BAB II

LANDASAN TEORI

Dalam merancang dan membangun aplikasi, sangatlah penting untuk mengetahui terlebih dahulu dasar-dasar teori yang digunakan. Dasar-dasar teori tersebut digunakan sebagai landasan berpikir dalam melakukan pembahasan lebih lanjut sehingga terbentuk suatu aplikasi yang sesuai dengan tujuan awal.

2.1 *Harbour Mobile Crane*

Harbour mobile crane (HMC) adalah sebuah jenis alat berat yang terdiri dari kerangka bahu (*boom*) dilengkapi tali penarik (*wayroof*) dan digerakkan oleh mesin di atas roda ban yang bisa berpindah-pindah di sekitar area pelabuhan. Alat berat ini memiliki kapasitas angkat lebih dari 115 ton dengan jangkauan sekitar radius 40 meter dari ruangan kabin operator, ini untuk *crane type* standar seperti LHM400. Mampu bekerja 24 jam tanpa berhenti dalam segala cuaca. Untuk mengoperasikan *crane* raksasa ini diperlukan operator yang handal dan terlatih, walaupun alat berat ini telah dilengkapi sistem keamanan (*safety*) yang tinggi untuk operasi tugas berat (@Warkop Aremania, 2012).

2.2 Mesin dan Kinerja Mesin

Menurut (Assuri, 2004), mesin adalah suatu peralatan yang digerakkan oleh suatu kekuatan atau tenaga yang dipergunakan untuk membantu manusia dalam mengerjakan produk atau bagian produk-produk tertentu. Mesin dikelompokkan menjadi dua, yaitu:

1. Mesin yang bersifat serbaguna (*general purpose machines*) merupakan mesin yang dibuat untuk mengerjakan pekerjaan-pekerjaan tertentu untuk berbagai jenis produk. Ciri-ciri dari *general purpose machines* adalah:
   a. Mesin ini diproduksi dalam bentuk standar dan atas dasar pasar (*ready stock*).
   b. Mesin ini memproduksi dalam *volume* yang besar, maka harganya relatif murah sehingga investasi dalam mesin lebih murah.
   c. Penggunaan mesin sangat fleksibel dan variasinya banyak.
   d. Dipergunakan kegiatan pengawasan atau inspeksi atas apa yang dikerjakan mesin tersebut.
   e. Biaya operasi lebih mahal.
   f. Biaya pemeliharaan lebih murah, karena bentuknya standar.
   g. Mesin ini tidak mudah ketinggalan jaman.
2. Mesin yang bersifat khusus (*special purpose machines*) merupakan mesin-mesin yang dibuat untuk mengerjakan satu atau beberapa jenis kegiatan yang sama. Ciri-ciri *special purpose machines* adalah:
   a. Mesin ini diuat atas dasar pesanan dan dalam jumlah kecil. Oleh karena itu harganya lebih mahal, sehingga investasi menjadi lebih mahal.
   b. Mesin ini biasanya otomatis, sehingga pekerjaan lebih cepat.
   c. Biaya pemeliharaan dari mesin lebih mahal karena dibutuhkan ahli khusus.
   d. Biaya produksi per unit relatif lebih rendah.
   e. Mesin ini mudah ketinggalan jaman.

Menurut Siringoringo & Sudiyantoro (2004), semakin seringnya mesin bekerja untuk memenuhi target yang kadang melebihi kapasitas dapat

menurunkan kemampuan mesin, menurunkan umur mesin dan sering membutuhkan pergantian komponen yang rusak. Apabila mesin atau peralatan yang digunakan mengalami kerusakan maka tujuan dari sebuah organisasi akan terhambat. Menurut Lazim & Ramayah (2010), untuk beroperasi secara efisien dan efektif, perusahaan perlu memastikan bahwa tidak terdapat gangguan mesin yang disebabkan oleh kerusakan, pemberhentian dan kegagalan mesin. Pada umumnya penyebab gangguan mesin dapat dikategorikan menjadi tiga, yaitu faktor manusia, mesin dan lingkungan. Faktor terpenting dari kondisi tersebut adalah kinerja mesin yang digunakan (Wahjudi, Tjitro, & Soeyono, 2009).

Berdasarkan hasil wawancara dengan Bapak Fanani, karyawan divisi Perencanaan Teknik & Administrasi PT. BJTI, kinerja alat atau mesin diukur berdasarkan ketersediaan (*availability*), kehandalan (*reliability*) dan penggunaan (*utilization*). Sedangkan menurut Warren (2011), kinerja mesin itu sendiri dapat diukur menggunakan *Key Performance Indicators* (KPI).

2.3 *Hour Meter Reading* (HRM)

Berdasarkan hasil wawancara dengan Bapak Fanani, karyawan divisi Perencanaan Teknik & Administrasi, *hour meter reading* (HRM) adalah piranti untuk mengukur penggunaan alat atau mesin tersebut.

2.4 Ketersediaan (*availability*)

Berdasarkan hasil wawancara dengan Bapak Fanani, karyawan divisi Perencanaan Teknik & Administrasi, ketersediaan (*availability*) adalah rasio tingkat kesiapan alat waktu akan dioperasi. Berikut ini adalah perhitungan ketersediaan (*availability*):

$$A = \frac{(TH - (BD + TM))}{TH}$$

Dimana:

A = *Availability* (ketersediaan).

TH = *Total Hours* (waktu yang diberikan terhadap mesin HMC).

B = *Breakdown* (waktu kegagalan mesin HMC).

TM = *Total Maintenance* (total waktu perawatan rutin mesin HMC).

2.5 Kehandalan (*reliability*)

Berdasarkan hasil wawancara dengan Bapak Fanani, karyawan divisi Perencanaan Teknik & Administrasi, kehandalan (*reliability*) adalah rasio terhadap tingkat ketahanan alat waktu beroperasi. Berikut ini adalah perhitungan kehandalan (*reliability*):

$$R = \frac{TH - BD}{TH} \times 100\%$$

Dimana:

R = *Reliability* (kehandalan).

TH = *Total Hours* (waktu yang diberikan terhadap mesin HMC).

B = *Breakdown* (waktu kegagalan mesin HMC).

2.6 Penggunaan (*utilization*)

Berdasarkan hasil wawancara dengan Bapak Fanani, karyawan divisi Perencanaan Teknik & Administrasi, rasio terhadap jam kerja (operasional) alat. Berikut ini adalah perhitungan penggunaan (*utilization*):

$$U = \frac{TO}{TH} \times 100\%$$

Dimana:

U = *Utilization* (penggunaan).

TH = *Total Hours* (waktu yang diberikan terhadap mesin HMC).

TO = *Total Operation* (waktu bekerja mesin HMC).

2.7 *Key Performance Indicator*

Key Performance Indicator (KPI) adalah pengukuran yang mengevaluasi bagaimana sebuah perusahaan menjalankan visi strategis (Warren, 2011).

Dalam setiap proses pengukuran kinerja dibutuhkan suatu ukuran untuk mengetahui tingkat keberhasilan atau capaian dari kinerja perusahaan tersebut. Salah satu ukuran yang digunakan dalam proses pengukuran kinerja adalah Indikator Kinerja Utama/*Key Performance Indicator* (KPI). Indikator Kinerja Utama/*Key Performance Indicator* (KPI) merupakan suatu indikator yang digunakan untuk mengetahui seberapa jauh strategi yang telah dilakukan oleh perusahaan sesuai dengan visi dan misi perusahaan (Moeheriono, 2012).

Menurut Darly (2005), *Key Performance Indicator* (KPI) adalah informasi yang dapat digunakan untuk memberikan panduan secara aktif terhadap kinerja bisnis yang dapat diakses oleh pengambil keputusan menggunakan dashboard. Menurut Anna & Martina (2012), *Key Performance Indicator* (KPI) juga dapat digunakan untuk mendukung proses pengambilan keputusan.

Berdasarkan hasil wawancara dengan Bapak Fanani, untuk indikator yang digunakan dalam pengukuran kinerja mesin yang ada divisi perencanaan teknik & administrasi yang merupakan bagian yang menangani tingkat ketersediaan alat atau mesin yang digunakan untuk kegiatan operasional PT. BJTI akan dijelaskan pada Tabel 2.1 di bawah ini.

Tabel 2.1 Tabel Indikator *Dashboard*

No.	Indikator	Unit Pengukuran
1.	Ketersediaan (*availability*)	Persen (%)
2.	Kehandalan (*reliability*)	Persen (%)
3.	Penggunaan (*utilization*)	Persen (%)
4.	Penggantian *Oli Engine*	HRM
5.	Penggantian *Oli Gear Box*	HRM
6.	Penggantian *Oli Hydraulic*	HRM
7.	Penggantian *Oli Transmission*	HRM
8.	*Corrective Maintenance* (Perawatan Rutin)	Jam

Sumber: Divisi Perencanaan Teknik & Administrasi PT. BJTI.

2.8 *Dashboard*

Dashboard adalah sebuah tampilan visual dari informasi terpenting yang dibutuhkan untuk mencapai satu atau lebih tujuan, digabungkan dan diatur pada sebuah layar, menjadi informasi yang dibutuhkan dan dapat dilihat secara sekilas. *Dashboard* itu sebuah tampilan pada satu monitor komputer penuh yang berisi informasi yang bersifat kritis, agar kita dapat mengetahui hal-hal yang perlu diketahui. Biasanya kombinasi teks dan grafik, tetapi lebih ditekankan pada grafik (Few, 2006).

Menurut Darly (2005), *dashboard* didefinisikan sebagai alat untuk memonitor organisasi dari hari ke hari. Informasi ditampilkan dalam sebuah antar muka tunggal, sehingga pengambil keputusan dapat mengakses *Key Performance Indicator* (KPI). Sebelum ditampilkan dalam sebuah antar muka tunggal, informasi diproses terlebih dahulu. Terdapat 5 tipe dalam proses informasi menurut Bocij, Chaffey, Greasly, & Hickie (2006), yaitu:

1. Klasifikasi : Ini melibatkan menempatkan data ke dalam kategori, misalnya, mengkategorikan beban pada saat baik atau biaya variabel tetap.

2. Menata ulang/menyortir : Ini melibatkan pengorganisasian data sehingga item dikelompokkan bersama-sama atau ditempatkan dalam urutan tertentu. Data karyawan, misalnya, mungkin akan diurutkan menurut nama belakang atau nomor gaji.
3. Agregat : Ini melibatkan meringkas data, misalnya, dengan menghitung rata-rata, total atau subtotal.
4. Melakukan perhitungan : Sebuah contoh mungkin menghitung gaji kotor karyawan dengan mengalikan jumlah jam kerja dengan tarif per jam dari gaji.
5. Seleksi : Ini melibatkan memilih atau membuang item data atas dasar seperangkat kriteria seleksi. Sebuah organisasi penjualan, misalnya, mungkin membuat daftar calon pelanggan dengan memilih orang-orang dengan pendapatan di atas tingkat tertentu.

2.8.1 Tujuan Penggunaan *Dashboard*

Tujuan penggunaan *dashboard* menurut Eckerson (A) (2006), yaitu:

1. Mengkomunikasikan Strategi

   Mengkomunikasikan strategi dan tujuan yang dibuat oleh eksekutif kepada semua pihak yang berkepentingan sesuai dengan peran dan levelnya dalam organisasi.
2. Memonitor dan Menyesuaikan Pelaksanaan Strategi

   Memonitor pelaksanaan dari rencana dan strategi yang telah dibuat. Memungkinkan eksekutif untuk mengidentifikasi permasalahan kritis dan membuat stategi untuk mengatasinya.

3. Menyampaikan Wawasan dan Informasi ke Semua Pihak

   Menyajikan informasi menggunakan grafik, simbol, bagan dan warna yang memudahkan pengguna dalam memahami dan mempersepsi informasi secara benar.

2.8.2 Jenis *Dashboard*

Dashboard bisa dikelompokkan seseuai dengan level manajemen yang didukungnya menurut Hariyanti (2008), yaitu:

1. *Strategic Dashboard*
   a. Mendukung manajemen level strategis.
   b. Informasi untuk membuat keputusan bisnis, memprediksi peluang, dan memberikan arahan pencapaian tujuan strategis.
   c. Fokus pada pengukuran kinerja *high-level* dan pencapaian tujuan strategis organisasi.
   d. Mengadopsi konsep *Balance Score Card*.
   e. Informasi yang disajikan tidak terlalu detail.
   f. Konten informasi tidak terlalu banyak dan disajikan secara ringkas.
   g. Informasi disajikan dengan mekanisme yang sederhana, melalui tampilan yang *unidirectional*.
   h. Tidak di desain untuk berinteraksi dalam melakukan analisis yang lebih detail.
   i. Tidak memerlukan data *real time*.
2. *Tactical Dashboard*
   a. Mendukung manajemen *tactical*.

b. Memberikan informasi yang diperlukan oleh analisis untuk mengetahui penyebab suatu kejadian.
c. Fokus pada analisis untuk menemukan penyebab dari suatu kondisi atau kejadian tertentu.
d. Dengan fungsi *drill down* dan navigasi yang baik.
e. Memiliki konten informasi yang lebih banyak (Analisis perbandingan, pola/tren, evaluasi kerja).
f. Menggunakan media penyajian yang "cerdas" yang memungkinkan pengguna melakukan analisis terhadap data yang kompleks.
g. Didesain untuk berinteraksi dengan data.
h. Tidak memerlukan data *real time*.

3. *Operational Dashboard*
   a. Mendukung manajemen level operasional.
   b. Memberikan informasi tentang aktivitas yang sedang terjadi, beserta perubahannya secara *real time* untuk memberikan kewaspadaan terhadap hal-hal yang perlu direspon secara cepat.
   c. Fokus pada monitoring aktifitas dan kejadian yang berubah secara konstan.
   d. Informasi disajikan spesifik, tingkat kedetailan yang cukup dalam.
   e. Media penyajian yang sederhana.
   f. *Alert* disajikan dengan cara yang mudah dipahami dan mampu menarik perhatian pengguna.
   g. Bersifat dinamis, sehingga memerlukan data *real time*.

h. Didesain untuk berinteraksi dengan data, untuk mendapatkan informasi yang lebih detail, maupun informasi pada level lebih atas (*Higher Level Data*).

2.8.3 Karakteristik *Dashboard*

Karakteristik *dashboard* menurut Eckerson (A) (2006), yaitu:

1. Model pemrosesan berdasarkan kejadian yaitu menangkap kejadian setiap saat dari beberapa sistem yang mencakup dan mempengaruhi proses bisnis.
2. Aturan bisnis yang kuat yaitu mengijinkan penggunanya membuat peringatan, target, ambang untuk menilai kinerja individu.
3. Dashboard bisnis yang user friendly yaitu mempebarui nilai sebagai aliran kejadian melalui sistem dan menempatkan nilai tersebut dalam hubungan dengan menghubungkan ke pencapaian bisnis.
4. Sebuah sistem aliran kerja yang bergabung dan bekerjasama yang mengijinkan penggunanya untuk memulai proses secara formal dan informal, yang dengan proses itu pengguna dapat berkolaborasi mendiskusikan hasilnya.

Beberapa karakteristik *dashboard* menurut Hariyanti (2008), yaitu:

1. Sinergi

   Ergonomis dan memiliki tampilan visual yang mudah dipahami oleh pengguna. *Dashboard* mensinergikan informasi dari berbagai aspek yang berbeda dalam satu layar.

2. Monitor

   Menampilkan KPI yang diperlukan dalam pembuatan keputusan dalam domain tertentu, sesuai dengan tujuan pembangunan *dashboard* tersebut.

3. Akurat

   Informasi yang disajikan harus akurat, dengan tujuan untuk mendapatkan kepercayaan dari penggunanya.

4. Responsif

   Merespon *threshold* yang telah didefinisikan, dengan memberikan *alert* (seperti bunyi alarm, *blinker*, email) untuk mendapatkan perhatian pengguna terhadap hal-hal yang kritis.

5. *Timely*

   Menampilkan informasi terkini yang diperlukan untuk pengambilan keputusan.

6. Interaktif

   Pengguna dapat melakukan *drilldown* dan mendapatkan informasi lebih detail, analisis sebab akibat dan sebagainya.

7. *More Data History*

   Melihat tren sejarah KPI contohnya perbandingan jumlah mahasiswa baru saat ini dengan beberapa tahun yang lalu, untuk mengetahui apakah kondisi sekarang lebih baik atau tidak.

8. *Personalized*

   Penyajian informasi spesifik untuk setiap jenis pengguna sesuai domain tanggung jawab, hak akses dan batasan akses data.

9. *Analitical*

   Fasilitas untuk melakukan analisis seperti sebab akibat.

10. *Collaborative*

    Fasilitas pertukaran catatan laporan antar pengguna mengenai hasil pengamatan *dashboard*-nya masing-masing yaitu sarana komunikasi dalam melakukan fungsi manajemen dan kontrol.

11. *Trackability*

    Memungkinkan setiap pengguna untuk mengkustomisasi nilai yang akan dilacaknya.

2.8.4 Komponen *Dashboard*

Untuk memahami perbedaan diantara ketiga jenis *dashboard* kinerja, perlu untuk mengetahui masing-masing komponen aplikasi yang digunakan. Meskipun tidak ada aturan keras dan cepat tentang penggunaan komponen, lihat Gambar 2.1 yang memberikan beberapa pedoman umum (Eckerson (B), 2006).

PERFORMANCE DASHBOARD COMPONENTS

	Operational Dashboard	Tactical Dashboard	Strategic Dashboard
Monitoring	Dashboard	BI Portal	Scorecard
Analysis	Statistical models Decision engines	OLAP analysis Interactive reporting Advanced visualization Scenario modeling	Time-series analysis Standard reports
Management	Alerts Agents	Workflow Usage monitoring Auditing	Meetings Annotations Strategy maps

The monitoring, analysis, and management components that most commonly comprise the three types of performance dashboards.

Gambar 2.1 Komponen *Dashboard* Kinerja. (Eckerson (B), 2006)

1. Komponen *Dashboard* Operasional

   Dashboard operasional menggunakan antarmuka *dashboard* untuk memantau proses operasional. *Dashboard* memberikan peringatan yang memberitahukan pengguna tentang kondisi pengecualian dalam proses yang

sedang mereka pantau sehingga mereka dapat bertindak cepat untuk memperbaiki masalah atau memanfaatkan peluang.

2. Komponen *Dashboard* Taktis

   Dashboard taktis sering menampilkan hasil dalam *business intelligence* (BI) portal yang berisi grafik dan tabel serta dokumen lainnya pengguna perlu untuk memantau proyek atau proses yang mereka kelola. Portal ini dibangun ke sebagian besar alat BI dan biasanya mengintegrasikan dengan portal komersial yang banyak digunakan perusahaan untuk menjalankan intranet perusahaan mereka.

3. Komponen *Dashboard* Strategis

   Dashboard Strategis menggunakan antarmuka *scorecard* untuk melacak kinerja terhadap tujuan strategis. Meskipun mereka mirip dengan antarmuka *dashboard*, *scorecard* umumnya melacak kemajuan kelompok secara bulanan daripada secara tepat waktu. *Scorecard* umumnya menampilkan lebih metrik seluruh spektrum yang lebih luas dari organisasi daripada *dashboard*, terutama di *scorecard* perusahaan. Informasi kinerja dalam antarmuka *scorecard* biasanya lebih diringkas dari dalam antarmuka *dashboard*.

2.9 *System Development Life Cycle*

Siklus Hidup Pengembangan Sistem, nama lain dari *System Development Life Cycle* (SDLC) ini merupakan suatu proses pengembangan atau perubahan suatu sistem perangkat lunak. Pengembangan atau perubahan tersebut dilakukan dengan menggunakan model-model dan metodologi yang digunakan oleh banyak orang, yang telah mengembangkan sistem-sistem perangkat lunak sebelumnya. Hal tersebut tentu berdasarkan *best practice* atau cara-cara yang telah teruji

dengan baik oleh banyak orang yang menggunakannya. SDLC memiliki beberapa model dalam penerapan tahapan prosesnya. Beberapa model SDLC tersebut antara lain yaitu Model *Waterfall, Spiral, Rapid Application Development, Agile dan Prototype*. Masing-masing model memiliki kelemahan dan kelebihan, sehingga hal yang terpenting adalah mengenali tipe pelanggan dan memilih menggunakan model SDLC yang sesuai dengan karakter pelanggan dan sesuai dengan karakter pengembang perangkat lunak (Kendall & Kendall, 2008).

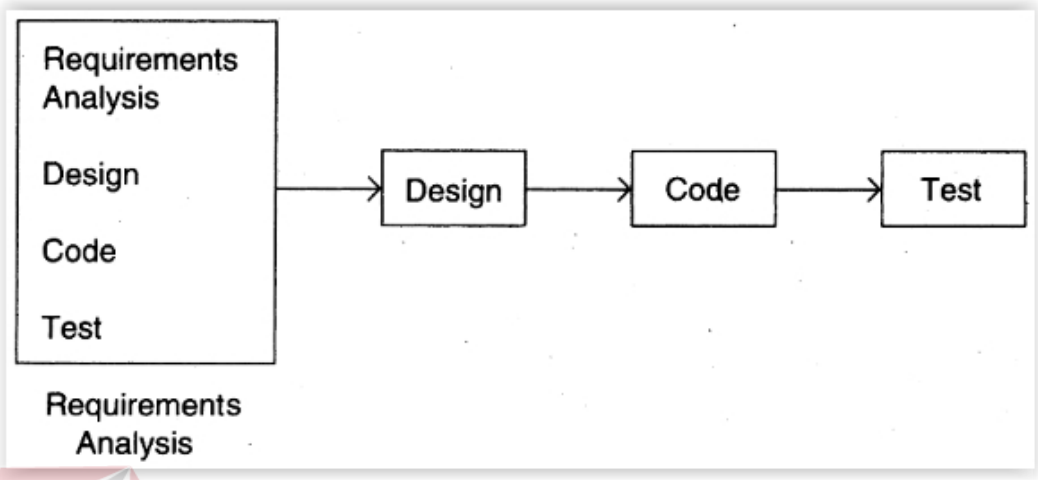
2.9.1 Metode *Prototyping*

Prototyping adalah salah satu pendekatan dalam rekayasa perangkat lunak yang secara langsung mendemonstrasikan bagaimana sebuah perangkat lunak atau komponen-komponen perangkat lunak akan bekerja dalam lingkungannya sebelum tahapan konstruksi aktual dilakukan (Howard, 1997).

Beberapa model *prototype* adalah sebagai berikut:

1. *Reusable prototype* : *Prototype* yang akan ditransformasikan menjadi produk *final*.
2. *Throwaway prototype* : *Prototype* yang akan dibuang begitu selesai menjalankan maksudnya.
3. *Input*/*output prototype* : *Prototype* yang terbatas pada antar muka pengguna (*user interface*).
4. *Processing prototype* : *Prototype* yang meliputi perawatan *file* dasar dan proses-proses transaksi.
5. *System prototype* : *Prototype* yang berupa model lengkap dari perangkat lunak.

Menurut Jalote (2008), metode *prototyping* digambarkan dalam model sebagai berikut:



Gambar 2.2 Metode *Prototyping.* (Jalote, 2008)

Metode *prototyping* biasanya dimulai ketika versi awal dari dokumen persyaratan spesifikasi telah dikembangkan. Pada tahap ini, ada pemahaman yang wajar dari sistem dan kebutuhan serta yang masih belum jelas atau mungkin akan berubah. Setelah prototipe dikembangkan, pengguna akhir dan klien diberi kesempatan untuk menggunakan dan mengeksplorasi prototipe. Berdasarkan pengalaman mereka, mereka memberikan umpan balik kepada para pengembang mengenai prototipe: apa yang benar, apa yang perlu diubah, apa yang hilang, apa yang tidak dibutuhkan, dll. Berdasarkan umpan balik, prototipe dimodifikasi untuk menggabungkan beberapa perubahan yang disarankan agar dapat dilakukan dengan mudah, dan kemudian pengguna dan klien lagi diizinkan untuk menggunakan sistem. Siklus ini berulang sampai, menurut penilaian para pengembang prototipe dan analis, keuntungan dari mengubah sistem dan mendapatkan umpan balik sebanding dengan biaya dan waktu yang dibutuhkan dalam membuat perubahan dan mendapatkan umpan balik. Berdasarkan umpan

balik, persyaratan awal yang dimodifikasi menghasilkan spesifikasi kebutuhan akhir, yang kemudian digunakan untuk mengembangkan sistem kualitas produksi.

2.10 *Unified Modeling Language* (UML)

Unified Modeling Language atau biasa disingkat UML merupakan suatu standar yang digunakan untuk memodelkan sistem yang akan digunakan untuk melakukan rancang bangun suatu aplikasi. Penggunaan UML diperlukan untuk memberikan gambaran ddalam bentuk diagram tentang bagaimana bentuk dan dokumentasi dari sistem yang akan dibangun (Stephens & Rosenberg, 2007). Komponen-komponen dari UML terdiri dari beberapa diagram, antara lain sebagai berikut:

1. *Domain Model*

   Domain model merupakan teknik pengidentifikasian *object-object* pada kata benda yang terdapat pada daftar *requirement* yang diklasifikasikan pada area (*domain*) permasalahan yang sama untuk dijadikan *candidate class* pada *class diagram*. Analisis yang harus pertama kali dilakukan adalah analisis *domain model* daripada *analisis use case diagram* bersifat *abstract* dan ambigu untuk dianalisa dan pada akhirnya *use case diagram* harus dibuat secara konkrit pada konteks model *object*. Bentuk domain model merupakan fondasi dari bagian statis dari sebuah model sistem, sedangkan *use case* merupakan bagian dinamis dari sebuah model sistem. *Domain mode*l mendeskripsikan struktur arsitektur dari sebuah sistem yang statis, sedangkan *use case diagram* mendeskripsikan fungsi atau tingkah laku dari sebuah sistem. Titik awal untuk memulai *domain model* adalah dari *requirement* atau kebutuhan sistem

dari *user/client*. Pengumpulan daftar kebutuhan *requirement* yang baik akan menghasilkan *domain model* yang baik pula.

2. *Use Case Diagram*

   Diagram ini menggambarkan cara pengguna interaksi dengan sistem dan bagaimana sistem akan merespon. Diagram ini menunjukkan interaksi antara *user* dengan sistem atau antara *external parties* dengan sistem. Diagram *use case* meperlihatkan *user* dari sebuah sistem dan proses-proses yang dapat mereka lakukan untuk berinteraksi dengan sistem tersebut.

3. *Robustness Diagram*

   Robustness diagram merupakan representasi bergambar dari perilaku (*behavior*) yang dideskripsikan oleh *use case*. Diagram *robustness* menunjukkan perilaku dari kelas kelas dan perilaku dari perangkat lunak. Pada diagram ini tidak digambarkan kelas mana yang bertanggung jawab terhadap perilaku tertentu. Walaupun demikian, diagram *robustness* dapat dibaca seperti diagram aktivitas (*activity diagram*) atau sebagai sebuah *flowchart* dalam arti suatu objek "berbicara" dengan objek lainnya. Simbol-simbol yang terlibat dalam diagram *robustness* *Boundary object* adalah antarmuka antara sistem dengan segala sesuatu di luar sistem. Contohnya adalah layar atau halaman *web*. *Entity object* merupakan kelas-kelas dari domain model. *Controller* adalah penghubung antara *boundary* dan *entity object*.

4. *Sequence Diagram*

   Sequence diagram digunakan untuk meng-*explore* desain dari sistem secara detail dengan menggunakan basis *scenario-by-scenario*. Pada tahapan ini

dilakukan *detailing* pada desain sistem. *Use case* harus sudah selesai dengan benar, detail dan jelas untuk digunakan sebagai acuan untuk membuat desain yang lebih detail.

5. *Class Diagaram*

   Class diagram merupakan penggambaran dari seluruh *method class* pada pemrograman berbasis obyek yang digunakan untuk membangun sebuah aplikasi. *Class diagram* menggambarkan keadaan (atribut/properti) suatu sistem, sekaligus menawarkan layanan untuk memanipulasi keadaan tersebut (metoda/fungsi).

2.11 *Database*

Menurut Marlinda (2004), *database* adalah suatu susunan/kumpulan data operasional lengkap dari suatu organisasi/perusahaan yang diorganisir/dikelola dan disimpan secara terintegrasi dengan menggunakan metode tertentu menggunakan komputer sehingga mampu menyediakan informasi optimal yang diperlukan pemakainya. Penyusunan satu *database* digunakan untuk mengatasi masalah-masalah pada penyusunan data yaitu redundansi dan inkonsistensi data, kesulitan pengaksesan data, isolasi data untuk standarisasi, banyak pemakai (*multiple user*), masalah keamanan (*security*), masalah kesatuan (*integration*), dan masalah kebebasan data (*data independence*).

2.12 *Oracle*

Oracle merupakan *software database* yang banyak dipakai di perusahaan-perusahaan besar di seluruh dunia saat ini. *Software* ini juga banyak diminati oleh para konsultan pembuat aplikasi yang berkaitan dengan *database*. Sistem keamanannya yang handal membuat para profesional yang berkecimpung

dalam dunia *database* lebih memilih *Oracle* sebagai perangkat untuk menunjang kegiatan bisnis mereka. Bukan hanya masalah *security*-nya saja yang handal, *Oracle* juga merupakan *software database* yang bisa menampung serta mengeola data dengan kapasitas yang sangat besar serta dapat mengaksesnya dengan sangat cepat pula. Sintak SQL-nya yang hampir seluruhnya telah memenuhi standar ANSI-92 lebih memudahkan para programer dalam membangun aplikasi baik dari sisi *back-end* maupun dari sisi *front-end*. Demikian pula bagi seorang administrator yang berkecimpung dalam menangani administrasi *database* serta bertanggung jawab terhadap keamanan *database* akan merasa diuntungkan serta dimudahkan dengan *software Oracle* yang lebih *establish* ini (Heryanto & Raharjo, 2006).

2.13 *Hypertext Preprocessor*

Menurut Firdaus (2007), PHP merupakan singkatan dari *Hypertext Preprocessor*, adalah sebuah bahasa *scripting* berbasis *server side scripting* yang terpasang pada HTML dan berada di *server* dieksekusi di *server* dan digunakan untuk membuat halaman *web* yang dinamis. Sebagian besar sintaksnya mirip dengan bahasa C atau *java*, ditambah dengan beberapa fungsi PHP yang spesifik. Tujuan utama bahasa ini adalah untuk memungkinkan perancang *web* menulis halaman *web* dinamis dengan cepat.

Halaman *web* biasanya disusun dari kode-kode HTML yang disimpan dalam sebuah *file* berekstensi .html. *File* HTML ini dikirimkan oleh *server* (atau *file*) ke *browser,* kemudian *browser* menerjemahkan kode-kode tersebut sehingga menghasilkan suatu tampilan yang indah. Lain halnya dengan program PHP, program ini harus diterjemahkan oleh *web server* sehingga menghasilkan kode

html yang dikirim ke *browser* agar dapat ditampilkan. Program ini dapat berdiri sendiri ataupun disisipkan di antara kode-kode HTML sehingga dapat langsung ditampilkan bersama dengan kode-kode HTML tersebut. Program php dapat ditambahkan dengan mengapit program tersebut di antara tanda <? dan ?>. Tanda-tanda tersebut biasanya digunakan untuk memisahkan kode php dari kode HTML. *File* HTML yang telah dibubuhi program php harus diganti ekstensi-nya menjadi .php atau .php3.

2.14 *Java Script*

Menurut Hakim (2010), *java script* merupakan bahasa *scripting* yang dapat bekerja di sebagian besar *web browser*. *Java script* dapat disisipkan di dalam *web* menggunakan *tag script*. *Java script* dapat digunakan untuk banyak tujuan, misalnya untuk membuat efek *roolover* baik gambar maupun *text*, dan untuk membuat AJAX *Java script* adalah bahasa yang digunakan untuk AJAX. Kode *java script* juga dapat diletakkan di *file* tersendiri yang berekstensi *java script* (.js). *Script* tersebut akan dieksekusi ketika dipanggil berdasarkan *trigger* pada *event* tertentu.

2.15 *Highcharts*

Highcharts adalah *library* pembuatan *chart* yang ditulis dalam *JavaScript* murni, menawarkan cara mudah untuk menambahkan grafik interaktif ke situs *web* atau aplikasi *web*. *Highcharts* saat ini mendukung *line, spline, area, area spline, column, bar, pie, scatter, angular gauges, area range, area spline range, column range, bubble, box plot, error bars, funnel, waterfall* dan *polar chart types* (Highcharts, 2016).