BAB 2

LANDASAN TEORI

2.1. Multimedia

2.1.1. Pengertian Multimedia

Multimedia adalah sebuah kombinasi yang saling berkaitan dari teks, foto dan gambar, suara, animasi, dan video yang dimanipulasi secara *digital* (Vaughan, 2011, p1). Sedangkan menurut Hofstetter (2001, p2), multimedia adalah implementasi penggunaan komputer dalam menyajikan dan menggabungkan teks, gambar, suara, dan video dengan *interface* yang mengizinkan pengguna untuk berinteraksi dengan sistem multimedia tersebut.

2.1.2. Elemen Multimedia

Lima Elemen utama multimedia menurut Vaughan (2011, p1) adalah:

1. Teks

Teks sudah digunakan selama ribuan tahun oleh manusia untuk berkomunikasi. Tetapi sebuah kata dapat memiliki banyak arti, sehingga kata-kata yang digunakan haruslah singkat, padat, dan tepat sehingga pesan dan data dapat disampaikan dengan baik. Teks umumnya digunakan untuk merancang judul, menu, dan *buttons* (Vaughan, 2011, p20).

Ada hal-hal penting yang perlu diperhatikan oleh perancang multimedia dalam menggunakan teks menurut Dastbaz (2003, p56) yaitu:

- Penggunaan *font* yang berbeda akan tampak berbeda di beberapa *platform*
- Penggunaan tipe *font* yang khusus membutuhkan pengaturan tertentu pada mesin *user*
- Dibutuhkan keseimbangan yang tepat antara ukuran teks, warna, dan efek khusus seperti *anti-alias*, dimana teks akan secara halus menyatu dengan *background* nya

2. Suara

Penggunaan suara dalam multimedia dapat menghasilkan sebuah perbedaan dari presentasi multimedia yang biasa dengan presentasi multimedia yang *professional*. Walaupun begitu, penggunaan suara yang tidak pada tempatnya dapat merusak presentasi tersebut. (Vaughan, 2011, p104). Ada dua macam suara yang biasa digunakan di dalam multimedia, yaitu:

- *Digital Audio*

  Digital audio adalah hasil konversi dari gelombang suara yang disimpan ke dalam informasi berbentuk *bits* atau *bytes*. Proses konversi ini disebut *digitizing*. Kualitas dari hasil *digitizing* ini bergantung pada seberapa sering sampel yang diambil atau disebut juga *sampling rate* dan berapa banyak angka yang

digunakan untuk merepresentasikan tiap-tiap sampel, atau disebut juga dengan *bitdepth* (Vaughan, 2011, p106).

- MIDI

  MIDI adalah singkatan dari *Musical Instrument Digital Interface*, merupakan jenis suara yang paling mudah diimplementasikan ke dalam sebuah multimedia. MIDI sendiri adalah bentuk konversi dari suara yang disimpan ke dalam bentuk numerik (Vaughan, 2011, p134).

3. Gambar

Ada dua jenis gambar yang dapat dihasilkan oleh komputer menurut Vaughan (2011, p70), yaitu:

- *Bitmap* yaitu sebuah gambar yang dibentuk dari sebuah matriks yang terdiri dari titik-titik warna. Variasi warna di dalam gambar *bitmap* ditentukan dengan bit yang ditampilkan, dimana *n-bit* gambar *bitmap* memiliki 2^n macam warna (Vaughan, 2011, pp71-72).
- *Vector drawing* adalah gambar yang dihasilkan dari perhitungan koordinat Carthesian oleh komputer yang biasanya digunakan untuk menghasilkan bentuk garis, persegi, lingkaran, *oval*, dan *polygon* (Vaughan, 2011, p80).

Menurut Dastbaz (2003, p58), secara umum gambar dibagi menjadi tiga kelompok, yaitu:

- *Colour graphics*: gambar-gambar yang merepresentasikan warna dalam bentuk bit

- *Gray Scale graphics*: gambar yang terdiri dari warna-warna di antara warna hitam dan putih yang direpresentasikan ke dalam berbagai tingkat kedalaman warna
- *Mono graphics*: gambar yang hanya mengandung warna hitam dan putih saja.

4. Video

Penggunaan video di dalam sebuah presentasi multimedia dapat menjadi sebuah media penyampaian pesan maupun informasi yang sangat efektif. Dalam sebuah proyek multimedia, penggunaan video dapat meningkatkan penyampaian pesan kepada pengguna secara efektif dan pengguna akan lebih mengingat apa yang telah mereka saksikan (Vaughan, 2011, p164). Video sendiri dapat didefinisikan sebagai penggabungan yang halus dari gambar yang bergerak dan suara (Dastbaz, 2003, p62).

5. Animasi

Animasi merupakan sumber utama dari sebuah aksi multimedia yang dinamis di dalam sebuah presentasi multimedia. Animasi sering digunakan untuk mempresentasikan sesuatu yang tidak terlalu banyak memerlukan interaksi penggunanya sehingga presentasi tersebut akan mengalir berjalan seperti sebuah film. Animasi juga digunakan dalam membantu sebuah presentasi, seperti efek transisi *slide* dan lainnya (Vaughan, 2011, p140).

Ada tiga bentuk animasi yang dijelaskan oleh Vaughan (2011, pp142-143), yaitu

- Animasi 2D adalah animasi yang paling mudah dibuat, dimana hanya menggunakan dua dimensi saja yaitu sumbu x dan y pada sumbu Carthesian.
- Animasi 2½ D adalah animasi 2D yang diberikan tambahan sebuah ilusi sumbu z dengan cara menambahkan efek bayangan pada gambar, tetapi secara keseluruhan gambar itu sendiri tetap pada bidang datar dua dimensi.
- Animasi 3D adalah bentuk ruang virtual yang memiliki 3 dimensi dan pergerakan objeknya dapat melalui tiga sumbu yaitu sumbu x, y, dan z, sehingga seolah-olah objek tersebut bergerak ke kiri, kanan, atas, bawah, dan menjauhi serta mendekati penontonnya.

2.1.3. Aplikasi Multimedia

Pada zaman sekarang ini, penggunaan aplikasi multimedia sudah merambah ke segala bidang kehidupan manusia. Berikut ini adalah bentuk-bentuk aplikasi multimedia yang ada menurut Dastbaz (2003, p9) pada bidang:

1. Pendidikan

Tidak diragukan lagi bahwa bidang pendidikan telah mendapatkan salah satu keuntungan dengan adanya teknologi multimedia. Penggunaan multimedia dalam bidang pendidikan

dapat memperkaya pembelajaran dari materi pendidikan tersebut. Dengan bantuan gambar, video, animasi, dan suara, materi presentasi dari sebuah mata pelajaran akan dapat lebih dimengerti. Contoh penggunaan aplikasi multimedia dalam bidang pendidikan adalah CAI, perangkat ajar, *E-Learning*, dan lain-lain.

2. Pelatihan

Dalam sebuah studi yang dilakukan Departemen Pertahanan Amerika Serikat, dinyatakan bahwa pelatihan yang menggunakan multimedia 40% lebih efektif daripada pelatihan biasanya. Selain itu, pelatihan dengan multimedia dapat meningkatkan fleksibilitas jadwal pelatihan dan mengurangi biaya pelaksanaan pelatihan. Fungsi multimedia seperti penggunaan audio, video, animasi, gambar, dan teks juga sangat membantu memperkaya materi pelatihan yang diberikan. Aplikasi multimedia dalam bidang pelatihan ini disebut juga dengan *E-Training*.

3. Informasi Penjualan

Aplikasi multimedia yang biasanya dipakai dalam bidang ini adalah kios informasi. Kios informasi ini disebutkan sebagai sebuah *hardware* yang dapat menampilkan gambar, audio, dan video dengan teknologi *touchscreen* sebagai alat *input*nya. Kios ini ditempatkan di tempat umum seperti di bandara atau di museum sehingga pengunjung dapat menerima informasi tentang tempat tersebut.

4. *News Delivery, Broadcasting,* dan Periklanan

Permintaan akan penggunaan media interaktif pada *broadcasting* dan periklanan meningkat pada awal tahun 1992, dan semakin berkembang hingga sekarang. Contoh penggunaan multimedia pada bidang ini antara lain semakin banyaknya *website* berita yang menampilkan berita-berita secara *up-to-date* setiap waktunya dengan *video streaming* ataupun *live broadcast streaming*.

5. Bisnis dan Penjualan

Teknologi aplikasi multimedia bersama dengan teknologi *World Wide Web* (WWW), telah memberikan dampak utama dalam perubahan cara berbisnis manusia. Teknologi tersebut telah menghilangkan batasan ruang dan waktu dalam berbisnis, sehingga proses bisnis dapat berjalan kapan saja dan di mana saja.

2.1.4. *Interactive Multimedia System Design and Development* (IMSDD)

IMSDD adalah sebuah siklus perancangan dan pengimplementasian multimedia. Tahapan-tahapan dalam IMSDD menurut Dastbaz (2003, p130-132) adalah sebagai berikut:

1. Identifikasi kebutuhan sistem

   Fungsi utama tahapan ini adalah:

   - Menentukan sistem yang tepat
   - Menentukan *hardware* dan *software* yang akan digunakan
   - Mengidentifikasi *user* dan kebutuhannya

- Menentukan *platform* pengiriman yang tepat untuk sistem yang digunakan

2. Identifikasi perancangan

   Fungsi utama tahapan ini adalah:

   - Metafora Desain: menentukan bentuk utama *interface* yang akan digunakan pada sistem
   - Bentuk Informasi: menentukan bentuk-bentuk informasi yang akan dimasukkan ke dalam sistem
   - Struktur Navigasi: menyusun struktur navigasi
   - Kontrol Sistem: menentukan tipe dan fitur kontrol untuk sistem

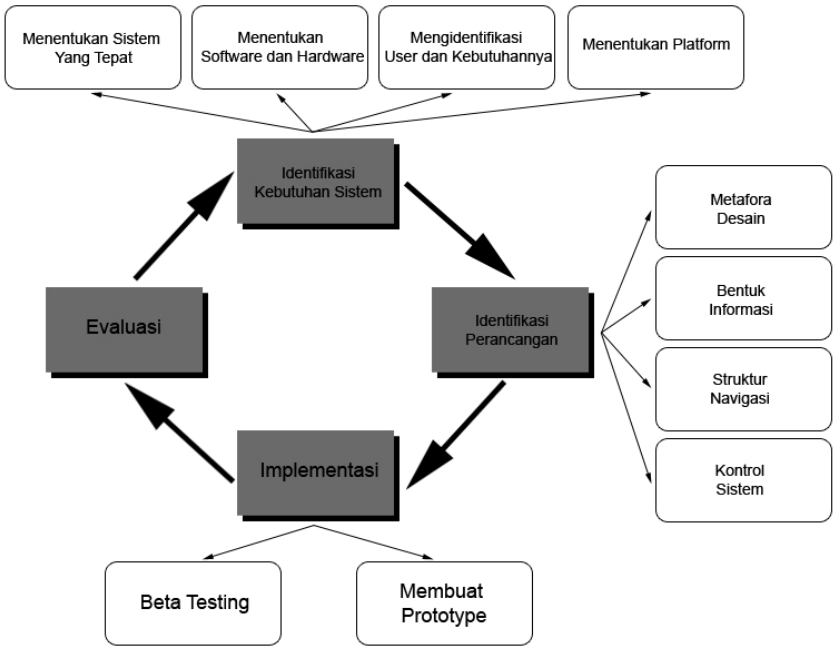
3. Implementasi

   Tahapan ini terdiri dari:

   - Membuat *prototype* dari sistem
   - Melakukan *beta testing* terhadap *prototype* untuk menguji apakah terdapat permasalahan pada kontrol dan rancangan

4. Percobaan dan evaluasi

   Pada tahapan ini, sistem dievaluasi terhadap sasaran yang telah ditetapkan sebelumnya.



Gambar 2.1 Siklus IMSDD

(Sumber: Dastbaz, 2003, p131)

2.1.5. Kios Informasi

2.1.5.1 Pengertian Kios Informasi

Kios informasi adalah sistem yang menyampaikan informasi dan jasa kepada masyarakat sesuai kebutuhan publik. Pada dasarnya, kios informasi adalah *booth* sederhana atau suatu kabinet kecil yang menawarkan layanan komputer yang mudah dioperasikan, biasanya menggunakan teknologi *touchscreen*. Kios informasi merupakan bentuk paling sederhana dibandingkan *platform* interaktif lainnya.

Kios informasi yang dibuat pertama kali adalah *Automated Teller Machine* (ATM) yang diperkenalkan oleh Chemical Bank pada tahun 1969. Sejak saat itu, kios informasi

telah digunakan dengan berbagai tujuan yang sama bermanfaatnya dengan ATM. Kios informasi biasanya diletakkan di tempat umum, baik dalam ruangan maupun di luar ruangan. Popularitas kios informasi tumbuh sejalan dengan turunnya harga komponen teknologi komputer dan bertambahnya kecepatan komputer, membuat kios informasi makin terjangkau dan berguna (Miller, 2004, p376).

2.1.5.2 Karakteristik Kios Informasi

Menurut Stone (2005, p386), karakter utama kios informasi adalah letaknya yang berada di tempat umum dan sifatnya yang menyediakan informasi sesuai kebutuhan masyarakat. Selain itu, karakteristik kios informasi adalah sebagai berikut:

- Dapat digunakan oleh *user* yang belum pernah menggunakan kios informasi tersebut sebelumnya ataupun melalui pelatihan tertentu.
- Memadukan elemen-elemen multimedia untuk menampilkan informasi dengan *user interface* yang interaktif.
- Menggunakan pendekatan nyata dengan menggunakan *metaphor* sehingga *user* dapat merasakan situasi yang sebenarnya. Contohnya untuk kios informasi daerah wisata, *user* diberi informasi tentang posisinya dan posisi tempat

yang dituju sehingga *user* dapat menyimpulkan dengan lebih mudah cara menuju tempat tersebut.

- Dapat berupa *point of information* yang menyediakan informasi atau *point of sale* yang menyajikan informasi untuk memasarkan dan menjual sesuatu (Carbonell, p389).

2.1.5.3 Pedoman Pengembangan Konten Kios Informasi

Menurut Miller (2005, p377), proses pengembangan konten kios informasi berbeda dengan *platform* interaktif lainnya. Perbedaan utamanya ada pada dua faktor: Pertama, kios informasi biasanya diletakkan di tempat umum, dan kedua, kios informasi digunakan oleh individual dari kelompok demografis yang luas. *User* berasal dari umur, latar belakang pendidikan dan kelompok ekonomi sosial yang berbeda yang memiliki sikap yang berbeda dalam menggunakan teknologi komputer. Dengan faktor-faktor tersebut, berikut pedoman utama mengembangkan konten kios informasi:

1. Kios informasi harus menarik perhatian orang yang lewat dengan visual layar yang atraktif.
2. Program harus mudah untuk digunakan. Program kios informasi seharusnya tidak memiliki manual atau instruksi tertulis dan *user* tidak memerlukan latihan sebelum menggunakan program.
3. *Interface* dan navigasi harus serasional mungkin. Saran dan bantuan harus dibuat untuk menolong *user* yang

mendapat kesulitan. Jika mungkin, saran berupa suara lebih diutamakan untuk dibuat daripada saran berupa tulisan karena instruksi tertulis harus dibuat seminimal mungkin.

4. Jika memungkinkan, *input* dari *user* tidak menggunakan *keyboard.* Selain karna *keyboard* rentan rusak, tidak semua *user* merasa nyaman dalam mengetik.
5. Konten harus bersifat melibatkan *user* dan dipresentasikan dengan visual dan interaktivitas sesopan mungkin.
6. Naskah dan teks harus ditulis dengan bahasa yang umum dan dapat dipahami oleh *user* dari berbagai latar belakang. Kosakata yang digunakan sederhana namun ekspresif.
7. Program harus didesain singkat dari awal sampai akhir dalam waktu beberapa menit saja. Harus diperhatikan bahwa *user* berhenti untuk menggunakan kios informasi pada saat mereka akan melakukan sesuatu yang lain. Selain itu, kemungkinan *user* lain menunggu untuk menggunakannya juga.
8. Sama dengan konten interaktif lainnya, program harus diuji ke kelompok *user* yang representatif terhadap populasi sebelum program tersedia untuk masyarakat umum.

2.2. Interaksi Manusia dan Komputer

2.2.1. Pengertian Interaksi Manusia dan Komputer

Interaksi Manusia dan Komputer adalah ilmu yang mempelajari implementasi dan perancangan interaksi manusia sebagai *user* dengan sistem komputer. Perancangan yang dilakukan di sini adalah dengan memanfaatkan kelebihan teknologi yang dimiliki oleh komputer sehingga dapat menjangkau kapasitas dan kebutuhan manusia sebagai *user* (Shneiderman, 2010, p22).

2.2.2. *User Interface*

User Interface menurut Johnson (1992) secara umum adalah sebuah bagian yang menghubungkan *user* dengan komputer, dan dapat melibatkan *hardware* dan *software* dari komputer itu sendiri. Sedangkan Lewis dan Rieman (1993) menyatakan bahwa sebuah bentuk *interface* yang paling dasar harus terdiri dari menu, *windows*, *keyboard*, *mouse*, bunyi *beep*, dan suara komputer lainnya. Dengan ini sebuah *user interface* haruslah mengandung semua bentuk informasi yang dapat membuat komputer dan penggunanya saling berkomunikasi (Dastbaz, 2003, p108).

2.2.3. Lima Faktor Manusia Terukur

Dalam melakukan perancangan sebuah *user interface*, diperlukan sebuah pengukuran yang tepat sehingga *interface* yang dirancang sesuai dengan kebutuhan *user*. Menurut Shneiderman (2010, p32) ada lima pengukuran yang dijadikan dasar pengukuran yaitu:

1. **Waktu Belajar**

   Pengukuran yang menentukan berapa lama waktu yang diperlukan oleh *user* untuk mempelajari langkah yang relevan untuk melakukan sebuah tugas.

2. **Kecepatan Kerja**

   Pengukuran yang menentukan berapa lama waktu yang diperlukan untuk menyelesaikan sebuah tugas.

3. **Tingkat Kesalahan *User***

   Pengukuran yang menentukan berapa banyak kesalahan dan kesalahan apa saja yang dilakukan oleh *user* dalam menyelesaikan tugas tersebut.

4. **Daya Ingat *User***

   Pengukuran yang menentukan berapa lama *user* dapat mengelola dan mempertahankan pengetahuannya dalam jangka waktu tertentu.

5. **Kepuasan Subjektif**

   Pengukuran yang menentukan tingkat kepuasan *user* akan *interface* yang dihadapinya.

2.2.4. *Eight Golden Rules of Interface Design*

Dalam perancangan sebuah *interface*, terdapat delapan aturan emas yang harus diperhatikan dalam melakukan perancangan menurut Shneiderman (2010, pp88-89). Kedelapan aturan emas tersebut adalah:

1. **Berusaha untuk konsisten**

   Konsistensi yang diperlukan dalam perancangan *interface* antara lain dengan menggunakan bentuk *layout* yang sama, *font* yang sama, warna yang sama, dan terminologi yang sama pada setiap halaman.

2. **Memenuhi kegunaan yang universal**

   Perancang *interface* harus mengetahui kebutuhan dari beragam jenis *user*. Kebutuhan tersebut bisa terdiri dari perbedaan tingkat kemahiran *user*, perbedaan usia *user*, kekurangan (kecacatan) *user*, dan lain sebagainya. Penambahan *interface* terhadap kebutuhan-kebutuhan yang beragam tersebut dapat memperkaya *interface* tersebut dan memperbaiki kualitas sistem.

3. **Memberikan umpan balik yang informatif**

   Umpan balik yang informatif dimaksudkan agar *user* dapat mengerti setiap aksi yang dilakukannya. Untuk aksi yang sering dilakukan atau aksi yang sederhana, umpan balik yang diberikan boleh umpan balik yang sederhana. Sedangkan untuk aksi yang jarang dilakukan atau aksi yang besar, umpan balik yang diberikan adalah umpan balik yang lebih rinci.

4. **Memberikan dialog untuk hasil akhir**

   Urutan aksi harus dikelompokkan ke dalam beberapa kelompok yaitu kelompok awal, tengah, dan akhir. Umpan balik yang diberikan pada saat *user* selesai melakukan sebuah aksi, dapat memberikan kepuasan dan rasa lega sehingga *user* dapat mempersiapkan diri untuk aksi selanjutnya.

5. **Memberikan penanganan kesalahan yang sederhana**

   Interface harus dirancang agar *user* tidak melakukan kesalahan yang besar, misalnya dengan menahan *input* numerik pada kolom isian alfabetis. *Interface* juga harus dirancang untuk memberikan penanganan kesalahan yang dilakukan oleh *user* secara sederhana.

6. **Memberikan pembalikan aksi yang mudah**

   Setiap aksi harus dirancang agar dapat melakukan pembalikan aksi sehingga dapat mengurangi kekhawatiran *user* ketika melakukan kesalahan karena kesalahan tersebut bisa diperbaiki.

7. **Memberikan tempat kontrol internal**

   User yang berpengalaman lebih menginginkan perasaan bahwa mereka yang memegang kendali atas *interface* dan *interface*nya memberikan respon atas kendali mereka. Gaines (1981) melengkapi aturan ini dengan pendapatnya bahwa *user* harus menjadi pelaku aksi, bukan perespon aksi.

8. **Mengurangi beban ingatan jangka pendek**

   Batas kemampuan ingatan jangka pendek manusia menyebabkan tampilan *interface* yang diberikan harus sederhana, tampilan yang jumlahnya banyak harus digabungkan, jumlah pergerakan *window* harus dikurangi, dan waktu belajar yang diberikan dialokasikan untuk mempelajari kode-kode dan urutan aksi.

2.3. Rekayasa Perangkat Lunak

2.3.1. Pengertian Rekayasa Perangkat Lunak

Menurut Pressman (2005, p13), rekayasa perangkat lunak adalah aplikasi dari pendekatan-pendekatan yang sistematik, disiplin, dan dapat dihitung untuk pengembangan, operasi dan pemeliharaan perangkat lunak. Sifat sistematik, disiplin dan dapat dihitung yang dikembangkan oleh satu tim pengembang perangkat lunak dapat memberatkan tim pengembang lain. Walaupun disiplin diperlukan, suatu perangkat lunak juga memerlukan kemampuan untuk beradaptasi dan juga ketangkasan.

Berbagai pendekatan teknik (termasuk rekayasa perangkat lunak) harus berdasarkan pada komitmen terhadap kualitas. Rekayasa perangkat lunak terdiri atas tiga lapisan yang didasarkan pada *quality focus*. Tiga lapisan tersebut adalah:

1. **Proses**

   Merupakan fondasi dari rekayasa perangkat lunak yang menghubungkan lapisan-lapisan teknologi yang ada dan membentuk pengembangan perangkat lunak yang rasional. Proses mendefinisikan *framework* yang harus dibuat demi keefektifan penyampaian rekayasa perangkat lunak. Proses perangkat lunak membentuk dasar dari kontrol manajemen dan mendirikan konteks dimana metode teknik diaplikasikan, pembentukan produk pekerjaan (model, dokumen, data, laporan, *form*, dan lain-lain), pembentukan *milestone*, penjaminan kualitas dan pengaturan perubahan secara tepat.

2. **Metode**

   Metode perangkat lunak menyediakan cara teknik untuk membangun perangkat lunak. Metode meliputi berbagai tugas yaitu komunikasi, analisa kebutuhan, model desain, konstruksi program, pengujian dan *support.*

3. ***Tools***

   Tools menyediakan *support* semi-otomatis atau otomatis untuk proses dan metode. Sebuat sistem yang berfungsi untuk mendukung perkembangan perangkat lunak, disebut *Computer-Aided Software Engineering (CASE)*, dibuat ketika *tools* diintegrasikan. Pengintegrasian *tool* bertujuan supaya informasi yang dibuat oleh satu *tool* dapat digunakan oleh *tool* lainnya.

2.3.2. Karakteristik Rekayasa Perangkat Lunak

Untuk lebih mendalami rekayasa perangkat lunak, pemahaman lebih dalam mengenai perangkat lunak perlu dilakukan. Perangkat lunak lebih cocok disebut elemen sistem logika dibanding elemen sistem fisik (Pressman, 2005, p4). Perangkat lunak memiliki tiga karakteristik utama, yaitu:

1. **Perangkat lunak dikembangkan ataupun direkayasa**

   Walaupun terdapat beberapa kesamaan antara pemroduksian perangkat lunak dan perangkat keras, dua aktivitas tersebut memiliki dasar yang berbeda. Pada kedua aktivitas tersebut, kualitas tertinggi perangkat dapat dicapai dengan desain yang baik. Perbedaannya adalah pada proses pembuatan perangkat keras,

masalah kualitas dapat diketahui dengan lebih mudah dibanding perangkat lunak. Biaya pembuatan perangkat lunak berkonsentrasi pada rekayasa.

2. Perangkat lunak tidak habis dipakai

Perangkat lunak tidak rentan terhadap keadaan lingkungan yang menyebabkan perangkat keras dapat habis dipakai. Ketika perangkat keras habis dipakai, komponen akan diganti dengan suku cadang baru. Pada perangkat lunak, tidak terdapat suku cadang. Setiap kesalahan perangkat lunak menyatakan *error* pada desain atau pada proses saat desain diterjemahkan ke dalam bentuk *code* yang dapat dilaksanakan mesin. Maka dari itu, perawatan perangkat lunak dipertimbangkan lebih kompleks dari perawatan perangkat keras.

3. Mayoritas perangkat lunak dibuat sesuai kriteria yang ditetapkan

Seiring berkembangnya disiplin teknis, sejumlah koleksi komponen desain standar diciptakan. Pemasangan standar dan sirkuit yang terintegrasi diluar susunan hanyalah dua dari ribuan komponen standar yang digunakan oleh teknisi mesin dan elektro untuk mendesain sistem baru. Komponen yang dapat dipergunakan berulang telah diciptakan sehingga teknisi dapat berkonsentrasi untuk masalah desain. Pada perangkat keras, komponen yang bisa dipergunakan berulang adalah bagian alami dari proses rekayasa

sementara pada perangkat lunak ini adalah awal untuk mencapai skala yang lebih besar.

Sebuah komponen perangkat lunak harus didesain dan diimplementasikan sehingga dapat digunakan di banyak program yang berbeda. Komponen modern yang dapat digunakan kembali berfungsi menyimpan data dan pemrosesan pada data, memungkinkan teknisi untuk menggunakan bagian yang dapat digunakan kembali untuk membuat aplikasi baru. Sebagai contoh, saat ini *user interface* interaktif dibuat menggunakan komponen yang bisa digunakan kembali yang memungkinkan dibuatnya *windows* grafik, menu *pull-down*, dan berbagai variasi mekanisme interaktif.

2.4. *Database*

Menurut Connoly & Begg (2010, p65), *database* adalah kumpulan data yang saling berhubungan secara logis dan dapat diakses oleh beberapa *user* dan departemen secara bersamaan. Pada *database*, data-data terintegrasi satu sama lain dengan sesedikit mungkin duplikasi antar data.

Menurut Indrajani (2008, p2), *database* mengkonsolidasi banyak catatan yang sebelumnya disimpan dalam *file* terpisah. *Database* dianalogikan sebagai tempat penyimpanan besar yang digunakan oleh banyak *user*.

2.4.1. *Database Management System* (DBMS)

Menurut Connoly & Begg (2010, p66), DBMS adalah *software* yang berinteraksi dengan program aplikasi milik *user* dan *database*. DBMS memungkinkan *user* untuk menetapkan, menciptakan, memelihara, dan mengontrol akses ke *database*. Pada umumnya DBMS menyediakan beberapa fasilitas yaitu DDL (*Data Definition Language*) untuk menetapkan dan menspesifikasikan data, DML (*Data Manipulation Language*) untuk memungkinkan *user* melakukan *insert*, *update*, *delete*, dan *retrieve* dari *database* dan menyediakan kontrol akses ke *database*.

Dengan fungsi tersebut, DBMS menjadi perangkat yang sangat berguna dan *powerful*. Namun DBMS memberikan tampilan yang lebih kompleks bagi *end-user* karena kini *end-user* mendapatkan lebih banyak data daripada yang mereka butuhkan. Mempertimbangkan masalah ini, DBMS menyediakan fasilitas yang disebut *view mechanism* yang memungkinkan setiap *user* memiliki tampilan *database*nya sendiri.

2.4.2 *Entity Relationship Diagram* (ERD)

2.4.2.1 Entitas

Menurut Connoly & Begg (2010, p372), entitas adalah konsep dasar dari ERD, yang merepresentasikan sekumpulan objek (orang, tempat, barang, konsep, *event*) dengan muatan yang sama di dalam *database*. Entitas diidentifikasi

mempunyai keberadaan yang independen oleh sebuah perusahaan dan perorangan.

2.4.2.2 *Relationship*

Menurut Connoly (2010, p374) *relationship* adalah sekumpulan hubungan yang berarti antara satu atau lebih entitas, dimana setiap tipe *relationship* diberi nama yang menggambarkan fungsinya. Sedangkan *relationship occurance* adalah hubungan yang dapat diidentifikasi secara unik.

2.4.2.3 Atribut

Menurut Connoly (2010, p379), atribut adalah sebuah sifat dari entitas atau *relationship*. Sebagai contoh, entitas staff mungkin dapat menjelaskan atribut sebagai berikut: noStaff, nama, posisi, dan gaji. Setiap atribut menyimpan nilai yang menjelaskan setiap *entry occurance* dan menggambarkan bagian utama dari data yang disimpan di dalam *database*.

2.5. *Unified Modelling Language* (UML)

Menurut Larman (2005, p10), UML adalah sebuah bahasa untuk menspefisikasikan, memvisualisasikan, menkonstruksikan dan mendokumentasikan bagian/komponen dari suatu sistem perangkat lunak. UML juga dapat dipergunkan untuk permodelan bisnis dan sistem-sistem non perangkat lunak.

Menurut Connolly & Begg (2010, p909), diagram utama UML terbagi menjadi dua kategori:

a.) *Structural Diagram*, menjelaskan hubungan statis antar komponen. Contoh:

1. *Class Diagram*
2. *Object Diagram*
3. *Component Diagram*
4. *Deployment Diagram*

b.) *Behavioral Diagram*, menjelaskan hubungan dinamis antar komponen. Contoh:

1. *Use Case Diagram*
2. *Sequence Diagram*
3. *Collaboration Diagram*
4. *Statechart Diagram*
5. *Activity Diagram*

2.5.1. *Use Case Diagram*

Menurut Whitten & Bentley (2007, p246), *Use Case Diagram* adalah diagram yang menggambarkan interaksi antara sistem dengan hal-hal eksternal dari sistem dan *user*. Dengan kata lain, secara grafis menjelaskan siapa yang akan mempergunakan sistem dan dengan cara apa *user* diharapkan untuk berinteraksi dengan sistem.

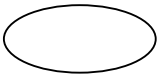
Menurut Wahono (2003, p4), yang ditekankan pada *Use Case Diagram* adalah "apa" yang diperbuat oleh sistem, dan bukan "bagaimana".

Use Case Diagram berguna untuk menentukan *requirement* sebuah sistem, mengkomunikasikan rancangan dengan klien, dan merancang *test case* untuk semua fitur yang ada pada sistem.

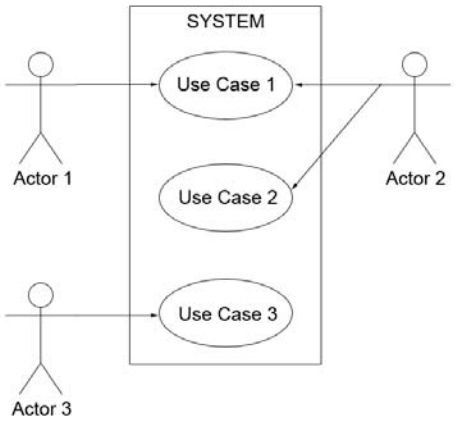
Berikut ini adalah simbol-simbol dalam *Use Case Diagram*:

Tabel 2.1 Tabel Simbol dalam *Use Case Diagram*

(Sumber: Whitten, Bentley, & Dittman, 2004, pp273-274)

Simbol	**Nama**
	Aktor
	Use Case
	Association

Berikut ini adalah contoh penggunaan *Use Case Diagram*:



Gambar 2.2 Contoh Penggunaan *Use Case Diagram*

(Sumber: Whitten & Bentley, 2007, p246)

2.5.2. *Class Diagram*

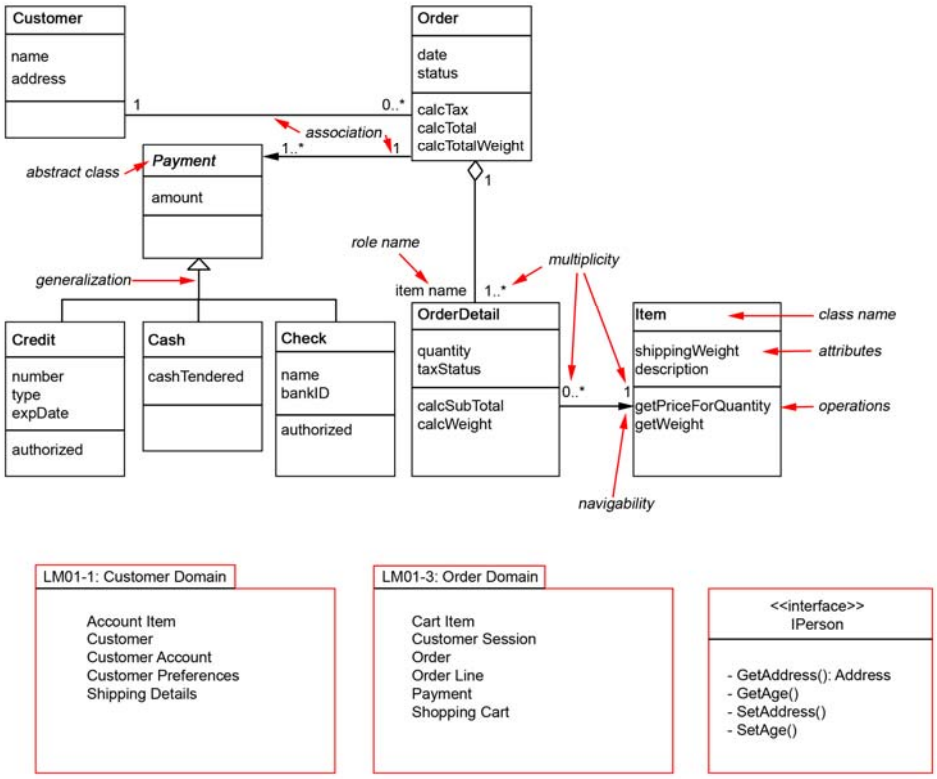
Menurut Wahono (2003, p5) *class* adalah spesifikasi yang jika diinisiasi akan menghasilkan sebuah objek dan merupakan inti dari pengembangan dan desain berorientasi objek. *Class* memiliki tiga area pokok yaitu nama, atribut/properti, dan metode, yang merupakan layanan untuk memanipulasi keadaan tersebut.

Menurut Whitten & Bentley (2007, p400) *Class Diagram* adalah penggambaran grafis dan deskripsi dari *class* dan objek, beserta hubungannya satu sama lain.

Menurut Wahono (2003, p6) hubungan antar *Class* terdiri dari 4 hubungan:

1. Asosiasi, hubungan statis antar *class*. Umumnya menggambarkan *class* yang memiliki atribut berupa *class* lain, atau *class* yang harus mengetahui eksistensi *class* lain. Panah menunjukan arah *query* antar *class*.
2. Agregasi, hubungan yang menyatakan bagian ("terdiri atas..")
3. Pewarisan, hubungan hirarkis antar *class*. Dapat diturunkan dari *class* lain dengan mewarisi semua atribut dari metode *class* asalnya dan menambahkan fungsionalitas baru, sehingga ia disebut anak dari *class* yang diwarisinya.
4. Hubungan dinamis, rangkaian *message* yang dipindahkan dari satu *class* ke *class* lain.

Berikut ini adalah contoh dari *Class Diagram*:



Gambar 2.3 Contoh *Class Diagram*

(Sumber: Wahono 2003, p6)

2.5.3. Activity Diagram

Menurut Whitten & Bentley (2007, p390), *Activity Diagram* adalah diagram yang dipergunakan untuk menggambarkan alur dari proses bisnis secara grafis, langkah-langkah dari *use case* dan logika dari karakteristik objek.

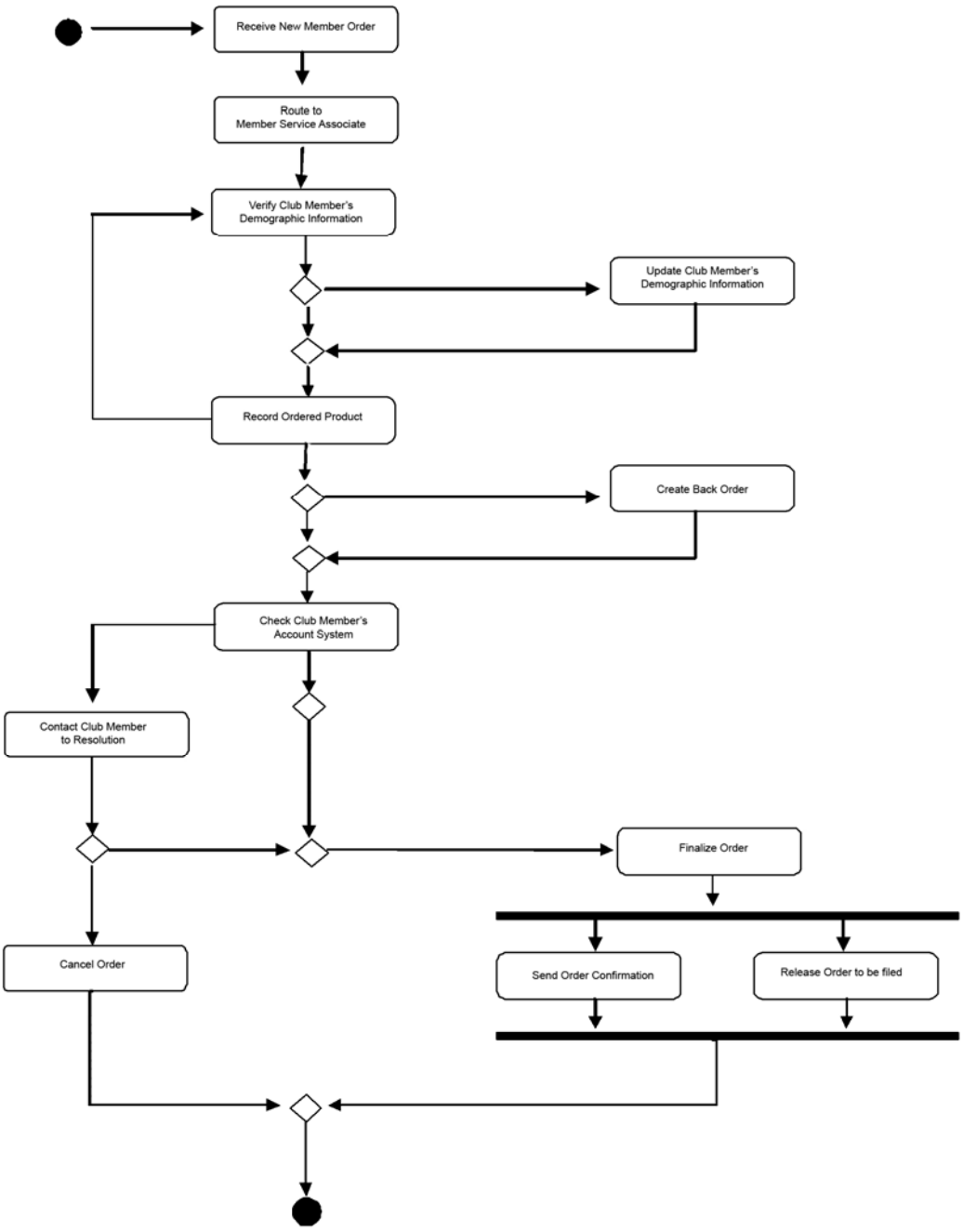
Menurut Wahono (2003, p7), yang digambarkan pada *Activity Diagram* meliputi bagaimana aliran data berawal, hasil yang mungkin terjadi, dan bagaimana mereka berakhir. *Activity Diagram* juga dapat menggambarkan proses paralel yang mungkin terjadi pada beberapa eksekusi. *Activity Diagram* tidak menggambarkan sifat internal sebuah sistem secara eksak, tetapi lebih menggambarkan proses-proses dan jalur-jalur aktivitas dari level atas secara umum.

Notasi-notasi yang digunakan dalam *Activity Diagram* adalah sebagai berikut:

Tabel 2.2 Notasi pada *Activity Diagram*

(Sumber: Whitten & Bentley, 2007, pp451-454)

Notasi	**Fungsi**
	Melambangkan awal dari sebuah proses
	Melambangkan *activity*
	Melambangkan *trigger*
	Melambangkan garis sinkronisasi
[ ]	Kata-kata di dalam [ ] melambangkan *trigger* yang menjadi hasil dari *decision activity*.
	Melambangkan *decision activity*
	Melambangkan akhir dari proses



Gambar 2.4 Contoh *Activity Diagram*

(Sumber: Whitten & Bentley, p392)