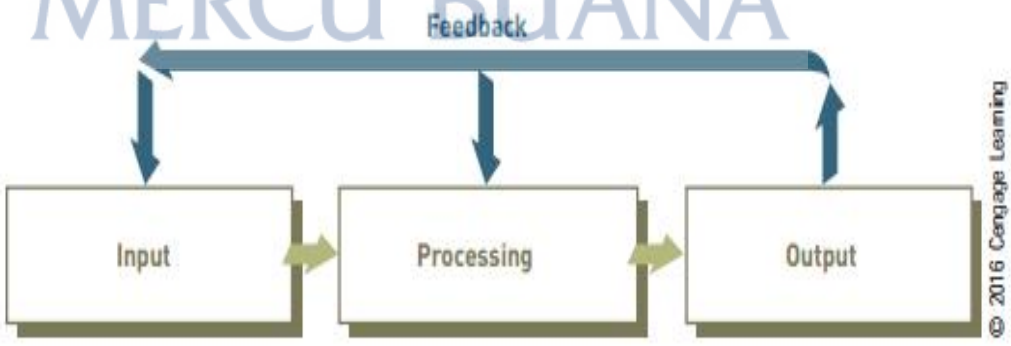
BAB II
LANDASAN TEORI

2.1 Sistem Informasi

Sistem informasi adalah sekumpulan elemen/komponen yang saling berhubungan dimana didalamnya terdapat proses mengumpulkan (*input*), memanipulasi (proses), menyimpan (*store*), dan menyebarkan (*output*) data dan informasi, serta mampu menyediakan reaksi korektif (*feedback mechanism*) terhadap sistem itu sendiri (Stair & Reynolds 2016).

Sistem informasi memiliki beberapa komponen pembentuk yaitu :

a. *input*, melibatkan aktifitas yang berhubungan dengan pengumpulan data mentah.
b. *Processing*, berkaitan dengan proses konfersi atau transformasi data menjadi output yang berguna.
c. *Output*, berkaitan dengan hasil pemrosesan data mentah menjadi informasi yang berguna, biasanya dalam bentuk dokumen – dokumen dan laporan – laporan.
d. *Feedback*, informasi dari sistem yang nantinya akan digunakan sebagai dasar untuk melakukan perubahan input atau aktifitas proses di sistem.
e. *Forecasting*, memprediksi kejadian – kejadian di masa yang akan datang untuk menghindari permasalahan – permasalahan yang mungkin akan terjadi.



Gambar 2.1 Diagram Sistem Informasi

2.2 Rekayasa Perangkat Lunak

Perangkat lunak atau *software* berbeda dengan program komputer.*Software* tidak hanya mencakup program, melainkan juga semua dokumentasi dan konfigurasi data yang berhubungan, yang dibutuhkan untuk membuat program dapat beroperasi dengan benar. Perangkat lunak merupakan instruksi-instruksi yang jika dijalankan akan menyediakan fungsi yang diperlukan, struktur data yang memungkinkan program untuk memanipulasi informasi, dokumen yang menyatakan operasi dan kegunaan program (Roger S. Pressman, 2010).

Sistem perangkat lunak terdiri dari :

a. Sejumlah program yang terpisah

b. File-file konfigurasi

c. Dokumentasi sistem

d. Dokumentasi *user*

Dari beberapa definisi tersebut dapat dilihat bahwa pemeliharaan untuk rekayasa perangkat lunak sangatlah penting.Definisi ini menganggap bahwa pendekatan yang digunakan harus sistematis, disiplin, dan terukur bukan hanya didasarkan pada ilmu komputer dan matematika.Selain itu juga mencakup *study* dalam rekayasa perangkat lunak, yaitu studi dan mencari metode, teknik, alat, dan lain sebagainya.

Penelitian terhadap rekayasa perangkat lunak sangat penting karena produksi perangkat lunas sulit, jauh lebih sulit daripada perkiraan banyak orang. Menurut (Roger S. Pressman, 2010) rekayasa perangkat lunak dikelompokkan dalam beberapa lapisan atau *layers*, antara lain:

Gambar 2.2 *Software Engineering Layers* (Roger S. Pressman, 2010)

Landasan paling dasar yang mendukung rekayasa perangkat lunak adalah fokus kepada kualitas atau *quality focus*. Yang menjadi pondasi suatu rekayasa perangkat lunak adalah lapisan proses. Proses merupakan perekat yang memegang lapisan teknologi bersama dan memungkinkan pengembangan rasional dan tepat waktu bagi perangkat lunak komputer. Proses mendefiniskan kerangka kerja untuk satu set area proses kunci

atau *key process areas (KPAs)* yang harus dibentuk untuk pengiriman efektif teknologi rekayasa perangkat lunak. Area proses kunci membentuk dasar bagi kontrol manajemen proyek piranti lunak dan menetapkan konteks dimana metode teknis yang diterapkan, pekerjaan produk (model, dokumen, data, laporan, formulir, dan lain sebaginya) diproduksi, kualitas dijamin, tonggak ditetapkan, dan perubahan dikelola dengan baik. Metode rekayasa perangkat lunak menyediakan teknis bagaimana membangun perangkat lunak.Metode tersebut mencakup tugas-tugas analis seperti kebutuhan, desain, konstruksi program, pengujian, dan metode rekayasa *support.*

Perangkat lunak bergantung pada sekumpulan prinsip dasar yang mengatur setiap area teknologi dan termasuk aktifitas *modelling* dan teknik deskriptif lainnya. Alat rekayasa perangkat lunak memberikan dukungan otomatis atau semi-otomatis untuk proses dan alat *methods*. Ketika alat yang digunakan terintegrasi maka informasi yang dibuat oleh satu alat, dapat digunakan oleh yang lain. Kasus menggabungkan *software, hardware*, dan rekayasa perangkat lunak *database* (repositori berisi informasi penting tentang analisis, desain, konstruksi program, dan pengujian) untuk menciptakan sebuah lingkungan piranti lunak rekayasa *analog* dengan *CAD (Computer Aided Design)* atau *CAE (Computer Aided Engineering)* untuk *hardware*. (Roger S. Pressman, 2010).

2.3 Metode PIECES

Menurut James Wetherbe 2012, Metode yang digunakan untuk membangun sistem ini adalah metode analisis PIECES (Performance, Information, Economy, Control, Efficiency dan Services). Adapun yang dilakukan pada analisis PIECES ini ada beberapa tahapan, yaitu Analisis Performance, Analisis Information, Analisis Economy, Analisis Control, Analisis Eficiency, dan Analisis Services.

a. *Analisis Performance*

   Masalah kinerja terjadi terjadi ketika tugas-tugas bisnis yang dijalankan tidak mencapai sasaran.kinerja diukur dengan jumlah produksi dan waktu tanggap. Jumlah adalah jumlah pekerjaan yang bisa diselesaikan selama jangka waktu tertentu.

b. *Analisis Information*

   Evaluasi terhadap kemampuan sistem dalam menghasilkan informasi yang bermanfaat perlu dilakukan untuk menyikapi peluang dan menangani masalah yang muncul. Analisis informasi ini memeriksa output sistem data yang tersimpan dalam sebuah sistem permasalahan yang dihadapi meliputi

- Data berlebihan dimana data yang selama ini ditangkap atau disimpan dibanyak tempat
- Kekakuan data dimana data ditangkap atau disimpan tetapi tidak diorganisasikan sedemikian rupa sehingga laporan dan pengujian tidak dapat atau sulit dilakukan.

c. *Analisis Economy*

Persoalan ekonomis dan peluang berkaitan dengan biaya dan keuntungan.

d. *Analisis Control*

Control dipasang untuk meningkatkan kinerja sistem, menjamin keamanan data dan informasi.

e. *Analisis Eficiency*

Efisiensi menyangkut bagaimana menghasilkan output sebanyak-banyaknya dengan input yang sekecil mungkin

f. *Analisis Services*

Berikut adalah beberapa kriteria penilaian dimana kualitas suatu sistem dikatakan buruk:

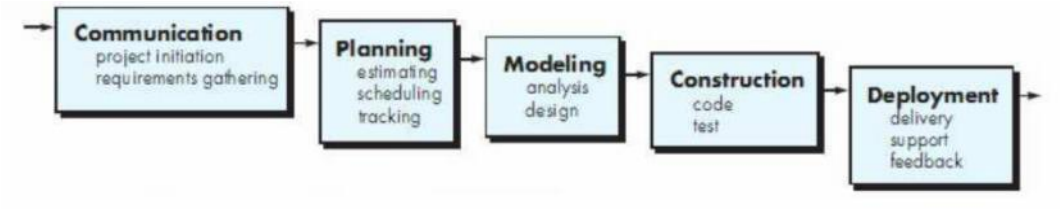
- Sistem menghasilkan produk yang tidak akurat
- Sistem menghasilkan produk yang tidak konsisten
- Sistem menghasilkan produk yang tidak dapat dipercaya
- Sistem tidak mudah dipelajari
- Sistem tidak mudah digunakan
- Sistem canggung untuk digunakan

2.4 Metode Waterfall

Menurut (Pressman, 2015) model *waterfall* adalah model klasik yang bersifat sistematis, berurutan dalam membangun *software*. Nama model ini sebenarnya adalah *"Linear Sequential Model"*. Model ini sering disebut juga dengan *"classic life cycle"* atau metode waterfall.

Model ini termasuk ke dalam model *generic* pada rekayasa perangkat lunak dan pertama kali diperkenalkan oleh Winston Royce sekitar tahun 1970 sehingga sering dianggap kuno, tetapi merupakan model yang paling banyak dipakai dalam *Software Engineering* (SE). Model ini melakukan pendekatan secara sistematis dan berurutan.Disebut dengan *waterfall* karena tahap demi tahap yang dilalui harus menunggu selesainya tahap sebelumnya dan berjalan berurutan.

Fase-fase dalam *Waterfall Model* menurut referensi Pressman :



Gambar 2.3 *Waterfall* Pressman (Pressman, 2015)

1. *Communication*

   Sebelum memulai pekerjaan yang bersifat teknis, sangat diperlukan adanya komunikasi dengan *customer* demi memahami dan mencapai tujuan yang ingin dicapai.Hasil dari komunikasi tersebut adalah inisialisasi proyek, seperti menganalisis permasalahan yang dihadapi dan mengumpulkan data-data yang diperlukan, serta membantu mendefinisikan fitur dan fungsi *software*.Pengumpulan data-data tambahan bisa juga diambil dari jurnal, artikel, dan internet.

2. *Planning*

   Tahap berikutnya adalah tahapan perencanaan yang menjelaskan tentang estimasi tugas-tugas teknis yang akan dilakukan, resiko-resiko yang dapat terjadi, sumber daya yang diperlukan dalam membuat sistem, produk kerja yang ingin dihasilkan, penjadwalan kerja yang akan dilaksanakan, dan *tracking* proses pengerjaan system.

3. *Modeling*

   Tahapan ini adalah tahap perancangan dan permodelan arsitektur sistem yang berfokus pada perancangan struktur data, arsitektur *software*, tampilan *interface*, dan algoritma program. Tujuannya untuk lebih memahami gambaran besar dari apa yang akan dikerjakan.

4. *Construction*

   Tahapan *Construction* ini merupakan proses penerjemahan bentuk desain menjadi kode atau bentuk/bahasa yang dapat dibaca oleh mesin. Setelah pengkodean selesai, dilakukan pengujian terhadap sistem dan juga kode yang sudah dibuat.Tujuannya untuk menemukan kesalahan yang mungkin terjadi untuk nantinya diperbaiki.

5. *Deployment*

   Tahapan *Deployment* merupakan tahapan implementasi *software* ke *customer*, pemeliharaan *software* secara berkala, perbaikan *software*, evaluasi *software*, dan pengembangan *software* berdasarkan umpan balik yang diberikan agar sistem dapat tetap berjalan dan berkembang sesuai dengan fungsinya.

   (Pressman, 2015)

Keuntungan menggunakan metode *waterfall* adalah prosesnya lebih terstruktur, hal ini membuat kualitas *software* baik dan tetap terjaga. Dari sisi *user* juga lebih menguntungkan, karena dapat merencanakan dan menyiapkan kebutuhan data dan proses yang diperlukan sejak awal. Penjadwalan juga menjadi lebih menentu, karena jadwal setiap proses dapat ditentukan secara pasti. Sehingga dapat dilihat jelas target penyelesaian pengembangan program. Dengan adanya urutan yang pasti, dapat dilihat pula perkembangan untuk setiap tahap secara pasti. Dari sisi lain, model ini merupakan jenis model yang bersifat dokumen lengkap sehingga proses pemeliharaan dapat dilakukan dengan mudah.

Kelemahan menggunakan metode *waterfall* adalah bersifat kaku, sehingga sulit melakukan perubahan di tengah proses. Jika terdapat kekurangan proses/prosedur dari tahap sebelumnya, maka tahapan pengembangan harus dilakukan mulai dari awal lagi. Hal ini akan memakan waktu yang lebih lama. Karena jika proses sebelumnya belum selesai sampai akhir, maka proses selanjutnya juga tidak dapat berjalan. Oleh karena itu, jika terdapat kekurangan dalam permintaan *user* maka proses pengembangan harus dimulai kembali dari awal. Karena itu, dapat dikatakan proses pengembangan *software* dengan metode *waterfall* bersifat lambat.

2.5 Pengertian Manajemen SDM

Manajemen Sumber Daya Manusia merupakan kegiatan yang mengatur tentang cara pengadaan tenaga kerja, melakukan pengembangan, memberikan kompensasi, integrasi, pemeliharaan, dan pemisahan tenaga kerja melalui proses-proses menajemen dalam rangka mencapai tujuan organisasi (Amirullah, 2015).

2.6 UML (*Unified Modelling Language*)

Menurut Alan dennis (2012), “UML is a standart set of diagramming techniques that provide a graphical representation rich enough to model any systems development project. From analysis through implementation”.

Jadi, UML merupakan suatu set standar teknik yang menyediakan representasi grafis yang cukup kaya untuk model disetiap proyek pengembangan sistem, dari analisis melalui implementasi diagram.

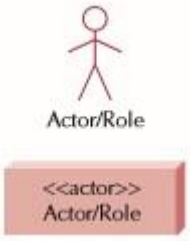
Tabel 2.1 Jenis - jenis Diagram Unifed Modeling Language(UML) (Alan Dennis, 2012)

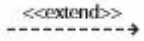
No	Diagram	Kegunaan
1	Activity	Behavior procedural dan parallel
2	Class	Class, fitur, dan hubungan – hubungan
3	Communication	Interaksi antar objek, penekanan pada jalur
4	Component	Struktur dan koneksi komponen
5	Composite Structure	Dekomposisi runtime sebuah class
6	Deployment	Pemindahan artifak ke node
7	Interaction Overview	Campuran sequence dan activity diagram
8	Object	Contoh konfigurasi dari contoh – contoh
9	Package	Struktur hirarki compile-time
10	Sequence	Interaksi antar objek, penekana pada sequence
11	State Machine	Bagaimana even mengubah objek selama aktif
12	Timing	Interaksi antar objek; penekanan pada timing
13	Use Case	Bagaimana user berinteraksi dengan sistem

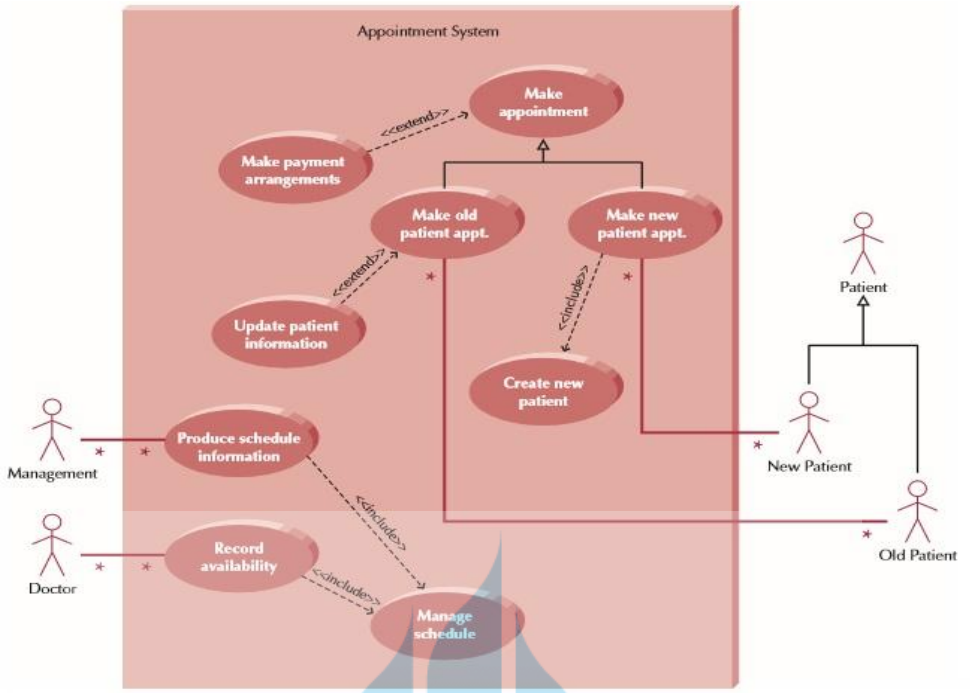
2.6.1 Use Case Diagram

Use case diagram ini bersifat statis dan merupakan penggambaran yang sederhana dari pengoperasian fungsi sistem yang dilakukan oleh user yang berbeda kemudian saling berinteraksi. Berikut notasi yang ada dalam use case diagram.

Tabel 2.2 Notasi Use case Diagram (Alan Dennis, 2012)

Nama	Notas	Deskripsi
Actor		• Adalah orang atau sistem yang berasal manfaat dari dan eksternal untuk subjek • Digambarkan baik sebagai tongkat (default) atau jika aktor non-manusia yang terlibat, sebagai persegi panjang dengan << aktor >> di dalamnya (alternatif) • Diberi label dengan perannya • Dapat dikaitkan dengan aktor-aktor

		lain menggunakan spesialisasi / asosiasi superclass, dilambangkan dengan anak panah dengan mata panah berongga
Use Case		• Merupakan bagian utama dari fungsi sistem • Dapat memperpanjang kasus penggunaan lain • Dapat mencakup kasus penggunaan lain • Ditempatkan di dalam batas sistem • Diberi label dengan frase kata kerja-kata benda deskriptif
Subject Boundary		• Termasuk nama subjek dalam atau di atas • Merupakan lingkup subjek, misalnya, sistem atau proses bisnis individu
Association Relationship		• Link aktor dengan use case (s) dengan yang berinteraksi
Include Relationship		• Merupakan masuknya fungsi satu kasus digunakan dalam lain • panah diambil dari kasus penggunaan dasar untuk kasus penggunaan termasuk
Extend Relationship		• Merupakan perpanjangan kasus penggunaan untuk memasukkan perilaku opsional • panah diambil dari kasus penggunaan ekstensi untuk kasus penggunaan dasar
Generalization Relationship		• Merupakan kasus penggunaan khusus ke yang lebih umum • panah diambil dari kasus penggunaan khusus untuk kasus penggunaan dasar



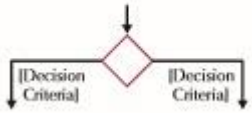
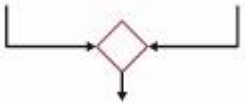
Gambar 2.4 Contoh Diagram Use Case (Alan Dennis, 2012)

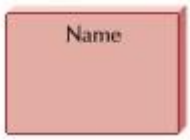
2.6.2 Activity Diagram

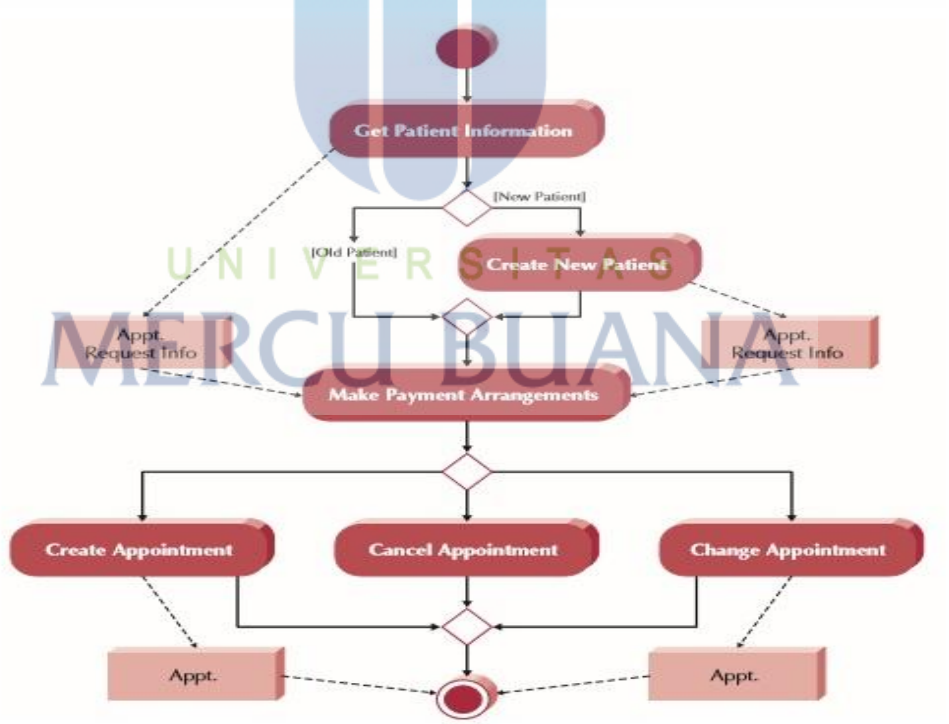
Menurut Alan Denis, 2012 Diagram aktivitas digunakan untuk model perilaku dalam proses bisnis independen dari objects. Dalam banyak hal, diagram aktivitas dapat dilihat sebagai data canggih flow diagram yang digunakan dalam hubungannya dengan analisis terstruktur. Namun, tidak seperti data yang diagram, diagram Activity termasuk notasi yang membahas pemodelan paralel, kegiatan bersamaan dan proses pengambilan keputusan yang kompleks. Dengan demikian, diagram aktivity dapat digunakan untuk model segala sesuatu dari sebuah karya bisnis tingkat tinggi yang melibatkan banyak kasus penggunaan yang berbeda.

Tabel 2.3 Notasi Dalam Activity Diagram (Alan Dennis, 2012)

Nama	Notasi	Deskripsi
Action Node		• Adalah sederhana, sepotong non-decomposable perilaku • Diberi label dengan namanya

Activity		• Digunakan untuk mewakili serangkaian tindakan • Diberi label dengan namanya
Object Node		• Digunakan untuk mewakili suatu objek yang terhubung ke satu set dari Obyek Arus • Diberi label dengan nama kelasnya
Control Flow		• Menunjukkan urutan eksekusi
Object Flow		• Menunjukkan aliran dari sebuah objek dari satu kegiatan (atau tindakan) untuk kegiatan lain (atau tindakan)
Initial Node		• Menggambarkan awal dari serangkaian tindakan atau kegiatan
Final Activity Node		• Digunakan untuk menghentikan semua arus kontrol dan arus objek dalam suatu kegiatan (atau tindakan)
Final Flow Node		• Digunakan untuk menghentikan aliran kontrol tertentu atau aliran objek
Decision Node		• Digunakan untuk mewakili kondisi tes untuk memastikan bahwa aliran kontrol atau objek mengalir hanya turun satu jalur • Diberi label dengan kriteria keputusan untuk terus menyusuri jalan tertentu
Merge Node		• Digunakan untuk membawa kembali jalur keputusan bersama yang berbeda yang dibuat menggunakan keputusan-node

Fork Node		• Digunakan untuk membagi perilaku menjadi satu set arus paralel atau bersamaan kegiatan (atau tindakan)
Join Node		• Digunakan untuk membawa kembali bersama satu set arus paralel atau bersamaan kegiatan (atau tindakan)
Swimlane		• Digunakan untuk memecah diagram aktivitas dalam baris dan kolom untuk menetapkan kegiatan individu (atau tindakan) kepada individu atau objek yang bertanggung jawab untuk melaksanakan kegiatan (atau tindakan)

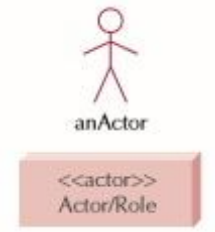
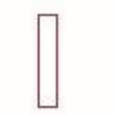


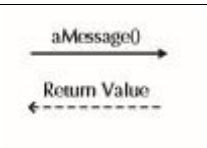
Gambar 2.5 Contoh Activity Diagram (Alan Dennis, 2012)

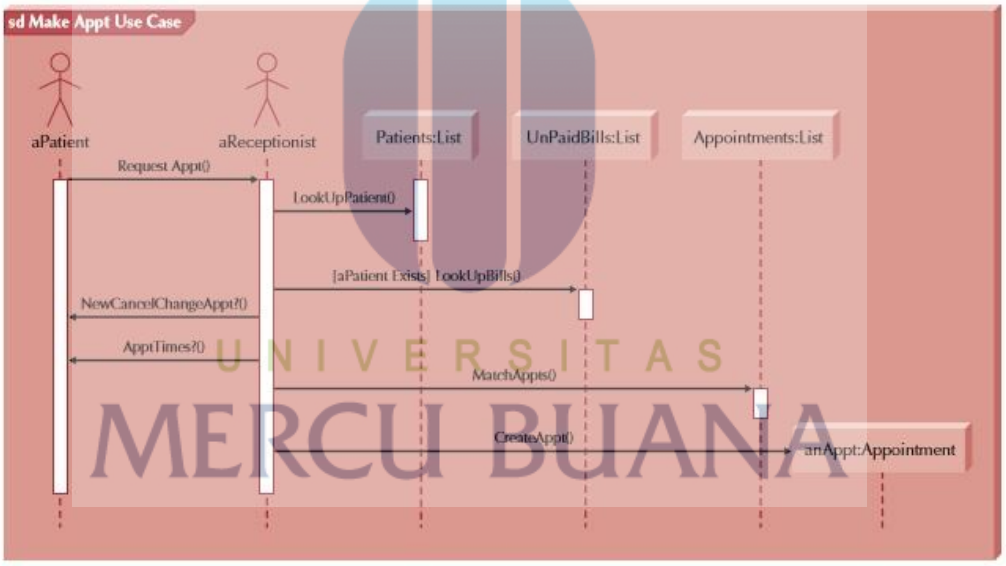
2.6.3 Sequence Diagram

Sequence diagram merupakan suatu diagram interaksi yang menggambarkan objek- objek berpartisipasi dalam bagian interaksi dan pesan yang di tukar dalam urutan waktu.

Tabel 2.4 Notasi Dalam Sequence Diagram (Alan Dennis, 2012)

Nama	Notasi	Deskripsi
Actor		• Adalah orang atau sistem yang berasal manfaat dari dan eksternal untuk sistem • Berpartisipasi secara berurutan dengan mengirim dan / atau menerima pesan • Ditempatkan di bagian atas diagram • Digambarkan baik sebagai tongkat (default) atau jika aktor non-manusia yang terlibat, sebagai persegi panjang dengan << aktor >> di dalamnya (alternatif)
Object		• Berpartisipasi secara berurutan dengan mengirim dan / atau menerima pesan • Ditempatkan di bagian atas diagram
Lifeline		• menandakan kehidupan obyek selama berurutan • Berisi "X" pada titik di mana kelas berinteraksi tidak lagi
Execution Occurrence		• Adalah persegi panjang yang panjang sempit ditempatkan di atas garis hidup • Menunjukkan ketika sebuah objek mengirim atau menerima pesan

Message		• Menyampaikan informasi dari satu objek satu sama lain • Panggilan operasi dilabeli dengan pesan yang dikirim dan panah padat, sedangkan kembali diberi label dengan nilai yang akan dikembalikan dan
Object Destruction	X	• X ditempatkan pada akhir garis hidup suatu objek untuk menunjukkan bahwa itu akan keluar dari eksistensi
Frame		• Menunjukkan konteks diagram urutan

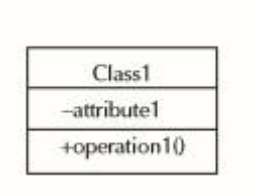
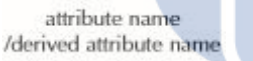
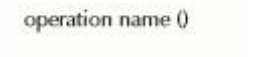


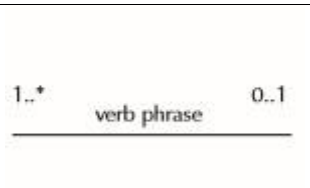
Gambar 2.6 Contoh Sequence Diagram (Alan Dennis, 2012)

2.6.4 Class Diagram

Class Diagram menunjukan kelas dan hubungan antara kelas yang tetap konstan dalam sistem, dari waktu ke waktu.

Tabel 2.5 Notasi Dalam Class Diagram (Alan Denis, 2012)

Nama	Notasi	Deskripsi
Class		• Merupakan jenis orang, tempat, atau hal tentang yang sistem akan perlu untuk menangkap dan menyimpan informasi • Telah nama diketik dalam huruf tebal dan berpusat di kompartemen atas nya • Memiliki daftar atribut di kompartemen tengah nya • Memiliki daftar operasi di bawahnya compartmentHas daftar atribut di kompartemen tengah nya • Tidak secara eksplisit menunjukkan operasi yang tersedia untuk semua kelas
Attribute		• Merupakan sifat yang menggambarkan negara dari sebuah objek • Dapat diturunkan dari atribut lainnya, ditunjukkan dengan menempatkan garis miring sebelum nama atribut
Operation		• Merupakan tindakan atau fungsi yang kelas dapat melakukan • Dapat diklasifikasikan sebagai konstruktor, query, atau operasi update • Termasuk tanda kurung yang mungkin berisi parameter atau informasi yang dibutuhkan untuk melakukan operasi

Association		• Merupakan hubungan antara beberapa kelas, atau kelas dan itu sendiri • Diberi label menggunakan frase kata kerja atau nama peran, mana yang lebih baik merupakan hubungan • Bisa ada di antara satu atau lebih kelasMengandung simbol multiplisitas, yang mewakili minimum dan maksimum kali contoh kelas dapat dikaitkan dengan contoh kelas terkait

2.7 Database

Menurut (Jubilee, 2014) *database* adalah suatu aplikasi yang menyimpan sekumpulan data.Setiap *database* mempunyai API tertentu untuk membuat, mengakses, mengatur, mencari dan menyalin data yang ada di dalamnya.

Untuk menampung dan mengatur data yang segitu banyaknya, anda dapat menggunakan *Relation Database Management Systems (DBMS).*Hal ini disebut *relation* database karena semua data disimpan dalam tabel-tabel yang berbeda dan dihubungkan berdasarkan relasinya dengan menggunakan primary key dan foreign key.

2.7.1 Database MySQL

Menurut (Jubilee, 2014), MySQL adalah DBMS yang cepat dan mudah digunakan, serta sudah banyak digunakan untuk berbagai kebutuhan.MySQL dikembangkan oleh MySQL AB Swedia.

Berikut ini hal-hal yang menyebabkan MySQL menjadi begitu populer :

- Berlisensi *open-source*, sehingga anda dapat menggunakanya secara gratis.
- Merupakan program yang powerful dan menyediakan fitur yang lengkap.
- Menggunakan bentuk standar bahasa data SQL.
- Dapat bekerja dengan banyak sistem operasi dan dengan bahasa-bahasa pemrograman seperti PHP, PERL, C, C++, JAVA dan lain-lain.
- Bekerja dengan cepat dan baik, bahakan dengan data set yang banyak.
- Sangat mudah digunakan dengan PHP untuk pengembangan aplikasi web

- Mendukung banyak database, sampai 50 juta baris atau lebih dalam suatu tabel.
- Dapat dikostumisasi sesuai dengan keinginan anda

2.8 PHP

PHP Hypertext Prepoccesor (PHP) merupakan bahasa interprenter yang mirip dengan bahasa C dan Perl yang memiliki kesederhanaan dalam perintah.PHP dapat digunakan bersamaan dengan WML sehingga pembangunan situs *Web* site dapat dilakukan dengan cepat dan mudah.PHP dapat digunakan untuk meng-update *database*, menciptakan *database*.

Menurut (I Komang Setia Buana, 2014) PHP *hypertext preprocessor* atau sering disebut PHP merupakan bahasa pemrograman berbasis *server-side* yang dapat melakukan parsing *script* web sehingga dari sisi client menghasilkan sesuatu tampilan yang menarik. PHPmerupakan pengembangan dari FI*FormInterface*yang dibuatoleh RasmusLerdoffpadatahun 1995.PHPsangatberperan besardalammembuatwebsitekerendandinamiskarenadapatmelakukanbanyakhal, seperti;membaca*file*,menulis*file*,menampilkangambar,animasiatau*movie*,danyang palingpokokadalahdapatmelakukankoneksiterhadap*database*.

2.9 Software Engineering

Menurut (Pressman, 2015)*Software Engineering*atau Rekayasa Perangkat Lunak adalah pembuatan dan penggunaan prinsip-prinsip keahlian teknik untuk mendapatkan perangkat lunak yang ekonomis yang handal dan bekerja secara efisien pada mesin yang sesungguhnya. Rekayasa Perangkat Lunak mendirikan suatu pondasi untuk suatu proses perangkat lunak yang lengkap dengan mengidentifikasi sejumlah aktivitas kerangka kerja yang berlaku untuk semua proyek perangkat lunak, terlepas dari hal ukuran dan kompleksitas.

Selain itu Menurut IEEE 610.12 Rekayasa perangkat lunak adalah sebuah studi dan aplikasi dari sebuah pendekatan kuantifiabel, disiplin, dan sistematis kepada pengembangan, operasi dan pemeliharaan perangkat lunak yang kesemuanya itu merupakan aplikasi rekayasa yang berkaitan dengan perangkat lunak.

Dari beberapa pengertian diatas dapat disimpulkan bahwa *Software engineering* adalah disiplin teknik yang berkaitan dengan semua aspek produksi perangkat lunak dari tahap awal spesifikasi sistem sampai pemeliharaan sistem setelah itu perangkat lunak sudah mulai digunakan. Dalam definisi ini, ada dua frase kunci:

1. Disiplin teknik membuat sesuatu bekerja. Mereka menerapkan teori, metode, dan alat-alat di mana keduanya tepat. Namun, mereka menggunakannya selektif dan selalu mencoba untuk menemukan solusi untuk masalah bahkan ketika tidak ada teori dan metode yang dapat aplikatif. Insinyur juga mengakui bahwa mereka harus bekerja untuk kendala organisasi dan keuangan sehingga mereka mencari solusi dalam kendala.

2. Semua aspek rekayasa perangkat lunak *production software* tidak hanya berkaitan dengan proses teknis dari pengembangan perangkat lunak. Ini juga mencakup kegiatan seperti manajemen proyek perangkat lunak dan pengembangan alat, metode, dan teori untuk mendukung produksi perangkat lunak. Teknik adalah tentang mendapatkan hasil kualitas yang diperlukan dalam jadwal dan anggaran.

2.10 Pengujian Perangkat Lunak

Menurut (Rizky, 2011) testing adalah sebuah proses yang diejawantahkan sebagai siklus hidup dan merupakan bagian dari proses rekayasaperangkat lunak secara terintegrasi demi memastikan kualitas dari perangkatlunak serta memenuhi kebutuhan teknis yang telah disepakati dari awal.

Pengujian adalah proses eksekusi suatu program untuk menentukan kesalahan (Simarmata, 2010). Dari beberapa definisi di atas maka dapat disimpulkan pengujian adalah suatu proses eksekusi dalam siklus hidup pengembangan perangkat lunak secara terintegrasi untuk memvalidasi dan memverifikasi guna menentukan kesalahan dan memenuhi harapan yang telah disepakati di awal.

2.10.1 Metode White Box

Dalam pengujian *white-box*, anda bisa tahu bagaimana sebuah metode bekerja.Anda kemudian menggunakan pengetahuan lebih untuk merancang sebuah tes untuk mencoba membuat sebuah metode rusak.Pengujian *white-box* memiliki keuntungan bahwa anda tahu bagaimana sebuah metode bekerja, jadi anda bisa mencoba untuk memilih kasus pengujian yang sulit.

Sayangnya ia memiliki kelemahan yang anda tahu bagaimana sebuah metode bekerja, jadi anda melewati beberapa kasus pengujian yang anda anggap bekerja. Misalnya, anda mungkin mengetahui bahwa sebuah metode akan dibingungkan dengan string kosong. Tapi anda tahu bahwa ketika anda menulis kode, jadi anda yang mengatasinya.Masalahnya, anda mungkin tidak menanganinya dengan benar. Jika anda

menangani semuanya dengan benar, maka tidak akan ada *bug* dan anda tidak akan perlu melakukan pengujian secara keseluruhan. Menggunakan pengujian *white-box* untuk membuat tes akan merepotkan, tapi jangan melewati tes yang anda "tahu" metodenya anda dapat tangani (RodStephend, 2015:188)

2.10.2 Metode Black Box

Dalam pengujian *black-box*, anda berpura-pura bahwa metode ini adalah *black-box* yang anda tidak dapat melihat kedalam. Anda tahu apa yang seharusnya dilakukan, tapi anda tidak tahu bagaimana ini bekerja. Anda kemudian mencoba segala macam masukan untuk melihat apa yang akan dilakukannya. Anda dapat memulai pengujian *black-box* dengan mengirim sekelompok masukan acak.Ingat bahwa anda perlu untuk melakukan tes ini hanya beberapa kali, tidak setiap kali program berjalan, jadi anda bisa menguji banyak nilai acak.

Contohnya, anda bisa mencoba untuk menebak nilai yang mungkin mengacaukannya.Biasanya, itu melibatkan nilai spesial seperti 0 untuk angka dan kosong untuk string.Mereka juga boleh termasuk nilai terbesar dan terkecil yang memungkinkan. Untuk string yang mungkin maksudnya adalah string yang semuanya kosong atau semua karakter. Misalnya, biasanya quicksort adalah salah satu algoritma pengurutan tercepat, tapi ia memberikan kinerja yang buruk jika item yang diurutkan memiliki nilai yang sama. Jika metode mengambil sejumlah variabel masukan, pastikan ia dapat mengatasi masukan 0 dan sejumlah angka yang sangat besar untuk masukan. Jika dibutuhkan sebuah array atau daftar parameter, lihat apakah ia lakukan jika array atau daftar parameter kosong atau hilang. Akhirnya, lihat pada batasan nilai. Jika metode menerapkan parameter float antara 0.0 dan 1.0, maka pastikan ia dapat menangani kedua nilai itu (Rod Stephend,2015:187).