)
3. Penulisan Sumber berada pada tengah bawah grafik

#### Bahasa

1. Bahasa yang dipakai

Tugas Akhir ditulis dalam bahasa Indonesia yang baku, sesuai dengan ketentuan ejaan yang disempurnakan.

1. Bentuk kalimat Kalimat ditulis tanpa menggunakan kata ganti orang.
2. Istilah
3. Sedapat mungkin digunakan istilah Indonesia atau yang sudah diIndonesiakan.
4. Jika terpaksa harus dipakai istilah asing harus dijelaskan dengan tanda khusus secara konsisten dan ditulis miring (italic).
5. Hal-hal yang perlu diperhatikan
6. Kata hubung, misalnya sehingga dan sedangkan, tidak boleh dipakai sebagai awal suatu kalimat.
7. Kata “dimana” dan “dari”, sebagai terjemahaan “where” dan “of” dalam bahasa Inggris sering kurang tepat pemakaiannya dalam bahasa Indonesia yang baku.
8. Awalan “ke” dan “di” harus dibedakan dengan kata depan “ke” dan “di”. d. Pemenggalan kata supaya disesuaikan dengan kaidah Bahasa Indonesia yang baik dan benar

#### Abstrak dan Abstract

1. Abstrak dan Abstract
2. Pengetikan Abstrak
3. Jarak spasi dalam pengetikan abstrak adalah satu spasi.
4. Jarak antara alinea yang satu dengan alinea yang lain adalah satu setengah spasi.
5. Alinea baru diketik menjorok ke dalam lima ketukan dari margin kiri teks.
6. Pengetikan Abstract

Pada dasarnya sama dengan butir a di atas, tetapi judul **ABSTRACT** dan seluruh teks diketik dengan huruf miring atau italic dengan mengunakan bahasa Inggris.

1. Panjang dan Isi Abstrak dan Abstract

Panjang abstrak dan abstract Tugas Akhir ditetapkan sekitar 150 - 250 kata. Abstrak dan abstract karya tulis sekurang-kurangnya berisi hal-hal berikut:

1. Nama jelas penulis
2. NPM Penulis
3. Judul Penelitian
4. Kegiatan kerja yang diobservasi/dilakukan atau masalah yang diteliti, kalau mungkin dalam satu kalimat;
5. Subyek/obyek penelitian, disertai karakteristik khususnya, seperti, misalnya, jumlah, tipe, usia, jenis kelamin, spesies, dan/atau karakteristik lainnya;
6. Pendekatan yang digunakan, termasuk peralatan/instrument (kalau ada), prosedur kerja/pengumpulan data, penggunaan perlakuan atau treatment (kalau ada)
7. Hasil observasi, termasuk taraf signifikansi statistik (kalau ada);
8. Kesimpulan dan implikasi, terapan, atau rekomendasi.
9. Kata kunci (keyword)

#### Halaman Kulit Luar/Sampul Luar

Bahan sampul luar/kulit luar sesuai dengan ketentuan pada butir 5.1.(b). Penulisan dan penempatan judul Tugas Akhir, anak judul (kalau ada), tulisan LAPORAN TUGAS AKHIR, nama dan NPM mahasiswa, simbol Universitas Informatika dan Bisnis Indonesia, program Diploma III, program studi, nama Perguruan Tinggi, kota dan tahun penyusunan Tugas Akhir, pada sampul luar dan sampul dalam, mengikuti ketentuan berikut.

1. **Judul dan Sub/Anak judul**

Penulisan judul dan anak judul diatur sebagai berikut:

1. Judul Tugas Akhir ditulis dibaris paling atas, dengan huruf kapital semua, dengan jarak dari tepi atas kertas sebesar 4 (empat) cm, dan tidak boleh data kata yang disingkat kecuali singkatan yang sudah baku (seperti PT, CV, dan yang lainnya).
2. Jenis huruf yang digunakan adalah Times New Roman berukuran 14 point dan di bold.
3. Judul yang panjang ditulis menjadi dua baris atau lebih, dengan pemotongan judul yang logis, sesuai dengan kaidah Bahasa Indonesia. Jarak antara kedua baris judul diatur agak rapat dan dianjurkan ditulis secara piramida terbalik.
4. Anak Judul (kalau ada) ditulis di bawah judul, dengan huruf kapital semua dengan ukuran 12 point dan di bold jenis huruf Tahoma, dengan diberi jarak dari judul sekitar 1,5 cm dari baris judul yang paling bawah.
5. Anak judul yang panjang ditulis menjadi dua baris atau lebih, dengan pemotongan subjudul yang logis, sesuai dengan kaidah Bahasa Indonesia. Jarak antara kedua baris diatur agak rapat dan dianjurkan ditulis secara piramida terbalik.
6. Judul maupun anak judul tidak diakhiri dengan tanda titik (.).
7. **Penjelasan**

Pada bagian tengah halaman kulit luar dicantumkan suatu penjelasan mengenai maksud penulisan Tugas Akhir yang ditulis secara piramida terbalik, dengan menggunakan huruf kapital pada setiap awal kata, kecuali kata sambung.

1. Tulisan TUGAS AKHIR ditulis dengan huruf kapital semua dan di- Bold, diletakkan ditengah, dengan jenis huruf Arial, dan besar huruf 12 point.
2. Letak tulisan TUGAS AKHIR sekitar 2,5 (dua setengah) cm di bawah anak judul. Kalau tidak ada anak judul, letak tulisan TUGAS AKHIR sekitar 5 (lima) cm dari baris judul yang paling bawah.
3. Di bawah tulisan TUGAS AKHIR, dengan jarak sekitar satu cm, dicantumkan kalimat penjelasan.
4. Kalimat penjelasan menggunakan huruf kapital pada setiap awal kata, kecuali kata sambung (seperti untuk, pada) diketik dengan jenis huruf Arial, besar huruf 11 point, dan spasi 1.
5. **Nama dan NPM Penulis**
6. Nama mahasiswa ditulis dengan huruf kapital semua, diletakkan di tengah setelah baris yang berisi kalimat **Disusun** **oleh** **:,** dengan jenis huruf **Times** **New** **Roman**, dengan di-**Bold** dan besar huruf yang sama dengan anak judul yaitu 14 point.
7. Letak tulisan nama mahasiswa sekitar 2,5 (dua setengah) cm di bawah tulisan Universitas Informatika dan Bisnis Indonesia pada butir 2 di atas.
8. NPM mahasiswa ditulis dengan huruf kapital semua, diletakkan di tengah, di bawah nama mahasiswa, dengan jenis dan besar huruf yang sama dengan anak judul. Baris NPM diatur agak rapat dengan baris nama mahasiswa.
9. **Lambang Universitas Informatika dan Bisnis Indonesia**

Simbol Universitas Informatika dan Bisnis Indonesia bergaris tengah (diameter) sekitar tiga setengah cm. Titik tengahnya terletak kira-kira di tengah-tengah diantara baris NPM mahasiswa dengan baris nama Universitas Informatika dan Bisnis Indonesia.

1. **Nama Program studi, Fakultas, Perguruan Tinggi, Kota Perguruan Tinggi, dan Tahun Penyusunan**
2. Tulisan nama Universitas Informatika dan Bisnis Indonesia, program Diploma III, program studi dan tahun penyusunan laporan Tugas Akhir ditulis dengan huruf kapital semua dan di-Bold, dengan jenis huruf Tahoma, 12 point, dipertebal, dengan spasi 1 (single) berurutan seperti contoh di bawah.
3. Tahun penyusunan Tugas Akhir yang ditulis paling bawah, diletakkan sekitar tiga setengah cm dari tepi bawah kertas.
4. Berturut-turut ke atas seperti pada contoh di bawah.

#### Halaman Judul Bagian Dalam/Sampul Dalam

Judul bagian dalam sama denan sampul luar/kulit luar, hanya dicetak pada kertas HVS, sesuai dengan ketentuan pada butir

#### Halaman Lembar Pengesahan

Pada dasarnya lay-out halaman persetujuan pembimbing mengacu pada butir 1.5.1 dan 1.5.2 Isinya adalah sebagai berikut:

1. Judul laporan Tugas Akhir diketik dengan jarak empat cm dari tepi kertas bagian atas. Semua kalimat judul diketik dengan huruf kapital, dengan jarak antar baris 1 spasi.
2. Baris subjudul diketik di bawah judul, dengan jarak sekitar satu cm dari baris terakhir judul. Semua diketik dengan huruf kapital.
3. Nama mahasiswa diketik di bawah subjudul, dengan jarak sekitar satu cm dari baris terakhir subjudul. Semua diketik dengan huruf kapital.
4. NPM diketik di bawah nama mahasiswa, dengan jarak rapat.
5. Waktu pemberian persetujuan hanya ditulis bulan dan tahunnya, diketik di belakang ‘Bandung’.
6. Lay-out ‘Menyetujui’, Pembimbing diatur dengan memperhatikan keseimbangan pada halaman ini. Nama pembimbing berjarak empat cm dari tepi kertas bagian bawah.

#### Halaman Pernyataan Keaslian Tugas Akhir

Halaman ini diketik sesuai dengan ketentuan pada butir 1.5.2 dengan spasi 1 (satu). Nama dan NPM dituliskan secara lengkap.

#### Daftar Pustaka

Pengetikan buku, jurnal, dan artikel yang digunakan sebagai bahan referensi, dilakukan seperti di bawah ini:

1. Jarak spasi yang digunakan untuk pengetikan daftar pustaka adalah satu spasi.
2. Baris kedua tiap buku (jurnal, artikel lain) referensi ditik menjorok ke dalam 5 (lima) ketukan.
3. Judul buku diketik miring (jika ada huruf miring atau italic), atau diberi garis bawah (jika tidak ada huruf miring atau italic); selain huruf pertama judul dan subjudul, seluruhnya ditik dengan huruf miring.
4. Jika judul artikel yang digunakan merupakan bagian dari suatu buku (misalnya, beberapa artikel dengan penulis berbeda diedit dalam satu buku), yang ditik miring adalah judul bukunya (judul artikelnya tetap ditik dengan huruf tegak diapit).
5. Jurnal buku tidak ditik dengan huruf miring, yang ditik dengan huruf miring adalah nama jurnalnya, hanya huruf pertama judul dan huruf pertama subjudul yang ditik dengan huruf kapital, lainnya dengan huruf kecil.
6. Jarak spasi baris akhir suatu buku (Jurnal, artikel lain) dengan baris pertama buku (Jurnal, artikel lain) berikutnya adalah satu setengah spasi.
7. Urutan pengetikan adalah sebagai berikut.
8. Buku (Format Penulisan Dapat Dilihat Pada Sub Judul 1.5.2.20)
9. Nama penulis, baik penulis Indonesia maupun bukan Indonesia, dimulai dengan nama belakang (ditik lengkap), diikuti nama depan (diketik singkatannya), diakhiri dengan tanda titik (.). Bila penulis lebih dari satu orang, maka nama penulis kedua dan seterusnya disusun sesuai dengan nama aslinya, yaitu nama disesuaikan dengan yang tercantum pada halaman judul referensi tanpa mengindahkan urutan alpabet, serta antara nama pertama dengan penulis kedua dipisahkan dengan tanda koma (,). Bila penulis lebih dari empat orang, setelah nama penulis pertama dicantumkan huruf dkk (singkatan dari dan kawan-kawan).
10. Tahun terbit, diakhiri dengan tanda titik.
11. Judul buku, ditik dengan huruf miring atau diberi garis bawah; selain huruf pertama judul dan subjudul, seluruhnya ditik dengan huruf kecil dan diakhiri dengan tanda titik (.).
12. Kota tempat penerbit atau negara bagian tempat penerbit (yang didahului dengan kota tempat penerbit), diakhiri dengan tanda titik dua (:).
13. Nama penerbit, diakhiri dengan tanda titik (.). Masing-masing dengan jarak dua ketukan, kecuali kota tempat penerbit berjarak satu ketukan.
14. Penulisan daftar pustaka yang melebihi satu baris, maka baris kedua dan seterusnya dimulai dengan indensi 5 (lima) (huruf pertama diketik pada ketukan keenam) dari batas huruf pertama pada baris pertama, dan jarak antar baris adalah satu spasi.
15. Artikel yang Diedit Dalam Suatu Buku (Format Penulisan Dapat Dilihat Pada Sub Judul 1.5.2.20)
16. Nama penulis, baik penulis Indonesia maupun bukan Indonesia, dimulai dengan nama belakang (ditik lengkap),diikuti nama depan (ditik singkatnya), diakhiri dengan tanda titik (.).
17. Tahun terbit, diakhiri dengan tanda titik (.).
18. Judul artikel tidak diketik dengan huruf miring (italic) atau diberi garis bawah; selain huruf pertama judul dan subjudul, seluruhnya ditik dengan huruf kecil dan diakhiri dengan tanda titik (.).
19. Nama editor (atau editor-editornya), ditik di belakang kata “Dalam” dan dimulai dengan nama depannya (ditik singkatnya), diikuti nama belakang (ditik lengkap), diakhiri dengan tanda titik dua (:).
20. Judul buku ditik dengan huruf miring atau italic atau diberi garis bawah, seluruhnya ditik dengan huruf kecil dan diakhiri dengan tanda titik (.).
21. Kota tempat penerbit atau negara bagian tempat penerbit (yang dapat didahului dengan kota tempat penerbit), diakhiri dengan tanda titik dua (.).
22. Nama penerbit, diakhiri dengan tanda titik (.).

Masing-masing dengan jarak dua ketukan, kecuali kota tempat penerbit dan nama penerbit jarak satu ketukan.

1. Jurnal (Format Penulisan Dapat Dilihat Pada Sub Judul 1.5.2.20)
2. Nama penulis, baik penulis Indonesia maupun bukan penulis Indonesia, dimulai dengan nama belakang (diketik lengkap), diikuti nama depan (diketik singkatannya), diakhiri dengan tanda titik (.);
3. Tahun terbit, diakhiri dengan tanda titik (.);
4. Judul artikel, tidak diketik dengan huruf miring (italic) atau diberi garis bawah, selain huruf pertama judul dan subjudul, seluruhnya diketik dengan huruf kecil dan diakhiri dengan tanda titik (.);
5. Nama jurnal, diketik dengan huruf miring (italic) atau diberi garis bawah, diakhiri dengan tanda titik (.);
6. Nomor halaman, tidak ditik dengan huruf miring (italic), nomor halaman ini ditik mulia dari halaman awal sampai dengan akhir artikel.
7. Artikel Lain (Format Penulisan Dapat Dilihat Pada Sub Judul 1.5.2.20) Pada prinsipnya mengikuti aturan seperti pada butir (a), (b), dan (c) di atas.
8. Apabila dua referensi atau lebih digunakan, nama penulisnya (atau penulis- penulis) sama, maka nama penulis pada referensi kedua (dan selanjutnya) tidak ditulis lagi, nama penulis diganti dengan garis bawah sebanyak tujuh ketukan-ketentuan.
9. Penulis sama, tahun sama, pakai a, b, c

#### Format Penulisan Pustaka

Penulisan daftar pustaka dapat mengikuti format berikut.

1. Buku

Contoh:

Bandura, A. 1977. *Socal* *Learning* *Theory*. Englewood Cliffs, New Jersey; Prentice

- Hall.

Koentjaraningrat. 1983. *Bunga Rampai Kebudayaan, Mentalitas, dan*

*Pembangunan*. Jakarta: Gramedia.

Lewin, K. 1935. *A* *Dynamic* *Theory* *of* *Personality*; *Selected* *papers*. New York:

McGraw - Hill.

Tarigan, Henry Guntur, 1958a. *Pengajaran* *Gaya* *Bahasa*, Bandung: Angkasa.

Atau

Bandura, A. 1977. Socal Learning Theory. Englewood Cliffs, New Jersey; Prentice

- Hall.

Koentjaraningrat. 1983. Bunga Rampai Kebudayaan, Mentalitas, dan

Pembangunan. Jakarta: Gramedia

Lewin, K. 1935. A Dynamic Theory of Personality; Selected papers. New York:

McGraw - Hill.

Tarigan, Henry Guntur, 1958a. Pengajaran Gaya Bahasa, Bandung: Angkasa

1. Artikel yang Diedit dalam Suatu Buku

Contoh;

Lewin, L. 1958. Group Decision and Social Change. Dalam E.E. Maccoby, T.M.

NewComb, & E.L. Hartley (Eds), *Reading in social Psychology*, 3rd edition.

New York: Holt, Rinenead, & winston.

. Atau

Lewin, L. 1958. Group Decision and Social Change. Dalam E.E. Maccoby, T.M.

NewComb, & E.L. Hartley (Eds), Reading in social Psychology, 3rd edition.

New York: Holt, Rinenead, & winston.

1. Jurnal

Contoh: (besarnya huruf disesuaikan dengan ketentuan)

Bell, S.M. 1970 The Development of the concept of the object as related to infant

- mother attachement. *Child Development*, 41,291-311.

Bower, G.H. 1981. Mood and Memory. *American Psychologist*, 36, 139-148.

MacLean, P.D. 1958. The limbic system with respect to self - preservation and the

preservation of species. *Journal or nervous Mental Disease*, 127, 1-11.

Atau

Bell, S.M. 1970 The Development of the concept of the object as related to infant

- mother attachement. Child Development, 41,291-311.

Bower, G.H. 1981. Mood and Memory. American Psychologist, 36, 139-148.

MacLean, P.D. 1958. The limbic system with respect to self - preservation and the

1. Artikel Lain

Mulder, N. 1984. Kebatinan dan hidup sehari-hari orang jawa: kelangsungan dan

perubahan kulturil. Diterjemahkan oleh A.A. Nugroho. Jakarta: Gramedia.

Hendrojuwono, W. 1990. Pengaruh ‘experiental learning’ terhadap peningkatan

ketahanan ego dan kontrol ego remaja: Suatu model perlakuan dan evaluasi fungsi ego. Universitas Informatika dan Bisnis Indonesia: yang tidak dipublikasikan.

Atau

Mulder, N. 1984. Kebatinan dan hidup sehari-hari orang jawa: kelangsungan dan

perubahan kulturil. Diterjemahkan oleh A.A. Nugroho. Jakarta: Gramedia.

Hendrojuwono, W. 1990. Pengaruh ‘experiental learning’ terhadap peningkatan

ketahanan ego dan kontrol ego remaja: Suatu model perlakuan dan evaluasi fungsi ego. Universitas Informatika dan Bisnis Indonesia: yang tidak dipublikasikan.

#### Dokumentasi

Laporan Tugas Akhir harus diperbanyak dan dijilid rapi setelah saran perbaikan dilaksanakan sebaik-baiknya dan setelah dinyatakan lulus dalam sidang ujian akhir program.

Laporan Tugas Akhir yang telah diperbanyak harus diserahkan kepada tim pembimbing, program studi dan UPT perpustakaan.

Atas persetujuan Program Studi, laporan Tugas Akhir dapat diberikan kepada instansi/lembaga lain, tempat mahasiswa melakukan Tugas Akhir. Tanpa izin ketua program studi atau pembimbing utama, laporan Tugas Akhir tidak boleh digunakan sebagai referensi mahasiswa lain dalam menyusun Tugas Akhirnya.

# BAB 2

# LANDASAN TEORI

## Perangkat Lunak (Software)

## Definisi Perangkat Lunak

Berdasarkan Kamus Webster’s New Intercollegiate, 1979 yang dikutip Pratap K.J. Mohapatra mengemukakan : “Software is the entire set of programs, procedures and related documentation associated with a system and especially a computer system.” (2010:9).

“Perangkat lunak adalah sekumpulan rangkaian program, prosedur dan dokumentasi yang terkait dengan suatu system dan terutama sistem komputer.”

Kamus Webster’s terbaru, 1981, yang dikutip Pratap K.J. Mohapatra menulis ulang definisi, berorientasi sepenuhnya ke komputer: “Software is the programs and programming support necessary to put a computer through its assigned tasks, as distinguished from the actual machine.” (2010:9)

“Perangkat lunak adalah program dan dukungan pemrograman yang diperlukan untuk menempatkan komputer melalui tugas yang ditugaskan, yang dibedakan dari mesin sebenarnya.”

Definisi yang lebih ketat tetapi fungsional diberikan oleh Blum (1992), yang dikutip Pratap K.J Mohapatra : “Software is the detailed instructions that control the operation of a computer system. Its functions are to (1) manage the computer resources of the organisation (2) provide tools for human beings to take advantage of these resources, and (3) act as intermediary between organisations and stored information.” (2010:9)

“Perangkat lunak adalah intruksi terperinci yang mengontrol operasi pada sistem komputer. Fungsinya adalah untuk (1) mengelola sumber daya organisasi komputer (2) menyediakan alat bagi manusia untuk mengambil keuntungan dari sumber daa ini, dan (3) bertindak sebagai perantara antara organisasi dan informasi yang tersimpan.”

Dikutip oleh Pratap K.J Mohapatra, Gilb (1977) mendefinisikan dua komponen utama perangkat lunak:

1. Logicware, urutan logis dari intruksi aktif yang mengendalikan urutan eksekusi (urutan pemrosesan data) yang dilakukan perangkat keras, dan

2. Dataware, fisik dari mana semua informasi (pasif), termasuk logicware, muncul di perangkat keras, dan yang diproses sebagai hasil dari logicware.

Menurut Ian Sommervile di Buku Software Engineering 9th Edition definisi software adalah: “Computer programs and associated documentation. Software products may be developed for a particular consumer or may be developed for a general market.” (2009:6)

“Program komputer dan dokumentasi terkait. Produk perangkat lunak dapat dikembangkan untuk konsumen tertentu atau dapat dikembangkan untuk pasar umum."

Menurut Roger S, Pressman di buku Software Engineering A PRACTITIONER’S APPROACH Eight Edition: “Software is: (1) Instructions (computer programs) that when executed provide desires features, function, and performance; (2) data structures that enable the programs to adequately manipulate information, and (3) descriptive information in both hard copy and virtual forms that describes the operation and use of the programs.” (2015:4)

“Perangkat lunak adalah: (1) Instruksi (program komputer) yang ketika dijalankan memberikan fitur, fungsi, dan kinerja yang diinginkan; (2) struktur data yang memungkinkan program untuk memanipulasi informasi secara memadai, dan (3) informasi deskriptif baik dalam bentuk cetak maupun virtual yang menggambarkan operasi dan penggunaan program." (2015:4)

## Karakteristik Perangkat Lunak

Perangkat lunak lebih logis daripada elemen sistem fisik. Oleh karena itu, perangkat lunak memiliki karakteristik yang sangat berbeda dari perangkat keras (Wolverton 1984, dan Pressman 1992), Beberapa perbedaan utama adalah sebagai berikut:

1. Perangkat lunak dikembangkan atau direkayasa, tidak diproduksi.

* Konsep 'bahan baku' tidak ada di sini. Lebih baik divisualisasikan sebagai suatu proses, daripada produk (Jensen dan Tonies, 1979)
* 'Elemen manusia' sangat tinggi dalam pengembangan perangkat lunak, dibandingkan dengan manufaktur.
* Produktivitas pengembangan sangat tidak pasti, bahkan dengan produk standar, sangat bervariasi dengan keterampilan pengembang.
* Alat, teknik, standar, dan prosedur pengembangan sangat bervariasi di dan di dalam suatu organisasi.
* Masalah kualitas dalam pengembangan perangkat lunak sangat berbeda dengan yang ada di manufaktur. Sedangkan karakteristik kualitas manufaktur dapat ditentukan secara objektif dan mudah

1. Pengembangan perangkat lunak hadir sebagai lingkungan manufaktur kecil

* Di sini setiap produk dibuat khusus dan karenanya unik.
* Tidak dapat dirakit dari komponen yang ada.
* Semua kerumitan bengkel kerja (yaitu, masalah desain, estimasi, dan penjadwalan) ada di sini.
* Keterampilan manusia, elemen paling penting dalam sebuah job shop, juga merupakan elemen paling penting dalam pengembangan perangkat lunak.

1. Waktu dan upaya untuk pengembangan perangkat lunak sulit untuk diperkirakan.

* Pekerjaan yang menarik dilakukan dengan mengorbankan pekerjaan yang membosankan, dan dokumentasi, menjadi pekerjaan yang membosankan, mendapatkan prioritas yang paling rendah.
* Melakukan pekerjaan dengan cara yang cerdas cenderung menjadi pertimbangan yang lebih penting daripada menyelesaikannya secara memadai, tepat waktu, dan dengan biaya yang masuk akal.
* Pemrogram cenderung optimis, tidak realistis, dan perkiraan waktu mereka untuk penyelesaian tugas mencerminkan kecenderungan ini.
* Pemrogram kesulitan berkomunikasi.

1. User Requreimenet sering kali tidak dipahami dengan baik: karena itu sebuah perangkat luna mengalami banyak modifikasi sebelum diterapkan dengan memuaskan
2. Hampir tidak ada standar objektif untuk mengukur kemajuan perangkat lunak
3. Menguji perangkat lunak sangat sulit, karena bahkan program berukuran sedang (<5000 pernyataan yang dapat di eksekusi) dapat memiliki banyak path yang dapat di eksekusi ( Cara untuk mendapatkan dari awal program hingga akhir) sehingga proses pengujian setiap path melalui program bisa menjadi sangat mahal.
4. Perangkat lunak tidak memiliki waktu kadaluarsa.

* Perangkat lunak biasanya tidak kehilangan fungsinya saat digunakan.
* Ini mungkin kehilangan fungsionalitasnya dalam waktu, namun, ketika persyaratan pengguna berubah.
* Ketika cacat ditemukan, mereka dihapus dengan menulis ulang kode yang relevan, bukan dengan menggantinya dengan kode yang tersedia. Itu berarti bahwa konsep mengganti kode yang rusak dengan kode cadangan sangat tidak biasa dalam pengembangan perangkat lunak.
* Ketika cacat dihilangkan, ada kemungkinan bahwa cacat baru diperkenalkan.

1. Hardware memiliki model fisik untuk digunakan dalam mengevaluasi keputusan desain. Evaluasi desain perangkat lunak, di sisi lain dan, bertumpu pada penilaian dan intuisi.
2. Perangkat keras, karena keterbatasan fisiknya, memiliki batasan praktis pada kompleksitas karena setiap desain perangkat keras harus diwujudkan sebagai implementasi fisik. Perangkat lunak, di sisi lain, bisa sangat kompleks sambil tetap memenuhi hampir semua kebutuhan.
3. Ada perbedaan besar antara manajemen proyek perangkat keras dan perangkat lunak. Kontrol tradisional untuk proyek perangkat keras mungkin kontraproduktif dalam proyek perangkat lunak. Misalnya, melaporkan persen yang diselesaikan dalam hal Line of Code dapat sangat menyesatkan.

### Jenis Perangkat Lunak

Menurut Ian Sommerville, Ada dua jenis produk perangkat lunak:

1. **Generic products**

Produk generik Ini adalah sistem yang berdiri sendiri yang diproduksi oleh organisasi pengembangan dan dijual di pasar terbuka kepada setiap pelanggan yang mampu membelinya. Contoh dari jenis produk ini termasuk perangkat lunak untuk PC seperti basis data, pengolah kata, paket gambar, dan alat manajemen proyek. Ini juga mencakup apa yang disebut aplikasi vertikal yang dirancang untuk beberapa tujuan tertentu seperti sistem informasi perpustakaan, sistem akuntansi, atau sistem untuk memelihara catatan gigi.

1. **Customized (or bespoke)**

produk Ini adalah sistem yang ditugaskan oleh pelanggan tertentu. Kontraktor perangkat lunak mengembangkan perangkat lunak terutama untuk pelanggan itu. Contoh perangkat lunak jenis ini meliputi sistem kontrol untuk perangkat elektronik, sistem yang ditulis untuk mendukung proses bisnis tertentu, dan sistem kontrol lalu lintas udara.

### Karakteristik Produk Perangkat Lunak

Menurut Ian Sommervile di buku Software engineering 9th edition

Ada beberapa karakteristik produk perangkat lunak, diantaranya:

* **Maintainability**

Perangkat lunak harus ditulis sedemikian rupa sehingga dapat berkembang untuk memenuhi perubahan kebutuhan pelanggan. Ini adalah atribut penting karena perubahan perangkat lunak merupakan persyaratan yang tak terhindarkan dari lingkungan bisnis yang berubah.

* **Dependability and security**

Ketergantungan perangkat lunak mencakup serangkaian karakteristik termasuk keandalan, keamanan, dan keselamatan. Perangkat lunak yang dapat diandalkan tidak boleh menyebabkan kerusakan fisik atau ekonomi jika terjadi kegagalan sistem. Pengguna jahat seharusnya tidak dapat mengakses atau merusak sistem.

* **Efficiency**

Perangkat lunak tidak boleh menggunakan sumber daya sistem secara boros seperti siklus memori dan prosesor. Efisiensi karena itu termasuk daya tanggap, waktu pemrosesan, pemanfaatan memori, dll.

* **Acceptability**

Perangkat lunak harus dapat diterima oleh pengguna sebagaimana perangkat lunak itu di desain untuk tipe pengguna . Ini berarti bahwa itu harus dapat dimengerti, dapat digunakan, dan kompatibel dengan sistem lain yang mereka gunakan.

## 2.1.5 Kategori Perangkat Lunak

Menurut Ian Pressman, Perangkat lunak memiliki 7 Kategori :

* **System Software (Perangkat Lunak Sistem)**

Kumpulan program yang ditulis untuk melayani program lain. Beberapa perangkat lunak sistem (mis., Kompiler, editor, dan utilitas manajemen file) memproses kompleks, tetapi menentukan, struktur informasi. Aplikasi sistem lainnya (mis., Komponen sistem operasi, drive, perangkat lunak jaringan, prosesor telekomunikasi) memproses sebagian besar data yang tidak ditentukan.

* **Application Software (Aplikasi perangkat lunak)**

Program yang berdiri sendiri yang menyelesaikan kebutuhan bisnis tertentu. Aplikasi dalam bidang ini memproses data bisnis atau teknis sedemikian rupa sehingga memudahkan operasi bisnis atau pengambilan keputusan teknis / teknis.

* **Engineering/scientific software (Rekayasa / perangkat lunak ilmiah)**

Beragam program pengolah angka yang berkisar dari astronomi hingga vulkanologi, dari analisis tegangan otomotif hingga dinamika orbital, dan dari desain berbantuan komputer hingga biologi molekuler, dari analisis genetik hingga meteorologi.

* **Embedded Software (Perangkat lunak tertanam)**

Berada di dalam produk atau sistem dan digunakan untuk mengimplementasikan dan mengontrol fitur dan fungsi untuk pengguna akhir dan untuk sistem itu sendiri. Perangkat lunak tertanam dapat melakukan fungsi terbatas dan esoteris (mis., Fungsi digital dalam mobil seperti kontrol bahan bakar, tampilan dasbor, dan sistem pengereman).

* **Product Line Software (Perangkat lunak lini produk)**

Dirancang untuk memberikan kemampuan spesifik untuk digunakan oleh banyak pelanggan yang berbeda. Perangkat lunak lini produk dapat fokus pada pasar yang terbatas dan esoteris (mis., Produk kontrol inventaris) atau alamat konsumen massal.

* **Web / Mobile Application (Aplikasi Web/Mobile)**

Kategori perangkat lunak yang berpusat pada jaringan ini mencakup beragam aplikasi dan mencakup aplikasi dan perangkat lunak berbasis browser yang berada di perangkat seluler.

* **Artificial intelligence software (Perangkat lunak kecerdasan buatan)**

Memanfaatkan algoritma nonnumerik untuk menyelesaikan masalah kompleks yang tidak sesuai dengan perhitungan atau analisis langsung. Aplikasi dalam area ini termasuk robotika, sistem pakar, pengenalan pola (gambar dan suara), jaringan saraf tiruan, pembuktian teorema, dan bermain game.

## Rekayasa Perangkat Lunak

### Definisi

Naur (Naur dan Randell 1969) yang ikut mengedit laporan tentang konferensi NATO yang terkenal di Garnish juga ikut menulis salah satu buku paling awal tentang masalah ini (Naur et al.1976). Dalam buku ini, ide-ide di balik rekayasa perangkat lunak diberikan sebagai berikut:

* + - 1. Mengembangkan produk perangkat lunak besar jauh lebih kompleks daripada mengembangkan program yang berdiri sendiri.
      2. Prinsip-prinsip desain teknik harus diterapkan pada tugas mengembangkan produk perangkat lunak besar.

Ada banyak definisi "Rekayasa Perangkat Lunak" seperti halnya penulis. Kami mencoba untuk melihat sekilas contoh definisi yang diberikan oleh eksponen di lapangan.

Bauer (1972) memberikan definisi paling awal untuk rekayasa perangkat lunak (Bauer 1972, p. 530):

"... Pembentukan dan penggunaan prinsip-prinsip rekayasa suara (metode) untuk mendapatkan perangkat lunak ekonomis yang dapat diandalkan dan bekerja pada mesin nyata."

Menurut Boehm (1976), rekayasa perangkat lunak adalah

"... aplikasi praktis dari pengetahuan ilmiah dalam desain dan konstruksi program komputer dan dokumentasi terkait yang diperlukan untuk mengembangkan, mengoperasikan, dan memeliharanya."

Boehm (1976) memperluas idenya dengan menekankan bahwa masalah pengembangan perangkat lunak yang paling mendesak adalah di bidang analisis persyaratan, desain, pengujian, dan pemeliharaan perangkat lunak aplikasi oleh teknisi dalam konteks yang didorong oleh ekonomi daripada di bidang desain terperinci dan pengkodean perangkat lunak sistem oleh para ahli dalam konteks yang relatif independen terhadap ekonomi.

DeRemer dan Kron (1976) mengakui rekayasa perangkat lunak untuk menangani pemrograman-in-thelarge, sementara Parnas (1978) berpandangan bahwa rekayasa perangkat lunak berkaitan dengan ‘pembangunan multi-orang untuk perangkat lunak multi-versi.’

IEEE (IEEE93a) telah mengembangkan definisi berikut untuk rekayasa perangkat lunak:

Rekayasa Perangkat Lunak: (1) Penerapan pendekatan yang sistematis, disiplin, kuantitatif untuk pengembangan, pengoperasian, dan pemeliharaan perangkat lunak: yaitu, penerapan rekayasa perangkat lunak. (2) Studi pendekatan seperti pada (1).

Definisi yang lebih baru oleh Wang dan King (2000) menganggap rekayasa perangkat lunak sebagai disiplin dan membuat prinsip-prinsip teknik dan atribut produk lebih eksplisit:

"Rekayasa perangkat lunak adalah disiplin yang mengadopsi pendekatan rekayasa seperti metodologi yang ditetapkan, proses, alat, standar, metode organisasi, metode manajemen, sistem jaminan kualitas, dan sejenisnya untuk mengembangkan perangkat lunak skala besar dengan produktivitas tinggi, biaya rendah, kualitas terkendali, dan jadwal pengembangan pengukuran. "

Menurut Ian Sommervile dalam buku Rekayasa Perangkat Lunak Edisi ke-9.

Rekayasa perangkat lunak adalah disiplin teknik yang berkaitan dengan semua aspek produksi perangkat lunak dari tahap awal spesifikasi sistem hingga pemeliharaan sistem setelah mulai digunakan. Dalam definisi ini, ada dua frase kunci:

1. Disiplin Insinyur Insinyur membuat sesuatu bekerja. Mereka menerapkan teori, metode, dan alat di mana ini sesuai. Namun, mereka menggunakannya secara selektif dan selalu berusaha menemukan solusi untuk masalah bahkan ketika tidak ada teori dan metode yang berlaku. Insinyur juga mengakui bahwa mereka harus bekerja dengan kendala organisasi dan keuangan sehingga mereka mencari solusi dalam kendala ini.
2. Semua aspek produksi perangkat lunak Rekayasa perangkat lunak tidak hanya berkaitan dengan proses teknis pengembangan perangkat lunak. Ini juga mencakup kegiatan seperti manajemen proyek perangkat lunak dan pengembangan alat, metode, dan teori untuk mendukung produksi perangkat lunak

Rekayasa adalah tentang mendapatkan hasil dari kualitas yang diperlukan dalam jadwal dan anggaran. Ini sering melibatkan kompromi — insinyur tidak bisa perfeksionis. Orang-orang yang menulis program untuk diri mereka sendiri, bagaimanapun, dapat menghabiskan waktu sebanyak yang mereka inginkan untuk pengembangan program.

Secara umum, insinyur perangkat lunak mengadopsi pendekatan yang sistematis dan terorganisir untuk pekerjaan mereka, karena ini seringkali merupakan cara paling efektif untuk menghasilkan perangkat lunak berkualitas tinggi. Namun, rekayasa adalah tentang memilih metode yang paling tepat untuk serangkaian keadaan sehingga pendekatan pengembangan yang lebih kreatif dan kurang formal mungkin efektif dalam beberapa keadaan. Pengembangan yang kurang formal sangat sesuai untuk pengembangan sistem berbasis web, yang membutuhkan perpaduan antara keterampilan perangkat lunak dan desain grafis.

Rekayasa perangkat lunak penting karena dua alasan:

1. Semakin banyak, individu dan masyarakat bergantung pada sistem perangkat lunak canggih. Kita harus mampu menghasilkan sistem yang andal dan dapat dipercaya secara ekonomis dan cepat.
2. Dalam jangka panjang, biasanya lebih murah untuk menggunakan metode dan teknik rekayasa perangkat lunak untuk sistem perangkat lunak daripada hanya menulis program seolah-olah itu adalah proyek pemrograman pribadi. Untuk sebagian besar jenis sistem, sebagian besar biaya adalah biaya untuk mengubah perangkat lunak setelah mulai digunakan.

### Etika Rekayasa Perangkat Lunak

Menurut Ian Sommervile dalam buku Software Engineering 9th Edition. Seperti disiplin teknik lainnya, rekayasa perangkat lunak dilakukan dalam kerangka sosial dan hukum yang membatasi kebebasan orang yang bekerja di bidang itu. Sebagai seorang insinyur perangkat lunak, Anda harus menerima bahwa pekerjaan Anda melibatkan tanggung jawab yang lebih luas daripada sekadar penerapan keterampilan teknis. Anda juga harus berperilaku etis dan bertanggung jawab secara moral jika Anda harus dihormati sebagai insinyur profesional.

Tak perlu dikatakan bahwa Anda harus menjunjung standar kejujuran dan integritas yang normal. Anda tidak boleh menggunakan keahlian dan kemampuan Anda untuk berperilaku tidak jujur ​​atau dengan cara yang akan menimbulkan kecemaran bagi profesi rekayasa perangkat lunak. Namun, ada area di mana standar perilaku yang dapat diterima tidak terikat oleh hukum tetapi oleh gagasan tanggung jawab profesional yang lebih lemah. Beberapa di antaranya adalah:

1. Kerahasiaan Anda biasanya harus menghormati kerahasiaan majikan atau klien Anda terlepas dari apakah perjanjian kerahasiaan formal telah ditandatangani atau tidak.
2. Kompetensi Anda tidak boleh salah menggambarkan tingkat kompetensi Anda. Anda seharusnya tidak secara sadar menerima pekerjaan yang berada di luar kompetensi Anda.
3. Hak kekayaan intelektual Anda harus mengetahui undang-undang setempat yang mengatur penggunaan kekayaan intelektual seperti paten dan hak cipta. Anda harus berhati-hati untuk memastikan bahwa kekayaan intelektual pengusaha dan klien dilindungi.
4. Penyalahgunaan komputer Anda tidak boleh menggunakan keterampilan teknis Anda untuk menyalahgunakan komputer orang lain. Penyalahgunaan komputer berkisar dari yang relatif sepele (bermain game di mesin majikan, katakanlah) hingga sangat serius (penyebaran virus atau malware lain).

## Proses Perangkat Lunak

### Definisi Proses Perangkat Lunak

Proses perangkat lunak adalah serangkaian kegiatan terkait yang mengarah pada produksi produk perangkat lunak. Kegiatan ini mungkin melibatkan pengembangan perangkat lunak dari awal dalam Bahasa pemrograman standar seperti Java atau C. Namun, aplikasi bisnis tidak perlu dikembangkan dengan cara ini. Perangkat lunak bisnis baru sekarang sering dikembangkan dengan memperluas dan memodifikasi system yang ada atau dengan mengkonfigurasi dan mengintegrasikan perangkan lunak atau komponen sistem yang tidak tersedia.

Ada banyak proses perangkat lunak yang berbeda tetapi semua harus mencakup empat kegiatan yang mendasar untuk rekayasa perangkat lunak:

* + - 1. Software specification Fungsionalitas perangkat lunak dan batasan operasinya harus ditentukan.
      2. Software design and implementation perangkat lunak yang di produksi harus memenuhi spesifikasi.
      3. Software validation perangkat lunak harus divalidasi untuk memastikan bahwa ia melakukan apa yang diinginkan pelanggan.
      4. Software evolution perangkat lunak harus berevolusi untuk memenuhi perubahan kebutuhan pelanggan.

Dalam beberapa bentuk, kegiatan ini adalah bagian dari semua proses perangkat lunak. Dalam praktiknya, tentu saja, mereka adalah kegiatan yang kompleks dalam dirinya sendiri dan mencakup sub kegiatan seperti validasi persyaratan, desain arsitektur, pengujian unit, dll. Ada juga kegiatan proses pendukung seperti dokumentasi dan manajemen konfigurasi perangkat lunak.

Ketika kami menggambarkan dan mendiskusikan proses, kami biasanya berbicara tentang kegiatan dalam proses ini seperti menentukan model data, merancang antarmuka pengguna, dll., Dan urutan aktifitas. Namun, seperti halnya aktifitas, deskripsi juga dapat mencakup:

* + - 1. Produk, yang merupakan hasil dari suatu kegiatan proses. Sebagai contoh, hasil dari aktivitas desain arsitektur dapat menjadi model arsitektur perangkat lunak.
      2. Peran, yang mencerminkan tanggung jawab orang-orang yang terlibat dalam proses. Contoh peran adalah manajer proyek, manajer konfigurasi, pemrogram, dll.
      3. Pra dan pasca kondisi, yaitu pernyataan yang benar sebelum dan sesudah suatu kegiatan proses diberlakukan atau produk diproduksi. Misalnya, sebelum desain arsitektur dimulai, prasyarat mungkin bahwa semua persyaratan telah disetujui oleh pelanggan; setelah kegiatan ini selesai, kondisi pasca mungkin bahwa model UML yang menggambarkan arsitektur telah ditinjau.

Proses perangkat lunak itu kompleks dan, seperti semua proses intelektual dan kreatif, bergantung pada orang yang membuat keputusan dan penilaian. Tidak ada proses yang ideal dan sebagian besar organisasi telah mengembangkan proses pengembangan perangkat lunak mereka sendiri. Proses telah berkembang untuk mengambil keuntungan dari kemampuan orang-orang dalam suatu organisasi dan karakteristik spesifik dari sistem yang sedang dikembangkan. Untuk beberapa sistem, seperti sistem kritis, proses pengembangan yang sangat terstruktur diperlukan. Untuk sistem bisnis, dengan persyaratan yang berubah dengan cepat, proses yang kurang formal dan fleksibel cenderung lebih efektif.

Terkadang, proses perangkat lunak dikategorikan sebagai proses yang digerakkan oleh rencana atau tangkas. Proses yang digerakkan oleh rencana adalah proses di mana semua kegiatan proses direncanakan sebelumnya dan kemajuan diukur terhadap rencana ini. Dalam proses gesit, yang saya bahas dalam Bab 3, perencanaan bersifat inkremental dan lebih mudah untuk mengubah proses untuk mencerminkan perubahan kebutuhan pelanggan. Seperti yang dibahas oleh Boehm dan Turner (2003), setiap pendekatan cocok untuk berbagai jenis perangkat lunak. Secara umum, Anda perlu menemukan keseimbangan antara proses yang digerakkan oleh rencana dan proses yang gesit.

Meskipun tidak ada proses perangkat lunak 'ideal', ada ruang untuk meningkatkan proses perangkat lunak di banyak organisasi. Proses dapat mencakup teknik yang sudah ketinggalan zaman atau mungkin tidak memanfaatkan praktik terbaik dalam rekayasa perangkat lunak industri. Memang, banyak organisasi masih tidak memanfaatkan metode rekayasa perangkat lunak dalam pengembangan perangkat lunak mereka.

Proses perangkat lunak dapat ditingkatkan dengan standarisasi proses di mana keragaman proses perangkat lunak di suatu organisasi berkurang. Ini mengarah pada peningkatan komunikasi dan pengurangan waktu pelatihan, dan membuat dukungan proses otomatis lebih ekonomis. Standardisasi juga merupakan langkah pertama yang penting dalam memperkenalkan metode dan teknik rekayasa perangkat lunak baru serta praktik rekayasa perangkat lunak yang baik.

### Model Proses Perangkat Lunak

Model proses perangkat lunak adalah representasi yang disederhanakan dari proses perangkat lunak. Setiap model proses mewakili proses dari perspektif tertentu, dan dengan demikian hanya memberikan informasi parsial tentang proses itu. Misalnya, model kegiatan proses menunjukkan kegiatan dan urutannya tetapi mungkin tidak menunjukkan peran orang yang terlibat dalam kegiatan ini. Di bagian ini, saya memperkenalkan sejumlah model proses yang sangat umum (kadang-kadang disebut 'paradigma proses') dan menyajikannya dari perspektif arsitektur. Artinya, kita melihat kerangka proses tetapi bukan rincian kegiatan tertentu.

Model generik ini bukan deskripsi pasti dari proses perangkat lunak. Sebaliknya, mereka adalah abstraksi dari proses yang dapat digunakan untuk menjelaskan berbagai pendekatan pengembangan perangkat lunak. Anda dapat menganggapnya sebagai kerangka proses yang dapat diperluas dan disesuaikan untuk menciptakan proses rekayasa perangkat lunak yang lebih spesifik.

Model proses yang saya bahas di sini adalah:

1. The Waterfall Model Ini mengambil kegiatan proses dasar spesifikasi, pengembangan, validasi, dan evolusi dan mewakili mereka sebagai fase proses terpisah seperti spesifikasi kebutuhan, desain perangkat lunak, implementasi, pengujian, dan sebagainya.
2. Incremental development. Pendekatan ini menghubungkan aktivitas spesifikasi, pengembangan, dan validasi. Sistem ini dikembangkan sebagai serangkaian versi (peningkatan), dengan setiap versi menambahkan fungsionalitas ke versi sebelumnya.
3. Reuse-oriented software engineering. Pendekatan ini didasarkan pada keberadaan sejumlah besar komponen yang dapat digunakan kembali. Proses pengembangan sistem berfokus pada pengintegrasian komponen-komponen ini ke dalam sistem daripada mengembangkannya dari awal.

#### The Waterfall Model

Model publikasi pertama dari proses pengembangan perangkat lunak berasal dari proses rekayasa sistem yang lebih umum (Royce, 1970). Karena riam dari satu fase ke fase lain, model ini dikenal sebagai 'model air terjun' atau siklus hidup perangkat lunak. Model air terjun adalah contoh dari proses yang digerakkan oleh rencana — pada prinsipnya, Anda harus merencanakan dan menjadwalkan semua kegiatan proses sebelum mulai mengerjakannya.

Tahapan utama dari model air terjun secara langsung mencerminkan kegiatan pembangunan mendasar:

1. Analisis dan definisi persyaratan. Layanan, kendala, dan tujuan sistem ditetapkan melalui konsultasi dengan pengguna sistem. Mereka kemudian didefinisikan secara rinci dan berfungsi sebagai spesifikasi sistem.
2. Desain sistem dan perangkat lunak Proses desain sistem mengalokasikan persyaratan untuk sistem perangkat keras atau perangkat lunak dengan membangun arsitektur sistem secara keseluruhan. Desain perangkat lunak melibatkan mengidentifikasi dan menggambarkan abstraksi sistem perangkat lunak mendasar dan hubungannya.
3. Implementasi dan pengujian unit Selama tahap ini, desain perangkat lunak diwujudkan sebagai satu set program atau unit program. Pengujian unit melibatkan verifikasi bahwa setiap unit memenuhi spesifikasinya.
4. Integrasi dan pengujian sistem Unit program individu atau program diintegrasikan dan diuji sebagai sistem yang lengkap untuk memastikan bahwa persyaratan perangkat lunak telah dipenuhi. Setelah pengujian, sistem perangkat lunak dikirim ke pelanggan.
5. Operasi dan pemeliharaan Biasanya (walaupun tidak harus), ini adalah fase siklus hidup terpanjang. Sistem diinstal dan digunakan secara praktis. Pemeliharaan melibatkan koreksi kesalahan yang tidak ditemukan pada tahap awal siklus hidup, meningkatkan implementasi unit sistem dan meningkatkan layanan sistem ketika persyaratan baru ditemukan.

Pada prinsipnya, hasil dari setiap fase adalah satu atau lebih dokumen yang disetujui ('ditandatangani'). Fase berikut tidak boleh dimulai sampai fase sebelumnya selesai. Dalam praktiknya, tahapan-tahapan ini saling tumpang tindih dan saling memberi informasi. Selama desain, masalah dengan persyaratan diidentifikasi. Selama pengkodean, masalah desain ditemukan dan seterusnya. Proses perangkat lunak bukanlah model linier sederhana tetapi melibatkan umpan balik dari satu fase ke fase lain. Dokumen yang diproduksi di setiap fase kemudian harus dimodifikasi untuk mencerminkan perubahan yang dilakukan.

Karena biaya memproduksi dan menyetujui dokumen, iterasi dapat menjadi mahal dan melibatkan pengerjaan ulang yang signifikan. Oleh karena itu, setelah sejumlah kecil iterasi, adalah normal untuk membekukan bagian dari pengembangan, seperti spesifikasi, dan untuk melanjutkan dengan tahap pengembangan selanjutnya. Masalah dibiarkan untuk resolusi nanti, diabaikan, atau diprogram di sekitar. Pembekuan persyaratan prematur ini dapat berarti bahwa sistem tidak akan melakukan apa yang diinginkan pengguna. Ini juga dapat menyebabkan sistem terstruktur dengan buruk karena masalah desain diatasi dengan trik implementasi.

Selama fase siklus hidup akhir (operasi dan pemeliharaan) perangkat lunak mulai digunakan. Kesalahan dan kelalaian dalam persyaratan perangkat lunak asli ditemukan. Kesalahan program dan desain muncul dan kebutuhan akan fungsionalitas baru diidentifikasi. Karena itu sistem harus berkembang agar tetap berguna. Membuat perubahan ini (pemeliharaan perangkat lunak) mungkin melibatkan pengulangan tahapan proses sebelumnya.

Model Waterfall konsisten dengan model proses rekayasa lainnya dan dokumentasi diproduksi pada setiap fase. Ini membuat proses terlihat sehingga manajer dapat memantau kemajuan terhadap rencana pengembangan. Masalah utamanya adalah partisi proyek yang tidak fleksibel menjadi beberapa tahap berbeda. Komitmen harus dibuat pada tahap awal dalam proses, yang membuatnya sulit untuk menanggapi perubahan kebutuhan pelanggan.

Pada prinsipnya, model Waterfall hanya boleh digunakan ketika persyaratan dipahami dengan baik dan tidak mungkin berubah secara radikal selama pengembangan sistem. Namun, model Waterfall mencerminkan jenis proses yang digunakan dalam proyek rekayasa lainnya. Karena lebih mudah untuk menggunakan model manajemen umum untuk seluruh proyek, proses perangkat lunak berdasarkan model Waterfall masih umum digunakan.

Varian penting dari model Waterfall adalah pengembangan sistem formal, di mana model matematika dari spesifikasi sistem dibuat. Model ini kemudian disempurnakan, menggunakan transformasi matematika yang mempertahankan konsistensinya, menjadi kode yang dapat dieksekusi. Berdasarkan asumsi bahwa transformasi matematis Anda benar, Anda dapat membuat argumen yang kuat bahwa program yang dihasilkan dengan cara ini konsisten dengan spesifikasinya.

Proses pengembangan formal, seperti yang didasarkan pada metode B (Schneider, 2001; Wordsworth, 1996) sangat cocok untuk pengembangan sistem yang memiliki persyaratan keselamatan, keandalan, atau keamanan yang ketat. Pendekatan formal menyederhanakan produksi kasus keselamatan atau keamanan. Ini menunjukkan kepada pelanggan atau regulator bahwa sistem benar-benar memenuhi persyaratan keselamatan atau keamanannya.

Proses yang didasarkan pada transformasi formal umumnya hanya digunakan dalam pengembangan sistem keselamatan-kritis atau keamanan-kritis. Mereka membutuhkan keahlian khusus. Untuk sebagian besar sistem, proses ini tidak menawarkan manfaat biaya yang signifikan atas pendekatan lain untuk pengembangan sistem.

#### Incremental Development

Incremental development didasarkan pada gagasan untuk mengembangkan implementasi awal, memaparkannya pada komentar pengguna dan mengembangkannya melalui beberapa versi hingga sistem yang memadai telah dikembangkan. Spesifikasi, pengembangan, dan kegiatan validasi disisipkan daripada dipisahkan, dengan umpan balik cepat di seluruh kegiatan.

Incremental development, yang merupakan bagian mendasar dari pendekatan gesit, lebih baik daripada pendekatan air terjun untuk sebagian besar bisnis, e-commerce, dan sistem pribadi. Pengembangan tambahan mencerminkan cara kita memecahkan masalah. Kami jarang mencari solusi masalah yang lengkap sebelumnya, tetapi bergerak menuju solusi dalam serangkaian langkah, mundur ketika kami menyadari bahwa kami telah membuat kesalahan. Dengan mengembangkan perangkat lunak secara bertahap, itu lebih murah dan lebih mudah untuk membuat perubahan dalam perangkat lunak saat ini sedang dikembangkan.

Setiap kenaikan atau versi sistem menggabungkan beberapa fungsi yang dibutuhkan oleh pelanggan. Secara umum, peningkatan awal sistem mencakup fungsi yang paling penting atau paling mendesak dibutuhkan. Ini berarti bahwa pelanggan dapat mengevaluasi sistem pada tahap yang relatif awal dalam pengembangan untuk melihat apakah itu memberikan apa yang diperlukan. Jika tidak, maka hanya kenaikan saat ini yang harus diubah dan, mungkin, fungsionalitas baru yang ditentukan untuk kenaikan selanjutnya.

Pengembangan tambahan memiliki tiga manfaat penting, dibandingkan dengan model air terjun:

1. Biaya mengakomodasi perubahan kebutuhan pelanggan berkurang. Jumlah analisis dan dokumentasi yang harus diulang jauh lebih sedikit daripada yang diperlukan dengan model air terjun.
2. Lebih mudah untuk mendapatkan umpan balik pelanggan tentang pekerjaan pengembangan yang telah dilakukan. Pelanggan dapat mengomentari demonstrasi perangkat lunak dan melihat berapa banyak yang telah diterapkan. Pelanggan merasa sulit untuk menilai kemajuan dari dokumen desain perangkat lunak.
3. Pengiriman yang lebih cepat dan penyebaran perangkat lunak yang bermanfaat bagi pelanggan adalah mungkin, bahkan jika semua fungsi belum dimasukkan. Pelanggan dapat menggunakan dan mendapatkan nilai dari perangkat lunak lebih awal dari yang dimungkinkan dengan proses air terjun.

Incremental development dalam beberapa bentuk sekarang merupakan pendekatan yang paling umum untuk pengembangan sistem aplikasi. Pendekatan ini dapat berupa rencana-didorong, gesit, atau, lebih biasanya, campuran dari pendekatan ini. Dalam pendekatan yang digerakkan oleh rencana, peningkatan sistem diidentifikasi sebelumnya; jika pendekatan tangkas diadopsi, kenaikan awal diidentifikasi tetapi pengembangan kenaikan kemudian tergantung pada kemajuan dan prioritas pelanggan.

Dari perspektif manajemen, pendekatan inkremental memiliki dua masalah:

1. Prosesnya tidak terlihat. Manajer membutuhkan kiriman reguler untuk mengukur kemajuan. Jika sistem dikembangkan dengan cepat, itu tidak hemat biaya untuk menghasilkan dokumen yang mencerminkan setiap versi sistem.
2. Struktur sistem cenderung menurun ketika penambahan baru ditambahkan. Kecuali jika waktu dan uang dihabiskan untuk refactoring untuk meningkatkan perangkat lunak, perubahan reguler cenderung merusak strukturnya. Menggabungkan perubahan perangkat lunak lebih lanjut menjadi semakin sulit dan mahal.

Masalah pengembangan inkremental menjadi sangat akut untuk sistem yang besar, kompleks, dan tahan lama, di mana tim yang berbeda mengembangkan bagian sistem yang berbeda. Sistem besar membutuhkan kerangka kerja atau arsitektur yang stabil dan tanggung jawab tim yang berbeda yang bekerja pada bagian sistem perlu didefinisikan dengan jelas sehubungan dengan arsitektur itu. Ini harus direncanakan terlebih dahulu daripada dikembangkan secara bertahap.

Anda dapat mengembangkan sistem secara bertahap dan memaparkannya kepada pelanggan untuk dikomentari, tanpa benar-benar mengirimkannya dan menggunakannya di lingkungan pelanggan. Pengiriman dan penyebaran tambahan berarti bahwa perangkat lunak digunakan dalam proses operasional yang nyata. Ini tidak selalu mungkin karena bereksperimen dengan perangkat lunak baru dapat mengganggu proses bisnis normal.

#### Reuse-oriented Software Engineering

Di sebagian besar proyek perangkat lunak, ada beberapa perangkat lunak yang digunakan kembali. Ini sering terjadi secara informal ketika orang yang bekerja di proyek mengetahui desain atau kode yang mirip dengan apa yang diperlukan. Mereka mencari ini, memodifikasinya sesuai kebutuhan, dan memasukkannya ke dalam sistem mereka.

Penggunaan kembali secara informal ini terjadi terlepas dari proses pengembangan yang digunakan. Namun, pada abad ke-21, proses pengembangan perangkat lunak yang berfokus pada penggunaan kembali perangkat lunak yang ada telah banyak digunakan. Pendekatan berorientasi-ulang bergantung pada basis besar komponen perangkat lunak yang dapat digunakan kembali dan kerangka kerja integrasi untuk komposisi komponen-komponen ini. Terkadang, komponen-komponen ini adalah sistem dengan haknya sendiri (COTS atau sistem komersial yang tersedia) yang dapat menyediakan fungsionalitas khusus seperti pengolah kata atau spreadsheet.

Meskipun tahap spesifikasi persyaratan awal dan tahap validasi dapat dibandingkan dengan proses perangkat lunak lain, tahap perantara dalam proses berorientasi-ulang berbeda. Tahapan-tahapan ini adalah:

* + - 1. Analisis komponen Mengingat spesifikasi persyaratan, pencarian dilakukan untuk komponen untuk mengimplementasikan spesifikasi itu. Biasanya, tidak ada kecocokan persis dan komponen yang dapat digunakan hanya menyediakan beberapa fungsi yang diperlukan.
      2. Persyaratan modifikasi Selama tahap ini, persyaratan dianalisis menggunakan informasi tentang komponen yang telah ditemukan. Mereka kemudian dimodifikasi untuk mencerminkan komponen yang tersedia. Jika modifikasi tidak dimungkinkan, aktivitas analisis komponen dapat dimasukkan kembali untuk mencari solusi alternatif.
      3. Desain sistem dengan menggunakan kembali Selama fase ini, kerangka kerja sistem dirancang atau kerangka kerja yang ada digunakan kembali. Para desainer mempertimbangkan komponen yang digunakan kembali dan mengatur kerangka kerja untuk memenuhi hal ini. Beberapa perangkat lunak baru mungkin harus dirancang jika komponen yang dapat digunakan kembali tidak tersedia.
      4. Pengembangan dan integrasi Perangkat lunak yang tidak dapat dibeli secara eksternal dikembangkan, dan komponen serta sistem COTS diintegrasikan untuk membuat sistem baru. Integrasi sistem, dalam model ini, mungkin menjadi bagian dari proses pengembangan daripada aktivitas terpisah.

Ada tiga jenis komponen perangkat lunak yang dapat digunakan dalam proses berorientasi ulang:

* + - 1. Layanan web yang dikembangkan sesuai dengan standar layanan dan yang tersedia untuk doa jarak jauh.
      2. Koleksi objek yang dikembangkan sebagai paket untuk diintegrasikan dengan kerangka kerja komponen seperti .NET atau J2EE.
      3. Sistem perangkat lunak yang berdiri sendiri yang dikonfigurasi untuk digunakan dalam lingkungan tertentu.

Rekayasa perangkat lunak berorientasi reuse memiliki keuntungan yang jelas dalam mengurangi jumlah perangkat lunak yang akan dikembangkan sehingga mengurangi biaya dan risiko. Biasanya juga mengarah pada pengiriman perangkat lunak yang lebih cepat. Namun, kompromi persyaratan tidak dapat dihindari dan ini dapat mengarah pada sistem yang tidak memenuhi kebutuhan nyata pengguna. Selain itu, beberapa kontrol atas evolusi sistem hilang karena versi baru dari komponen yang dapat digunakan kembali tidak di bawah kendali organisasi yang menggunakannya

## Process Activities

Proses perangkat lunak yang nyata adalah rangkaian kegiatan teknis, kolaboratif, dan manajerial yang saling terkait dengan tujuan keseluruhan untuk menentukan, merancang, mengimplementasikan, dan menguji sistem perangkat lunak. Pengembang perangkat lunak menggunakan berbagai alat perangkat lunak yang berbeda dalam pekerjaan mereka. Alat sangat berguna untuk mendukung pengeditan berbagai jenis dokumen dan untuk mengelola volume besar informasi terperinci yang dihasilkan dalam proyek perangkat lunak besar.

Empat kegiatan dasar proses specification, development, validation, dan evolution diatur secara berbeda dalam proses pengembangan yang berbeda. Dalam model Waterfall, mereka diatur secara berurutan, sedangkan dalam Incremental Development mereka disisipkan. Bagaimana kegiatan ini dilakukan tergantung pada jenis perangkat lunak, orang, dan struktur organisasi yang terlibat. Dalam pemrograman ekstrim, misalnya, spesifikasi ditulis di kartu. Tes dapat dieksekusi dan dikembangkan sebelum program itu sendiri. Evolusi dapat melibatkan restrukturisasi atau refactoring sistem yang substansial.

### Software Specification

Software Specification atau Spesifikasi perangkat lunak atau rekayasa persyaratan adalah proses memahami dan menentukan layanan apa yang diperlukan dari sistem dan mengidentifikasi kendala pada operasi dan pengembangan sistem. Rekayasa kebutuhan adalah tahap yang sangat penting dalam proses perangkat lunak karena kesalahan pada tahap ini tidak dapat dihindari akan menimbulkan masalah di kemudian hari dalam desain dan implementasi sistem.

Proses rekayasa persyaratan bertujuan untuk menghasilkan dokumen persyaratan yang disepakati yang menetapkan sistem yang memenuhi persyaratan pemangku kepentingan. Persyaratan biasanya disajikan pada dua tingkat detail. Pengguna akhir dan pelanggan membutuhkan pernyataan persyaratan tingkat tinggi; pengembang sistem memerlukan spesifikasi sistem yang lebih rinci.

Ada empat kegiatan utama dalam proses rekayasa persyaratan:

* 1. Studi kelayakan Perkiraan dibuat apakah kebutuhan pengguna yang diidentifikasi dapat dipenuhi menggunakan perangkat lunak dan teknologi perangkat keras saat ini. Studi ini mempertimbangkan apakah sistem yang diusulkan akan hemat biaya dari sudut pandang bisnis dan jika dapat dikembangkan dalam batasan anggaran yang ada. Studi kelayakan harus relatif murah dan cepat. Hasilnya harus menginformasikan keputusan apakah akan melanjutkan analisis yang lebih terperinci atau tidak.
  2. Persyaratan elisitasi dan analisis Ini adalah proses memperoleh persyaratan sistem melalui pengamatan sistem yang ada, diskusi dengan pengguna potensial dan pengadaan, analisis tugas, dan sebagainya. Ini mungkin melibatkan pengembangan satu atau lebih model sistem dan prototipe. Ini membantu Anda memahami sistem yang akan ditentukan.
  3. Spesifikasi persyaratan Spesifikasi persyaratan adalah kegiatan menerjemahkan informasi yang dikumpulkan selama kegiatan analisis ke dalam dokumen yang mendefinisikan serangkaian persyaratan. Dua jenis persyaratan dapat dimasukkan dalam dokumen ini. Persyaratan pengguna adalah pernyataan abstrak dari persyaratan sistem untuk pelanggan dan pengguna akhir sistem; persyaratan sistem adalah deskripsi yang lebih rinci tentang fungsi yang akan disediakan.
  4. Validasi persyaratan. Kegiatan ini memeriksa persyaratan untuk realisme, konsistensi, dan kelengkapan. Selama proses ini, kesalahan dalam dokumen persyaratan pasti ditemukan. Kemudian harus dimodifikasi untuk memperbaiki masalah ini.

Tentu saja, kegiatan dalam proses persyaratan tidak hanya dilakukan dalam urutan yang ketat. Analisis persyaratan berlanjut selama definisi dan spesifikasi dan persyaratan baru terungkap sepanjang proses. Oleh karena itu, kegiatan analisis, definisi, dan spesifikasi saling terkait. Dalam metode tangkas, seperti pemrograman ekstrem, persyaratan dikembangkan secara bertahap sesuai dengan prioritas pengguna dan pemunculan persyaratan berasal dari pengguna yang merupakan bagian dari tim pengembangan.

### Software Design and Implementation

Tahap implementasi pengembangan perangkat lunak adalah proses mengubah spesifikasi sistem menjadi sistem yang dapat dieksekusi. Itu selalu melibatkan proses desain perangkat lunak dan pemrograman tetapi, jika pendekatan tambahan untuk pengembangan digunakan, mungkin juga melibatkan penyempurnaan spesifikasi perangkat lunak.

Desain perangkat lunak adalah deskripsi struktur perangkat lunak yang akan diimplementasikan, model data dan struktur yang digunakan oleh sistem, antarmuka antara komponen sistem dan, kadang-kadang, algoritma yang digunakan. Desainer tidak segera sampai pada desain yang sudah jadi tetapi mengembangkan desain secara iteratif. Mereka menambahkan formalitas dan detail saat mereka mengembangkan desain mereka dengan backtracking terus-menerus untuk memperbaiki desain sebelumnya.

Sebagian besar antarmuka perangkat lunak dengan sistem perangkat lunak lain. Ini termasuk sistem operasi, basis data, middleware, dan sistem aplikasi lainnya. Ini membentuk 'platform perangkat lunak', lingkungan di mana perangkat lunak akan dieksekusi. Informasi tentang platform ini merupakan input penting untuk proses desain, karena desainer harus memutuskan cara terbaik untuk mengintegrasikannya dengan lingkungan perangkat lunak. Spesifikasi persyaratan adalah deskripsi fungsionalitas yang harus disediakan oleh perangkat lunak serta persyaratan kinerja dan keandalannya. Jika sistem memproses data yang ada, maka deskripsi data tersebut dapat dimasukkan dalam spesifikasi platform; jika tidak, deskripsi data harus menjadi input untuk proses desain sehingga organisasi data sistem harus didefinisikan.

Aktivitas dalam proses desain bervariasi, tergantung pada jenis sistem yang dikembangkan. Sebagai contoh, sistem real-time membutuhkan desain waktu tetapi mungkin tidak termasuk database sehingga tidak ada desain database yang terlibat. bagian dari proses desain untuk sistem informasi:

1. Desain arsitektur, di mana Anda mengidentifikasi keseluruhan struktur sistem, komponen utama (kadang-kadang disebut sub-sistem atau modul), hubungan mereka, dan bagaimana mereka didistribusikan.
2. Desain antarmuka, tempat Anda menentukan antarmuka antara komponen sistem. Spesifikasi antarmuka ini harus jelas. Dengan antarmuka yang tepat, suatu komponen dapat digunakan tanpa komponen lain harus tahu cara penerapannya. Setelah spesifikasi antarmuka disetujui, komponen dapat dirancang dan dikembangkan secara bersamaan
3. Desain komponen, di mana Anda mengambil setiap komponen sistem dan desain bagaimana itu akan beroperasi. Ini mungkin pernyataan sederhana dari fungsionalitas yang diharapkan untuk diimplementasikan, dengan desain khusus diserahkan kepada programmer. Atau, itu mungkin daftar perubahan yang harus dilakukan untuk komponen yang dapat digunakan kembali atau model desain terperinci. Model desain dapat digunakan untuk secara otomatis menghasilkan implementasi.
4. Perancangan basis data, tempat Anda mendesain struktur data sistem dan bagaimana ini diwakili dalam basis data. Sekali lagi, pekerjaan di sini tergantung pada apakah database yang ada akan digunakan kembali atau database baru akan dibuat. Kegiatan-kegiatan ini mengarah pada serangkaian output desain, yang juga ditunjukkan pada Gambar 2.5. Detail dan representasi ini sangat bervariasi. Untuk sistem kritis, dokumen desain terperinci yang menguraikan deskripsi sistem secara tepat dan akurat harus dihasilkan. Jika pendekatan berbasis model digunakan, output ini sebagian besar mungkin berupa diagram. Di mana metode pengembangan tangkas digunakan, output dari proses desain mungkin bukan dokumen spesifikasi yang terpisah tetapi dapat diwakili dalam kode program.

Metode terstruktur untuk desain dikembangkan pada 1970-an dan 1980-an dan merupakan pendahulu untuk UML dan desain berorientasi objek (Budgen, 2003). Mereka mengandalkan pembuatan model grafis sistem dan, dalam banyak kasus, secara otomatis menghasilkan kode dari model ini. Model-driven development (MDD) atau model-driven engineering (Schmidt, 2006), di mana model perangkat lunak dibuat pada berbagai tingkat abstraksi, merupakan evolusi dari metode terstruktur. Dalam MDD, ada penekanan yang lebih besar pada model arsitektur dengan pemisahan antara implementasi abstrak model independen dan model implementasi spesifik. Model dikembangkan dalam detail yang cukup sehingga sistem yang dapat dieksekusi dapat dihasilkan dari mereka.

Pengembangan program untuk mengimplementasikan sistem mengikuti secara alami dari proses desain sistem. Meskipun beberapa kelas program, seperti sistem yang kritis terhadap keselamatan, biasanya dirancang secara terperinci sebelum implementasi dimulai, itu lebih umum untuk tahap selanjutnya dari desain dan pengembangan program untuk disatukan. Alat pengembangan perangkat lunak dapat digunakan untuk menghasilkan program kerangka dari suatu desain. Ini termasuk kode untuk mendefinisikan dan mengimplementasikan antarmuka, dan, dalam banyak kasus, pengembang hanya perlu menambahkan rincian operasi setiap komponen program.

Pemrograman adalah kegiatan pribadi dan tidak ada proses umum yang biasanya diikuti. Beberapa programmer mulai dengan komponen yang mereka pahami, kembangkan, dan kemudian beralih ke komponen yang kurang dipahami. Yang lain mengambil pendekatan yang berlawanan, membiarkan komponen-komponen yang dikenal sampai akhir karena mereka tahu bagaimana mengembangkannya. Beberapa pengembang suka mendefinisikan data di awal proses kemudian menggunakannya untuk mendorong pengembangan program; yang lain meninggalkan data yang tidak ditentukan selama mungkin.

Biasanya, programmer melakukan beberapa pengujian terhadap kode yang telah mereka kembangkan. Ini sering mengungkapkan cacat program yang harus dihapus dari program. Ini disebut debugging. Pengujian dan debugging yang cacat adalah proses yang berbeda. Pengujian menetapkan adanya cacat. Debugging berkaitan dengan mencari dan memperbaiki cacat ini.

Ketika Anda debugging, Anda harus membuat hipotesis tentang perilaku yang dapat diamati dari program kemudian menguji hipotesis ini dengan harapan menemukan kesalahan yang menyebabkan anomali output. Menguji hipotesis mungkin melibatkan melacak kode program secara manual. Mungkin memerlukan kasus uji baru untuk melokalisasi masalah. Alat debugging interaktif, yang menunjukkan nilai tengah variabel program dan jejak pernyataan yang dieksekusi, dapat digunakan untuk mendukung proses debugging.

### Software Validation

Software Validation atau Validasi perangkat lunak atau, lebih umum, verifikasi dan validasi (V&V) dimaksudkan untuk menunjukkan bahwa suatu sistem memenuhi spesifikasi dan memenuhi harapan pelanggan sistem. Pengujian program, di mana sistem dijalankan menggunakan data uji simulasi, adalah teknik validasi utama. Validasi juga dapat melibatkan proses pemeriksaan, seperti inspeksi dan ulasan, pada setiap tahap proses perangkat lunak dari definisi kebutuhan pengguna hingga pengembangan program. Karena dominasi pengujian, sebagian besar biaya validasi dikeluarkan selama dan setelah implementasi.

Kecuali untuk program kecil, sistem tidak boleh diuji sebagai unit tunggal, monolitik. proses pengujian tiga tahap di mana komponen sistem diuji kemudian sistem terintegrasi diuji dan, akhirnya, sistem diuji dengan data pelanggan. Idealnya, kerusakan komponen ditemukan pada awal proses, dan masalah antarmuka ditemukan saat sistem terintegrasi. Namun, saat ditemukan kerusakan, program harus di-debug dan ini mungkin memerlukan tahapan lain dalam proses pengujian untuk diulang. Kesalahan dalam komponen program, misalnya, mungkin terungkap selama pengujian sistem. Oleh karena itu proses ini merupakan proses berulang dengan informasi diumpan balikkan dari tahap selanjutnya ke bagian proses sebelumnya.

Tahapan dalam proses pengujian adalah:

* 1. Pengujian pengembangan Komponen yang membentuk sistem diuji oleh orang-orang yang mengembangkan sistem. Setiap komponen diuji secara independen, tanpa komponen sistem lainnya. Komponen dapat berupa entitas sederhana seperti fungsi atau kelas objek, atau mungkin pengelompokan yang koheren dari entitas ini. Alat otomatisasi pengujian, seperti JUnit (Massol dan Husted, 2003), yang dapat menjalankan kembali pengujian komponen ketika versi baru komponen dibuat, biasanya digunakan.
  2. Pengujian sistem Komponen sistem terintegrasi untuk membuat sistem yang lengkap. Proses ini berkaitan dengan menemukan kesalahan yang dihasilkan dari interaksi yang tidak terduga antara komponen dan masalah antarmuka komponen. Ini juga berkaitan dengan menunjukkan bahwa sistem memenuhi persyaratan fungsional dan non-fungsional, dan menguji sifat-sifat sistem yang muncul. Untuk sistem besar, ini mungkin merupakan proses multi-tahap di mana komponen diintegrasikan untuk membentuk subsistem yang diuji secara individual sebelum sub-sistem ini sendiri terintegrasi untuk membentuk sistem akhir.
  3. Pengujian penerimaan. Ini adalah tahap terakhir dalam proses pengujian sebelum sistem diterima untuk penggunaan operasional. Sistem diuji dengan data yang diberikan oleh pelanggan sistem daripada dengan data uji simulasi. Pengujian penerimaan dapat mengungkapkan kesalahan dan kelalaian dalam definisi persyaratan sistem, karena data nyata menggunakan sistem dengan cara yang berbeda dari data pengujian. Pengujian penerimaan juga dapat mengungkapkan masalah persyaratan di mana fasilitas sistem tidak benar-benar memenuhi kebutuhan pengguna atau kinerja sistem tidak dapat diterima.

Biasanya, pengembangan komponen dan proses pengujian saling terkait. Pemrogram membuat data pengujian mereka sendiri dan secara bertahap menguji kode saat dikembangkan. Ini adalah pendekatan yang masuk akal secara ekonomi, karena programmer mengetahui komponen dan oleh karena itu orang yang terbaik untuk menghasilkan kasus uji.

Jika pendekatan inkremental untuk pengembangan digunakan, setiap kenaikan harus diuji saat dikembangkan, dengan tes ini didasarkan pada persyaratan untuk kenaikan itu. Dalam pemrograman ekstrim, tes dikembangkan bersama dengan persyaratan sebelum pengembangan dimulai. Ini membantu penguji dan pengembang untuk memahami persyaratan dan memastikan bahwa tidak ada keterlambatan saat kasus uji dibuat.

Saat proses perangkat lunak yang digerakkan oleh rencana digunakan (mis., Untuk pengembangan sistem kritis), pengujian didorong oleh serangkaian paket pengujian. Tim penguji independen bekerja dari rencana pengujian pra-formulasi ini, yang telah dikembangkan dari spesifikasi dan desain sistem.

Tes penerimaan kadang-kadang disebut 'pengujian alfa'. Sistem khusus dikembangkan untuk satu klien. Proses pengujian alfa berlanjut sampai pengembang sistem dan klien setuju bahwa sistem yang disampaikan adalah implementasi yang dapat diterima dari persyaratan.

Ketika suatu sistem akan dipasarkan sebagai produk perangkat lunak, proses pengujian yang disebut 'pengujian beta' sering digunakan. Pengujian beta melibatkan pengiriman sistem ke sejumlah pelanggan potensial yang setuju untuk menggunakan sistem itu. Mereka melaporkan masalah kepada pengembang sistem. Ini memperlihatkan produk untuk penggunaan nyata dan mendeteksi kesalahan yang mungkin tidak diantisipasi oleh pembangun sistem. Setelah umpan balik ini, sistem dimodifikasi dan dirilis baik untuk pengujian beta lebih lanjut atau untuk penjualan umum.

### Software Evolution

Fleksibilitas sistem perangkat lunak adalah salah satu alasan utama mengapa semakin banyak perangkat lunak dimasukkan dalam sistem yang besar dan kompleks. Begitu keputusan telah dibuat untuk memproduksi perangkat keras, sangat mahal untuk melakukan perubahan pada desain perangkat keras. Namun, perubahan dapat dilakukan ke perangkat lunak kapan saja selama atau setelah pengembangan sistem. Bahkan perubahan ekstensif masih jauh lebih murah daripada perubahan yang sesuai dengan perangkat keras sistem.

Secara historis, selalu ada pemisahan antara proses pengembangan perangkat lunak dan proses evolusi perangkat lunak (pemeliharaan perangkat lunak). Orang menganggap pengembangan perangkat lunak sebagai aktivitas kreatif di mana sistem perangkat lunak dikembangkan dari konsep awal hingga sistem kerja. Namun, mereka terkadang menganggap pemeliharaan perangkat lunak sebagai sesuatu yang membosankan dan tidak menarik. Meskipun biaya pemeliharaan seringkali beberapa kali lipat dari biaya pengembangan awal, proses pemeliharaan kadang-kadang dianggap kurang menantang daripada pengembangan perangkat lunak asli.

Perbedaan antara pengembangan dan pemeliharaan ini semakin tidak relevan. Hampir tidak ada sistem perangkat lunak yang sepenuhnya merupakan sistem baru dan jauh lebih masuk akal untuk melihat pengembangan dan pemeliharaan sebagai sebuah rangkaian. Daripada dua proses terpisah, lebih realistis untuk berpikir tentang rekayasa perangkat lunak sebagai proses evolusi di mana perangkat lunak terus berubah sepanjang masa hidupnya dalam menanggapi perubahan kebutuhan dan kebutuhan pelanggan.

## Requirements Engineering

Persyaratan untuk suatu sistem adalah uraian tentang apa yang harus dilakukan oleh sistem — layanan yang disediakannya dan kendala dalam operasinya. Persyaratan ini mencerminkan kebutuhan pelanggan untuk sistem yang melayani tujuan tertentu seperti mengendalikan perangkat, melakukan pemesanan, atau mencari informasi. Proses mencari tahu, menganalisis, mendokumentasikan, dan memeriksa layanan dan kendala ini disebut persyaratan teknik (RE).

Istilah 'persyaratan' tidak digunakan secara konsisten dalam industri perangkat lunak. Dalam beberapa kasus, persyaratan hanyalah pernyataan abstrak tingkat tinggi dari suatu layanan yang harus disediakan oleh suatu sistem atau kendala pada suatu sistem. Pada ekstrem yang lain, ini adalah definisi formal dan terperinci dari fungsi sistem. Davis (1993) menjelaskan mengapa perbedaan ini ada:

 Jika sebuah perusahaan ingin membiarkan sebuah kontrak untuk proyek pengembangan perangkat lunak besar, ia harus mendefinisikan kebutuhannya secara abstrak sehingga solusi tidak ditentukan sebelumnya. Persyaratan harus ditulis sehingga beberapa kontraktor dapat mengajukan penawaran untuk kontrak, menawarkan, mungkin, berbagai cara untuk memenuhi kebutuhan organisasi klien. Setelah kontrak diberikan, kontraktor harus menulis definisi sistem untuk klien secara lebih rinci sehingga klien memahami dan dapat memvalidasi apa yang akan dilakukan perangkat lunak. Kedua dokumen ini dapat disebut dokumen persyaratan untuk sistem.

Beberapa masalah yang muncul selama proses rekayasa persyaratan adalah hasil dari kegagalan untuk membuat pemisahan yang jelas antara berbagai tingkatan deskripsi. Saya membedakan antara mereka dengan menggunakan istilah 'persyaratan pengguna' yang berarti persyaratan abstrak tingkat tinggi dan 'persyaratan sistem' berarti deskripsi terperinci tentang apa yang harus dilakukan sistem. Persyaratan pengguna dan persyaratan sistem dapat didefinisikan sebagai berikut:

1. Persyaratan pengguna adalah pernyataan, dalam bahasa alami plus diagram, tentang layanan apa yang diharapkan disediakan sistem untuk pengguna sistem dan kendala yang harus digunakannya.

2. Persyaratan sistem adalah uraian lebih rinci tentang fungsi, layanan, dan kendala operasional sistem perangkat lunak. Dokumen persyaratan sistem (kadang-kadang disebut spesifikasi fungsional) harus menentukan dengan tepat apa yang harus diimplementasikan. Ini mungkin merupakan bagian dari kontrak antara pembeli sistem dan pengembang perangkat lunak.

Persyaratan fungsional dan non-fungsional

Persyaratan sistem perangkat lunak sering diklasifikasikan sebagai persyaratan fungsional atau persyaratan nonfungsional:

1. Persyaratan fungsional Ini adalah pernyataan layanan yang harus disediakan oleh sistem, bagaimana sistem harus bereaksi terhadap input tertentu, dan bagaimana sistem harus berperilaku dalam situasi tertentu. Dalam beberapa kasus, persyaratan fungsional juga dapat secara eksplisit menyatakan apa yang seharusnya tidak dilakukan oleh sistem.

2. Persyaratan non-fungsional Ini adalah kendala pada layanan atau fungsi yang ditawarkan oleh sistem. Mereka termasuk kendala waktu, kendala pada proses pengembangan, dan kendala yang ditentukan oleh standar. Persyaratan non-fungsional sering berlaku untuk sistem secara keseluruhan, daripada fitur atau layanan sistem individual.

* + 1. Functional and Non-Functional Requirements

Persyaratan Fungsional

Persyaratan fungsional untuk sistem menggambarkan apa yang harus dilakukan sistem. Persyaratan ini tergantung pada jenis perangkat lunak yang dikembangkan, pengguna perangkat lunak yang diharapkan, dan pendekatan umum yang diambil oleh organisasi ketika menulis persyaratan. Ketika dinyatakan sebagai kebutuhan pengguna, persyaratan fungsional biasanya dijelaskan secara abstrak yang dapat dipahami oleh pengguna sistem. Namun, persyaratan sistem fungsional yang lebih spesifik menggambarkan fungsi sistem, input dan outputnya, pengecualian, dll., Secara rinci.

Persyaratan sistem fungsional bervariasi dari persyaratan umum yang mencakup apa yang harus dilakukan sistem terhadap persyaratan yang sangat spesifik yang mencerminkan cara kerja lokal atau sistem yang ada di organisasi.

Persyaratan Non Fungsional

Persyaratan non-fungsional, seperti namanya, adalah persyaratan yang tidak berkaitan langsung dengan layanan spesifik yang diberikan oleh sistem kepada penggunanya. Mereka mungkin berhubungan dengan sifat-sifat sistem yang muncul seperti keandalan, waktu respons, dan hunian toko. Atau, mereka dapat mendefinisikan kendala pada implementasi sistem seperti kemampuan perangkat I / O atau representasi data yang digunakan dalam antarmuka dengan sistem lain. Persyaratan non-fungsional, seperti kinerja, keamanan, atau ketersediaan, biasanya menentukan atau membatasi karakteristik sistem secara keseluruhan.

Persyaratan non-fungsional seringkali lebih penting daripada persyaratan fungsional individu. Pengguna sistem biasanya dapat menemukan cara untuk mengatasi fungsi sistem yang tidak benar-benar memenuhi kebutuhan mereka. Namun, gagal memenuhi persyaratan non-fungsional dapat berarti bahwa keseluruhan sistem tidak dapat digunakan. Misalnya, jika sistem pesawat terbang tidak memenuhi persyaratan keandalannya, maka sistem tersebut tidak akan disertifikasi aman untuk dioperasikan; jika sistem kontrol tertanam gagal memenuhi persyaratan kinerjanya, fungsi kontrol tidak akan beroperasi dengan benar.

Meskipun sering kali mungkin untuk mengidentifikasi komponen sistem mana yang menerapkan persyaratan fungsional spesifik (mis., Mungkin ada komponen pemformatan yang menerapkan persyaratan pelaporan), seringkali lebih sulit untuk menghubungkan komponen dengan persyaratan non-fungsional. Implementasi persyaratan ini dapat disebarkan ke seluruh sistem. Ada dua alasan untuk ini:

1. Persyaratan non-fungsional dapat mempengaruhi keseluruhan arsitektur sistem daripada komponen individu. Misalnya, untuk memastikan bahwa persyaratan kinerja terpenuhi, Anda mungkin harus mengatur sistem untuk meminimalkan komunikasi antar komponen.

2. Persyaratan non-fungsional tunggal, seperti persyaratan keamanan, dapat menghasilkan sejumlah persyaratan fungsional terkait yang menentukan layanan sistem baru yang diperlukan. Selain itu, dapat juga menghasilkan persyaratan yang membatasi persyaratan yang ada.

* + 1. The Software Requirements Document

Dokumen persyaratan perangkat lunak (kadang-kadang disebut spesifikasi persyaratan perangkat lunak atau SRS) adalah pernyataan resmi tentang apa yang harus diterapkan oleh pengembang sistem. Ini harus mencakup persyaratan pengguna untuk suatu sistem dan spesifikasi terperinci dari persyaratan sistem. Terkadang, persyaratan pengguna dan sistem diintegrasikan ke dalam satu deskripsi. Dalam kasus lain, persyaratan pengguna didefinisikan dalam pengantar spesifikasi persyaratan sistem. Jika ada sejumlah besar persyaratan, persyaratan sistem terperinci dapat disajikan dalam dokumen terpisah.

Dokumen persyaratan sangat penting ketika kontraktor luar mengembangkan sistem perangkat lunak. Namun, metode pengembangan gesit berpendapat bahwa persyaratan berubah begitu cepat sehingga dokumen persyaratan sudah kedaluwarsa begitu ditulis, sehingga sebagian besar upaya sia-sia. Daripada dokumen formal, pendekatan seperti Extreme Programming (Beck, 1999) mengumpulkan persyaratan pengguna secara bertahap dan menuliskannya pada kartu sebagai cerita pengguna. Pengguna kemudian memprioritaskan persyaratan untuk implementasi dalam peningkatan sistem selanjutnya.

Untuk sistem bisnis di mana persyaratannya tidak stabil, saya pikir pendekatan ini bagus. Namun, saya pikir masih berguna untuk menulis dokumen pendukung singkat yang mendefinisikan bisnis dan persyaratan ketergantungan untuk sistem; mudah untuk melupakan persyaratan yang berlaku untuk sistem secara keseluruhan ketika berfokus pada persyaratan fungsional untuk rilis sistem berikutnya

* + 1. Requirements Specification

Spesifikasi persyaratan adalah proses penulisan persyaratan pengguna dan sistem dalam dokumen persyaratan. Idealnya, persyaratan pengguna dan sistem harus jelas, tidak ambigu, mudah dipahami, lengkap, dan konsisten. Dalam praktiknya, ini sulit untuk dicapai karena pemangku kepentingan menafsirkan persyaratan dengan cara yang berbeda dan sering ada konflik dan inkonsistensi yang melekat dalam persyaratan.

Persyaratan pengguna untuk suatu sistem harus menggambarkan persyaratan fungsional dan nonfungsional sehingga persyaratan tersebut dapat dipahami oleh pengguna sistem yang tidak memiliki pengetahuan teknis terperinci. Idealnya, mereka harus menentukan hanya perilaku eksternal sistem. Dokumen persyaratan tidak boleh menyertakan perincian arsitektur atau desain sistem. Akibatnya, jika Anda menulis persyaratan pengguna, Anda tidak boleh menggunakan jargon perangkat lunak, notasi terstruktur, atau notasi formal. Anda harus menulis persyaratan pengguna dalam bahasa alami, dengan tabel sederhana, formulir, dan diagram intuitif.

Persyaratan sistem adalah versi yang diperluas dari persyaratan pengguna yang digunakan oleh insinyur perangkat lunak sebagai titik awal untuk desain sistem. Mereka menambahkan detail dan menjelaskan bagaimana persyaratan pengguna harus disediakan oleh sistem. Mereka dapat digunakan sebagai bagian dari kontrak untuk implementasi sistem dan oleh karena itu harus menjadi spesifikasi lengkap dan terperinci dari keseluruhan sistem.

Idealnya, persyaratan sistem harus hanya menggambarkan perilaku eksternal sistem dan kendala operasionalnya. Mereka seharusnya tidak peduli dengan bagaimana sistem harus dirancang atau diimplementasikan. Namun, pada tingkat perincian yang diperlukan untuk sepenuhnya menentukan sistem perangkat lunak yang rumit, praktis tidak mungkin untuk mengecualikan semua informasi desain. Ada beberapa alasan untuk ini:

1. Anda mungkin harus merancang arsitektur awal sistem untuk membantu menyusun spesifikasi kebutuhan. Persyaratan sistem diatur sesuai dengan berbagai sub-sistem yang membentuk sistem.

2. Dalam kebanyakan kasus, sistem harus berinteroperasi dengan sistem yang ada, yang membatasi desain dan memaksakan persyaratan pada sistem baru.

3. Penggunaan arsitektur tertentu untuk memenuhi persyaratan non-fungsional mungkin diperlukan. Regulator eksternal yang perlu menyatakan bahwa sistem tersebut aman dapat menentukan bahwa desain arsitektur yang telah disertifikasi dapat digunakan.

Natural Specification Language

Bahasa alami telah digunakan untuk menulis persyaratan untuk perangkat lunak sejak awal rekayasa perangkat lunak. Ini ekspresif, intuitif, dan universal. Ini juga berpotensi kabur, ambigu, dan maknanya tergantung pada latar belakang pembaca. Akibatnya, ada banyak proposal untuk cara alternatif menulis persyaratan. Namun, tidak satu pun dari ini telah diadopsi secara luas dan bahasa alami akan terus menjadi cara yang paling banyak digunakan untuk menentukan persyaratan sistem dan perangkat lunak. Untuk meminimalkan kesalahpahaman saat menulis persyaratan bahasa alami, saya sarankan Anda mengikuti beberapa panduan sederhana:

Spesifikasi Terstruktur

Bahasa alami terstruktur adalah cara menulis persyaratan sistem di mana kebebasan persyaratan penulis dibatasi dan semua persyaratan ditulis dengan cara standar. Pendekatan ini mempertahankan sebagian besar ekspresi dan pemahaman bahasa alami tetapi memastikan bahwa beberapa keseragaman dikenakan pada spesifikasi. Notasi bahasa terstruktur menggunakan template untuk menentukan persyaratan sistem. Spesifikasi dapat menggunakan konstruksi bahasa pemrograman untuk menunjukkan alternatif dan iterasi, dan dapat menyoroti elemen kunci menggunakan font shading atau berbeda.

The Robertsons (Robertson dan Robertson, 1999), dalam buku mereka tentang metode rekayasa persyaratan VOLERE, merekomendasikan bahwa persyaratan pengguna pada awalnya ditulis pada kartu, satu persyaratan per kartu. Mereka menyarankan sejumlah bidang pada setiap kartu, seperti alasan persyaratan, ketergantungan pada persyaratan lain, sumber persyaratan, bahan pendukung, dan sebagainya.

Untuk menggunakan pendekatan terstruktur untuk menentukan persyaratan sistem, Anda mendefinisikan satu atau lebih templat standar untuk persyaratan dan mewakili templat ini sebagai formulir terstruktur. Spesifikasi dapat disusun di sekitar objek yang dimanipulasi oleh sistem, fungsi yang dilakukan oleh sistem, atau peristiwa yang diproses oleh sistem.

Ketika formulir standar digunakan untuk menentukan persyaratan fungsional, informasi berikut harus dimasukkan:

1. Deskripsi fungsi atau entitas yang ditentukan.

2. Deskripsi inputnya dan dari mana ini berasal.

3. Deskripsi outputnya dan kemana perginya.

4. Informasi tentang informasi yang diperlukan untuk perhitungan atau entitas lain dalam sistem yang digunakan (bagian ‘membutuhkan’).

5. Deskripsi tindakan yang akan diambil.

6. Jika pendekatan fungsional digunakan, pra-kondisi menetapkan apa yang harus benar sebelum fungsi dipanggil, dan post-kondisi menentukan apa yang benar setelah fungsi dipanggil.

7. Deskripsi efek samping (jika ada) dari operasi.

* + 1. Requirements Elicitation and Analysis

Dalam kegiatan ini, insinyur perangkat lunak bekerja dengan pelanggan dan pengguna akhir sistem untuk mencari tahu tentang domain aplikasi, layanan apa yang harus disediakan sistem, kinerja sistem yang diperlukan, kendala perangkat keras, dan sebagainya.

Elisitasi dan analisis persyaratan dapat melibatkan berbagai jenis orang dalam suatu organisasi. Stakeholder sistem adalah siapa saja yang memiliki pengaruh langsung atau tidak langsung terhadap persyaratan sistem. Stakeholder termasuk pengguna akhir yang akan berinteraksi dengan sistem dan siapa pun dalam organisasi yang akan terpengaruh olehnya. Pemangku kepentingan sistem lainnya mungkin adalah insinyur yang mengembangkan atau memelihara sistem terkait lainnya, manajer bisnis, pakar domain, dan perwakilan serikat pekerja.

Kegiatan proses adalah:

1. Penemuan kebutuhan. Ini adalah proses berinteraksi dengan pemangku kepentingan sistem untuk menemukan persyaratan mereka. Persyaratan domain dari pemangku kepentingan dan dokumentasi juga ditemukan selama kegiatan ini. Ada beberapa teknik pelengkap yang dapat digunakan untuk penemuan kebutuhan, yang akan saya bahas nanti di bagian ini.

2. Klasifikasi dan organisasi persyaratan Kegiatan ini mengambil kumpulan persyaratan yang tidak terstruktur, persyaratan terkait kelompok, dan mengorganisasikannya ke dalam kelompok yang koheren. Cara paling umum dari pengelompokan persyaratan adalah dengan menggunakan model arsitektur sistem untuk mengidentifikasi sub-sistem dan untuk mengaitkan persyaratan dengan masing-masing sub-sistem. Dalam praktiknya, persyaratan teknik dan desain arsitektur tidak dapat sepenuhnya merupakan kegiatan yang terpisah.

3. Prioritas persyaratan dan negosiasi. Tidak dapat dihindari, ketika banyak pemangku kepentingan terlibat, persyaratan akan bertentangan. Kegiatan ini berkaitan dengan memprioritaskan persyaratan dan menemukan dan menyelesaikan konflik persyaratan melalui negosiasi. Biasanya, para pemangku kepentingan harus bertemu untuk menyelesaikan perbedaan dan menyepakati persyaratan kompromi.

4. Spesifikasi persyaratan Persyaratan didokumentasikan dan dimasukkan ke dalam putaran spiral berikutnya.

Siklus proses dimulai dengan penemuan persyaratan dan berakhir dengan dokumentasi persyaratan. Pemahaman analis tentang persyaratan meningkat dengan setiap putaran siklus. Siklus berakhir ketika dokumen persyaratan selesai.

Menuntut dan memahami persyaratan dari pemangku kepentingan sistem adalah proses yang sulit karena beberapa alasan:

1. Stakeholder sering tidak tahu apa yang mereka inginkan dari sistem komputer kecuali dalam istilah yang paling umum; mereka mungkin merasa sulit untuk mengartikulasikan apa yang mereka ingin sistem lakukan; mereka mungkin membuat tuntutan yang tidak realistis karena mereka tidak tahu apa yang layak dan tidak layak.

2. Stakeholder dalam suatu sistem secara alami mengungkapkan persyaratan dalam istilah mereka sendiri dan dengan pengetahuan implisit tentang pekerjaan mereka sendiri. Insinyur persyaratan, tanpa pengalaman dalam domain pelanggan, mungkin tidak memahami persyaratan ini.

3. Pemangku kepentingan yang berbeda memiliki persyaratan yang berbeda dan mereka dapat mengekspresikannya dengan cara yang berbeda. Persyaratan Para insinyur harus menemukan semua sumber persyaratan potensial dan menemukan kesamaan dan konflik.

4. Faktor-faktor politik dapat mempengaruhi persyaratan suatu sistem. Manajer dapat menuntut persyaratan sistem tertentu karena ini akan memungkinkan mereka untuk meningkatkan pengaruh mereka dalam organisasi.

5. Lingkungan ekonomi dan bisnis tempat analisis berlangsung dinamis. Ini pasti akan berubah selama proses analisis. Pentingnya persyaratan tertentu dapat berubah. Persyaratan baru dapat muncul dari pemangku kepentingan baru yang awalnya tidak diajak berkonsultasi.

Tidak dapat dihindari, para pemangku kepentingan yang berbeda memiliki pandangan yang berbeda tentang pentingnya dan prioritas persyaratan dan, kadang-kadang, pandangan ini saling bertentangan. Selama proses, Anda harus mengatur negosiasi pemangku kepentingan secara teratur sehingga kompromi dapat dicapai. Mustahil untuk sepenuhnya memuaskan setiap pemangku kepentingan, tetapi jika beberapa pemangku kepentingan merasa bahwa pandangan mereka belum dipertimbangkan dengan baik, maka mereka mungkin dengan sengaja berupaya merusak proses ET.

Pada tahap spesifikasi persyaratan, persyaratan yang telah diperoleh sejauh ini didokumentasikan sedemikian rupa sehingga dapat digunakan untuk membantu dengan penemuan persyaratan. Pada tahap ini, versi awal dokumen persyaratan sistem dapat dibuat dengan bagian yang hilang dan persyaratan yang tidak lengkap. Atau, persyaratan dapat didokumentasikan dengan cara yang sama sekali berbeda (mis., Dalam spreadsheet atau pada kartu). Persyaratan penulisan pada kartu bisa sangat efektif karena mudah bagi pemegang saham untuk menangani, mengubah, dan mengatur.

* + 1. Requirements Validation

Validasi persyaratan adalah proses memeriksa bahwa persyaratan benar-benar menentukan sistem yang benar-benar diinginkan pelanggan. Ini tumpang tindih dengan analisis karena berkaitan dengan menemukan masalah dengan persyaratan. Validasi persyaratan penting karena kesalahan dalam dokumen persyaratan dapat menyebabkan biaya pengerjaan ulang yang luas ketika masalah ini ditemukan selama pengembangan atau setelah sistem dalam layanan.

Biaya memperbaiki masalah persyaratan dengan membuat perubahan sistem biasanya jauh lebih besar daripada memperbaiki kesalahan desain atau pengkodean. Alasan untuk ini adalah bahwa perubahan pada persyaratan biasanya berarti bahwa desain dan implementasi sistem juga harus diubah. Selanjutnya sistem kemudian harus diuji ulang.

Selama proses validasi persyaratan, berbagai jenis pemeriksaan harus dilakukan pada persyaratan dalam dokumen persyaratan. Pemeriksaan ini meliputi:

1. Pemeriksaan validitas Seorang pengguna mungkin berpikir bahwa suatu sistem diperlukan untuk melakukan fungsi-fungsi tertentu. Namun, pemikiran dan analisis lebih lanjut dapat mengidentifikasi fungsi tambahan atau berbeda yang diperlukan. Sistem memiliki pemangku kepentingan yang beragam dengan kebutuhan yang berbeda dan serangkaian persyaratan tidak dapat dihindari adalah kompromi di seluruh komunitas pemangku kepentingan.

2. Pemeriksaan Konsistensi Persyaratan dalam dokumen tidak boleh bertentangan. Artinya, seharusnya tidak ada kendala yang bertentangan atau deskripsi yang berbeda dari fungsi sistem yang sama.

3. Pemeriksaan kelengkapan Dokumen persyaratan harus mencakup persyaratan yang mendefinisikan semua fungsi dan batasan yang dimaksudkan oleh pengguna sistem.

4. Pemeriksaan realisme Dengan menggunakan pengetahuan teknologi yang ada, persyaratan harus diperiksa untuk memastikan bahwa mereka benar-benar dapat diimplementasikan. Pemeriksaan ini juga harus memperhitungkan anggaran dan jadwal untuk pengembangan sistem.

5. Dapat diverifikasi Untuk mengurangi potensi perselisihan antara pelanggan dan kontraktor, persyaratan sistem harus selalu ditulis sehingga dapat diverifikasi. Ini berarti bahwa Anda harus dapat menulis serangkaian tes yang dapat menunjukkan bahwa sistem yang dikirimkan memenuhi setiap persyaratan yang ditentukan.

Ada sejumlah teknik validasi persyaratan yang dapat digunakan secara individu atau bersama-sama:

1. Tinjauan persyaratan Persyaratan dianalisis secara sistematis oleh tim peninjau yang memeriksa kesalahan dan ketidakkonsistenan.

2. Prototyping Dalam pendekatan validasi ini, model sistem yang dapat dieksekusi yang diperlihatkan diperlihatkan kepada pengguna akhir dan pelanggan. Mereka dapat bereksperimen dengan model ini untuk melihat apakah itu memenuhi kebutuhan nyata mereka.

3. Persyaratan pembuatan kasus uji harus dapat diuji. Jika tes untuk persyaratan dirancang sebagai bagian dari proses validasi, ini sering mengungkapkan masalah persyaratan. Jika suatu tes sulit atau tidak mungkin untuk dirancang, ini biasanya berarti bahwa persyaratan akan sulit untuk diterapkan dan harus dipertimbangkan kembali. Mengembangkan tes dari persyaratan pengguna sebelum kode apa pun ditulis adalah bagian integral dari pemrograman ekstrem.

Anda tidak boleh meremehkan masalah yang terlibat dalam validasi persyaratan. Pada akhirnya, sulit untuk menunjukkan bahwa seperangkat persyaratan memang memenuhi kebutuhan pengguna. Pengguna perlu menggambarkan sistem dalam operasi dan membayangkan bagaimana sistem itu akan cocok dengan pekerjaan mereka. Sulit bahkan bagi para profesional komputer yang terampil untuk melakukan jenis analisis abstrak ini dan lebih sulit lagi bagi pengguna sistem. Akibatnya, Anda jarang menemukan semua masalah persyaratan selama proses validasi persyaratan. Tidak dapat dihindari bahwa akan ada perubahan persyaratan lebih lanjut untuk memperbaiki kelalaian dan kesalahpahaman setelah dokumen persyaratan telah disepakati.

* + 1. Requirements Management

Persyaratan untuk sistem perangkat lunak yang besar selalu berubah. Salah satu alasannya adalah bahwa sistem ini biasanya dikembangkan untuk mengatasi masalah 'jahat' — masalah yang tidak dapat sepenuhnya didefinisikan. Karena masalah tidak dapat sepenuhnya ditentukan, persyaratan perangkat lunak pasti tidak lengkap. Selama proses perangkat lunak, pemahaman pemangku kepentingan tentang masalah terus berubah. Persyaratan sistem kemudian harus juga berkembang untuk mencerminkan pandangan masalah yang berubah ini.

Setelah suatu sistem telah diinstal dan digunakan secara teratur, persyaratan baru pasti muncul. Sulit bagi pengguna dan pelanggan sistem untuk mengantisipasi apa efek sistem baru akan memiliki pada proses bisnis mereka dan cara kerja dilakukan. Setelah pengguna akhir memiliki pengalaman sistem, mereka akan menemukan kebutuhan dan prioritas baru. Ada beberapa alasan mengapa perubahan tidak bisa dihindari:

1. Lingkungan bisnis dan teknis sistem selalu berubah setelah instalasi. Perangkat keras baru dapat diperkenalkan, mungkin perlu untuk menghubungkan sistem dengan sistem lain, prioritas bisnis dapat berubah (dengan konsekuensi perubahan dalam dukungan sistem diperlukan), dan undang-undang dan peraturan baru mungkin diperkenalkan bahwa sistem harus mematuhi.

2. Orang-orang yang membayar suatu sistem dan pengguna sistem itu jarang orang yang sama. Pelanggan sistem memberlakukan persyaratan karena kendala organisasi dan anggaran. Ini mungkin bertentangan dengan persyaratan pengguna akhir dan, setelah pengiriman, fitur baru mungkin harus ditambahkan untuk dukungan pengguna jika sistem ingin mencapai tujuannya.

3. Sistem besar biasanya memiliki komunitas pengguna yang beragam, dengan banyak pengguna memiliki persyaratan dan prioritas yang berbeda yang mungkin saling bertentangan atau bertentangan. Persyaratan sistem terakhir adalah kompromi di antara mereka dan, dengan pengalaman, sering ditemukan bahwa keseimbangan dukungan yang diberikan kepada pengguna yang berbeda harus diubah.

Manajemen persyaratan adalah proses memahami dan mengendalikan perubahan terhadap persyaratan sistem. Anda perlu melacak persyaratan individu dan menjaga hubungan antara persyaratan yang tergantung sehingga Anda dapat menilai dampak perubahan persyaratan. Anda perlu membuat proses formal untuk membuat proposal perubahan dan menghubungkannya dengan persyaratan sistem. Proses formal manajemen persyaratan harus dimulai segera setelah versi konsep dokumen persyaratan tersedia. Namun, Anda harus mulai merencanakan cara mengelola perubahan kebutuhan selama proses elisitasi persyaratan.

## Android

Definisi

Bahasa Pemrograman

Tools

## 