BAB II
LANDASAN TEORI

2.1 Sejarah Singkat Kantor Pengawasan dan Pelayanan Bea dan Cukai Tipe Madya Pabean A Bandung

Direktorat Jenderal Bea dan Cukai (DJBC) merupakan organisasi vertikal di bawah Kementerian Keuangan yang melaksanakan sebagian tugas pokok Kementerian Keuangan di bidang kepabeanan dan cukai berdasarkan kebijaksanaan yang ditetapkan oleh Menteri Keuangan. Bertugas mengamankan kebijaksanaan pemerintah yang berkaitan dengan lalu lintas barang yang masuk atau keluar Daerah Pabean dan pemungutan Bea Masuk dan Cukai serta pungutan negara lainnya berdasarkan peraturan perundang-undangan yang berlaku. Berdasarkan tugas pokok tersebut maka Direktorat Jenderal Bea dan Cukai (DJBC) melaksanakan pemungutan, pertama, Bea Masuk dan Bea Keluar atas barang-barang berdasarkan peraturan perundang-undangan pabean yang berlaku dan pungutan lainnya, kedua, Cukai atas barang-barang tertentu yang peredarannya dibatasi berdasarkan peraturan perundang-undangan yang berlaku.

Nomenklatur Kantor Pengawasan dan Pelayanan Bea Cukai (KPPBC) Tipe Madya Pabean (TMP) A Bandung pada tahun 1982 semula bernama Kantor Inspeksi Bea dan Cukai Tipe C Bandung, berlokasi di Gedung Keuangan Negara (GKN) di Jalan Asia Afrika, dan masih menginduk kepada Kantor Bea dan Cukai Halim, dengan fokus pengawasan pada bidang cukai dan Kantor Pos Lalu Bea dengan wilayah kerja hingga Purwakarta dan Bekasi. Pada tahun 1990 kantor ini berubah dari kantor Bea dan Cukai Tipe C menjadi Tipe B, dan bersamaan dengan itu juga dibentuk Kantor Wilayah Bea dan Cukai Bandung yang membawahi seluruh wilayah Jawa Barat [4].

Pada tahun 1990 mulai dibangun Terminal Peti Kemas (TPKB) Bandung yang berlokasi di Gede Bage sehingga Kantor Inspeksi Tipe B Bandung pindah dari gedung GKN ke gedung Apekti di Jalan Soekarno Hatta sambil menunggu pembangunan gedung milik sendiri yang berada di Jalan Rumah Sakit, Gede Bage. Setelah pembangunan selesai, kurang lebih 2 tahun kemudian akhirnya

Kantor Inspeksi Bea dan Cukai Tipe B Bandung menempati gedung yang baru di Jalan Rumah Sakit. Sekitar tahun 1995 nama Kantorpun berubah menjadi Kantor Pelayanan Bea dan Cukai Tipe A Bandung, seiring dengan mulai berdirinya beberapa perusahaan pengguna fasilitas Kawasan Berikat. Pada tahun 1999 Purwakarta berpisah dari Bandung dan terbentuk Kantor Pelayanan yang berdiri sendiri, demikian juga Bekasi pada tahun 2002 memiliki Kantor Pelayanan sendiri.

Setelah ditempati kurang lebih 10 tahun, dilakukan renovasi gedung dan untuk sementara kantorpun berpindah kembali ke gedung Apekti. Tahun 2003 setelah renovasi selesai, kantor kembali menempati gedung semula di Jalan Rumah Sakit dan bidang pelayanan dan pengawasannya bertambah setelah ekspor fasilitas Bapeksta penanganannya dikembalikan ke Direktorat Jenderal Bea dan Cukai, dan nama Kantorpun kembali berubah menjadi Kantor Pelayanan Bea dan Cukai Tipe A3 Bandung. Tidak berselang lama kemudian nama kantor berubah kembali menjadi Kantor Pengawasan dan Pelayanan Bea dan Cukai Tipe A2 Bandung.

Dengan adanya reformasi birokrasi di Direktorat Jenderal Bea dan Cukai dan melihat perkembangan volume kerja yang semakin meningkat, maka berdasarkan Peraturan Menteri Keuangan nomor 74/PMK.01/2009 tanggal 8 April 2009 nama kantor berubah menjadi Kantor Pengawasan dan Pelayanan Bea dan Cukai Tipe Madya Pabean Bandung yang implementasinya berlaku secara efektif setelah diresmikan pada tanggal 30 September 2009. Berdasarkan Peraturan menteri keuangan nomor 131/PMK.01/2011 maka penamaan kantor mengalami perubahan kembali sehingga menjadi Kantor Pengawasan dan Pelayanan Bea dan Cukai Tipe Madya Pabean A Bandung.

2.1.1 Visi dan Misi Direktorat Jenderal Bea dan Cukai

Visi dan misi Museum Geologi adalah Direktorat Jenderal Bea dan Cukai (DJBC) sejajar dengan institusi Kepabeanan dan Cukai dunia di bidang kinerja dan citra sesuai dengan Standar Internasional serta mengamankan hak keuangan

negara, memfasilitasi perdagangan, mendukung industri dan melindungi masyarakat.

Direktorat Jenderal Bea dan Cukai (DJBC) menetapkan misi yang saling terkait yaitu:

1. Memungut penerimaan negara dari sektor perdagangan internasional dan cukai.
2. Memberikan pelayanan terbaik di bidang kepabeanan dan cukai yang sederhana dengan berbasis teknologi informasi.
3. Mengembangkan pengawasan yang efektif dalam rangka penegakan hukum di bidang kepabeanan dan cukai serta perlindungan masyarakat.
4. Mengembangkan institusi kepabeanan dan cukai yang berdaya guna dan berhasil guna.
5. Mendorong terciptanya iklim usaha yang kondusif bagi pertumbuhan industri dan investasi.
6. Mengembangkan kerjasama internasional di bidang kepabeanan dan cukai.

2.1.2 Strategi dan Komitmen Harian KPPBC TMP A Bandung

Strategi KPPBC Tipe Madya Pabean A Bandung Jawa Barat dalam pencapaian visi dan misi adalah ”Profesionalisme sumber daya manusia, efisiensi dalam organisasi dan pelayanan.”

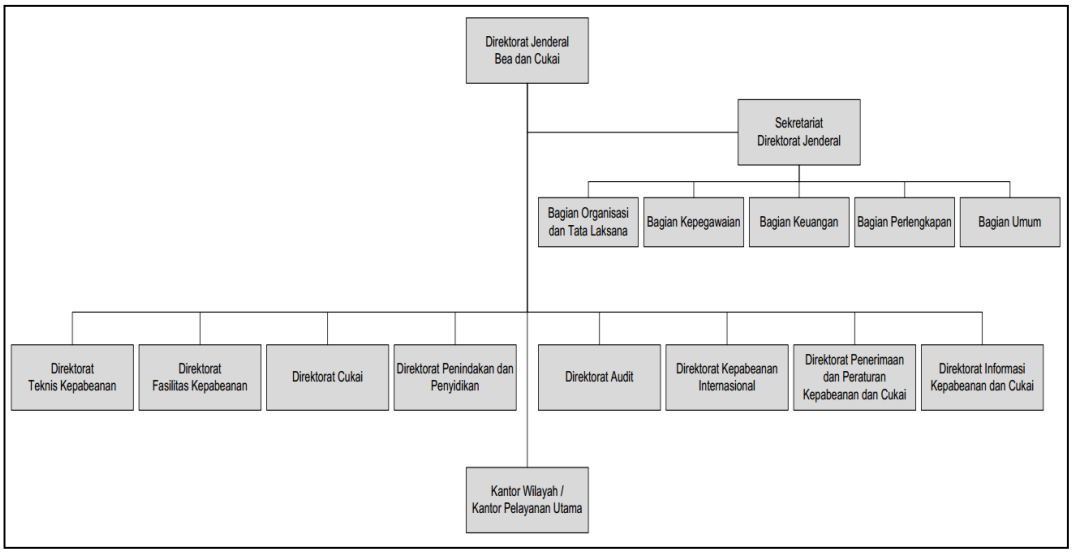
Pencapaian visi dan misi KPPBC Tipe Madya Pabean A Bandung dirumuskan ke dalam strategi KPPBC Tipe Madya Pabean A Bandung. Agar strategi tersebut dapat berjalan optimal, ditetapkan lima komitmen KPPBC Tipe Madya Pabean A Bandung, yaitu:

1. Tingkatkan pelayanan
2. Tingkatkan transparansi keadilan dan konsistensi
3. Pastikan pengguna jasa bekerja sesuai ketentuan
4. Hentikan perdagangan ilegal
5. Tingkatkan integritas

2.1.3 Struktur Organisasi KPPBC Tipe Madya Pabean A Bandung

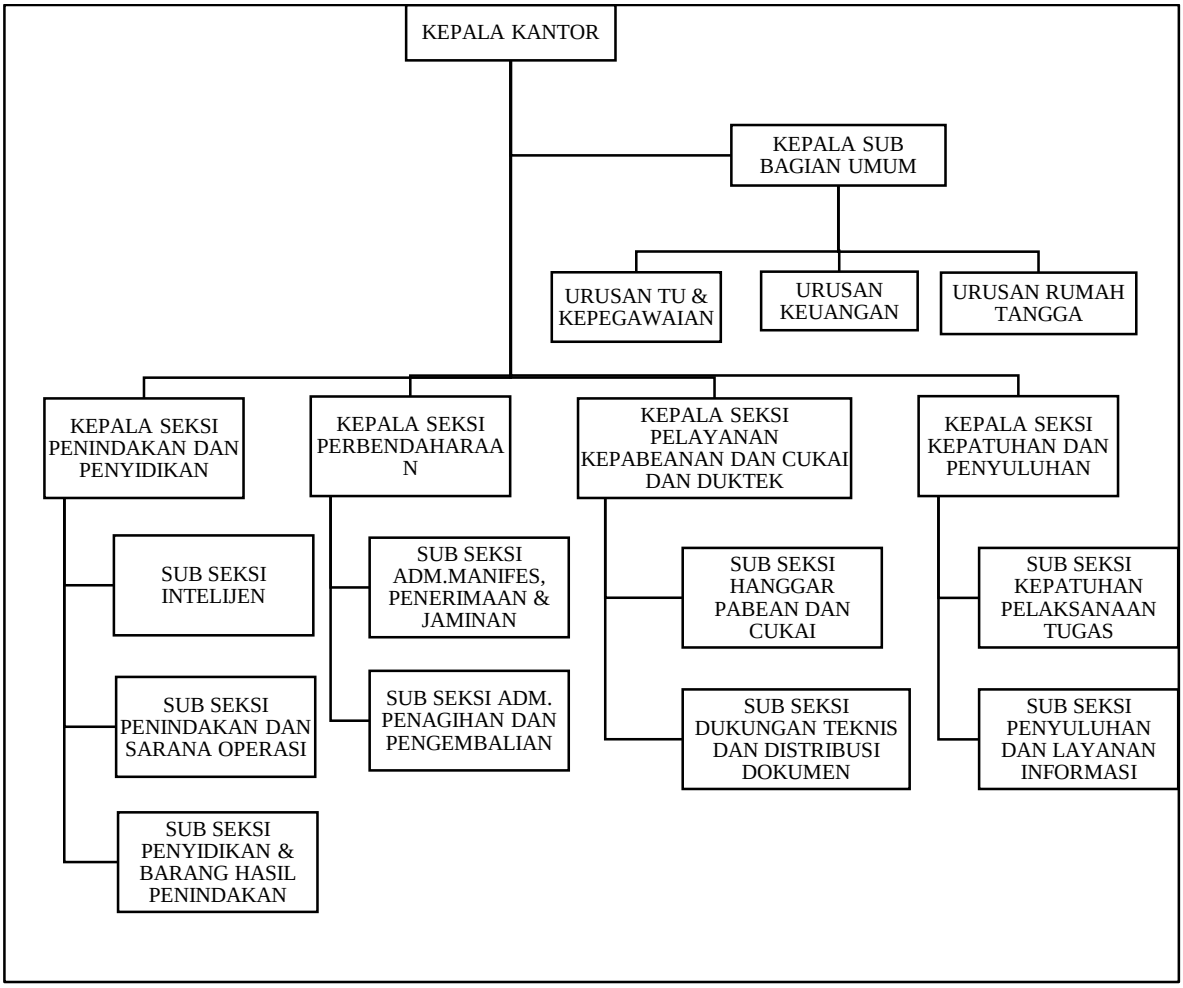
Struktur organisasi merupakan susunan yang terdiri atas fungsi-fungsi dan hubungan-hubungan yang menyatakan keseluruhan kegiatan untuk mencapai suatu sasaran. Struktur organisasi secara fisik dinyatakan dalam bentuk gambar grafik dan atau bagan yang memperlihatkan unit-unit organisasi dan garis-garis wewenang.

Direktorat Jenderal Bea dan Cukai (DJBC) merupakan unit vertikal eselon I dari Kementerian Keuangan Republik Indonesia. Untuk melaksanakan sebagian tugas pokok Kementerian Keuangan dalam pemungutan bea masuk dan cukai, Direktorat Jenderal Bea dan Cukai (DJBC) membentuk unit vertikal eselon II hingga eselon V di lingkungannya. KPPBC Tipe Madya Pabean A Bandung merupakan salah satu unit vertikal eselon II Direktorat Jenderal Bea dan Cukai (DJBC) tersebut. Struktur organisasi Direktorat Jenderal Bea dan Cukai (DJBC) mulai dari unit vertikal eselon I sampai unit vertikal eselon II ditunjukan pada Gambar 2.1.



Gambar 2.1 Struktur Organisasi Direktorat Jenderal Bea dan Cukai

Struktur organisasi Kantor KPPBC TMP A Bandung ditunjukan pada Gambar 2.2.



Gambar 2.2 Struktur Organisasi KPPBC TMP A Bandung

2.1.4 Uraian Tugas (*Job Description*) KPPBC Tipe Madya Pabean A Bandung

Uraian tugas pada KPPBC Tipe Madya Pabean A Bandung adalah sebagai berikut :

1. Kepala Kantor Wilayah DJBC

   Melaksanakan koordinasi, bimbingan teknis, pengendalian, evaluasi dan pelaksanaan tugas di bidang kepabeanan dan cukai dalam wilayah kerjanya berdasarkan peraturan perundang-undangan yang berlaku.

2. Kepala Bagian Umum dan Kepatuhan Internal

   Melaksanakan urusan kepegawaian, keuangan, ketatausahaan dan rumah tangga, penyiapan koordinasi dan pelaksanaan pengawasan pelaksanaan tugas, dan evaluasi kinerja serta penyuluhan dan publikasi peraturan perundang-undangan kepabeanan dan cukai.

   a. Kepala Subbagian Kepegawaian dan Kepatuhan Internal

      Melakukan urusan kepegawaian, penyiapan bahan koordinasi dan pelaksanaan pengawasan pelaksanaan tugas dan evaluasi kinerja, pemantauan dan pelaporan tindak lanjut hasil pemeriksaan aparat pengawasan fungsional dan pengawasan masyarakat.

   b. Kepala Subbagian Hubungan Masyarakat dan Rumah Tangga

      Melakukan penyiapan bahan penyuluhan dan publikasi peraturan perundang-undangan kepabeanan dan cukai, hubungan masyarakat, urusan rumah tangga, dan perlengkapan.

   c. Kepala Subbagian Tata Usaha dan Keuangan

      Melakukan urusan tata persuratan, kearsipan, penyusunan rencana kerja dan laporan akuntabilitas, serta urusan keuangan, anggaran, dan kesejahteraan pegawai.

3. Kepala Bidang Kepabeanan dan Cukai

   Melaksanakan bimbingan teknis, pengendalian, dan evaluasi pelaksanaan peraturan perundang-undangan, pemberian perijinan, pelaksanaan penelitian atas keberatan terhadap penetapan di bidang kepabeanan dan cukai, serta pelaksanaan pengolahan data, penyajian informasi, dan laporan di bidang kepabeanan dan cukai.

   a. Kepala Seksi Pabean dan Cukai

      Melakukan penyiapan bahan penyusunan rencana, pemantauan dan evaluasi realisasi, dan penyusunan laporan penerimaan di bidang kepabeanan dan cukai, bimbingan teknis, penyiapan bahan pengendalian dan evaluasi pelaksanaan tatalaksana impor dan ekspor,klasifikasi barang, nilai pabean, penyiapan bahan rekomendasi dan perijinan di

bidang impor dan ekspor, serta evaluasi pelaksanaan tatalaksana dan fasilitas di bidang cukai, penyiapan bahan rekomendasi dan perijinan, dan fasilitas di bidang cukai.

b. Kepala Seksi Keberatan dan Banding

Melakukan penelitian atas keberatan terhadap penetapan di bidang kepabeanan dan cukai dan penyiapan administrasi urusan banding.

c. Kepala Seksi Informasi Kepabeanan dan Cukai

Melakukan penyiapan bahan koordinasi dan pelaksanaan pengolahan data, penyajian informasi dan pelaporan kepabeanan dan cukai.

4. Kepala Bidang Fasilitas Kepabeanan

Melaksanakan bimbingan teknis pelaksanaan peraturan perundang-undangan kepabeanan di bidang tempat penimbunan, melaksanakan pengendalian pelaksanaan peraturan perundang-undangan kepabeanan dan evaluasi pelaksanaan serta fasilitasi di bidang kepabeanan.

a. Kepala Seksi Fasilitas Pabean

Melakukan penyiapan bahan bimbingan teknis, pengendalian dan evaluasi pelaksanaan peraturan perundang-undangan kepabeanan di bidang tempat penimbunan, pelaksanaan pemberian perijinan di bidang tempat penimbunan serta pemberian fasilitas di bidang kepabeanan lainnya.

b. Kepala Seksi Kemudahan Impor Tujuan Ekspor

Melakukan pemberian fasilitas kemudahan impor tujuan ekspor.

5. Kepala Bidang Penindakan dan Penyidikan

Melaksanakan bimbingan teknis, pengendalian, dan evaluasi pelaksanaan peraturan perundang-undangan, pemberian perijinan, pelaksanaan penelitian atas keberatan terhadap penetapan di bidang kepabeanan dan cukai.

a. Kepala Seksi Intelijen

Melakukan penyiapan bahan bimbingan teknis, pengendalian, evaluasi dan koordinasi pelaksanaan intelijen di bidang kepabeanan dan cukai, pengumpulan, analisis, penyajian, dan penyebaran informasi intelijendan hasil intelijen, serta pengelolaan pangkalan data intelijen.

b. Kepala Seksi Penindakan

Melakukan penyiapan bahan bimbingan teknis, pengendalian, evaluasi dan koordinasi pelaksanaan patroli dan operasi pencegahan pelanggaran peraturan perundang-undangan, penindakan di bidang kepabeanan dan cukai, pengendalian tindak lanjut hasil penindakan,serta pengelolaan dan pemeliharaan sarana operasi, sarana komunikasi dan senjata api Kantor Wilayah.

c. Kepala Seksi Penyidikan dan Barang Hasil Penindakan

Melakukan penyiapan bahan bimbingan teknis, pengendalian, evaluasi dan koordinasi pelaksanaan penyidikan tindak pidana kepabeanan dan cukai, pemantauan tindak lanjut hasil penyidikan, pengumpulan data pelanggaran dan data penyelesaian pelanggaran peraturan perundang-undangan kepabeanan dan cukai, penatausahaan dan pengurusan barang hasil penindakan, barang bukti, pelelangan dan premi.

6. Kepala Bidang Audit

Melaksanakan perencanaan dan pelaksanaan audit serta evaluasi hasil audit di bidang kepabeanan dan cukai.

a. Kepala Seksi Perencanaan Audit

Melakukan penyiapan bahan penyusunan rencana pelaksanaan audit di bidang Kepabeanan dan cukai.

b. Kepala Seksi Pelaksanaan Audit

Melakukan penatausahaan pelaksanaan audit, pemantauan pelaksanaan audit, pelaporan pelaksanaan audit di bidang Kepabeanan dan cukai.

c. Kepala Seksi Evaluasi Audit

Melakukan evaluasi hasil audit di bidang kepabeanan dan cukai.

2.1.5 Aspek Kegiatan KPPBC TMP A Bandung

Kegiatan yang dilakukan pada Kantor Wilayah Direktorat Jenderal Bea dan Cukai, adalah sebagai berikut:

1. Perumusan kebijaksanaan teknis di bidang kepabeanan dan cukai, sesuai dengan kebijaksanaan yang ditetapkan oleh Menteri dan peraturan perundang-undangan yang berlaku.
2. Perencanaan, pelaksanaan, pengendalian, evaluasi dan pengamanan teknis operasional kebijaksanaan pemerintah yang berkaitan dengan pengawasan atas lalu lintas barang yang masuk atau keluar daerah pabean, sesuai dengan kebijaksanaan yang ditetapkan oleh Menteri dan berdasarkan peraturan perundang-undangan yang berlaku.
3. Perencanaan, pelaksanaan, pengendalian, evaluasi dan pengamanan teknis operasional di bidang pemungutan bea masuk dan cukai serta pungutan lainnya yang pemungutannya dibebankan kepada Direktorat Jenderal Bea dan Cukai (DJBC) berdasarkan peraturan perundang-undangan yang berlaku.
4. Perencanaan, pembinaan dan bimbingan di bidang pemberian pelayanan, perijinan, kemudahan, ketatalaksanaan dan pengawasan di bidang kepabeanan dan cukai berdasarkan peraturan perundang-undangan yang berlaku.
5. Pencegahan pelanggaran peraturan perundang-undangan kepabeanan dan cukai dan penindakan di bidang kepabeanan dan cukai serta penyidikan tindak pidana kepabeanan dan cukai sesuai peraturan perundang-undangan yang berlaku.

2.2 Ekspor

Pengertian Ekspor barang pada umumnya adalah kegiatan mengeluarkan/mengirim barang ke luar negeri, biasanya dalam jumlah besar untuk tujuan perdagangan, dan melibatkan *Custom* (Bea Cukai) baik di negara asal maupun negara tujuan. Bea Cukai bertugas sebagai pengawas keluar masuknya / lalu lintas barang dalam suatu Negara [5].



Gambar 2.3 Kegiatan Ekspor Pengangkutan Peti Kemas di Pelabuhan

Berikut prosedur atau mekanisme proses ekspor dari Indonesia ke luar negeri :

1. Mencari tahu terlebih dahulu apakah barang yang akan diekspor tersebut termasuk barang yang dilarang untuk diekspor, diperbolehkan untuk diekspor tetapi dengan pembatasan, atau barang yang bebas diekspor (Menurut undang-undang dan peraturan di Indonesia).
2. Memastikan barang diperbolehkan untuk masuk ke negara tujuan ekspor.
3. Jika sudah mendapatkan pembeli (*buyer*), menentukan sistem pembayaran, menentukan *quantity* dan spek barang, dll, maka selanjutnya kita mempersiapkan barang yang akan diekspor dan dokumen-dokumennya sesuai kesepakatan dengan *buyer*.
4. Melakukan pemberitahuan pabean kepada pemerintah (Bea Cukai) dengan menggunakan dokumen Pemberitahuan Ekspor Barang (PEB) beserta dokumen pelengkapnya.
5. Setelah eksportasi disetujui oleh Bea Cukai, maka akan diterbitkan dokumen NPE (Nota Persetujuan Ekspor)**.** Jika sudah terbit NPE, maka secara hukum barang sudah dianggap sebagai barang ekspor.

6. Melakukan *stuffing* dan mengapalkan barang menggunakan moda transportasi udara (*air cargo*), laut (*sea cargo*), atau darat.
7. Mengasuransikan barang / kargo (jika menggunakan term CIF).
8. Mengambil pembayaran di Bank (Jika menggunakan LC atau pembayaran di akhir).

2.2.1 Jenis Ekspor

Kegiatan ekspor terbagi menjadi 2 jenis, yaitu:

1. Ekspor langsung

   Ekspor langsung adalah cara menjual barang atau jasa melalui perantara/ eksportir yang bertempat di negara lain atau negara tujuan ekspor. Penjualan dilakukan melalui distributor dan perwakilan penjualan perusahaan. Keuntungannya, produksi terpusat di negara asal dan kontrol terhadap distribusi lebih baik. Kelemahannya, biaya transportasi lebih tinggi untuk produk dalam skala besar dan adanya hambatan perdagangan serta proteksionisme.
2. Ekspor tidak langsung

   Ekspor tidak langsung adalah teknik dimana barang dijual melalui perantara/eksportir negara asal kemudian dijual oleh perantara tersebut. Melalui, perusahaan manajemen ekspor (*export management companies*) dan perusahaan pengekspor (*export trading companies*). Kelebihannya, sumber daya produksi terkonsentrasi dan tidak perlu menangani ekspor secara langsung. Kelemahannya, kontrol terhadap distribusi kurang dan pengetahuan terhadap operasi di negara lain kurang. Umumnya, industri jasa menggunakan ekspor langsung sedangkan industri manufaktur menggunakan keduanya.

2.2.2 Kepabeanan dan Pemberitahuan Ekspor Barang (PEB)

Kepabeanan adalah segala sesuatu yang berhubungan dengan pengawasan atas lalu lintas barang yang masuk atau keluar daerah pabean serta pemungutan bea masuk dan bea keluar. Daerah pabean adalah wilayah Republik Indonesia

yang meliputi wilayah darat, perairan dan ruang udara di atasnya, serta tempat-tempat tertentu di Zona Ekonomi Eksklusif dan landas kontinen yang di dalamnya berlaku Undang-Undang Kepabeanan. Kawasan pabean adalah kawasan dengan batas-batas tertentu di pelabuhan laut, bandar udara, atau tempat lain yang ditetapkan untuk lalu lintas barang yang sepenuhnya berada di bawah pengawasan Direktorat Jenderal Bea dan Cukai. Kantor pabean adalah kantor dalam lingkungan Direktorat Jenderal Bea dan Cukai tempat dipenuhinya kewajiban pabean sesuai dengan ketentuan Undang-Undang Kepabeanan. Pos pengawasan pabean adalah tempat yang digunakan oleh pejabat bea dan cukai untuk melakukan pengawasan terhadap lalu lintas barang impor dan ekspor.

Kewajiban pabean adalah semua kegiatan di bidang kepabeanan yang wajib dilakukan untuk memenuhi ketentuan dalam Undang-Undang Kepabeanan. Pemberitahuan pabean adalah pernyataan yang dibuat oleh orang dalam rangka melaksanakan kewajiban pabean dalam bentuk dan syarat yang ditetapkan dalam Undang-Undang Kepabeanan. Bea masuk adalah pungutan negara berdasarkan Undang-Undang Kepabeanan yang dikenakan terhadap barang yang diimpor. Bea keluar adalah pungutan negara berdasarkan Undang-Undang Kepabeanan yang dikenakan terhadap barang ekspor.

Tempat penimbunan sementara adalah bangunan dan/atau lapangan atau tempat lain yang disamakan dengan itu di kawasan pabean untuk menimbun barang, sementara menunggu pemuatan atau pengeluarannya. Tempat penimbunan berikat adalah bangunan, tempat, atau kawasan yang memenuhi persyaratan tertentu yang digunakan untuk menimbun barang dengan tujuan tertentu dengan mendapatkan penangguhan bea masuk. Tempat penimbunan pabean adalah bangunan dan/atau lapangan atau tempat lain yang disamakan dengan itu, yang disediakan oleh pemerintah di kantor pabean, yang berada di bawah pengelolaan Direktorat Jenderal Bea dan Cukai untuk menyimpan barang yang dinyatakan tidak dikuasai, barang yang dikuasai negara, dan barang yang menjadi milik negara berdasarkan Undang-Undang Kepabeanan.

Barang tertentu adalah barang yang ditetapkan oleh instansi teknis terkait sebagai barang yang pengangkutannya di dalam daerah pabean diawasi. Audit

kepabeanan adalah kegiatan pemeriksaan laporan keuangan, buku, catatan dan dokumen yang menjadi bukti dasar pembukuan, surat yang berkaitan dengan kegiatan usaha termasuk data elektronik, surat yang berkaitan dengan kegiatan di bidang kepabeanan, dan/atau sediaan barang dalam rangka pelaksanaan ketentuan peraturan perundang-undangan di bidang kepabeanan. Tarif adalah klasifikasi barang dan pembebanan bea masuk atau bea keluar.

Pemberitahuan Ekspor Barang (PEB) adalah dokumen pabean yang digunakan untuk memberitahukan pelaksanaan ekspor barang. PEB dibuat oleh eksportir atau kuasanya, dimana barang yang akan diekspor wajib diberitahukan ke Kantor Bea dan Cukai dengan menggunakan PEB ini. PEB diajukan untuk memperoleh respon Persetujuan Ekspor (PE). Barulah kemudian PE digunakan sebagai surat jalan untuk memasukkan barang ekspor ke kawasan pabean/ kawasan dalam pengawasan bea cukai yang dipersiapkan untuk ekspor.

2.3 Aplikasi *Mobile*

Aplikasi adalah program yang digunakan orang untuk melakukan sesuatu pada sistem komputer. *Mobile* dapat diartikan sebagai perpindahan yang mudah dari satu tempat ke tempat yang lain, misalnya telepon *mobile* berarti bahwa terminal telepon yang dapat berpindah dengan mudah dari satu tempat ke tempat lain tanpa terjadi pemutusan atau terputusnya komunikasi. Sistem aplikasi *mobile* merupakan aplikasi yang dapat digunakan walaupun pengguna berpindah dengan mudah dari satu tempat ketempat lain lain tanpa terjadi pemutusan atau terputusnya komunikasi. Aplikasi ini dapat diakses melalui perangkat nirkabel seperti *pager*, seperti telepon seluler dan PDA. Adapun karakteristik perangkat *mobile* yaitu [6]:

1. Ukuran yang kecil

   Perangkat *mobile* memiliki ukuran yang kecil. Konsumen menginginkan perangkat yang terkecil untuk kenyamanan dan mobilitas mereka.

2. Memory yang terbatas

   Perangkat *mobile* juga memiliki memory yang kecil, yaitu primary (RAM) dan *secondary* (*disk*).

3. Daya proses yang terbatas

   Sistem *mobile* tidaklah setangguh rekan mereka yaitu *desktop*.

4. Mengkonsumsi daya yang rendah

   Perangkat *mobile* menghabiskan sedikit daya dibandingkan dengan mesin *desktop*.

5. Kuat dan dapat diandalkan

   Karena perangkat *mobile* selalu dibawa kemana saja, mereka harus cukup kuat untuk menghadapi benturan-benturan, gerakan, dan sesekali tetesan-tetesan air.

6. Konektivitas yang terbatas

   Perangkat *mobile* memiliki *bandwith* rendah, beberapa dari mereka bahkan tidak tersambung.

7. Masa hidup yang pendek

   Perangkat-perangkat konsumen ini menyala dalam hitungan detik kebanyakan dari mereka selalu menyala.

2.4 Cross *Platform*

Dalam era ini, memiliki aplikasi *mobile* merupakan keharusan bagi bisnis kontemporer. Hal ini menjadi alasan mengapa ada permintaan yang besar dari jasa pengembangan aplikasi *mobile* saat ini. Perusahaan yang beralih ke pengembangan *cross mobile service platform* untuk mendapatkan aplikasi *mobile* bisnis yang kompatibel di seluruh *platform mobile*, baik itu sistem operasi Android, Windows Phone maupun IOS. *Cross Platform* adalah istilah dalam teknologi informasi mengenai sebuah perangkat lunak yang dapat digunakan di beberapa sistem operasi yang berbeda (Microsoft Windows, Linux, Mac OS, BSD dan lain sebagainya). Umumnya perangkat lunak yang memiliki kemampuan lintas *platform* adalah perangkat lunak bebas. Misalnya Photoshop dan Microsoft Office yang keduanya dapat berjalan di sistem operasi Windows dan Macintosh. Oleh karena itu Photoshop dan Microsoft Office salah satu software *cross platform*.

Cross platform biasanya di gunakan untuk pengembangan software komputer, dan juga dapat merujuk ke perangkat keras komputer. Misalnya, seperti *keyboard, mouse, printer, scanner* dan kamera digital yang dapat berjalan di lebih dari satu sistem operasi komputer. Jika sebuah *software* yang di kembangkan untuk sistem operasi lebih dari satu, maka *software* tersebut termasuk dalam *multi platform*. Keuntungan dari penggunaan teknologi *cross platform* yaitu efektivitas biaya untuk membangun solusi *cross platform* daripada membangun aplikasi khusus untuk berbagai *platform*, akses ke plugin yang mudah yang menyediakan link umum untuk API yang sama, kecepatan dalam mengembangkan aplikasi dimana hanya satu basis *code* yang akan dibuat untuk aplikasi *mobile platform* ini [7].

Aplikasi *native* dapat memberikan pengalaman yang baik bagi pengguna, namun mungkin perusahaan tidak memiliki *budget* yang cukup besar atau keahlian untuk mengembangkan *native*. Pendekatan *hybrid* menawarkan solusi sederhana untuk mengembangkan aplikasi untuk *smartphone* dan *tablet*. Dengan menulis code program sekali dan menyebar ke sistem operasional yang berbeda akan membantu perusahaan untuk mendapatkan aplikasi *mobile* secara cepat dan mengurangi biaya pemeliharaan (*maintenance*). Struktur *hybrid* adalah pilihan yang cocok dalam penggunaan aplikasi untuk bisnis ataupun pendidikan. Disarankan di masa depan dapat menjadi solusi dalam peningkatan kinerja dan penghematan yang dihasilkan untuk aplikasi hibrida sesuai dengan kebutuhan dan ukuran yang berbeda [8].

Analisis dari beberapa solusi cross-*platform* menunjukkan PhoneGap adalah lebih banyak digunakan. Ponsel aplikasi *Web* menawarkan entri cepat dan sederhana menjadi pengembangan *cross-platform*. Singkatnya, kematangan pendekatan ini mengungkapkan bahwa pembangunan *native* tidak diperlukan ketika mengimplementasikan sistem informasi untuk *mobile*. Bahkan jika hanya satu *platform* yang akan didukung, pendekatan *cross platform* dapat membuktikan sebagai metode yang paling efisien dikarenakan hambatan yang rendah. HTML, CSS, dan JavaScript sejalan dengan paradigma *Web* sangat cocok untuk

mengembangkan aplikasi *cross-platform* karena standar, populer, cukup sederhana namun kuat dan didukung dengan baik [9].

Perbandingan dilakukan untuk menganalisis OS iOS dan ekosistem Android yang kita pada dasarnya bisa port beberapa fungsi *platform* iOS pada subsistem Android. Juga membantu kita untuk mengubah masalah untuk model matematika melalui mana bisa mulai bekerja pada fungsi dari IDE. Akhirnya setelah menyimpulkan semua ulasan yang terkait dengan topik *Cross Platform* Pengembangan Aplikasi *Mobile,* konsep yang berkaitan dengan sistem operasi *mobile* dan bagaimana sistem operasi *mobile* yang tampaknya bekerja sangat berbeda dari satu sama lain. Setelah semua percobaan ini adalah kita dapat membangun lingkungan pengembangan terpadu seperti Eclipse atau Netbeans yang akan membantu dalam mencapai tujuan pengembang.

Pada dasarnya dalam IDE ini pengembang akan diberikan pilihan untuk mengembangkan aplikasi-nya baik di Java atau Objective C. Setelah desain GUI dan bekerja pada bagian *coding* dan setelah itu debugs dan memastikan aplikasi nya selesai lalu akan diberi pilihan untuk menyebarkan aplikasi untuk iOS atau Android. Pada dasarnya IDE kami akan melakukan semua konversi kode, perubahan semantis dan perubahan logis diperlukan untuk menyebarkan aplikasi untuk *platform* yang berbeda. IDE juga akan menampilkan *benchmarking* atau menganalisis alat yang sangat canggih yang akan membantu pengembang untuk *debug* programnya, memeriksa kebocoran memori dan melakukan hal-hal lain yang berkaitan dengan bagian pengujian. IDE ini juga akan menjadi *platform* independen cukup yaitu, akan ada varian untuk sistem berbasis Linux dan sistem berbasis Macintosh juga [10].

2.5 Android

Android adalah sebuah sistem operasi pada *handphone* yang bersifat terbuka dan berbasis pada sistem operasi Linux. Android bisa digunakan oleh setiap orang yang ingin menggunakannya pada perangkat mereka. Android menyediakan *platform* terbuka bagi para pengembang untuk menciptakan aplikasi mereka sendiri yang akan digunakan untuk bermacam peranti bergerak. Awalnya, Google

Inc. membeli Android Inc., pendatang baru yang membuat peranti lunak untuk ponsel. Kemudian untuk mengembangkan Android, dibentuklah Open Handset Alliance, konsorsium dari 34 perusahaan peranti keras, peranti lunak, dan telekomunikasi, termasuk Google, HTC, Intel, Motorola, Qualcomm, T- *Mobile*, dan Nvidia. Pada saat perilisan perdana Android, 5 November 2007, Android bersama Open Handset Alliance menyatakan mendukung pengembangan standar terbuka pada perangkat seluler. Di lain pihak, Google merilis kode–kode Android di bawah lisensi Apache, sebuah lisensi perangkat lunak dan standar terbuka perangkat seluler. Berikut merupakan kelebihan dan kekurangan dari Android [11].

1. Kelebihan Android
   a. *Multitasking*

      Android yang mampu membuka beberapa aplikasi sekaligus tanpa harus menutup salah satunya.
   b. Kemudahan dalam Notifikasi

      Setiap ada SMS, Email, atau bahkan artikel terbaru dari RSS Reader, akan selalu ada notifikasi di *home screen* Ponsel Android dengan Lampu LED Indikator yang berkedip-kedip.
   c. Akses Mudah Aplikasi Android lewat Google Android App Market

      Dengan Google Android App Market dapat mendownload berbagai aplikasi dengan gratis maupun berbayar. Ada banyak ribuan aplikasi dan games yang siap untuk didownload di ponsel Android.
   d. Pilihan Ponsel yang beraneka ragam

      Android tersedia di ponsel dari berbagai produsen, mulai dari Sony Ericsson, Motorola, HTC sampai Samsung.Setiap pabrikan ponsel pun menghadirkan ponsel Android dengan gaya masing-masing.
   e. Bisa menginstal ROM yang dimodifikasi

      banyak Costum ROM yang bisa dipakai di ponsel Android guna merubah tampilan.

f. Widget

Dengan adanya Widget di *homescreen* pengguna bisa dengan mudah mengakses berbagai *setting* dengan cepat dan mudah.

g. Google Maniac

Kelebihan Android lainnya jika pengguna setia layanan Google mulai dari Gmail sampai Google Reader, ponsel Android telah terintegrasi dengan layanan Google, sehingga dapat dengan cepat mengecek email dari Gmail.

2. Kelemahan Android

a. Koneksi Internet yang terus menerus

Ponsel berbasis system ini memerlukan koneksi internet yang simultan/terus menerus aktif. Koneksi internet GPRS selalu aktif setiap waktu, dimana pengguna harus siap berlangganan paket GPRS yang sesuai dengan kebutuhan.

b. Iklan

Aplikasi di Ponsel Android memang bisa didapatkan dengan mudah dan gratis, namun konsekuensinya di setiap Aplikasi tersebut, akan selalu Iklan yang terpampang, entah itu bagian atas atau bawah aplikasi.

2.6 Windows Phone

Teknologi Windows Phone merupakan sistem operasi ponsel terbaru. Windows Phone atau Microsoft Windows Phone merupakan sistem operasi untuk perangkat *mobile* yang dikembangkan oleh Microsoft. Sistem operasi (OS) ini telah ditanamkan pada *smartphone* (ponsel pintar) yang diproduksi oleh Nokia. Versi terbaru Windows Phone 10 merupakan versi yang dipersiapkan untuk melawan Android secara langsung, versi ini telah menjalin kontrak kesepakatan dengan vendor ponsel Nokia dengan ponsel andalan seri Nokia Lumia.

Aplikasi yang dimiliki oleh Windows Phone hampir sama dengan Windows OS pada umumnya yang dapat mengoperasikan hampir semua aplikasi

yang dimiliki Windows seperti Microsoft Office, Windows Live, games plus nada dering pribadi. Window phone yang paling terbaru adalah Windows Phone 10 berikut ini adalah beberapa kelebihan dari Windows Phone[12] :

1. *Design User Interface* yang keren

   User Interface (UI) di Windows phone memiliki tampilan design yang cantik meski dengan halaman yang penuh dengan *shortcut*. *Shortcut* di sini selain menggantikan fungsi windows juga bertujuan untuk memudahkan user untuk mengakses fitur-fitur yang tersedia.

2. Desain arsitektur prosesor lengkap

   Windows 10 dilengkapi oleh ARM dan x86. ARM dan x86 adalah desain arsitektur prosesor. ARM lebih efisien terhadap daya, jadi lebih hemat baterai. Sedangkan x86 memiliki kekuatan yang ada di chip Intel pada umumnya, kompatibel dengan *software* yang didesain untuk Windows 2007 dan lebih tua dari itu.

3. *Multitasking*

   Windows telah lama dikenal sebagai sistem operasi yang memiliki fitur *multi-tasking* yang lengkap dan kompeten sehingga dapat menghemat waktu anda dalam bekerja dan bersosialiasi di dunia maya.

4. Waktu *booting* dan *loading* cepat

   Perangkat iOS dan Android membutuhkan waktu setidaknya setengah menit untuk hidup dan siap digunakan. Sedangkan Windows 10 hanya membutuhkan waktu 8 detik untuk *booting*. Jauh lebih singkat dibanding *booting* di Windows versi sebelumnya.

5. Microsoft Office

   Dengan adanya dukungan sepenuhnya dari Microsoft Office, anda bisa membuat dan mengedit dokumen baik dalam bentuk Word, Excel dan PowerPoint. Selain itu anda juga dapat mengunduh berbagai aplikasi seru.

2.7 IOS

IOS (sebelumnya iPhoneOS) adalah sistem operasi perangkat bergerak yang dikembangkan dan didistribusikan oleh Apple Inc. Sistem operasi

ini pertama diluncurkan tahun 2007 untuk iPhone dan iPod Touch, dan telah dikembangkan untuk mendukung perangkat Apple lainnya seperti iPad dan Apple TV.

Antarmuka pengguna iOS didasarkan pada konsep manipulasi langsung menggunakan gerakan multisentuh. Elemen kontrol antarmukanya meliputi slider, switch, dan tombol. Interaksi dengan SO ini mencakup gerakan seperti geser, sentuh, jepit, dan jepit buka, masing-masing memiliki arti tersendiri dalam konteks sistem operasi iOS dan antarmuka multisentuhnya. Akselerometer internalnya dipakai oleh sejumlah aplikasi agar bisa merespon terhadap pengguncangan alat (misalnya membatalkan tindakan) atau memutarnya dalam tiga dimensi (misalnya beralih dari mode potret ke landskap) [13].

IOS diturunkan dari OS X, yang memiliki fondasi Darwin dan karena itu iOS merupakan sistem operasi Unix. iOS adalah versi bergerak dari sistem operasi OS X yang dipakai di komputer-komputer Apple. Di iOS, ada empat lapisan abstraksi, yaitu Core OS, Core Services, Media, dan Cocoa Touch. Versi terbaru sistem operasi ini (iOS 7) menyisihkan 1,5 s.d. 2 GB memori perangkat bergerak untuk partisi sistem dengan memakai 800 MB partisi (tergantung model) untuk iOS-nya saja.

2.8 Apache Cordova

Apache Cordova adalah *platform* untuk membangun *native mobile applications* menggunakan HTML, CSS dan JavaScript. *Native mobile application* yang didukung antara lain Android, iOS, Windows Phone dan Blackberry. Apache Cordova berisi sekumpulan API (*Application Programming Interface*) untuk mengakses device dari perangkat *mobile*. Device itu antara lain kamera, GPS (*Global Positioning System*), *storage* dan lain-lain. Dengan menunggunakan UI (*User Interface*) framework seperti jQuery *Mobile*, Dojo *Mobile* atau Sencha Touch, maka dapat mengakses API ini. Dengan kata lain pembangun dapat membangun aplikasi hanya menggunakan HTML, CSS dan Javascript.

Dengan menggunakan API dari Cordova, tidak perlu membangun aplikasi menggunakan native code seperti Java, Objective-C dan lainnya. Pengembang hanya menggunakan teknologi *web* dengan bahasa pemrograman *web* dan diinstall pada perangkat *mobile* yang bersangkutan. Karena menggunakan Javascript, aplikasi yang dibangun pada suatu *platform mobile* dapat digunakan di *platform* lain dengan sedikit atau tanpa perubahan. Aplikasi yang dihasilkan dari Cordova adalah dikemas dalam aplikasi menggunakan SDK masing-masing *platform* dan dapat diterapkan ke dalam *platform* lain menggunakan SDK *platform* tersebut. Cordova digunakan untuk beberapa fungsional diantaranya [14]:

1. Seorang programmer untuk perangkat *mobile* yang multi *platform,* cukup membuat sekali coding tanpa harus melakukan implementasi ulang pada setiap *platform*.
2. Seorang *web* developer di mana ada aplikasi yang perlu dijalankan pada perangkat *mobile* untuk mengakses aplikasi yang dibuatnya.
3. Seorang programmer perangkat *mobile* yang tertarik membangun plugin dalam *WebView*.

2.9 *JavaScript Object Notation* (JSON)

JavaScript Object Notation (JSON) adalah format pertukaran data yang ringan, mudah dibaca dan ditulis oleh manusia, serta mudah diterjemahkan dan dibuat (*generate*) oleh komputer. JSON merupakan format teks yang tidak bergantung pada bahasa pemprograman apapun karena menggunakan gaya bahasa yang umum digunakan oleh programmer keluarga C termasuk C, C++, C#, Java, JavaScript, Perl, Python dll. Oleh karena sifat-sifat tersebut, menjadikan JSON ideal sebagai bahasa pertukaran-data. JSON terbuat dari dua struktur [15]:

1. Kumpulan pasangan nama/nilai. Pada beberapa bahasa, hal ini dinyatakan sebagai objek (*object*), rekaman (*record*), struktur (*struct*), kamus (*dictionary*), tabel *hash* (*hash table*), daftar berkunci (*keyed list*), atau *associative array*.

2. Daftar nilai terurutkan (*an ordered list of values*). Pada kebanyakan bahasa, hal ini dinyatakan sebagai larik (*array*), vektor (*vector*), daftar (*list*), atau urutan (*sequence*).

Struktur-struktur data ini disebut sebagai struktur data universal. Pada dasarnya, semua bahasa pemprograman moderen mendukung struktur data ini dalam bentuk yang sama maupun berlainan. Hal ini pantas disebut demikian karena format data mudah dipertukarkan dengan bahasa-bahasa pemrograman yang juga berdasarkan pada struktur data ini. JSON menggunakan bentuk sebagai berikut:

1. Objek

   Objek adalah sepasang nama/nilai yang tidak terurutkan. Objek dimulai dengan { (kurung kurawal buka) dan diakhiri dengan } (kurung kurawal tutup). Setiap nama diikuti dengan : (titik dua)dan setiap pasangan nama/nilai dipisahkan oleh , (koma).

2. Larik

   Larik adalah kumpulan nilai yang terurutkan. Larik dimulai dengan [(kurung kotak buka) dan diakhiri dengan] (kurung kotak tutup). Setiap nilai dipisahkan oleh , (koma).

3. Nilai

   Nilai (*value*) dapat berupa sebuah *string* dalam tanda kutip ganda, atau *angka*, atau *true* atau *false* atau *null*, atau sebuah objek atau sebuah larik. Struktur-struktur tersebut dapat disusun bertingkat.

4. *String*

   String adalah kumpulan dari nol atau lebih karakter Unicode, yang dibungkus dengan tanda kutip ganda. Di dalam string dapat digunakan *backslash escapes* "\" untuk membentuk karakter khusus. Sebuah karakter mewakili karakter tunggal pada string. String sangat mirip dengan string C atau Java.

5. Angka

   Angka adalah sangat mirip dengan angka di C atau Java, kecuali format oktal dan heksadesimal tidak digunakan.

6. Spasi kosong (*whitespace*)

   Spasi kosong dapat disisipkan di antara pasangan tanda-tanda tersebut, kecuali beberapa detil *encoding* yang secara lengkap dipaparkan oleh bahasa pemprograman yang bersangkutan.

2.10 *Android Studio*

Android Studio adalah sebuah IDE untuk pengembangan aplikasi di *platform* Android. Saat ini usia Android Studio masih tergolong muda, baru versi 0.2.3 (masih *early access preview*). SDK sebelumnya di-*bundle* bersama dengan Eclipse, sementara Android Studio menggunakan IntelliJ IDEA Community Edition. Beberapa pendukung IntelliJ IDEA sering mengatakan bahwa Eclipse terlalu rumit bagi pemula. Android Studio menggunakan Gradle untuk manajemen proyeknya. Gradle adalah *build automation tool* yang dapat dikonfigurasi melalui DSL berbasis Groovy. Ini yang membedakan Gradle dari Ant atau Maven yang memakai XML. Penggunaan DSL berbasis Groovy menyebabkan Gradle lebih fleksibel dan dapat diprogram dengan mudah.

Android Studio build system digunakan untuk membangun, tes, menjalankan, dan membuat paket dari aplikasi yang dibangun. *Build system* tidak tergantung (*independent*) dari Android Studio, jadi *developer* dapat memanggilnya dalam Android Studio atau dengan menggunakan *command line*. Setelah menuliskan code pada aplikasi yang akan dibangun, *developer* dapat menggunakan fitur *build system* untuk [16] :

1. Kostumisasi, konfigurasi, dan meng-*extends* proses build
2. Membuat beberapa apk untuk aplikasi android dengan fitur yang berbeda menggunakan project yang sama
3. Menggunakan ulang *code* dan *resources*

Fleksibilitas dari system pengembangan Android Studio memungkinkan untuk mencapai keseluruhan hal ini tanpa harus memodifikasi file inti dari projek. Android Studio ini adalah lingkungan pengembangan baru dan terintegrasi penuh, yang baru saja dirilis oleh Google untuk sistem operasi Android. Android Studio dirancang untuk menjadi peralatan baru dalam pengembangan aplikasi dan juga memberi alternative lain selain Eclipse yang saat ini menjadi IDE yang paling dipakai. Saat memulai proyek baru dengan Android Studio, struktur proyek akan muncul bersama dengan hampir semua berkas yang ada di dalam direktori SDK, peralihan ke sistem manajemen berbasis Gradle ini memberikan fleksibilitas yang lebih besar pada proses pembangunannya.

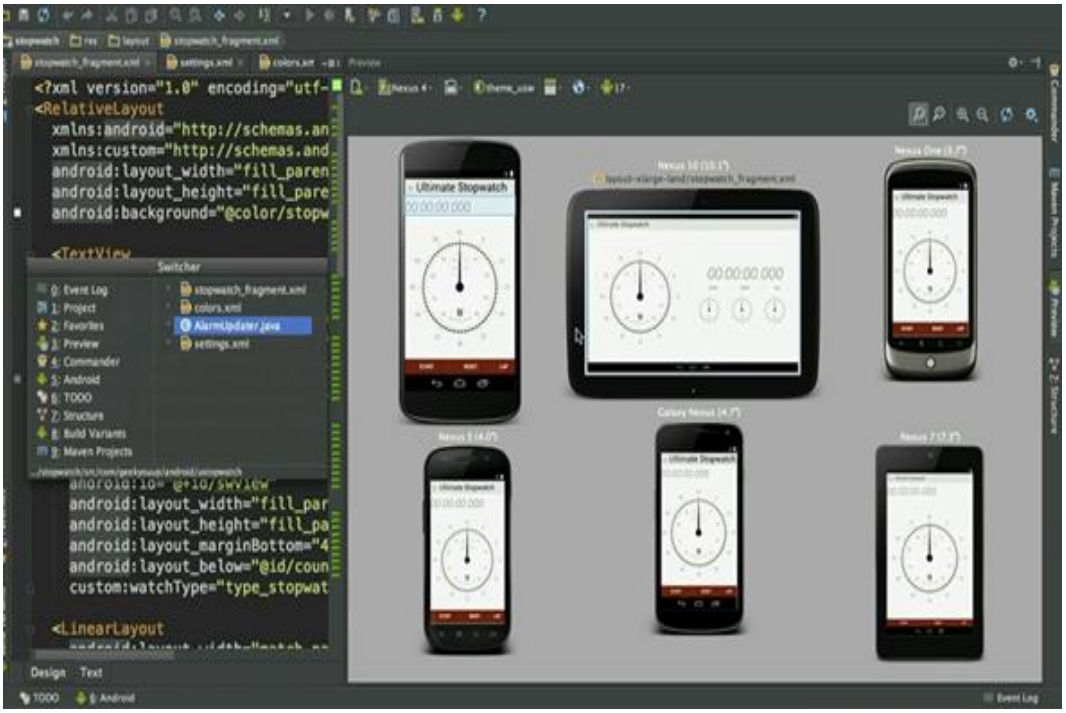
Android Studio developer dapat melihat perubahan visual apapun yang dilakukan pada aplikasi secara langsung.Dapat terlihat perbedaannya jika dipasang pada beberapa perangkat Android berbeda, termasuk konfigurasi dan resolusi secara bersamaan. Fitur lain di Android Studio adalah alat alat baru untuk *packing* dan memberi label *code*. Dengan begitu mengijinkan *developer* tetap menjadi yang teratas ketika berurusan dengan banyak *code*. Program ini juga menggunakan sistem *drag and drop* untuk memindahkan komponen melalui *interface user*.

Selain itu, lingkungan baru ini juga mendukung Google Cloud Messaging. Sebuah fitur yang mengijinkan untuk mengirim data dari server ke perangkat Android melalui *cloud*, cara terbaik untuk mengirim *Reminder* pada apps. Program ini juga membantu untuk melokalisasi aplikasi anda, memberi gambaran visual untuk tetap memprogram sambil mengontrol alur dari aplikasi. Kemudahan yang diberikan oleh Android Studio adalah sebagai berikut :

1. Lingkungan pengembangan yang mantap dan bersifat langsung.
2. Cara termudah untuk menguji performa pada perangkat dengan tipe lain.
3. Wizard dan template berisi elemen – elemen umum yang ada di semua pemrograman Android.
4. Editor dengan fitur lengkap dengan banyak peralatan ekstra untuk mempercepat pengembangan aplikasi anda. *Layout editor* yang

memungkinkan untuk *drag and drop* komponen UI, pratinjau *layout* pada beberapa konfigurasi layar, dan banyak lagi.

5. Berbasis Gradle.
6. Android spesifik *refactoring* dan perbaikan yang cepat.
7. Alat Lint untuk menangkap kinerja, kegunaan, versi kompatibilitas dan masalah lainnya.
8. ProGruard dan *app-signature*.
9. *Built-in* dukungan untuk Google Cloud *platform*, sehingga mudah untuk mengintegrasikan Google Cloud Messaging dan App Engine sebagai komponen *server-side*.



Gambar 2.4 *Android Studio*

2.11 Windows Phone SDK

Windows Phone SDK, merupakan suatu alat pengembangan yang bertujuan untuk membantu membuat aplikasi dan game untuk Windows Phone 8.0 atau 7.5. Pada Windows Phone SDK sudah termasuk alat–alat dasar dan dokumentasi yang diperlukan untuk melakukan pengkodean program pada perangkat bergerak (*mobile*). Windows phone SDK akan menyediakan berbagai macam *header files*,

libraries, contoh – contoh, serta berbagai macam *tools* yang memberikan dukungan dalam pembuatan dan pengembangan aplikasi, khususnya aplikasi perangkat *mobile* yaitu Windows Phone. Dalam perjalanannya, *tools* yang dikembangkan oleh perusahaan IT besar Microsoft ini memiliki versi terbaru, yaitu Windows Phone SDK 8.1.

Windows Phone 8.1 menggunakan bahasa pemograman XAML untuk UI (*User Interface*) dan C# atau Visual Basic untuk proses. Selain kedua bahasa itu juga terdapat bahasa C++ yang dapat digunakan khususnya dalam pengembangan Direct3D Windows Phone 8.1 sehingga pengembang akan mendapatkan banyak keuntungan dalam bidang grafis di aplikasi yang dibuat nantinya. Windows Phone SDK ini dapat menggunakan Silverlight dan XNA Framework dalam satu aplikasi atau game yang dengan kata lain kelengkapan model navigasi didapat dari *silverlight* dan *graphics* banyak didapat pada aplikasi XNA Framework. Kebutuhan sistem yang menjadi syarat minimal agar dapat menggunakan Windows Phone SDK ini adalah :

1. Komputer dengan arsitektur prosesor x64 *based* (64bit)
2. Besar RAM 4 GB
3. Besar ruang penyimpanan kosong (HDD) minimal 4 GB
4. Menggunakan sistem operasi Windows 8 Pro 64-bit atau yang lebih tinggi

Pada pengembangannya dari berbagai versi, fitur dan kelebihan yang diberikan oleh Windows Phone SDK antara lain [17]:

1. Lebih banyak pengembangan fungsi – fungsi baru yang dapat digunakan untuk memaksimalkan aplikasi yang dibuat. Dapat bekerja pada Windows Phone OS 8 maupun versi sebelumnya, seperti Windows Phone 7.1. Dapat melakukan *multitasking*, karena sudah adanya dukungan untuk fitur tersebut. Dapat membuat aplikasi dengan memanfaatkan perangkat *sound* dan *timer* yang ada pada perangkat *mobile*. Seperti membuat suatu aplikasi alarm *reminder* atau yang lainnya.
2. Aplikasi berjalan lebih cepat saat melakukan *switching* antar aplikasi maupun *multitasking* aplikasi. Terdapat *Background Agent* yang dapat

bekerja di belakang walaupun aplikasi berjalan di depan. Terdapat *Background Audio* yang memungkinkan audio bekerja saat aplikasi lain sedang bekerja. Menggunakan tampilan 32 bit dan memiliki kualitas video yang lebih baik dengan penggunaan batrai lebih hemat. Menggunakan sensor akselometer yang mengetahui arah dan kemiringan sehingga memungkinkan untuk membuat aplikasi maupun game fisik yang terpengaruh kemiringan ataupun letak *mobile*.

3. Aplikasi dapat terhubung melalui TCP dan UDP yang memungkinkan adanya hubungan 2 arah sehingga dapat melakukan pesan *instant* dan *multi-player gaming*. Mendapatkan infomasi mengenai akses *network* yang digunakan dan juga tampilan *network*. Notifikasi pada tiap aplikasi. Dapat memberikan analisa performa aplikasi.
4. Mensupport Visual Basic pada silverlight maupun XNA Framework. Mempermudah pembuatan dan penjualan aplikasi dengan disediakannya ads dari Microsoft Advertising. *Web*Browser support IE 9 dan HTML 5. Terdapat fitur Device Status, contohnya adalah saat pengguna aplikasi menggunakan baterai atau *charger*.
5. Adanya *Local Database* yang memungkinkan untuk menggunakan LINQ ke SQL untuk mengoprasikan database. Adanya *command line* pada Windows Phone SDK yang memungkinkan untuk mengisolasi direktori dan file ke dalam sebuah penyimpanan. Pada Windows Phone SDK menyediakan berbagai *Launchers*. *Read only* akses pada kontak yang ada dan data-data pada kalender.
6. Disediakannya API untuk melakukan enkripsi pada berbagai macam aplikasi yang akan dibuat. Mempermudah dalam mendesain aplikasi karena disediakan berbagai macam fitur bantuan untuk mengatur tampilan. Bisa mendapatkan akses ke *App Hub* dengan berbagai macam persyaratan yang telah tersedia di MSDN *library*.Windows Phone SDK menyediakan *market place Test Kit* untuk menguji aplikasi pengembang agar mendapat persetujuan yang nantinya dapat di-*deploy* langsung ke *App Hub*.

7. Windows Phone SDK memberikan akses ke kamera sehingga dapat dihubungkan dengan aplikasi yang dibuat. Ada dua ukuran tampilan Aplikasi yang dapat digunakan yakni *default* dan *minimized*. Fitur *On-screen keyboard*. Adanya *tray* sistem dan indikator *progress*. Windows Phone SDK memungkinkan untuk berhubungan langsung dengan *proxy*. Developer dapat membuat aplikasi *trial*. Adanya Windows phone market yang tersedia.

2.12 Xcode

Xcode merupakan sebuah IDE atau dikenal *suite of tools* yang diciptakan oleh perusahaan Apple untuk mengembangkan basis *platform*nya yakni Mac OS X dan IOS, yang mengindikasikan sebagai *platform* utama bagi pengembangan *software* khusus Apple. *Platform* ini bersifa *native app* yang tersedia di seri Mac OS generasi terbaru. Xcode adalah *platform* native yang dikembangkan langsung oleh Apple untuk mengembangkan OS X dan iOS . Xcode merupakan integrated Development Environment (IDE) yang juga disertakan dalam generasi OS X terkini . dirilis pada tahun 2003 melalui seri 1x ,dan tersedia di seri terkini yakni versi 5.0. Xcode mendukung langsung pengembangan untuk basis smartphone Apple yakni Ipad dan Iphone .

Xcode hadir pertama pada tahun 2003 .seri pertama ini dikenal dengan versi 1.x . saat itu *platform* ini berbasiskan *project builder*, namun sudah memiliki kemampuan UI, ZeroLink, Fix&Continue, *support* pengembangan distribusi, dan *code sense indexing*. Setelah itu berturut-turut diikuti versi terbaru yakni 2.x *series*,3.x *series*, 4.x *series* dan terkini versi 8.0. *Generator* kode yang digunakan oleh Xcode adalah LLVM,dan LLDB sebagai *debugger default*(versi 4.3). seri Xcode pertama hadir di OS Mac OS X 10.3 yakni versi 1.x, setelah itu hadir sebagai aplikasi yang dapat digunakan untuk OS Mac generasi selanjutnya .

Pada perkembangan awal, Xcode mendukung distribusi dari beberapa sistem yakni *workgroup* guild. Menariknya xcode juga menyediakan fasilitas untuk membuat *Dedicated Network Builds*, namun pada versi terbarunya tidak lagi mendukung fasilitas ini. Xcode meruapakan perangkat *Web*Objects yang

mampu menjadi framework pembuatan aplikasi *web*. tampilan instrumen Xcode adalah GUI dan berjalan di atas Dtrace. Untuk mengembangkan aplikasi pada Iphone dan Ipad disediakan Xcode emulator, namun sebelumnya perlu men-download iOS yang disesuaikan dengan perangkat pendukungnya [18].

Platform ini menggunakan bahasa Objective-C sebagai pengembangannya. bahasa pengembangan dari C, yang sekaligus menjadi bahasa *native* untuk membuat aplikasi mac. Salah satu ciri bahasa ini adalah penggunaan *style* dari Smalltalk. Bahasa dari *platform* ini bersifat *objek oriented* ,namun tetap mendukung bahasa terstruktur. Selain itu *platform* ini juga mendukung bahasa C, C++ , Objective-C, Objective-C++, Objective-C++, Java, AppleScript, python dan bahasa pengembangan model dari yang sudah dikenal. pengembang ketiga juga mensupport bahasa seperti Pascal,free pascal , c# ,perl. Sebagai Aplikasi *Native* dari Apple, Xcode dapat di download di apple-Xcode sebagai development software. Apple menyediakan Xcode secara gratis, selain itu terdapat juga Xcode Developer Preview, yang ditujukan kepada *Registered Developers*.

2.13 PhoneGap

PhoneGap adalah sebuah kerangka kerja/*framework* pembuat aplikasi *mobile* berbasis *open source* yang dikeluarkan oleh sebuah perusahaan di Amerika yang bernama Nitobi dengan lisensi MIT License. *Framework PhoneGap* memungkinkan seseorang untuk mengembangkan aplikasi native *mobile* dengan menggunakan keahlian *HTML, CSS* dan *Javascript*. *PhoneGap* menjadi suatu solusi yang ideal untuk seorang *web developer* yang tertarik dalam pembuatan aplikasi di *smartphone*. Sebuah aplikasi yang dibuat dengan menggunakan *framework PhoneGap* dapat dideploy ke berbagai *platform* sekaligus. Saat ini *PhoneGap* mendukung 7 *platform*, yaitu sebagai berikut :

1. iOS : iPhone, iPad
2. Android
3. Windows *Mobile*
4. Blackberry
5. *Web*OS

6. Symbian
7. Beda

Dari ketujuh *platform* diatas PhoneGap mendukung fitur-fitur, seperti *Accelorometer, Camera, Compass, Contact, File, Geolocation, Media, Network, Notification* dan *Storage*. *PhoneGap Storage* merupakan salah fitur yang dapat digunakan untuk menyimpan dan mengakses data ke dalam database lokal. *PhoneGap* adalah *library* yang digunakan untuk *porting* dari *web apps* ke *native apps*. *PhoneGap* menghasilkan *native web apps* sehingga hasil pengembangan *PhoneGap* bisa dimasukkan ke dalam *Store*, misalnya *iOS Apps Store*, *Android Market*, dan sebagainya [19].

Aplikasi berdasarkan kerangka PhoneGap yang dikembangkan dalam makalah ini memiliki keuntungan berikut terhadap aplikasi tradisional perangkat pintar :

1. Menggunakan semua teknologi sudah terkenal di seluruh dunia membuatnya mudah memulai.
2. Dengan cara sederhana untuk mengembangkan , total waktu pembangunan menjadi lebih pendek . Hal ini benar-benar bermanfaat untuk aplikasi seperti majalah perlu dipublikasikan per bulan.
3. Cross-*platform* adalah titik paling menarik dari pendekatan ini ; Anda dapat dengan mudah membuat aplikasi Anda berjalan perangkat lain jika memiliki browser apapun. Benar-benar Anda butuhkan untuk menyesuaikan aplikasi Anda untuk perangkat yang berbeda dengan ukuran layar yang berbeda atau *browser* [20].

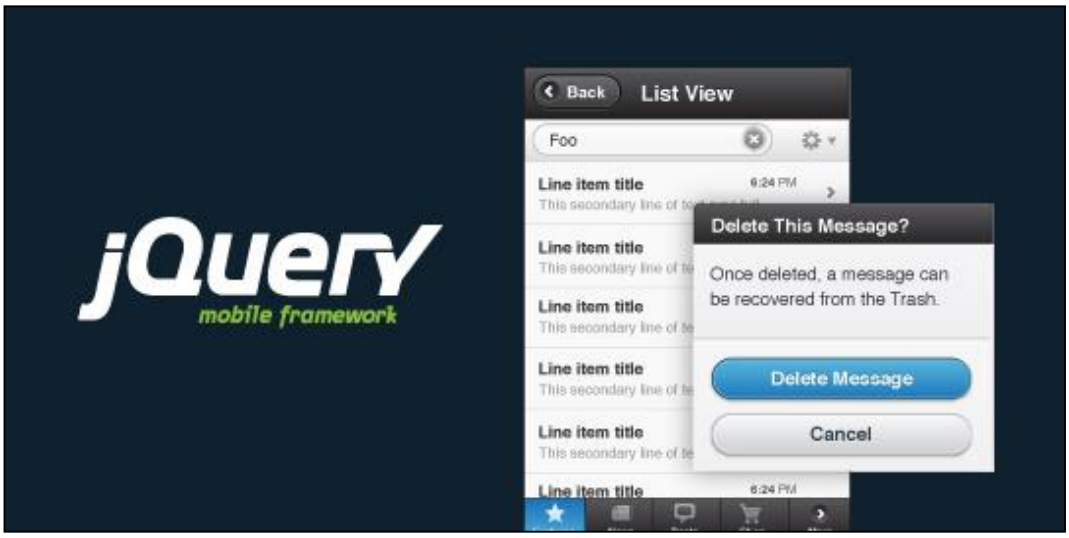


Gambar 2.5 *PhoneGap* menggunakan *HTML5, CSS3,* dan *JavaScript*

2.14 Jquery*Mobile*

Jquery *Mobile* adalah *web* framework yang dioptimalkan untuk perangkat-perangkat layar sentuh(*touch -optimized*). Sebagai *library*, Jquery yang dikenal sebagai *jquery javascript* maupun framework *mobile* ini dikembangkan oleh tim jquery Project. Tujuan Pengembangannya adalah berfokus pada pembuatan *framework* yang kompatibel dengan beraneka ragam perangkat *smartphone* dan tablet [21].

Jquery*Mobile* merupakan *system interface* berbasis HTML5 dan pengembangan dari *library jquery* dan *jquery* UI yang sedang popular saat ini. Jquery *Mobile* sangat mungkin untuk membuat aplikasi *web mobile* yang penampilan dan interaktifitasnya konsisten di semua perangkat yang memiliki kemampuan *User Interface* (UI) yang canggih.



Gambar 2.6 *JqueryMobile Framework*

Jquery *Mobile* dapat melakukan beberapa hal sesuai dengan kegunaannya, diantaranya adalah sebagai berikut ini :

1. Jquery *mobile* memudahkan pengembangan user interface untuk *mobile web apps.*
2. *The interface configuration is marup-driven*, yang berarti dapat membuat seluruh aplikasi antarmuka dasar HTML, tanpa perlu menulis satu baris javascript.
3. Menyediakan serangkaian *custom events* baru,mendeteksi *events mobile device* dan *touchscreen* seperti *tap,tap-and-hold,swipe*, dan *orientation change(i.e. rotating the device).*
4. Memastikan bahwa *user interface* bekerja pada *web browser*.
5. Menggunakan *theme* untuk memudahkan *costumized* tampilan aplikasi.

Manfaat dari jquery *mobile* bagi seorang desainer *web*site adalah :

1. Membuat pengguna javascript lebih sederhana karena cukup memakai fungsi dari *library javascript* yang tersedia.
2. Meringkas kode javascript dalam membuat sebuah *website*, jika dibandingkan apabila harus memulai sebuah javascript mulai dari awal.

3. Setiap fungsi yang disediakan di dokumentasi dengan sistematis bersama contoh penggunaannya. Hal ini mempermudahkan dalam mempelajari jquery.
4. Dapat menambahkan animasi kedalam *website*.
5. Mendukung CSS-3selector, yang membuat *web*site lebih *flexible* antar halaman depan *website* dengan interaksinya.
6. Umumnya *website* yang dibuat menggunakan jquery menjadi lebih cantik, menarik dan interaktif.

Kelebihan Jquery *Mobile* adalah sebagai berikut :

1. Dokumentasi yang lebih baik

   Dokumentasi yang disediakan pada *web* Jquery *Mobile* sangatlah mudah dipahami,dokumentasi juga dikemas dalam bentuk yang menyenangkan, yaitu dengan melampirkan contoh tampilan yang di hasilkan.
2. Mendesain *Template* dengan ThemeRoller

   Pengguna Jquery *Mobile* juga di manjakan dengan fasilitas ThemeRoller, ThemeRoller digunakan untuk mengganti warna dasar desain tampilan sesuai kebutuhan dengan cara *drag* and *drop*. Script hasil modifikasi yang dihasilkan juga dapat di unduh untuk dapat dikembangkan lebih lanjut.
3. Fitur-fitur Terbaik

   Jquery *Mobile* dikembangkan dengan *core* Jquery yang terkenal dengan desain UI nya yang ringan dan lembut, Sehingga bisa menggantikan teknologi *flash*. Kemampuan yang baik untuk layar sentuh seperti pada perangkat perangkat dengan browser IOS, Android, Blackberry, Palm*Web*OS, Symbian, Windows Phone, Meego, Opera *Mobile*, Firefox *Mobile*, Kindle, Nook dan lainnya yang menambah daya tarik jquery *mobile*. Jquery*Mobile* juga telah menggunakan markup dasar HTML5 dan CSS3. Desain tampilan yang *responsive* sangat membantu ketika ditampilkan pada *smartphone* yang mendukung fitur *accelerometer* (tampilan otomatis *vertical* dan *horizontal*)

4. Jquery*Mobile* Maps

   Penggunaan peta dalam perangkat bergerak sangatlah penting.

   Kekurangan Jquery *Mobile* adalah sebagai berikut :

1. Device gadget yang tidak universal, karena jquery*mobile* memang dirancang untuk *mobile web* bagi perangkat yang memiliki layar sentuh.
2. Tidak tepat untuk mendesain *web* yang berbasis *desktop*.

2.15 MySQL

MySQL adalah sebuah perangkat lunak sistem manajemen basis data SQL (bahasa Inggris: *database management system*) atau DBMS yang *multithread*, *multi-user*, dengan sekitar 6 juta instalasi di seluruh dunia. MySQL AB membuat MySQL tersedia sebagai perangkat lunak gratis dibawah lisensi GNU General Public License (GPL), tetapi mereka juga menjual dibawah lisensi komersial untuk kasus-kasus dimana penggunaannya tidak cocok dengan penggunaan GPL.

Tidak sama dengan proyek-proyek seperti Apache, dimana perangkat lunak dikembangkan oleh komunitas umum, dan hak cipta untuk kode sumber dimiliki oleh penulisnya masing-masing, MySQL dimiliki dan disponsori oleh sebuah perusahaan komersial Swedia MySQL AB, dimana memegang hak cipta hampir atas semua kode sumbernya. Kedua orang Swedia dan satu orang Finlandia yang mendirikan MySQL AB adalah: David Axmark, Allan Larsson, dan Michael "Monty" Widenius.

MySQL adalah sebuah implementasi dari sistem manajemen basis data relasional (RDBMS) yang didistribusikan secara gratis dibawah lisensi GPL (General Public License). Setiap pengguna dapat secara bebas menggunakan MySQL, namun dengan batasan perangkat lunak tersebut tidak boleh dijadikan produk turunan yang bersifat komersial. MySQL sebenarnya merupakan turunan salah satu konsep utama dalam basisdata yang telah ada sebelumnya, SQL (Structured Query Language). SQL adalah sebuah konsep pengoperasian

basis data, terutama untuk pemilihan atau seleksi dan pemasukan data, yang memungkinkan pengoperasian data dikerjakan dengan mudah secara otomatis.

Kehandalan suatu sistem basis data (DBMS) dapat diketahui dari cara kerja pengoptimasi-nya dalam melakukan proses perintah-perintah SQL yang dibuat oleh pengguna maupun program-program aplikasi yang memanfaatkannya. Sebagai peladen basis data, MySQL mendukung operasi basisdata transaksional maupun operasi basisdata non-transaksional. Pada modus operasi non-transaksional, MySQL dapat dikatakan unggul dalam hal unjuk kerja dibandingkan perangkat lunak peladen basis data kompetitor lainnya. Namun pada modus non-transaksional tidak ada jaminan atas reliabilitas terhadap data yang tersimpan, karenanya modus non-transaksional hanya cocok untuk jenis aplikasi yang tidak membutuhkan reliabilitas data seperti aplikasi blogging berbasis *web* (*wordpress*), CMS, dan sejenisnya. Untuk kebutuhan sistem yang ditujukan untuk bisnis sangat disarankan untuk menggunakan modus basis data transaksional, hanya saja sebagai konsekuensinya unjuk kerja MySQL pada modus transaksional tidak secepat unjuk kerja pada modus non-transaksional [22].

2.16 JavaScript

JavaScript merupakan bahasa skrip yang populer di internet dan dapat bekerja di sebagian besar penjelajah *web* populer seperti Internet Explorer (IE), Mozilla Firefox, Netscape dan Opera. Kode JavaScript dapat disisipkan ke dalam halaman *web* menggunakan tag SCRIPT.JavaScript adalah bahasa (pemrograman) yang hebat, sulit dan kurang begitu dipahami pada bidang bahasa pemrograman di komputer. Akan tetapi, kemampuan inti yang dimiliki oleh JavaScript untuk membuat aplikasi-aplikasi sehingga membuat perubahan dari cara pandang pemakaian Internet pada dekade ini. Salah satu contoh yang baik dari aplikasi tersebut adalah Google Maps [23].

Keunggulan JavaScript, yang juga dikenal sebagai ECMAScript, hadir pada semua *web* browser, sehingga memiliki kemampuan untuk menghasilkan keluaran yang sama pada semua *platform* yang didukung oleh browser. Seperti yang sudah disebutkan di atas, Google Maps, yang dapat berjalan di atas Linux, Windows dan

Mac OS. Dengan pertumbuhan pustaka (library) javascript baru-baru ini, hal ini memudahkan untuk menavigasi dokumen, memilih elemen DOM, membuat animasi, menangani event, dan mengembangkan aplikasi Ajax. Berbeda teknologi terkenal lain yang didorong oleh kepentingan komersial, JavaScript benar-benar teknologi yang cross-*platform*, bahasa pemrograman *client-side* yang bersifat bebas (untuk dimodifikasi dan gratis tentunya) dan diadopsi secara universal.

JavaScript dieksekusi pada *client side* (komputer pengguna): Sebuah *server website* mengirim javascript ke peramban milik pengguna, dan browser tersebut menginterpretasikan dan menjalankan kodenya. Semua ini terjadi dalam sebuah *sandbox*, yang menjaga agar javascript tidak menyentuh internal sistem, sehingga mencegah *malicious code* (kode jahat) menginfeksi komputer pengguna.

2.17 PHP

PHP (*Hypertext Prepocessor),* merupakan bahasa pemrograman yang digunakan secara luas untuk penanganan pembuatan dan pengembangan sebuah situs *web* dan bisa digunakan bersamaan dengan HTML. PHP diciptakan oleh Rasmus Lerdorf pertama kali tahun 1994. Pada awalnya PHP adalah singkatan dari "*Personal Home Page Tools*". Selanjutnya diganti menjadi FI ("*Forms Interpreter*"). Sejak versi 3.0, nama bahasa ini diubah menjadi "PHP: Hypertext *Prepocessor*" dengan singkatannya "PHP". PHP versi terbaru adalah versi ke-5. Berdasarkan survey Netcraft pada bulan Desember 1999, lebih dari sejuta site menggunakan PHP, di antaranya adalah NASA, Mitsubishi, dan RedHat.

PHP adalah bahasa pemrograman script *server-side* yang didesain untuk pengembangan *web*, tetapi juga bisa digunakan sebagai bahasa pemrograman umum *(wikipedia). PHP* pertama kali di kembangkan pada tahun 1995 oleh *Rasmus Lerdorf,* namun sekarang dikelola oleh *The PHP Group*. Situs resmi PHP beralamat di http://www.php.net. Pada awalnya PHP adalah singkatan dari *Personal Home Page*, namun karena dalam perkembangannya PHP tidak hanya digunakan untuk membuat halaman *web* pribadi, PHP saat ini merupakan singkatan dari PHP: Hypertext Preprocessor, sebuah kepanjangan rekursif, yakni permainan kata dimana kepanjangannya berisi juga singkatan itu sendiri.

PHP dirilis dalam lisensi *PHP License*, sedikit berbeda dengan lisensi *GNU General Public License (GPL)*yang biasa digunakan untuk proyek Open Source. Namun penggunaan PHP tetap tidak dikenakan biaya (gratis).Kemudahan dan kepopuleran PHP sudah menjadi standar bagi programmer *web* di seluruh dunia. Menurut wikipedia pada februari 2014, sekitar 82% dari *web server* di dunia menggunakan PHP, dan menjadi dasar dari program *CMS (Content Management System)* populer seperti *Joomla, Drupal*,dan *WordPress* [24].

System kerja dari PHP diawali dengan permintaan yang beasal dari halaman *web*site oleh browser. Berdasarkan URL atau alamat *web*site dalam jaringan internet, browser akan menemukan sebuah alamat dari *web server*, mengidentifikasi halaman yang dikehendaki, dan menyampaikan segala informasi yang dibutuhkan oleh *web server*. Selanjutnya *web server* akan mencarikan berkas yang diminta dan menampilkan isinya di *browser*. *Browser* yang mendapatkan isinya segera menerjemahkan kode HTML dan menampilkannya.

Pada prinsipnya sama dengan memanggil kode HTML, namun pada saat permintaan dikirim ke *web*-server, *web-server* akan memeriksa tipe *file* yang diminta user. Jika tipe file yang diminta adalah PHP, maka akan memeriksa isi script dari halaman PHP tersebut. Apabila dalam file tersebut tidak mengandung script PHP, permintaan user akan langsung ditampilkan ke *browser*, namun jika dalam file tersebut mengandung script PHP, maka proses akan dilanjutkan ke modul PHP sebagai mesin yang menerjemahkan script-script PHP dan mengolah script tersebut, sehingga dapat dikonversikan ke kode-kode HTML lalu ditampilkan ke *browser user*.

2.18 HTML

HyperText Markup Language (HTML) adalah sebuah bahasa markah yang digunakan untuk membuat sebuah halaman *web*, menampilkan berbagai informasi di dalam sebuah penjelajah *web* Internet dan pemformatan hiperteks sederhana yang ditulis dalam berkas format ASCII agar dapat menghasilkan tampilan wujud yang terintegerasi. Dengan kata lain, berkas yang dibuat dalam perangkat lunak pengolah kata dan disimpan dalam format ASCII normal sehingga

menjadi halaman *web* dengan perintah-perintah HTML. Bermula dari sebuah bahasa yang sebelumnya banyak digunakan di dunia penerbitan dan percetakan yang disebut dengan SGML (*Standard Generalized Markup Language*), HTML adalah sebuah standar yang digunakan secara luas untuk menampilkan halaman *web*. HTML saat ini merupakan standar Internet yang didefinisikan dan dikendalikan penggunaannya oleh *World Wide Web Consortium* (W3C). HTML dibuat oleh kolaborasi Caillau TIM dengan Berners-lee Robert ketika mereka bekerja di CERN pada tahun 1989 (CERN adalah lembaga penelitian fisika energi tinggi di Jenewa).

Tahun 1980, IBM memikirkan pembuatan suatu dokumen yang akan mengenali setiap elemen dari dokumen dengan suatu tanda tertentu. IBM kemudian mengembangkan suatu jenis bahasa yang menggabungkan teks dengan perintah-perintah pemformatan dokumen. Bahasa ini dinamakan Markup Language, sebuah bahasa yang menggunakan tanda-tanda sebagai basisnya. IBM menamakan sistemnya ini sebagai Generalized Markup Language atau GML.

Tahun 1986, ISO menyatakan bahwa IBM memiliki suatu konsep tentang dokumen yang sangat baik, dan kemudian mengeluarkan suatu publikasi (ISO 8879) yang menyatakan markup language sebagai standar untuk pembuatan dokumen-dokumen. ISO membuat bahasa ini dari GML milik IBM, tetapi memberinya nama lain, yaitu SGML (Standard Generalized Markup Language). ISO dalam publikasinya meyakini bahwa SGML akan sangat berguna untuk pemrosesan informasi teks dan sistem-sistem perkantoran. Tetapi di luar perkiraan ISO, SGML dan terutama subset dari SGML, yaitu HTML juga berguna untuk menjelajahi internet. Khususnya bagi mereka yang menggunakan *World Wide Web*. Versi terakhir dari HTML saat ini adalah HTML5.

HTML dokumen tersebut mirip dengan dokumen tulisan biasa, hanya dalam dokumen ini sebuah tulisan bisa memuat instruksi yang ditandai dengan kode atau lebih dikenal dengan TAG tertentu. Sebagai contoh jika ingin membuat tulisan ditampilkan menjadi tebal seperti: TAMPIL TEBAL, maka penulisannya dilakukan dengan cara: `< b> TAMPIL TEBAL</b>`. Tanda `< b>` digunakan untuk mengaktifkan instruksi cetak tebal, diikuti oleh tulisan yang ingin

ditebalkan, dan diakhiri dengan tanda untuk menonaktifkan cetak tebal tersebut.

HTML lebih menekankan pada penggambaran komponen-komponen struktur dan format di dalam halaman *web* daripada menentukan penampilannya. Sedangkan penjelajah *web* digunakan untuk menginterpretasikan susunan halaman ke gaya built-in penjelajah *web* dengan menggunakan jenis tulisan, tab, warna, garis, dan perataan text yang dikehendaki ke komputer yang menampilkan halaman *web*. Salah satu hal Penting tentang eksistensi HTML adalah tersedianya *Lingua franca* (bahasa Komunikasi) antar komputer dengan kemampuan berbeda. Pengguna Macintosh tidak dapat melihat tampilan yang sama sebagaimana tampilan yang terlihat dalam pc berbasis Windows. Pengguna Microsoft Windows pun tidak akan dapat melihat tampilan yang sama sebagaimana tampilan yang terlihat pada pengguna yang menggunakan Produk-produk Sun Microsystems. namun demikian pengguna-pengguna tersebut dapat melihat semua halaman *web* yang telah diformat [25].

HTML5 dan JavaScript telah menjadi sangat menarik (dan menyenangkan) dengan munculnya kerangka kerja dan lanskap *mobile*. PhoneGap adalah sebuah *framework open source* untuk cepat membangun aplikasi cross-*platform mobile* menggunakan HTML5, JavaScript dan CSS. Prosedur pengembangan aplikasi cross-*platform* adalah sebagai berikut: 1) Bungkus aplikasi Anda dengan PhoneGap; 2) Menyebarkan ke *platform mobile*. Membangun aplikasi untuk setiap perangkat membutuhkan kerangka kerja dan bahasa yang berbeda. PhoneGap memecahkan ini dengan menggunakan standar berbasis teknologi *Web* untuk menjembatani aplikasi *Web* dan perangkat *mobile*. Sebuah aplikasi *Web* tradisional hanya tidak bekerja tanpa jaringan. Salah satu solusi untuk masalah ini adalah dengan menggunakan dua fitur dari HTML5 Standar:

1. *Offline* aplikasi *Web*
2. Penyimpanan *database Client-side*.

Pengguna dapat menggunakan fungsi awan pada perangkat *mobile*, bekerja secara *offline* dengan aplikasi dikerahkan secara lokal pada *database* lokal, dan berbagi data dengan sisa awan ketika akan online lagi [26].

2.19 Node. Js

Nodejs pertama kali dikembangkan oleh Ryan Dahl. Framework ini dibuat untuk memperoleh *performance* aplikasi *web* yang tinggi dan mengoptimalisasikan lingkungan *concurrent* yang tinggi pula. *Framework* Nodejs memungkinkan bekerja dengan keandalan tinggi dan dengan I/O *non-blocking*. Tujuan dari *non-blocking* ini untuk menghindari proses *waiting* yang lama dan mengurangi beban CPU.Secara teknis, Nodejs dibangun menggunakan C/C++ dan berjalan di atas *framework* Javascript V8, *framework* yang dikembangkan oleh Google dan bagian dari sistem inti browser Chrome. Nodejs berisi *framework* yang didalamnya sudah terdapat *web server* sehingga kita sudah tidak perlu menggunakan *web server* lainnya seperti Apache dan IIS. Bahasa pemrograman yang digunakan pada Nodejs adalah Javascript. Portal resmi Nodejs dapat dikunjungi di alamat *web*site http://www.nodejs.org.

Secara umum, Nodejs berjalan di atas V8 Javascript Interpreter untuk menjalankan kode Javascript. Untuk menjaga performance tetap tinggi, Nodejs menerapkan solusi *event-driven*. Fitur-fitur yang ditawarkan oleh Nodejs antara lain [27]:

1. Konsumsi memori yang kecil.

   Hampir sebagian besar *web server* menghadapi masalah *concurrent request* yang tinggi untuk menangani user secara bersama-sama. Untuk penyelesaian masalah ini, umumnya *web* server menerapkan *multiple thread* sebagai solusi yang memungkinkan satu koneksi ditangani oleh satu *server thread*. Kita sudah mengetahui bahwa penggunaan *thread* ini akan mengalokasi *memory stack*. Semakin banyak *thread* yang berjalan, maka penggunaan konsumsi memori juga semakin meningkat. Jika jumlah *user* meningkat, hal ini akan menjadi tidak bagus. Nodejs mempunyai cara berbeda dalam masalah ini. Pendekatan yang dilakukan adalah *event driven*

berbasis infinite *event loop* dalam satu *thread*. Dengan solusi ini, penggunaan memori akan berkurang.

2. Operasi *Non-Blocking*

   Arsitektur Node yang dibangun dengan pendekatan *non-blocking* merupakan fitur utama dari *framework* Nodejs. Dengan pendekatan ini, ketika kita ingin mengakses *database*, membaca data pada file, atau mengirim data melalui *Socket*, kita harus membuat fungsi *callback*. Setelah melakukan eksekusi kode, Nodejs akan melakukan pekerjaan lain sampai data tersedia, baik pekerjaan itu sukses atau tidak, sehingga kita dapat mengontrolnya kembali.

3. Penggunaan CPU yang lebih kecil

   Nodejs sangat peduli terhadap kinerja CPU. Pada operasi yang melibatkan banyak *thread*, *server* akan meningkatkan penggunaan CPU ketika melayani permintaan dari *user*. Pendekatan inilah yang menimbulkan penggunaan CPU yang tinggi. Nodejs menghindari permasalah di atas dengan *single thread* yang memanfaatkan *event loop* dan *callback*. Nodejs dapat melakukan skalabilitas dengan memanfaatkan *load balancer* atau *proxy server*.

2.20 *Web*

Website merupakan halaman situs sistem informasi yang dapat diakses secara cepat. *Website* ini didasari dari adanya perkembangan teknologi informasi dan komunikasi. Melalui perkembangan teknologi informasi, tercipta suatu jaringan antar komputer yang saling berkaitan. Jaringan yang dikenal dengan istilah internet secara terus-menerus menjadi pesan–pesan elektronik, termasuk e-mail, transmisi file, dan komunikasi dua arah antar individu atau komputer.

2.20.1 Web Service

Web service adalah aplikasi sekumpulan data (*database*), perangkat lunak (*software*) atau bagian dari perangkat lunak yang dapat diakses secara remote oleh berbagai piranti dengan sebuah perantara tertentu. Secara umum,*web*

service dapat diidentifikasikan dengan menggunakan URL seperti hanya *web* pada umumnya. Namun yang membedakan *web service* dengan *web* pada umumnya adalah interaksi yang diberikan oleh *web service*. Berbeda dengan URL *web* pada umumnya, URL *web service* hanya menggandung kumpulan informasi, perintah, konfigurasi atau sintaks yang berguna membangun sebuah fungsi-fungsi tertentu dari aplikasi [28].

Web service dapat diartikan juga sebuah metode pertukaran data, tanpa memperhatikan dimana sebuah*database* ditanamkan, dibuat dalam bahasa apa sebuah aplikasi yang mengkonsumsi data, dan di *platform* apa sebuah data itu dikonsumsi. *Web service* mampu menunjang interoperabilitas. Sehingga *web service* mampu menjadi sebuah jembatan penghubung antara berbagai sistem yang ada.

Menurut W3C *Web services Architecture Working Group* pengertian *Web service* adalah sebuah sistem *software* yang di desain untuk mendukung interoperabilitas interaksi mesin ke mesin melalui sebuah jaringan. Interface*web service* dideskripsikan dengan menggunakan format yang mampu diproses oleh mesin (khususnya WSDL). Sistem lain yang akan berinteraksi dengan *web service* hanya memerlukan SOAP, yang biasanya disampaikan dengan HTTP dan XML sehingga mempunyai korelasi dengan standar *Web (Web Services Architecture Working Group*, 2004).

Web pada umumnya digunakan untuk melakukan *respon* dan *request* yang dilakukan antara *client* dan *server*. Sebagai contoh, seorang pengguna layanan *web* tertentu mengetikan alamat *url web* untuk membentuk sebuah*request*. *Request* akan sampai pada *server*, diolah dan kemudian disajikan dalam bentuk sebuah respon. Dengan singkat kata terjadilah hubungan *client-server* secara sederhana.

Sedangkan pada *web service* hubungan antara *client* dan *server* tidak terjadi secara langsung. Hubungan antara *client* dan *server* dijembatani oleh file *web service* dalam format tertentu. Sehingga akses terhadap *database* akan ditanggani tidak secara langsung oleh *server*, melainkan melalui perantara yang

disebut sebagai *web service*. Peran dari *web service* ini akan mempermudah distribusi sekaligus integrasi database yang tersebar di beberapa *server* sekaligus.

2.21 Data dan Informasi

Hubungan antara data dan informasi sangatlah erat sebagaimana hubungan antara sebab dan akibat. Bahwa data merupakan bentuk dasar dari sebuah informasi, sedangkan informasi merupakan elemen yang dihasilkan dari suatu bentuk pengolahan data.

2.21.1 Data

Secara konseptual, data adalah deskripsi tentang benda, kejadian, aktivitas dan transaksi, yang tidak mempunyai makna atau tidak berpengaruh secara langsung kepada pemakai. Data sering kali disebut sebagai bahan mentah informasi.

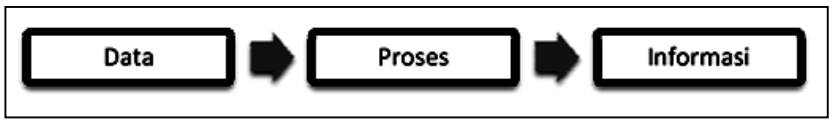
Berikut adalah kutipan pengertian data dari sudut pandang yang berbeda :

1. Menurut kamus bahasa Inggris-Indonesia, data diterjemahkan sebagai istilah yang berasal dari kata "*datum*" yang berarti fakta atau bahan-bahan keterangan.
2. Dari sudut pandang bisnis, terdapat pengertian data bisnis sebagai berikut: *"Bussines data is an organization's description of things* (*resources*) *and events (transactions) that it faces"*. Jadi data, dalam hal ini disebut sebagai data bisnis, merupakan deskripsi organisasi tentang sesuatu (*resource*) dan kejadian (*transaction)* yang terjadi.
3. Menurut Jogiyanto, data merupakan bentuk jamak dari datum (kenyataan) yang berupa fakta-fakta, angka-angka, gambar-gambar yang dapat ditarik kesimpulannya.
4. Gordon B. Davis dalam bukunya *Management Informations System : Conceptual Foundations, Structure, and Development* menyebut data sebagai bahan mentah dari informasi, yang dirumuskan sebagai sekelompok lambang-lambang tidak acak yang menunjukkan jumlah atau tindakan atau hal-hal lain.

Dari keempat pengertian diatas, dapat diambil kesimpulan bahwa data adalah bahan baku informasi, didefinisikan sebagai kelompok teratur simbol-simbol yang mewakili kuantitas, tindakan, benda,dan sebagainya. Data terbentuk dari karakter, dapat berupa alphabet, angka, maupun simbol khusus seperti *, $, dan /. Data disusun untuk diolah dalam bentuk struktur data, struktur file, dan basis data [29].

2.21.2 Informasi

Informasi adalah data yang telah diproses menjadi bentuk yang memiliki arti bagi penerima dan dapat berupa fakta, suatu nilai yang bermanfaat. Jadi ada suatu proses transformasi data menjadi suatu informasi (input – proses – output).



Gambar 2.7 Pemrosesan data menjadi informasi

Definisi umum untuk informasi dalam sistem informasi menurut Jogiyanto H.M (1990; 11) : "Informasi adalah bentuk data yang dapat diolah yang lebih berguna dan berarti bagi yang menerimanya". Menurut Robert G. Munik (1973 ; 12) : "Informasi adalah data yang telah diolah menjadi suatu bentuk yang berarti bagi penerimanya dan bermanfaat dalam pengambilan keputusan saat ini atau mendatang.Berikut pengertian informasi dari berbagai sumber :

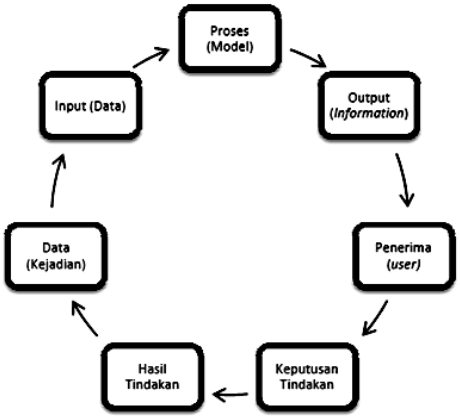
1. Menurut Gordon B. Davis dalam bukunya *"Management Informations System: Conceptual Foundations, Structures, and Development* menyebut informasi sebagai data yang telah diolah menjadi bentuk yang berguna bagi penerimanya dan nyata, berupa nilai yang dapat dipahami di dalam keputusan sekarang maupun masa depan.
2. Menurut Berry E. Cushing dalam bukunya*"Accounting Information System and Business Organization"* dikatakan bahwa informasi merupakan sesuatu yang menunjukkan hasil pengolahan data yang diorganisasi dan berguna kepada orang yang menerimanya.

3. Menurut Robert N. Anthony dan John Dearden dalam bukunya *"Managemet Control Systems"*, menyebut informasi sebagai suatu kenyataan, data, item, yangmenambah pengetahuan bagi penggunanya.
4. Menurut Stephan A. Moscove dan Mark G. Simkin dalambukunya *"Accounting Information Systems: Concepts and Practise"* mengatakan informasi sebagai kenyataan atau bentuk-bentuk yang berguna yang dapat digunakan untuk pengambilan keputusan bisnis.

Dari keempat pengertian seperti tersebut di atas dapat disimpulkan bahwa informasi hasil dari pengolahan data menjadibentuk yang lebih berguna bagi yang menerimanya yang menggambarkan suatu kejadian-kejadian nyata dan dapat digunakan sebagai alat bantu untuk pengambilan suatu keputusan.

2.21.3 Sirkulasi Informasi

Untuk memperoleh informasi yang bermanfaat bagi penerimanya, perlu untuk dijelaskan bagaimana siklus yang terjadi atau dibutuhkan dalam menghasilkan informasi. Pertama-tama data dimasukkan ke dalam model yang umumnya memiliki urutan proses tertentu dan pasti, setelah diproses akan dihasilkan informasi tertentu yang membuat suatu keputusan atau melakukakn tindakan tertentu. Dari keputusan atau tindakan tersebut akan menghasilkan atau diperoleh kejadian-kejadian tertentu yang akan digunakan kembali sebagai data yang nantinya akan dimasukkan ke dalam model (proses), begitu seterusnya. Dengan demikian akan membentuk suatu siklus informasi (*information cycle*) atau siklus pengolahan data (*data processing cycles*), seperti gambar berikut :



Gambar 2.8 Siklus Informasi

2.22 Basis Data

2.22.1 Konsep Dasar Basis Data

Basis data dapat dibayangkan sebagai sebuah lemari arsip yang ditempatkan secara berurutan untuk memudahkan dalam pengambilan kembali data tersebut. Data menunjukan sekumpulan data yang dipakai dalam suatu lingkungan perusahaan atau instansi – instansi. Penerapan basis data dalam sistem informasi disebut sistem basis data.

2.22.2 Pengertian Basis Data

Basis data terdiri dari kata *basis* dan *data.* Basis data dapat diartikan gudang atau tempat bersarang, sedangkan data berarti representasi fakta dunia nyata yang mewakili suatu objek seperti manusia, hewan, peristiwa, konsep dan sebagainya yang direkam dalam bentuk angka, huruf, simbol, teks, gambar, bunyi, atau kombinasi.

Dapat disimpulkan bahwa basis data merupakan kumpulan data (arsip) yang saling berhubungan yang disimpan secara bersama sedemikian rupa dan tanpa pengulangan (*redundancy*) yang tidak perlu, untuk memenuhi berbagai kebutuhan. Basis data dapat diartikan sebagai kumpulan file/tabel/arsip yang saling berhubungan yang disimpan dalam media penyimpanan elektronis [30].

2.22.3 Model Basis Data

Model basis data adalah sekumpulan konsep terintegrasi yang dipakai untuk menjabarkan data, hubungan antardata, dan kekangan terhadap data yang digunakan untuk menjagakonsistensi. Kadang model data disebut dengan stuktur data logis.Model data yang umum pada saat ini ada empat macam,diantaranya sebagai berikut :

1. Model data hierarki.
2. Model data jaringan.
3. Model data relasional.
4. Model data berbasis objek.

Tiga model di atas yang selain model data berbasis objek disebut model data yang berbasis rekaman *(record-based data model).*

2.23 Media Penyimpanan *(Storage)*

Media penyimpan data *(data storage)* adalah sebuah kemampuan dari sebuah benda yang memungkinkan data tersimpan dalam jumlah besar. Seiring dengan perkembangan dan penelitian para ilmuan-ilmuan dunia mencoba untuk membentuk sebuah media penyimpanan data yang efisien. Dalam artian memiliki ruang penyimpanan yang besar tapi ukuran dengan ukuran yang kecil [31].

2.24 Konsep Perancangan Berorientasi Objek

OOP (*Object Oriented Programming*) adalah suatu metode pemrograman yang berorientasi kepada objek. Tujuan dari OOP diciptakan adalah untuk mempermudah pengembangan program dengan cara mengikuti model yang telah ada di kehidupan sehari-hari. Jadi setiap bagian dari suatu permasalahan adalah objek, nah objek itu sendiri merupakan gabungan dari beberapa objek yang lebih kecil lagi. Contoh Pesawat, Pesawat adalah sebuah objek. Pesawat itu sendiri terbentuk dari beberapa objek yang lebih kecil lagi seperti mesin, roda, baling-baling, kursi, dan lain - lain. Pesawat sebagai objek yang terbentuk dari objek-objek yang lebih kecil saling berhubungan, berinteraksi, berkomunikasi dan saling mengirim pesan kepada objek-objek yang lainnya. Begitu juga dengan program,

sebuah objek yang besar dibentuk dari beberapa objek yang lebih kecil, objek-objek itu saling berkomunikasi, dan saling berkirim pesan kepada objek yang lain [32].

Konsep OOP, terbagi menjadi 4 kelas, yaitu sebagai berikut:

1. Kelas Abstrak (*Class Abstract*)
   a. Kelas merupakan deskripsi abstrak informasi dan tingkah laku dari sekumpulan data.
   b. Kelas dapat diilustrasikan sebagai suatu cetak biru (*blueprint*) atau prototipe yang digunakan untuk menciptakan objek.
   c. Kelas merupakan tipe data bagi objek yang mengenkapsulasi data dan operasi pada data dalam suatu unit tunggal.
   d. Kelas mendefinisikan suatu struktur yang terdiri atas data kelas (*data field*), prosedur atau fungsi (*method*), dan sifat kelas (*property*).

2. Enkapsulasi (*encapsulation*)
   a. Istilah enkapsulasi sebenarnya adalah kombinasi data dan fungsionalitas dalam sebuah unit tunggal sebagai bentuk untuk menyembunyikan detail informasi.
   b. Proses enkapsulasi memudahkan kita untuk menggunakan sebuah objek dari suatu kelas karena kita tidak perlu mengetahui segala hal secara rinci.
   c. Enkapsulasi menekankan pada antarmuka suatu kelas, atau dengan kata lain bagaimana menggunakan objek kelas tertentu.
   d. Contoh: kelas mobil menyediakan antarmuka fungsi untuk menjalankan mobil tersebut, tanpa kita perlu tahu komposisi bahan bakar, udara dan kalor yang diperlukan untuk proses tersebut.

3. Pewarisan (*Inheritance*)
   a. Kita dapat mendefinisikan suatu kelas baru dengan mewarisi sifat dari kelas lain yang sudah ada.

b. Penurunan sifat ini bisa dilakukan secara bertingkattingkat, sehingga semakin ke bawah kelas tersebut menjadi semakin spesifik.
c. Sub kelas memungkinkan kita untuk melakukan spesifikasi detail dan perilaku khusus dari kelas supernya.
d. Dengan konsep pewarisan, seorang programmer dapat menggunakan kode yang telah ditulisnya pada kelas super berulang kali pada kelas-kelas turunannya tanpa harus menulis ulang semua kode - kode itu.

4. Polimorfisme (*polymorphism*)
   a. Polimorfisme merupakan kemampuan objek - objek yang berbeda kelas namun terkait dalam pewarisan untuk merespon secara berbeda terhadap suatu pesan yang sama.
   b. Polimorfisme juga dapat dikatakan kemampuan sebuah objek untuk memutuskan method mana yang akan diterapkan padanya, tergantung letak objek tersebut pada jenjang pewarisan.
   c. *Method overriding*.
   d. *Method name overloading*.

Karakteristik OOP (*Object Oriented Programming*) adalah sebagai berikut:

a. Semua adalah objek.
b. Komputasi dilakukan dengan komunikasi antar objek. Setiap objek berkomunikasi dengan objek yang lain melalui pengiriman dan penerimaan pesan.
c. Sebuah pesan merupakan permintaan atas sekumpulan aksi dengan semua argumen yang diperlukan untuk menyelesaikan suatu tugas tertentu.
d. Setiap objek memiliki memori sendiri, yang dapat terdiri dari objek-objek lainnya.
e. Setiap objek adalah wakil atau representasi dari suatu kelas. Sebuah kelas dapat mewakili sekelompok objek yang sama.

f. Kelas merupakan kumpulan tingkah laku yang berkaitan dengan suatu objek. Jadi, semua objek yang merupakan wakil dari kelas yang sama dapat melakukan aksi yang sama pula.
g. Kelas-kelas diorganisasikan ke dalam struktur pohon yang berakar tunggal, yang dinamakan dengan jenjang pewarisan (*inheritance hierarchy*).
h. Setiap objek pada umumnya memiliki tiga sifat, yaitu keadaan, operasi dan identitas objek.
i. Operasi merupakan tindakan yang dapat dilakukan oleh sebuah objek.
j. Keadaan objek merupakan koleksi dari seluruh informasi yang dimiliki oleh objek pada suatu saat.
k. Informasi yang terkandung pada objek tersebut pada akhirnya memberikan identitas khusus yang membedakan suatu objek dengan objek lainnya.

2.25 Unified Modeling Language (UML)

UML (*Unified Modeling Language*) adalah sebuah bahasa yang berdasarkan grafik/gambar untuk memvisualisasi, menspesifikasikan, membangun, dan pendokumentasian dari sebuah sistem pengembangan software berbasis OO (*Object-Oriented*). UML tidak hanya merupakan sebuah bahasa pemograman visual saja, namun juga dapat secara langsung dihubungkan ke berbagai bahasa pemograman, seperti JAVA, C++, Visual Basic, atau bahkan dihubungkan secara langsung ke dalam sebuah object-oriented database. Bagian-bagian utama dari UML adalah view, diagram, model element, dan general mechanism [33].

1. *View*

   View digunakan untuk melihat sistem yang dimodelkan dari beberapa aspek yang berbeda. *View* bukan melihat grafik, tapi merupakan suatu abstraksi yang berisi sejumlah diagram. Beberapa jenis view dalam UML antara lain : *use case view, logical view, component view, concurrency view,* dan *deployment view*.

2. Use case View

   Mendeskripsikan fungsionalitas sistem yang seharusnya dilakukan sesuai yang diinginkan *external actors*. *Actor* yang berinteraksi dengan sistem dapat berupa *user* atau sistem lainnya. *View* ini digambarkan dalam *use case diagrams* dan kadang-kadang dengan *activity diagrams*. View ini digunakan terutama untuk pelanggan, perancang (*designer*), pengembang (*developer*), dan penguji sistem (*tester*).

3. *Logical View*

   Mendeskripsikan bagaimana fungsionalitas dari sistem, struktur statis (*class, object,* dan *relationship*) dan kolaborasi dinamis yang terjadi ketika *object* mengirim pesan ke *object* lain dalam suatu fungsi tertentu. *View* ini digambarkan dalam *class diagrams* untuk struktur statis dan dalam *state, sequence, collaboration,* dan *activity diagram* untuk model dinamisnya. *View* ini digunakan untuk perancang (*designer*) dan pengembang (*developer*).

4. *Component View*

   Mendeskripsikan implementasi dan ketergantungan modul. Komponen yang merupakan tipe lainnya dari *code module* diperlihatkan dengan struktur dan ketergantungannya juga alokasi sumber daya komponen dan informasi *administrative* lainnya. View ini digambarkan dalam *component view* dan digunakan untuk pengembang (*developer*).

5. *Concurrency View*

   Membagi sistem ke dalam proses dan prosesor. View ini digambarkan dalam diagram dinamis (*state, sequence, collaboration,* dan *activity diagrams*) dan diagram implementasi (*component* dan *deployment diagrams*) serta digunakan untuk pengembang (*developer*), pengintegrasi (*integrator*), dan penguji (*tester*).

6. *Deployment View*

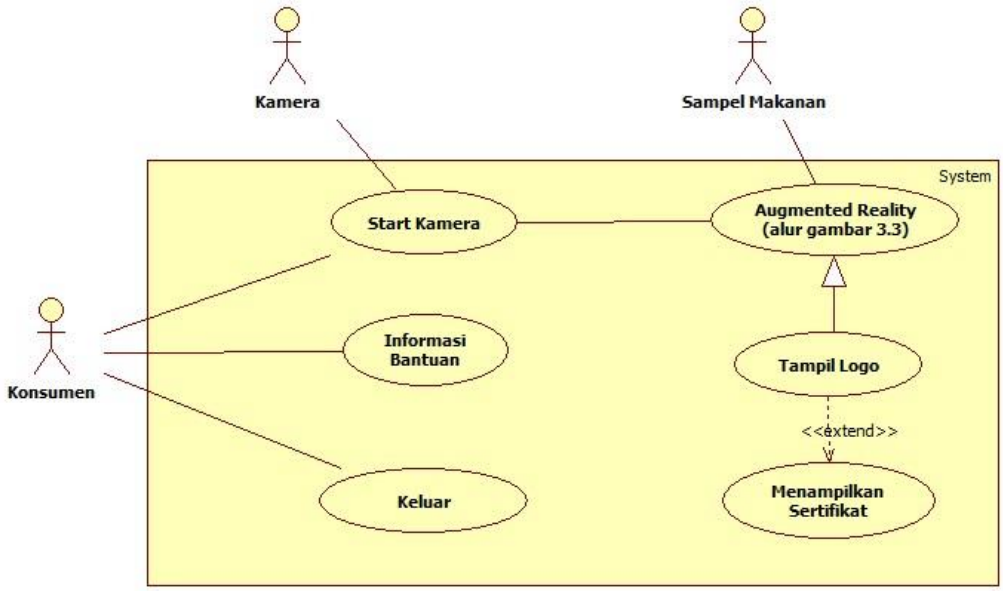
   Mendeskripsikan fisik dari sistem seperti komputer dan perangkat (*nodes*) dan bagaimana hubungannya dengan yang lain. *View* ini digambarkan

dalam *deployment diagrams* dan digunakan untuk pengembang (*developer*), pengintegrasi (*integrator*), dan penguji (*tester*).

Berikut ini menjelaskan tentang diagram-diagram yang ada di UML:

1. *Use Case Diagram*

   Use case adalah abstraksi dari interaksi antara *system* dan *actor*. Use case bekerja dengan cara mendeskripsikan tipe interaksi antara *user* sebuah *system* dengan sistemnya sendiri melalui sebuah cerita bagaimana sebuah *system* dipakai. *Use case* merupakan konstruksi untuk mendeskripsikan bagaimana *system* akan terlihat di mata *user*. Sedangkan *use case diagram* memfasilitasi komunikasi diantara analis dan pengguna serta antara analis dan *client*.



Gambar 2.9 *Use Case Diagram*

Diagram *Use Case* berguna dalam tiga hal :

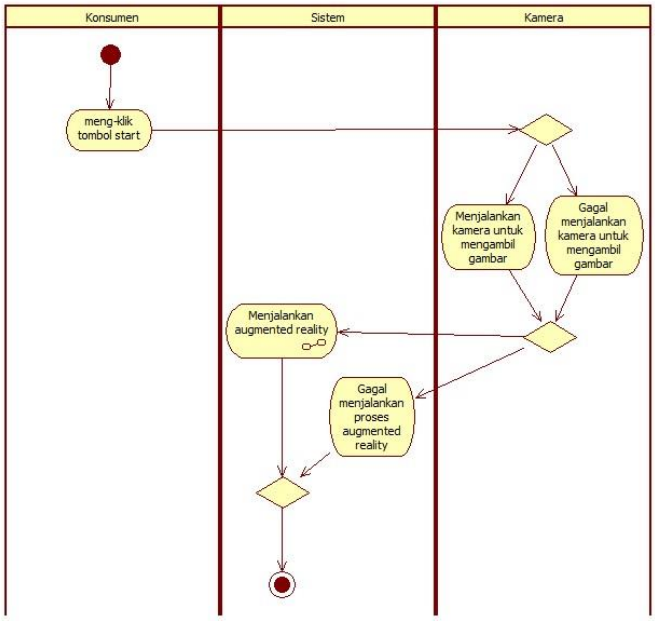
a. Menjelaskan fasilitas yang ada (*requirements*) : *Use Case* baru selalu menghasilkan fasilitas baru ketika sistem di analisa, dan *design* menjadi lebih jelas.

b. Komunikas dengan klien : Penggunaan notasi dan simbol dalam diagram *Use Case* membuat pengembang lebih mudah berkomunikasi dengan klien-kliennya.

c. Membuat *test* dari kasus-kasus secara umum : Kumpulan dari kejadian-kejadian untuk *Use Case* bisa dilakukan test kasus layak untuk kejadian-kejadian tersebut.

2. *Activity* Diagram

Pada dasarnya diagram *Activity* sering digunakan oleh *flowchart*. Diagram ini berhubungan dengan diagram *Statechart*. Diagram *Statechart* berfokus pada obyek yang dalam suatu proses (atau proses menjadi suatu obyek), diagram *Activity* berfokus pada aktifitas-aktifitas yang terjadi yang terkait dalam suatu proses tunggal. Jadi dengan kata lain, diagram ini menunjukkan bagaimana aktifitas-aktifitas tersebut bergantung satu sama lain. Lihat gambar di bawah ini, agar lebih jelas :



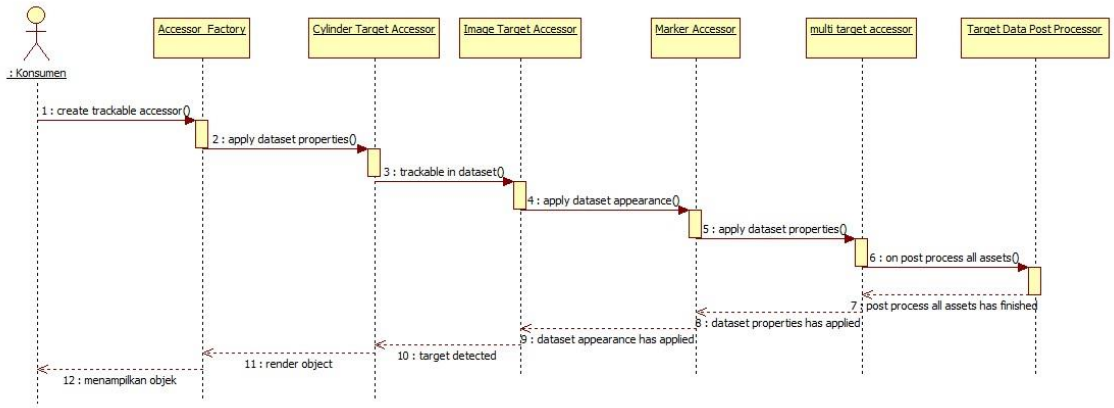
Gambar 2.10 *Activity Diagram*

Diagram Activity dapat dibagi menjadi beberapa jalur kelompok yang menunjukkan obyek mana yang bertanggung jawab untuk suatu

aktifitas. Peralihan tunggal (*single transition*) timbul dari setiap adanya *activity* (aktifitas), yang saling menghubungi pada aktifitas berikutnya. Sebuah *transition* (transisi) dapat membuat cabang ke dua atau lebih percabangan *exclusive transition* (transisi eksklusif). Label *Guard Expression* (ada didalam [ ]) yang menerangkan *output* (keluaran) dari percabangan. Percabangan akan menghasilkan bentuk menyerupai bentuk intan. T*ransition* bisa bercabang menjadi beberapa aktifitas paralel yang disebut *Fork***.** *Fork* beserta *join* (gabungan dari hasil *output fork*) dalam diagram berbentuk *solid bar* (batang penuh).

3. *Sequence* Diagram

Diagram *Class* dan diagram *Object* merupakan suatu gambaran *model statis*. Namun ada juga yang bersifat *dinamis*, seperti Diagram *Interaction*. Diagram *sequence* merupakan salah satu diagram *Interaction* yang menjelaskan bagaimana suatu operasi itu dilakukan; *message* (pesan) apa yang dikirim dan kapan pelaksanaannya. Diagram ini diatur berdasarkan waktu. Obyek-obyek yang berkaitan dengan proses berjalannya operasi diurutkan dari kiri ke kanan berdasarkan waktu terjadinya dalam pesan yang terurut.



Gambar 2.11 *Sequence Diagram*

Pemanggilan diri sendiri disebut dengan iterasi. *Expression* yang dikurung dengan "[ ]", adalah *condition* (keadaan kondisi). Pada diagram

dapat dibuat *note* (catatan). Pada gambar, terlihat seperti selembar kertas yang berisikan teks. *Note* bisa diletakan dimana saja pada diagram UML.

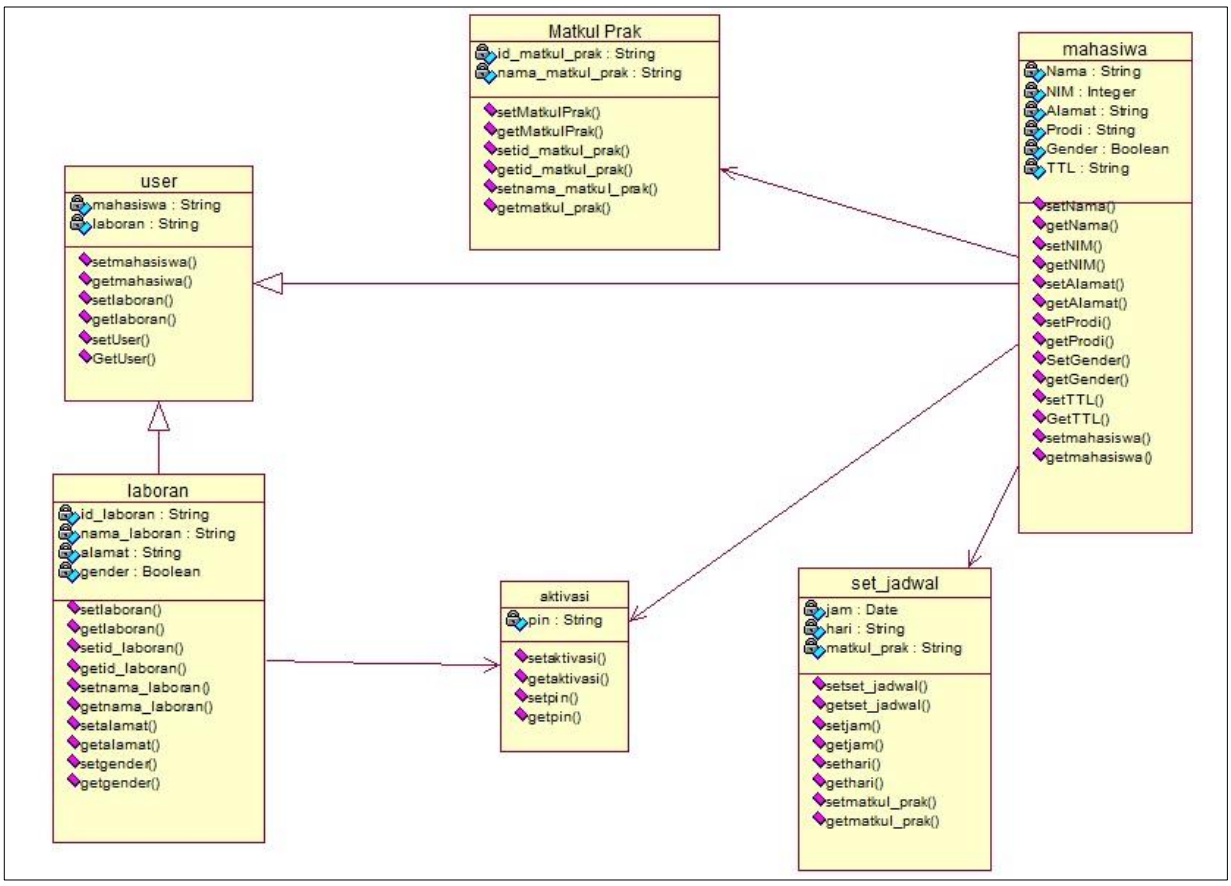
4. *Class* Diagram

Class adalah sebuah spesifikasi yang jika diinstansiasi akan menghasilkan sebuah objek dan merupakan inti dari pengembangan dan desain berorientasi objek. Class menggambarkan keadaan (atribut/properti) suatu sistem, sekaligus menawarkan layanan untuk memanipulasi keadaan tersebut (metoda/fungsi). Class diagram menggambarkan struktur dan deskripsi *class*, *package* dan objek beserta hubungan satu sama lain seperti *containment* , pewarisan, asosiasi, dan lain-lain.Class memiliki tiga area pokok yaitu nama (dan *stereotype*), Atribut, dan Metoda.

Atribut dan metoda dapat memiliki salah satu sifat berikut :

a. *Private*, tidak dapat dipanggil dari luar *class* yang bersangkutan
b. *Protected*, hanya dapat dipanggil oleh *class* yang bersangkutan dan anak-anak yang mewarisinya
c. *Public*, dapat dipanggil oleh siapa saja

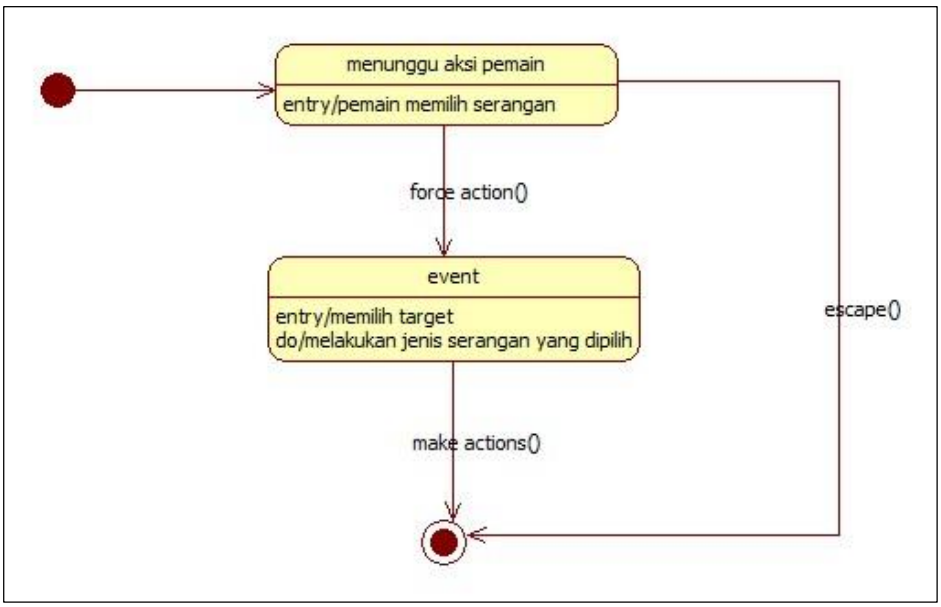
Class dapat merupakan implementasi dari sebuah *interface*, yaitu *class* abstrak yang hanya memiliki metoda. *Interface* tidak dapat langsung diinstansiasikan, tetapi harus diimplementasikan dahulu menjadi sebuah *class*. Dengan demikian *interface* mendukung resolusi metoda pada saat *run-time*. Sesuai dengan perkembangan *class* model, class dapat dikelompokkan menjadi *package*. Kita juga dapat membuat diagram yang terdiri atas *package*.



Gambar 2.12 Class Diagram

5. *State Machine* Diagram (*Statechart* diagram in versi 1.x)

Statechart diagram menggambarkan transisi dan perubahan keadaan (dari satu *state* ke*state* lainnya) suatu objek pada sistem sebagai akibat dari *stimuli* yang diterima. Pada umumnya *statechart diagram* menggambarkan *class* tertentu (satu *class* dapat memiliki lebih dari satu *statechart diagram*). Dalam UML, *state* digambarkan berbentuk segiempat dengan sudut membulat dan memiliki nama sesuai kondisinya saat itu. Transisi antar *state*umumnya memiliki kondisi *guard* yang merupakan syarat terjadinya transisi yang bersangkutan, dituliskan dalam kurung siku. *Action* yang dilakukan sebagai akibat dari *event* tertentu dituliskan dengan diawali garis miring. Titik awal dan akhir digambarkan berbentuk lingkaran berwarna penuh dan berwarna setengah.



Gambar 2.13 State Machine Diagram

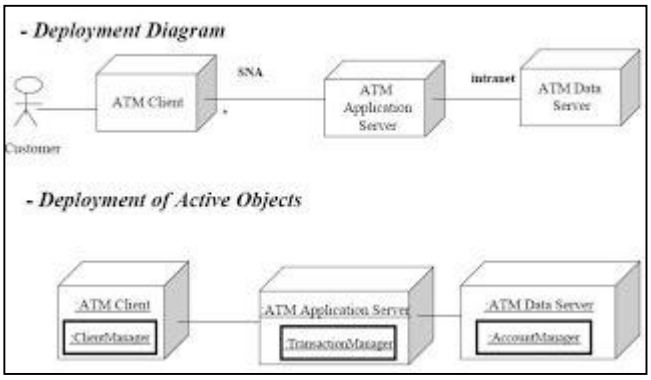
6. *Component* Diagram

Component diagram menggambarkan struktur dan hubungan antar komponen piranti lunak, termasuk ketergantungan (*dependency*) di antaranya. Komponen piranti lunak adalah modul berisi *code* , baik berisi *source code* maupun *binary code* , baik *library* maupun*executable* , baik yang muncul pada *compile time*, *link time* , maupun *run time* . Umumnya komponen terbentuk dari beberapa *class* dan/atau *package* , tapi dapat juga dari komponen-komponen yang lebih kecil. Komponen dapat juga berupa *interface* , yaitu kumpulan layanan yang disediakan sebuah komponen untuk komponen lain.

7. Deployment Diagram

Deployment/physical diagram menggambarkan detail bagaimana komponen di-*deploy* dalam infrastruktur sistem, di mana komponen akan terletak (pada mesin, *server* atau piranti keras apa), bagaimana kemampuan jaringan pada lokasi tersebut, spesifikasi *server*, dan hal-hal lain yang bersifat fisikal Sebuah *node* adalah server,*workstation* , atau piranti keras lain yang digunakan untuk men- *deploy* komponen dalam

lingkungan sebenarnya. Hubungan antar *node* (misalnya TCP/IP) dan *requirement* dapat juga didefinisikan dalam diagram ini.



Gambar 2.14 *Deployment* Diagram

Tujuan Penggunaan UML adalah sebagai berikut:

1. Memberikan bahasa pemodelan yang bebas dari berbagai bahas pemrograman dan proses rekayasa.
2. Menyatukan praktek-praktek terbaik yang terdapat dalam pemodelan.
3. Memberikan model yang siap pakai, bahsa pemodelan visual yang ekspresif untuk mengembangkan dan saling menukar model dengan mudah dan dimengerti secara umum.
4. UML bisa juga berfungsi sebagai sebuah (*blueprint*) cetak biru karena sangat lengkap dan detail. Dengan cetak biru ini maka akan bisa diketahui informasi secara detail tentang *coding* program atau bahkan membaca program dan menginterpretasikan kembali ke dalam bentuk diagram (*reserve enginering*).

2.26 Metode Pengujian *Black Box*

Metode pengujian adalah cara atau teknik untuk menguji perangkat lunak, mempunyai mekanisme untuk menentukan data uji yang dapat menguji perangkat lunak secara lengkap dan mempunyai kemungkinan tinggi untuk menemukan kesalahan. Salah satunya yaitu metode pengujian *black box*.

Pengujian black box merupakan pendekatan komplementer dari teknik *white box*, karena pengujian *black box* diharapkan mampu mengungkap kelas

kesalahan yang lebih luas dibandingkan teknik *white box*. Pengujian *black* box berfokus pada pengujian persyaratan fungsional perangkat lunak, untuk mendapatkan serangkaian kondisi input yang sesuai dengan persyaratan fungsional suatu program. Pengujian *black box* adalah pengujian aspek fundamental sistem tanpa memperhatikan struktur logika internal perangkat lunak. Metode ini digunakan untuk mengetahui apakah perangkat lunak berfungsi dengan benar [34].

Pengujian *black box* merupakan metode perancangan data uji yang didasarkan pada spesifikasi perangkat lunak. Data uji dibangkitkan, dieksekusi pada perangkat lunak dan kemudian keluaran dari perangkat lunak dicek apakah telah sesuai dengan yang diharapkan. Pengujian *black box* berusaha menemukan kesalahan dalam kategori :

1. fungsi-fungsi yang tidak benar atau hilang
2. kesalahan *interface*
3. kesalahan dalam struktur data atau akses *database* eksternal
4. kesalahan kinerja
5. inisialisasi dan kesalahan terminasi.

Pengujian *black box* cenderung diaplikasikan selama tahap akhir pengujian. Pengujian *black box* harus dapat menjawab :

1. Bagaimana validitas fungsional diuji
2. Kelas input apa yang akan membuat kasus pengujian menjadi lebih baik
3. Apakah *system* akan sangat sensitive terhadap harga input tertentu
4. Bagaimana batasan dari suatu data diisolasi
5. Kecepatan data apa dan volume data apa yang akan ditoleransi oleh *system*
6. Apa pengaruh kombinasi tertentu dari data terhadap *system* operasi.

2.26.1 *Graf* Based Testing

Langkah pertama pada pengujian black box adalah memahami objek yang terdapat dalam model perangkat lunak dan menentukan hubungan yang dimiliki antara objek-objek tersebut. Pengujian berbasiskan model graf dilakukan terhadap

perilaku *system*. *Graf Based Testing* menggambarkan graf yang mewakili hubungan antar objek pada modul sehingga tiap objek dan hubungannya dapat diuji. Pengujian ini dimulai dari mendefinisikan semua simpul dan bobot simpul, dimana objek dan atribut diidentifikasikan, serta memberikan indikasi titik mulai dan berhenti.Terdapat tiga pola *link weight*, yaitu :

1. Transitivitas, yaitu hubungan antara tiga objek atau lebih yang menentukan bagaimana pengaruh hubungan tersebut menyebar pada objek yang ditentukan.
2. Simetris, yaitu hubungan antara dua objek secara dua arah.
3. Refleksif, yaitu hubungan yang mengarah pada node itu sendiri atau *loop null.*

2.26.2 *Equivalence Partitioning* (Partisi ekuivalensi)

Partisi ekuivalensi adalah metode yang membagi domain input dari suatu program ke dalam kelas data, menentukan kasus pengujian dengan mengungkapkan kelas-kelas kesalahan, sehingga akan mengurangi jumlah keseluruhan kasus pengujian. Bila suatu link weight mempunyai pola transitivitas, simetris, dan refleksif maka akan terdapat kelas ekuivalensi. Kelas ekuivalensi merepresentasikan serangkaian kondisi valid dan invalid untuk kondisi inputan. Secara khusus, suatu kondisi input dapat berupa harga *numeric*, suatu rentang harga, serangkaian harga yang terkait, atau suatu kondisi *Boolean*.

Penentuan Kelas Ekuivalensi :

- Bila kondisi input menentukan suatu range, maka satu kelas ekuvalensi valid dan dua yang invalid ditentukan
- Bila suatu kondisi input memerlukan suatu harga khusus, maka satu kelas ekuivalensi valid dan dua yang invalid ditentukan
- Bila suatu kondisi menentukan anggota suatu himpunan, maka satu kelas ekuivalensi valid atau dua yang invalid ditentukan
- Bila suatu kondisi input adalah *Boolean*, maka satu kelas valid dan satu yang lain ditentukan.

2.26.3 *Boundary Value Analysis* (Analisis Nilai Batas)

Analisis nilai batas adalah teknik desain proses yang melengkapi partisi ekuivalensi, dengan berfokus pada domain *output*. Pedoman untuk menentukan analisis nilai batas :

1. Bila suatu kondisi input mengkhususkan suatu range dibatasi oleh nilai a dan b, maka pengujian harus didesain dengan nilai a dan b, persis di atas dan di bawah a dan b secara bersesuaian
2. Bila suatu kondisi input mengkhususkan sejumlah nilai, maka pengujian harus dikembangkan dengan menggunakan jumlah minimum dan maksimum. Nilai tepat di atas dan di bawah minimum dan maksimum juga diuji.
3. Pedoman 1 dan 2 juga diaplikasikan ke kondisi output.
4. Bila struktur data program telah memesan suatu batasan, maka pengujian akan dilakukan sesuai dengan batasan struktur data tersebut.

2.26.4 Comparison Testing

Pengujian perbandingan adalah metode pembangkitan data uji yang dilakukan pada perangkat lunak yang dibuat redundan. Perangkat lunak yang redundan mempunyai dua tim pengembang yang masing-masing mengembangkan perangkat lunak sendiri-sendiri untuk spesifikasi yang sama. Metode pengujian perbandingan digunakan untuk membangkitkan data uji untuk salah satu perangkat lunak, yang kemudian digunakan sebagai masukan pada pengujian perangkat lunak yang lain.

Comparison Testing digunakan untuk *system* yang menganut *redundancy*, kasus uji yang dirancang untuk satu versi perangkat lunak dijadikan masukan pada pengujian versi perangkat lunak lainnya, dan diharapkan hasil kedua versi perangkat lunak harus sama. Jika hasil pengujian kedua perangkat lunak tersebut berbeda maka kedua perangkat lunak itu akan dicek untuk mencari yang salah.

2.27 Definisi Kualitas Perangkat Lunak

Menurut definisi dalam Steve McConnell's *Code Complete* membagi perangkat lunak ke dalam dua hal yaitu internal dan *external quality characteristics*. Karakteristik kualitas eksternal merupakan bagian-bagian dari suatu produk yang berhubungan dengan para pemakainya, sedangkan karakteristik kualitas internal tidak secara langsung berhubungan dengan pemakai. *Software Quality* didefinisikan sebagai kesesuaian yang diharapkan pada semua software yang dibangun dalam hal fungsi software yang diutamakan dan unjuk kerja software, standar pembangunan software yang terdokumentasi dan karakteristik yang ditunjukkan oleh software [35].

Definisi ini menekankan pada 3 hal yaitu:

1. Kebutuhan software adalah fondasi ukuran kualitas *software*, jika *software* Tidak sesuai dengan kebutuhan yang ditentukan maka kualitas pun kurang.
2. Jika menggunakan suatu standar untuk pembangunan *software* maka jika *software* tidak memenuhi standar tersebut maka dianggap kurang berkualitas.
3. Seringkali ada kualitas yang secara langsung diutarakan (tersirat) seperti kemudahan penggunaan dan pemeliharaan yang baik. Kualitas *software* dipertanyakan jika tidak memenuhi kebutuhan ini.

Hal lain yang perlu diperhatikan dalam kualitas perangkat lunak adalah *quality assurarance* (QA) yang merupakan *activity of providing revidence*. Selain itu harus adanya *software quality management* (SQM). Tujuan dari SQM adalah untuk mengembangkan suatu pemahaman kuantitatif dari kualitas proyek produk perangkat lunak dan mencapai tujuan spesifik kualitasnya. Menurut IEEE, kualitas perangkat lunak didefinisikan sebagai derajat dari sebuah sistem, komponen atau proses yang memenuhi kebutuhan-kebutuhan yang dispesifikasikan dan kebutuhan pengguna atau harapan pengguna.

Menurut Pressman, kualitas perangkat lunak adalah kesesuaian antara kebutuhan fungsional dan kebutuhan performansi yang terdefinisi, standar pengembangan yang terdokumentasi secara ekplisit dan karakteristik-karakteristik

yang tersirat yang diharapkan dari perangkat lunak yang dikembangkan secara profesional.

2.28 Skala *Likert*

Skala *likert* digunakan untuk mengukur sikap, pendapat dan persepsi seseorang atau sekelompok orang tentang kejadian atau gejala sosial. Dengan menggunakan skala likert, maka variabel dijabarkan menurut urutan variabel, sub variabel, indikator, deskriptor. Dan deskriptor ini dapat dijadikan titik tolak untuk membuat butir instrumen berupa pernyataan atau pertanyaan yang perlu dijawab oleh responden.

Item item likert menyediakan repon dengan kategori yang berjenjang. Biasanya banyaknya jenjang adalah lima, yaitu sangat setuju, setuju, ragu ragu, tidak setuju dan sangat tidak setuju. Setiap kategori respon selanjutnya diberi skor. Untuk item positif skor terbesarnya adalah lima, skor terendah adalah satu dan sebaliknya jika item negatif. Untuk menghitung total skor tiap responden adalah dengan cara menjumlahkan skor skor item yang diperoleh reponden. Oleh karena itu, prosedur penskalaan likert sering disebut sebagai *Likert's Summeted Rating.*

Langkah langkah menyusunnya adalah sebagai berikut [36]:

1. Tentukan secara tegas sikap terhadap topik apa yang akan diukur.
2. Tentukan secara tegas sub varibel/dimensi yang menyusun sikap tersebut, kognitif, afektif dan konatif.
3. Susun pernyataan/pertanyaan atau item yang merupakan alat pengukur dimensi yang menyusun sikap yang akan diukur sesuai dengan indikator.
4. Setiap item diberi respon yang sifatnya tertutup.
5. Untuk setiap respon, jawaban diberi skor berdasarkan kriteria sebagai berikut: apabila item positif maka angka terbesar diletakkan pada respon sangat setuju, sedangkan bila item negatif maka angka terbesar diletakkan pada respon sangat tidak setuju.
6. Untuk mengetahui posisi setiap responden tentang suatu variabel, tentukan skor maksimal dan skor minimal yang mungkin dicapai oleh responden.

Langkah langkah untuk mengetahui bagaimana sikap tiap responden terhadap kualitas produk adalah :

1. Menentukan skor maksimal, yaitu skor jawaban terbesar dikali banyak item.
2. Menentukan skor minimal, yaitu skor jawaban terkecil dikali banyak item.
3. Menentukan nilai median, yaitu hasil penjumlahan skor maksimal dengan skor minimal dibagi dua.
4. Menentukan nilai kuartil 1, yaitu hasil penjumlahan skor minimal dengan median dibagi dua.
5. Menentukan nilai kuartil 3, yaitu hasil penjumlahan skor maksimal dengan median dibagi dua.
6. Membuat skala yang menggambarkan skor minimal, nilai kuartil 1, median, nilai kuartil 3 dan skor maksimal.
7. Mencari batas batas skor untuk masing masing kategori sikap.
8. Membuat tabel distribusi frekuensi sikap tiap responden terhadap kualitas produk.

Langkah langkah untuk mengetahui bagaimana sikap responden secara keseluruhan terhadap kualitas produk adalah :

1. menentukan skor maksimal, yaitu skor maksimal yang diperoleh tiap responden dikali banyaknya responden.
2. Menentukan skor minimal, yaitu skor minimal yang diperoleh tiap responden dikali banyaknya responden.
3. Menentukan nilai median, yaitu hasil penjumlahan total skor maksimal dengan total skor minimal dibagi dua.
4. Menentukan nilai kuartil 1, yaitu hasil penjumlahan total skor minimal dengan median dibagi dua.
5. Menentukan nilai kiartil 3, yaitu hasil penjumlahan total skor maksimal dengan median dibagi dua.
6. Membuat skala yang menggambarkan total skor minimal, nilai kuartil 1, median, kuartil 3 dan total skor maksimal.
7. Mencari batas batas skor untuk masing masing kategori sikap.
8. Menentukan skor total yang diperoleh seluruh responden.

9. Mengintrepretasikan skor total responden dengan skala pada point.

2.29 Model Metriks Kualitas Perangkat Lunak

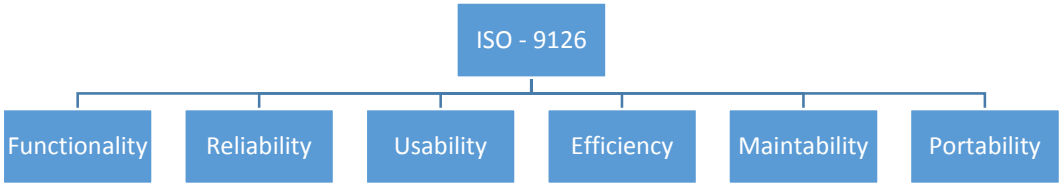
Dalam keilmuan rekayasa perangkat lunak terdapat beberapa model kualitas perangkat lunak. Pada setiap model ini terdapat beberapa faktor yang menjadi poin-poin utama dalam penilaian kualitas sebuah perangkat lunak. Berikut adalah penjelasan tentang model kualitas perangkat lunak yang akan diterapkan. Model ISO-9126 dikenalkan pertama kali pada tahun 1991 sebagai standarisasi kualitas produk perangkat lunak. Standarisasi ini dibuat karena banyaknya model kualitas yang ditawarkan sebagai faktor kualitas perangkat lunak [37].

Dalam dokumen pertama model ISO-9126 terdiri dari beberapa bagian model kualitas untuk kualitas sebuah produk perangkat lunak, di antaranya:

1. *Internal quality*
2. *External quality model*
3. *Quality in use model*

2.29.1 Model Kualitas Eksternal

Bagian pertama dari kualitas model tersebut menentukan 6 karakteristik yang mereka bagi kedalam 21 sub karakteristik untuk kualitas internal dan kualitas eksternal. Berikut ini merupakan pengertian dari masing-masing faktor kualitas dan sub pada ISO-9126.



Gambar 2.15 Faktor ISO 9126

1. *Functionality*

Faktor ini berhubungan dengan seberapa jauh sebuah perangkat lunak dapat memenuhi kebutuhan penggunanya.

a. *Suitability*

Sub-faktor *suitability* berhubungan dengan tingkat kemampuan dan kelayakan dari sebuah perangkat lunak untuk dapat menyediakan fungsionalitas untuk kebutuhan yang spesifik.

Tabel 2.1 Suitability Metriks

Nama Metriks	**Pengukuran**	**Intrepretasi nilai**
Functional adequacy	X = 1-A/B A = jumlah fungsi yang terdapat masalah dalam evaluasi B = Jumlah fungsi yang di evaluasi	0 <= X <= 1 Semakin dekat ke 1.0 semakin baik
Functional implementation completeness	X = 1-A/B A = jumlah fungsi yang hilang terdeteksi dalam evaluasi B = Jumlah fungsi yang dijelaskan dalam spesifikasi kebutuhan	0 <= X <= 1 Semakin dekat ke 1.0 semakin baik
Functional implementation coverage	X = 1-A/B A = jumlah yang dilakasanakan dengan benar atau fungsi yang hilang dalam evaluasi B = jumlah fungsi yang dijelaskan dalam	0 <= X <= 1 Semakin dekat ke 1.0 semakin baik

	spesifikasi kebutuhan	
Functional specification stability (volatility)	X = 1-A/B A = Jumlah fungsi perubahan setelah memasuki operasi dimulai B = Jumlah fungsi yang dijelaskan dalam spesifikasi kebutuhan	0 <= X <= 1 Semakin dekat ke 1.0 semakin baik

b. Accuracy

Accuracy berhubungan dengan kemampuan sebuah perangkat lunak untuk dapat memberikan hasil yang benar dan sesuai dengan fungsi yang dijalankan.

Tabel 2.2 *Accuracy* Metriks

Nama Metriks	**Pengukuran**	**Intrepretasi nilai**
Accuracy to expectation	X = A / T A = jumlah kasus yang dihadapi oleh pengguna dengan perbedaan terhadap hasil yang diharapkan T = Waktu operasi	0 <= X Semakin dekat ke 0 semakin baik
Computational Accuracy	X = A/T A = Jumlah perhitungan akurasi yang dihadapi oleh pengguna T = Waktu operasi	0 <= X Semakin dekat ke 0 semakin baik
Precision	X = A/T	0 <= X

	A = Jumlah hasil yang dihadapi oleh pengguna dengan tingkat presisi yang berbeda dari yang dibutuhkan waktu operasi	Semakin dekat ke 0 semakin baik

c. *Interoperability*

Sub-faktor *interoperability* menggambarkan tentang kemampuan sebuah perangkat lunak untuk dapat berinteraksi dengan sistem yang lain atau dengan sistem tertentu.

Tabel 2.3 *Interoperability* Metriks

Nama Metriks	**Pengukuran**	**Intrepretasi nilai**
Data exchangeability (Data format based)	X = A/B A = jumlah format yang data yang disetujui untuk ditukar dengan perangkat lunak lain atau sistem pengujian bursa data B = Total jumlah format data yang akan diperlukan	0 <= X <= 1 Semakin dekat ke 1.0 semakin baik
Data exchangeability (User's success attempt based)	X = 1-A/B A = Jumlah kasus dimana pengguna gagal untuk pertukaran data dengan perangkat lunak atau sistem lain B = Jumlah kasus di mana pengguna upaya	0 <= X <= 1 Semakin dekat ke 1.0 semakin baik

	untuk pertukaran data	

d. Security

Sub-faktor ini berhubungan dengan keamanan yang dimiliki oleh sebuah perangkat lunak. Keamanan yang dimaksud dapat berupa pemberian hak akses kepada penggunanya

Tabel 2.4 *Security* Metriks

Nama Metriks	Pengukuran	Intrepretasi nilai
Access auditability	X = A/B A = Jumlah "pengguna akses ke sistem data" tercatat dalam database sejarah akses B = Jumlah "pengguna akses ke sistem dan data" dilakukan selama evaluasi	0 <= X <= 1 Semakin dekat ke 1.0 semakin baik
Access controllability	X = A/B A = Jumlah yang terdeteksi berbagai jenis operasi ilegal B = Jumlah jenis operasi ilegal seperti dalam spesifikasi	0 <= X <= 1 Semakin dekat ke 1.0 semakin baik
Data corruption prevention	X = 1 – A/N A = Jumlah kali bahwa peristiwa korupsi data terbesar yang terjadi N = Jumlah uji coba menyebabkan acara korupsi data	0 <= X <= 1 Semakin dekat ke 1.0 semakin baik

e. Functionality Compliance

Functionality compliance menggambarkan tentang kesesuaian antara fungsionalitas yang ada pada sebuah perangkat lunak dengan aturan standar yang berlaku pada perangkat lunak lainnya yang sejenis.

Tabel 2.5 *Functionality* Metriks

Nama Metriks	Pengukuran	Intrepretasi nilai
Functional compliance	X = 1 - A/B A = Jumlah item kepatuhan fungsi tertentu yang belum dilakasankan selama pengujian B = Total jumlah item kepatuhan fungsi tertentu	0 <= X <= 1 Semakin dekat ke 1.0 semakin baik
Interface standard compliance	X = A/B A = Jumlah interface diterapkan dengan benar sebagaimana ditentukan B = jumlah interface yang membutuhkan pemenuhan	0 <= X <= 1 Semakin dekat ke 1.0 semakin baik

2. Reliability

Faktor *reliability* berhubungan dengan keandalan atau seberapa baik tingkat pelayanan sebuah perangkat lunak apabila digunakan dalam kondisi tertentu.

a. Maturity

Maturity berhubungan dengan kelayakan sebuah perangkat lunak dalam menangani kegagalan atau kesalahan yang terdapat didalamnya.

Tabel 2.6 *Maturity* Metriks

Nama Metriks	**Pengukuran**	**Intrepretasi nilai**
Estimated latent fault density	X = {ABS(A1 – A2)}/B (X:perkiraan yang kepadatan kesalahan laten) ABS ()= Absolut A1 = Jumlah kesalahan laten memprediksi dalam produk perangkat lunak A2 = Jumlah sebenarnya terprediksi kesalahan. B = Produk Ukuran	0 <= X Hal ini tergantung pada tahap pengujian. Pada tahap selanjutnya lebih kecil lebih baik.
Failure density against test cases	X = A1/A2 A1 = Jumlah kegagalan terdeteksi A2 = Jumlah kasus uji yang dilakukan	0 <= X Hal ini tergantung pada tahap pengujian. Pada tahap selanjutnya lebih kecil lebih baik.
Fault density	X = A/B A = Jumlah kesalahan yang terdeteksi B = Ukuran Produk	0 <= X Hal ini tergantung pada tahap pengujian. Pada tahap selanjutnya lebih kecil lebih baik.
Fault removal	a).X = A1/A2 A1= Jumlah kesalahan dikoreksi A2= Jumlah Sebenarnya terdeteksi kesalahan. b).Y = A1/A3 A= Jumlah kesalahan	0 <= X <= 1 Semakin dekat 1.0 semakin lebih baik sebagai kesalahan yang lebih sedikit 0 <= Y

	laten memprediksi dalam produk perangkat lunak.	Semakin dekat 1.0 semakin lebih baik sebagai kesalahan yang lebih sedikit
Mean time between failures (MTBF)	a).X = T1/A b).Y = T2 / A T1=Waktu operasi T2=Jumlah interval waktu yang berturut-turut mengalami kegagalan A=Jumlah sebenarnya terdeteksi kegagalan,(kegagalan terjadi selama waktu operasi)	0 <= X, Y Semakin lama semakin baik, seperti waktu yang lebih lama dapat diharapkan anatara kegagalan
Test coverage (Specified operation scenario testing coverage)	X = A / B A=Jumlah kasus uji benerbener dilakukan mewakili skenario operasi selama pengujian. B=Jumlah kasus uji yang akan dilakukan untuk menutupi kebutuhan	0 <= X <= 1 Semakin mendekati 1.0 semakin lebih baik
Test maturity	X = A / B A=Jumlah kasus uji lulus selama pengujian atau operasi B=Jumlah kasus uji yang akan dilakukan untuk menutupi kebutuhan	0 <= X <= 1 Semakin mendekati 1.0 semakin lebih baik

b. *Fault Tolerance*

Fault tolerance merupakan sub-faktor *reliability* yang menggambarkan tentang seberapa jauh sebuah perangkat lunak dapat memberikan toleransi kegagalan atas kesalahan yang terjadi saat digunakan pada kondisi tertentu.

Tabel 2.7 *Fault tolerance* Metriks

Nama Metriks	**Pengukuran**	**Intrepretasi nilai**
Breakdown avoidance	X = 1-A/B A = Jumlah kerusakan B = Jumlah kegagalan	0 <= X <= 1 Semakin dekat ke 1.0 semakin baik
Failure avoidance	X = A/B A = Jumlah kejadian kegagalan kritis dan serius terhadap kasus uji dari pola kesalahan B=Jumlah kasus uji dieksekusi dari pola kesalahan.	0 <= X <= 1 Semakin dekat ke 1.0 semakin baik
Incorrect operation avoidance	X = A/B A = Jumlah kejadian kegagalan kritis dan serius terhadap kasus uji dari pola kesalahan B=Jumlah kasus uji dieksekusi dari pola kesalahan.	0 <= X <= 1 Semakin dekat ke 1.0 semakin baik

c. *Recoverability*

Sub-faktor ini berhubungan dengan kemampuan sebuah perangkat lunak untuk dapat membangun kembali kinerja dan memulihkan data baik secara langsung maupun secara tidak langsung.

Tabel 2.8 *Recoverability* Metriks

Nama Metriks	**Pengukuran**	**Intrepretasi nilai**
Availability	a).X={To/(To+Tr)} To=Waktu untuk operasi Tr=Waktu untuk memperbaiki A1=Total kasus yang tersedia pengguna software pengguna berhasil ketika pengguna mencoba untuk menggunakan b).Y=A1/A2 A2=Jumlah kasus upaya pengguna untuk menggunakan perangkat lunak selama waktu pengamatan	0 <= X <= 1 Semakin dekat ke 1.0 semakin baik, karena pengguna dapat menggunakan perangkat lunak untuk lebih banyak waktu 0 <= Y <= 1 Semakin dekat ke 1.0 semakin baik
Mean down time	X=T/N T= Jumlah down time N=Jumlah kerusakan yang diamati kasus terburuk atau distribusi down time harus diukur	0 < X Semakin dekat ke 1.0 semakin baik
Mean recovery time	X=Sum(T)/N	0 < X

	T=Waktu untuk pemulihan sistem perangkat lunak pada setiap kesempatan B=Jumlah kasus yang diamati sistem software masuk recovery	Semakin dekat ke 1.0 semakin baik
Restartability	X=A/B A=Jumlah restart yang bertemu untuk waktu yang dibutuhkan selama pengujian atau operasi pengguna B=Total jumlah restart selama pengujian atau operasi pengguna	0 <= X <= 1 Semakin dekat ke 1.0 semakin baik, karena pengguna dapat restart dengan mudah.
Restorability	X=A/B A=Jumlah kasus restorasi berhasil dilakukan B=Jumlah kasus restorasi diuji sesuai kebutuhan	0 <= X <= 1 Semakin dekat ke 1.0 semakin baik, karena produk lebih mampu untuk mengembangkan dalam kasus tertentu.
Restore effectiveness	X=A/B A=Jumlah kasus berhasil dipublikasikan memenuhi terget mengembalikan waktu B=Jumlah kasus yang dilakukan	0 <= X <= 1 Semakin dekat ke 1.0 semakin baik, karena proses restorasi di produk lebih efektif.

d. *Reliability Compliance*

Reliability compliance berhubungan dengan kesesuaian antara tingkat pelayanan yang diberikan oleh perangkat lunak dengan aturan standar yang digunakan pada setiap perangkat lunak yang memiliki tipe yang sama.

Tabel 2.9 *Reliability compliance* Metriks

Nama Metriks	**Pengukuran**	**Intrepretasi nilai**
Reliability compliance	X = 1 - A / B A = Jumlah item kepuasan keandalan ditentukan yang belum dilaksanakan selama pengujian B = Total jumlah item kepatuhan keandalan ditentukan	0 <= X <= 1 Semakin dekat ke 1.0 semakin baik

3. Usability

Usability berhubungan dengan seberapa baik perangkat lunak dapat dipahami, dipelajari, dan digunakan.

a. *Understandability*

Sub-faktor ini berhubungan dengan seberapa jauh sebuah perangkat lunak dapat dipahami oleh pengguna baik dari secara konsep logis dan penerapan penggunaan perangkat lunak tersebut.

Tabel 2.10 *Understandability* Metriks

Nama Metriks	**Pengukuran**	**Intrepretasi nilai**
Completeness of description	X = A/B A =Jumlah fungsi (jenis	0 <= X <= 1 Semakin dekat ke 1.0

	fungsi) dipahami B = Total jumlah (jenis fungsi)	semakin baik.
Demonstration accessibility	X = A/B A = Jumlah demonstrasi / tutorial yang pengguna berhasil mengaksesnya B =Jumlah demonstrasi /tutorial yang tersedia	0 <= X <= 1 Semakin dekat ke 1.0 semakin baik.
Demonstration accessibility in use	X = A/B A =Jumlah kasus di mana pengguna berhasil melihat demonstrasi ketika pengguna mencoba untuk meliha demonstrasi B =Jumlah kasus di mana pengguna mencoba untuk melihat demonstrasi selama pengamatan	0 <= X <= 1 Semakin dekat ke 1.0 semakin baik.
Demonstration effectiveness	X = A/B A =Jumlah fungsi yang di operasikan berhasil B = Jumlah demonstrasi / tutorial yang di akses	0 <= X <= 1 Semakin dekat ke 1.0 semakin baik.
Evident functions	X = A/B A =Jumlah fungsi (jenis fungsi) yang didentifikasi oleh pengguna B =Total jumlah fungsi	0 <= X <= 1 Semakin dekat ke 1.0 semakin baik.

	yang sebenarnya	
Function understand-ability	X = A/B A =Jumlah fungsi antarmuka yang tujuanya dengan benar dijalankan oleh pengguna B =Jumlah fungsi yang tersedia dari antarmuka	0 <= X <= 1 Semakin dekat ke 1.0 semakin baik.
Understandable input and output	X = A/B A =Jumlah input dan output data item yang pengguna berhasil mengerti B =Jumlah item data input dan output yang tersedia dari antarmuka.	0 <= X <= 1 Semakin dekat ke 1.0 semakin baik.

b. *Learnability*

Learnability menggambarkan tentang sebuah perangkat lunak dapat dipelajari dengan baik oleh penggunanya.

Tabel 2.11 *Learnability* Metriks

Nama Metriks	**Pengukuran**	**Intrepretasi nilai**
Ease of function learning	T= Berarti waktu yang dibutuhkan untuk pembelajaran menggunakan fungsi dengan benar	0 < T Semakin pendek semakin baik
Ease of learning to perform a task in use	T = Jumlah waktu operasi pengguna sampai waktu	0 < T Semakin pendek semakin

	dicapai untuk melakukan tugas tertentu dalam waktu singkat	baik
Effectiveness of the user documentation and/or help system	X = A/B A = Jumlah tugas yang berhasil diselesaikan setelah mengakses bantuan online atau dokumentasi B =Jumlah tugas diuji	0 <= X <= 1 Semakin dekat ke 1.0 semakin baik

c. *Operability*

Operability berhubungan dengan seberapa jauh perangkat lunak dapat dioperasikan oleh penggunanya.

Tabel 2.12 *Operability* Metriks

Nama Metriks	**Pengukuran**	**Intrepretasi nilai**
Operational consistency in use	a).X = 1-A/B A = Jumlah pesan atau fungsi uang menemukan data yang tidak konsisten dengan ekspektasi pengguna B = Jumlah pesan atau fungsi b).Y=N/UOT N= Jumlah pesan atau fungsi yang menemukan data yang tidak konsisten	0 <= X <= 1 Semakin dekat ke 1.0 semakin baik 0 <= Y Semakin dekat ke 0.0 semakin baik

	dengan ekspektasi pengguna UOT = Pengguna waktu operasi selama sistem berjalan.	

d. *Attractiveness*

Sub-faktor ini menggambarkan tentang bagaimana sebuah perangkat lunak dapat menarik perhatian bagi penggunanya.

Tabel 2.13 *Attractiveness* Metriks

Nama Metriks	**Pengukuran**	**Intrepretasi nilai**
Attractive interaction	Kuesioner untuk menilai daya tarik dari antarmuka pengguan, setelah pengguna mengunakan.	Tergantung pada metode kuesioner scoringnya
Interface appearance customisability	X = A/B A = Jumlah elemen antarmuka disesuaikan dalam penampilan untuk kepuasan pengguna B = Jumlah antarmuka pengguna yang ingin menyesuaikan	0 <= X <= 1 Semakin dekat ke 1.0 semakin baik

e. *Usability Compliance*

Sub-faktor ini berhubungan dengan kesesuaian antara kegunaan perangkat lunak dengan standar yang digunakan oleh perangkat lunak sejenis lainnya.

Tabel 2.14 *Usability* Metriks

Nama Metriks	**Pengukuran**	**Intrepretasi nilai**
Usability compliance	X = 1-A/B A = Jumlah item kepuasan kegunaan tertentu yang belum dilaksanakan selama pengujian B = Total jumlah item kepuasan kegunaan tertentu	0 <= X <= 1 Semakin dekat ke 1.0 semakin baik

4. Efficiency

Faktor efficiency berhubungan dengan efisiensi dari seberapa besar sumber daya yang digunakan oleh sebuah perangkat lunak.

a. Time Behaviour

Time behaviour berhubungan dengan waktu yang dibutuhkan oleh sebuah perangkat lunak untuk menjalankan suatu fungsi atau tujuan tertentu.

Tabel 2.15 *Time behaviour* Metriks

Nama Metriks	**Pengukuran**	**Intrepretasi nilai**
Response time	T = (Waktu untuk mendapatkan hasil)-(saat masuk perintah selesai)	0 < T Semakin cepat semakin baik.
Response time (Mean time to response)	Tmean = (Ti) ∑/ N, (untuk i = 1 sampai N) TXmean = diperlukan waktu respon rata-rata Ti = waktu respon untuk evaluasi ke-i (shot)	0 <= X Semakin dekat ke 1.0 dan kurang dari 1.0 lebih baik

	N = jumlah evaluasi (tembakan sampel)	
Response time (Worst case response time ratio)	X = Tmax / Rmax Tmax = MAX (Ti) (untuk i = 1 sampai N) Rmax = diperlukan waktu respons maksimum MAX (Ti) = waktu respons maksimum antara evaluasi N = jumlah evaluasi (tembakan sampel) Ti = waktu respon untuk evaluasi ke-i (shot)	0 <= X Semakin ke 1 dan kurang dari 1 akan lebih baik

b. Resource Utilization

Sub-faktor ini menggambarkan tentang jumlah sumber daya yang digunakan oleh perangkat lunak untuk melakukan fungsi atau tujuan tertentu.

Tabel 2.16 *Resource* Metriks

Nama Metriks	**Pengukuran**	**Intrepretasi nilai**
I/O devices utilisation	X = A/B A = Waktu I/O perangkat diduduki B = Ditentukan waktu yang dirancang untuk menempati	0 <= X <= 1 Semakin dekat ke 1.0 semakin baik
I/O loading limits	X = Amax/Rmax	0 <= X

	Amax = MAX (Ai), (untuk i = 1 sampai N) Rmax = maksimum yang diperlukan I / O pesan MAX (Ai) = Jumlah maksimum pesan I / O dari 1 untuk evaluasi ke-i. N = jumlah evaluasi.	Semakin kecil semakin baik
I/O related errors	X = A/T A= Jumlah pesan peringatan atau kegagalan sistem T= Penggunaan waktu operasi selam pengamatan pengguna	0 <= X Semakin kecil semakin baik
Mean I/O fulfillment ratio	X = Amean/Rmean Amean = ∑ (Ai) / N Rmean = diperlukan rata jumlah pesan I /O Ai = jumlah pesan kesalahan I / O untuk i th evaluasi N = jumlah evaluasi	0 <= X Semakin kecil semakin baik
User waiting time of I/O devices utilisation	T= Waktu yang dihabiskan untu menugguakhir dari perangkat I/O operasi	0 < T Semakin kecil semakin baik

c. *Efficiency Compliance*

Efficiency compliance berhubungan dengan kesesuaian antara sumber daya yang digunakan oleh sebuah perangkat lunak dengan standar penggunaan sumber daya pada perangkat lunak lainnya yang sejenis.

Tabel 2.17 *Efficiency compliance* Metriks

Nama Metriks	**Pengukuran**	**Intrepretasi nilai**
Efficiency Compliance	X = 1-A/B (X:Rasio item kepatuhan berkaitan denga efisiensi) A = Jumlah efisiensi item kepatuahan ditentukan yang belum dilaksanakan selama pengujian B = Total jumlah item kepatuhan efisiensi dintentukan	0 <= X <= 1 Semakin dekat ke 1.0 semakin baik

5. Maintainability

Maintainability menggambarkan tentang pemeliharaan sebuah perangkat lunak, seberapa baik perangkat lunak tersebut dapat dipertahankan.

a. *Analyzability*

Analyzability berhubungan dengan seberapa jauh sebuah perangkat lunak dapat di analisis, hal ini diperlukan untuk analisis kekurangan atau penyebab kegagalan agar dapat diketahui bagian mana yang perlu dimodifikasi.

Tabel 2.18 *Analyzability* Metriks

Nama Metriks	**Pengukuran**	**Intrepretasi nilai**
Audit trail capability	X = A/B	0 <= X

Nama Metriks	Pengukuran	Intrepretasi nilai
	A = Jumlah data yang sebenarnya direkam selama operasi B = Jumlah data direncanakan akan cukup direkam untuk memantau status perangkat lunak selama operasi	Semakin dekat ke 1.0 semakin baik
Diagnostic function support	X = 1-A/B A = Jumlah kegagalan yang pemelihara dapat diagnose (menggunakan fungsi diagnostik) untuk memahami penyebab - kapal hubungan efek B = Total jumlah kegagalan terdaftar	0 <= X <= 1 Semakin dekat ke 1.0

b. *Changeability*

Changeability berhubungan dengan seberapa baik perangkat lunak dapat diubah, upaya ini diperlukan untuk modifikasi, penghapusan kesalahan atau perubahan lingkungan.

Tabel 2.19 *Changeability* Metriks

Nama Metriks	Pengukuran	Intrepretasi nilai
Change cycle efficiency	Average Time : Tav = Sum(Tu)/N Tu= Trc – Tsn Tsn = Waktu di mana	0 < Tav Semakin pendek lebih baik kecuali dari jumlah versi revisi itu besar

	pengguna selesai untuk mengirim permintaan untuk pemeliharaan untuk pemasok dengan laporan masalah Trc = Waktu di mana pengguna menerima rilis versi revisi (atau laporan status) N = Jumlah versi revisi	
Change implementation elapse time	Average Time : Tav = Sum(Tm) / N Tm=T out – T in T out = Waktu di mana penyebab kegagalan dihapus dengan mengubah perangkat lunak (atau status dilaporkan kembali ke pengguna) Tin = Waktu di mana penyebab kegagalan yang menemukan N = Jumlah kegagalan terdaftar dan dihapus	0 < Tav Semakin pendek lebih baik kecuali dari jumlah versi revisi itu besar
Modification complexity	T = Sum (A / B) / N A = Kerja dihabiskan untuk mengubah B = Ukuran perubahan software	0 < T Semakin dekat ke 1.0 semakin baik

	N = Jumlah perubahan	
Parameterised modifiability	X=1- A / B A = Jumlah kasus yang pemelihara gagal untuk mengubah perangkat lunak dengan menggunakan parameter B = Jumlah kasus yang maintainer mencoba untuk mengubah perangkat lunak dengan menggunakan parameter	0 <= X <= 1 Semakin dekat ke 1.0 semakin baik
Software change control capability	X= A / B A = Jumlah data log perubahan benar-benar dicatat B = Jumlah data perubahan log rencananya akan direkam cukup untuk melacak perubahan perangkat lunak	0 <= X <= 1 Semakin dekat ke 1.0 semakin baik

c. *Stability*

Sub-faktor ini berhubungan dengan stabilitas dari sebuah perangkat lunak yang memungkinkan untuk menyimpulkan resiko efek tak terduga yang disebabkan oleh modifikasi

Tabel 2.20 *Stability* Metriks

Nama Metriks	**Pengukuran**	**Intrepretasi nilai**
Change Success Ratio	X = Na/Ta Y={(Na/Ta)/(Nb/Nb)} Na = Jumlah kasus yang pengguna pertemuan kegagalan selama operasi setelah software berubah Nb = Jumlah kasus yang pengguna pertemuan kegagalan selama operasi sebelum perangkat lunak berubah Waktu ta = Operasi selama periode observasi ditentukan setelah perangkat lunak berubah Tb waktu = Operasi selama periode pengamatan ditentukan sebelum perangkat lunak berubah	0 <= X,Y Semakin dekat ke 1.0 semakin baik
Modification impact localisation (Emerging failure after change)	X=A/N A = Jumlah kegagalan muncul setelah kegagalan diselesaikan oleh perubahan selama periode tertentu N = Jumlah kegagalan diselesaikan	0<=X Semakin dekat ke 0 semakin baik

d. *Testability*

Testability menggambarkan tentang bagaimana perangkat lunak dapat diuji, hal ini untuk menyimpulkan tentang upaya yang diperlukan untuk memvalidasi perangkat lunak dan cakupan pengujian.

Tabel 2.21 *Testability* Metriks

Nama Metriks	**Pengukuran**	**Intrepretasi nilai**
Availability of built-in test function	X = A/B A = Jumlah kasus dimana pengelola dapat mengunakan sesuia buit-in fungsi tes B =Jumlah kasus peluang uji	0 <= X <= 1 Semakin dekat ke 1.0 semakin baik
Re-test efficiency	X = Sum(T)/N T = Waktu yang dihabiskan untuk menguji untuk memastikan apakah melaporkan kegagalan itu diselesaikan atau tidak N = Jumlah kegagalan diselesaikan	0 <= X Semakin dekat ke 1.0 semakin baik
Test restartability	X = A/B A = Jumlah kasus di mana pengelola dapat menghentikan sebentar dan restart melaksanakan uji coba pada titik-titik yang diinginkan untuk	0 <= X Semakin dekat ke 1.0 semakin baik

	memeriksa langkah demi langkah B = Jumlah kasus jeda melaksanakan uji coba	

e. Maintainability Compliance

Hal ini berhubungan dengan kesesuaian antara pemeliharaan yang dilakukan terhadap perangkat lunak dengan standarisasi yang terdapat pada pemeliharaan perangkat lunak lainnya yang sejenis.

Tabel 2.22 *Maintainability Compliance* Metriks

Nama Metriks	**Pengukuran**	**Intrepretasi nilai**
Maintainability compliance	X = 1-A/B A = Jumlah item kepatuhan rawatan ditentukan yang belum dilaksanakan selama pengujian B = Total jumlah item kepatuhan rawatan ditentukan	0 <= X <= 1 Semakin dekat ke 1.0 semakin baik

6. *Portability*

Portability menggambarkan tentang kemampuan sebuah perangkat lunak untuk dapat berpindah dari sebuah lingkungan atau sistem ke sistem lainnya.

a. *Adaptability*

Sub-faktor ini berhubungan dengan seberapa jauh sebuah perangkat lunak dapat beradaptasi dengan perubahan lingkungan atau sistem yang berbeda.

Tabel 2.23 *Adaptability* Metriks

Nama Metriks	**Pengukuran**	**Intrepretasi nilai**
Adaptability of data structures	X = A/B A = Jumlah data yang beroperasi dan tetapi tidak diamati karena operasi tidak lengkap disebabkan oleh keterbatasan adaptasi B = Jumlah data yang diharapkan dapat beroperasi dalam lingkungan yang perangkat lunak disesuaikan	0 <= X <= 1 Semakin dekat ke 1.0 semakin baik
Hardware environmental adaptability	X = 1-A/B A = Jumlah fungsional operasi tugas tidak selesai atau tidak cukup menghasilkan untuk memenuhi selama combined pengujian operasi dengan hardware lingkungan B = Total jumlah fungsi yang diuji	0 <= X <= 1 Semakin dekat ke 1.0 semakin baik

b. *Installability*

Installability menggambarkan tentang seberapa baik perangkat lunak dapat digunakan dalam lingkungan atau sistem tertentu.

Tabel 2.24 *Instability* Metriks

Nama Metriks	**Pengukuran**	**Intrepretasi nilai**
Ease of installation	X = A/B A = Jumlah kasus yang pengguna berhasil mengubah menginstal operasi untuk / kenyamanan nya B = Total jumlah kasus yang pengguna mencoba untuk mengubah menginstal operasi untuk / kenyamanan nya	0 <= X <= 1 Semakin dekat ke 1.0 semakin baik
Ease of Setup Retry	X = 1-A/B A = Jumlah kasus di mana pengguna gagal di coba selama set - up operasi B = Total jumlah kasus di mana pengguna upaya untuk kembali mencoba setup selama set - up operasi	0 <= X <= 1 Semakin dekat ke 1.0 semakin baik

c. Co-existence

Co-existence berhubungan dengan bagaimana perangkat lunak dapat berdampingan dengan produk atau perangkat lunak lain pada suatu lingkungan atau sistem yang sama untuk mengetahui tentang dependensi, perilaku, atau efek samping yang ditimbulkan.

Tabel 2.25 *Co-existence* Metriks

Nama Metriks	**Pengukuran**	**Intrepretasi nilai**
Available coexistence	X = A/T A = Jumlah hambatan (kegagalan) tak terduga yang pengguna temukan selama operasi dengan software lain T = Waktu durasi bersamaan operasi perangkat lunak lain	0 <= X Semakin dekat ke 1.0 semakin baik

d. *Replaceability*

Replaceability berhubungan dengan bagaimana sebuah perangkat lunak dapat menggantikan perangkat lunak lain apakah ada kebergantungan kepada perangkat lunak lain saat perangkat lunak tersebut digunakan.

Tabel 2.26 *Replaceability* Metriks

Nama Metriks	**Pengukuran**	**Intrepretasi nilai**
Continued use of data	X = A/B A = jumlah data yang digunakan dalam perangkat lunak lain yang akan diganti dan menegaskan bahwa mereka mampu terus menerus digunakan B = jumlah data yang digunakan dalam perangkat lunak lain	0 <= X <= 1 Semakin dekat ke 1.0 semakin baik

	untuk diganti dan rencananya akan terus digunakan kembali	
Function inclusiveness	X = A/B A = jumlah fungsi yang menghasilkan hasil yang sama seperti sebelumnya diproduksi dan di mana perubahan tidak akan en diperlukan B = jumlah fungsi diuji yang mirip dengan fungsi yang disediakan oleh perangkat lunak lain yang akan diganti	0 <= X <= 1 Semakin dekat ke 1.0 semakin baik
User support functional consistency	X = 1-A1/A2 A = Jumlah fungsi baru yang ditemukan pengguna , konsisten dengan ekspektasi pengguna B = Jumlah fungsi baru	0 <= X Semakin dekat ke 1.0 semakin baik

e. Portability Compliance

Portability compliance berhubungan dengan kesesuaian antara perubahan yang dapat dilakukan oleh sebuah perangkat lunak dengan standarisasi portability yang terdapat pada perangkat lunak lain yang sejenis.

Tabel 2.27 *Portability compliance* Metriks

Nama Metriks	**Pengukuran**	**Intrepretasi nilai**
Portability compliance	X = 1-A/B A = Jumlah portabilitas item kepatuhan ditentukan yang belum dilaksanakan selama pengujian B = Total jumlah portabilitas item kepatuhan ditentukan	0 <= X <= 1 Semakin dekat ke 1.0 semakin baik

2.29.2 Model *Quality In Use*

Model *Quality In Use* mempunyai 4 Faktor yaitu *Effectiveness, Productivity, Safety dan Satisfaction*. Masing masing faktor juga mempunyai beberapa metriks. Berikut penjelasannya :

1. *Effectiveness* (Efektivitas)

   Effectiveness berhubungan dengan seberapa efektif suatu perangkat lunak terhadap penggunanya.

Tabel 2.28 *Effectiveness*

Nama Metriks	**Pengukuran**	**Intrepretasi nilai**
Effectiveness	*Task completion*	X = A/B A = jumlah tugas diselesaikan B = jumlah tugas yang dicoba
	Error frequency	X = A/T A = jumlah kesalahan yang dibuat oleh pengguna T = jumlah tugas dicoba

2. *Productivity* (Produktivitas)

   Productivity berhubungan dengan seberapa produktif kah perangkat lunak terhadap penggunanya.

Tabel 2.29 *Productivity*

Nama Metriks	**Pengukuran**	**Intrepretasi nilai**
Productivity	*Task time*	X = A/B A = jumlah tugas diselesaikan B = jumlah tugas yang dicoba
	Task efficiency	X = A/T A = jumlah kesalahan yang dibuat oleh pengguna T = jumlah tugas dicoba
	Economic productivity	X = M1 / C M1 = efektivitas tugas C = total biaya tugas
	Productive proportion	X = Ta / Tb Ta = waktu produktif = waktu tugas – waktu bantu - waktu error - waktu pencarian Tb = waktu tugas
	Relative user efficiency	X = A / B A = efisiensi tugas pengguna biasa ini B = efisiensi tugas ahli pengguna

3. *Safety*

   Safety (Keamanan) berhubungan dengan seberapa aman perangkat lunak yang dianalisis terhadap penggunanya.

Tabel 2.30 *Safety*

Nama Metriks	**Pengukuran**	**Intrepretasi nilai**
Safety	*Safety of people affected by use of the system*	X = 1-A / B A = jumlah orang menaruh di hazard B = Jumlah orang yang berpotensi terkena dampak oleh sistem
	Economic damage	X = 1-A / B A = jumlah kejadian kerusakan ekonomi B = Jumlah situasi penggunaan
	Software damage	X = 1-A / B A = jumlah kejadian perangkat lunak corrupt B = Jumlah situasi penggunaan

4. *Satisfaction*

   Satisfaction (Kepuasan) berhubungan dengan kepuasan pelanggan terhadap perangkat lunak yang digunakan.

Tabel 2.31 *Satisfaction*

Nama Metriks	**Pengukuran**	**Intrepretasi nilai**
Satisfaction	*Discretionary usage*	X = A / B A = berapa kali fungsi perangkat lunak tertentu / aplikasi / sistem yang digunakan B = jumlah pengguna menggunakan
	Satisfaction questionnaire	X = A / n A = menanggapi pertanyaan n = jumlah tanggapan

2.30 Metode Pengukuran Kualitas Perangkat Lunak

Pengukuran kualitas perangkat lunak merupakan hal yang terpenting dari praktek rekayasa perangkat lunak yang baik. Serta pengukuran aktivitas manusia menambah nilai dan terlibat aktif dalam proses informasi. Pengukuran ini

membantu untuk membuat karakteristik khusus dari proses dan produk informasi. Pengukuran meliputi evaluasi kuantitatif yang biasanya menggunakan metriks serta ukuran dapat menggunakan secara langsung menentukan pencapaian tujuan kualitas numerik.

` *Metriks Internal* Dapat digunakan untuk membantu memprediksi apakah perangkat lunak akan memenuhi kebutuhan keandalan organisasi selama pengembangan sistem perangkat lunak. *Maturity Metrik* menunjukkan satu set atribut untuk menilai kematangan suatu software. Deteksi kesalahan adalah Sebuah ukuran seberapa banyak *defects* terdeteksi dalam produk perangkat lunak. Metrik ini dikumpulkan dengan menghitung jumlah terdeteksi *bug* yang ditemukan dalam review dan membandingkan angka itu dengan jumlah yang diperkirakan. dapat ditemukan dalam tahap pengujian statis. Metriks yang dihitung oleh rumus X = A / B , dimana A adalah jumlah absolut bug terdeteksi (dari laporan review) dan B adalah perkiraan jumlah *expected*. Nilai tinggi untuk metriks ini menyiratkan baik tingkat kualitas produk. Penghapusan kesalahan adalah Sebuah ukuran seberapa banyak cacat yang ditemukan selama review dihapus (dikoreksi) selama desain dan implementasi tahap. Metrik dapat dihitung dengan rumus X = A / B. dimana A adalah jumlah bug tetap selama desain dan coding dan B adalah Jumlah yang ditemukan selama review. Semakin dekat nilai adalah 1, yang lebih baik. A Nilai penghapusan kesalahan orang akan berarti bahwa setiap cacat terdeteksi telah dihapus.

Metriks Eksternal ISO 9126 mempunyai beberapa metrik eksternal. Metriks eksternal didefinisikan dalam ISO 9126-2 dan diukur selama pengujian dinamis. Kategori metrik ini sesuai dengan yang internal yaitu *maturity, fault 38 tolerance, recoverability, and compliance*. Metriks Kematangan mengukur kebebasan sistem perangkat lunak dari kegagalan disebabkan oleh bug di sistem itu sendiri. Faktor eksternal ini lebih menilai dari segi program kepada pengguna sedangkan internal lebih menilai dari segi program tersebut. Perhitungan Metriks Pada metriks 9126 terdapat 6 faktor dan beberapa subfaktor yang akan digunakan sebagai perhitungan yang tentunya akan dilihat dari karakteristik perangkat lunak

tersebut. Setiap faktor karakteristik mempunyai hitungan tersendiri yang sudah ditentukan oleh metriks 9126[37].

Tabel 2.32 Subfaktor Metriks

Funcionality Metrics	*Suitability metrics* *Accuracy metrics* *Interoperability metrics* *Security metrics* *Functionality compliance metrics*
Reliability metrics	*Maturity metrics* *Fault tolerance metrics* *Recoverability metrics* *Reliability compliance metrics*
Usability Metrics	*Understandability metrics* *Learnability metrics* *Operability metrics* *Attractiveness metrics* *Usability compliance metrics*
Efficiency metrics	*Time behaviour metrics* *Resource utilisation metrics* *Efficiency compliance metrics*
Maintainability metrics	*Analysability metrics* *Changeability metrics* *Stability metrics* *Testability metrics* *Maintainability compliance metrics*
Portability metrics	*Adaptability metrics* *Installability metrics* *Co-existence metrics* *Replaceability metrics* *Portability compliance metrics*

2.31 Pengukuran Perangkat Lunak

Pengukuran perangkat lunak adalah faktor utama untuk mengetahui seberapa baik perangkat lunak yang akan dinilai berdasarkan metode yang digunakan. Pengukuran perangkat lunak dilakukan dengan cara mencari nilai kriteria perangkat lunak tersebut dan juga mencari bobot kepentingan berdasarkan metode yang akan digunakan. Penilaian keseluruhan dapat diperoleh dari hasil nilai kriteria dan juga nilai bobot. Perhitungan bobot dan nilai kriteria bisa bervariasi tergantung dengan metode apa yang digunakan.

Metode Pembobotan Rank Order Centroid (ROC) didasarkan pada tingkat kepentingan atau prioritas dari kriteria. Menurut Jeffreys dan Cockfield dalam Afiefah Rahma, teknik ROC memberikan bobot pada setiap kriteria sesuai dengan ranking yang dinilai berdasarkan tingkat prioritas. Biasanya dibentuk dengan pernyataan "Kriteria 1 lebih penting dari kriteria 2, yang lebih penting dari kriteria 3" dan seterusnya hingga kriteria ke n, ditulis . Untuk menentukan bobotnya, diberikan aturan yang sama yaitu dimana merupakan bobot untuk kriteria[37]. Atau dapat dijelaskan sebagai berikut:

Jika Cr1 ≥ Cr2 ≥ Cr3 ≥ ... ≥ Crn

Maka W1 ≥ W2 ≥ W3 ≥ ... ≥ Wn

Selanjutnya, jika k merupakan banyaknya kriteria, maka

$$W1 = \frac{1+\frac{1}{2}+\frac{1}{3}+\cdots+1/k}{k}$$

$$W2 = \frac{0+\frac{1}{2}+\frac{1}{3}+\cdots+1/k}{k}$$

$$W3 = \frac{0+0+\frac{1}{3}+\cdots+1/k}{k}$$

$$W4 = \frac{0+0+0+\cdots+1/k}{k}$$

Secara umum pembobotan ROC dapat dirumuskan sebagai berikut :

$$Wk = 1/k \sum_{t=i}^{k} (1/i)$$

2.32 Metode Pengolahan Kuisioner

Kuesioner merupakan sekumpulan pertanyaan yang diajukan kepada responden dengan tujuan dan maksud tertentu untuk memperoleh data, sehingga data tersebut dapat digunakan atau diolah kembali. Untuk mengolah data dari kuesioner tersebut dapat menggunakan berbagai macam metode pengolahan data sesuai dengan rumus metriks yang digunakan pada ISO 9126 Kuisioner ini diajukan untuk memenuhi nilai A, B maupun T yang dibutuhkan untuk menghitung nilai metriks dai setiap kriteria yang sudah ada. Dari hasil jawaban yang diberikan oleh responden akan dicari nilai rata rata, hasil jawaban responden akan dijumlahkan dan totalnya akan dibagi dengan jumlah responden. Maka akan didapat nilai A, B ataupun T [37].

2.33 Metode Penilaian

Perhitungan nilai sub faktor , faktor maupun nilai keseluruhan menjadi faktor penting untuk mengetahui hasil akhir sebuah nilai. menentukan nilai kriteria per faktor nya dan nilai akhir dengan rumus perhitungan *Weight Summation*. Nilai w adalah nilai metriks yang diperoleh dari hasil pengolahan kuisioner dan m adalah bobot dari sub faktor atau faktor yang sudah diperoleh dari hasil perhitungan dengan menggunakan metode *Rank Order Centroid* (ROC) [37].

Nilai Kriteria

$$\text{Kriteria (a)} = \sum_{j=1}^{M} W_j . m_{ij} \ldots\ldots$$

$$= w_1 m_1 + w_2 m_2 + \ldots.. \text{wnmn}$$