BAB 2

TINJAUAN PUSTAKA

Bab ini berisikan tentang teori – teori yang digunakan sebagai landasan dari pembuatan aplikasi *e*-klinik. Berikut adalah teori – teori yang digunakan:

2.1 Landasan Teori

2.1.1. *Information Technology*

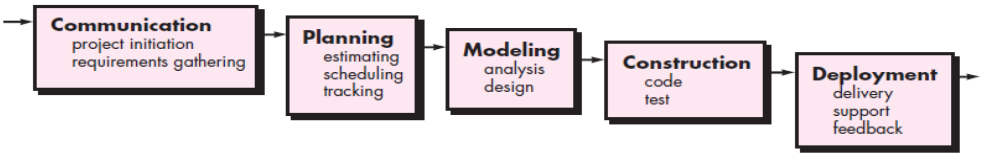
2.1.1.1. Pengertian *Information Technology*

Menurut ND *Century Code* (*Chapter* 54.59.01), *Information Technology* berarti penggunaan *hardware*, *software*, jasa, dan infrastruktur lain yang mendukung untuk mengatur dan mengirim informasi melalui suara, data, dan *video.*

2.1.1.2. SDLC

System Development Life Cycle (SDLC) digunakan untuk membantu para pengembang aplikasi agar hasil dari aplikasinya sesuai dengan yang diharapkan.

Menurut Pressman (2010: 39), model *Waterfall* menyarankan suatu pendekatan secara bertahap dan tersusun dalam pengembangan software, yang dimulai dengan spesifikasi kebutuhan *user* dan dilanjutkan dengan perencanaan, modeling, pengembangan, pemasaran produk, hingga pelayanan berkelanjutan dari produk akhirnya.



Gambar 2.1 *Waterfall Model*

Dari gambar di atas, dapat disimpulkanbahwa terdapat 5 tahap*waterfall model*, yaitu:

1) *Communication*

Pada tahap ini, pengembang aplikasi memulai dengan mendefinisikan tujuan proyek dan mencari kebutuhan *user*.

2) *Planning*

Tahap ini dilakukan dengan merencanakan jadwal pengembangan proyek serta memperhitungkan biaya dan waktu yang dibutuhkan.

3) *Modeling*

Tahap ini difokuskan pada perancangan desain sistem yang akan digunakan. Desain ini disesuaikan dengan kebutuhan *user.*

4) *Construction*

Pada tahap ini, aplikasi mulai dikembangkan sesuai dengan rancangan yang sudah dibuat, dan dilakukan uji coba aplikasi.

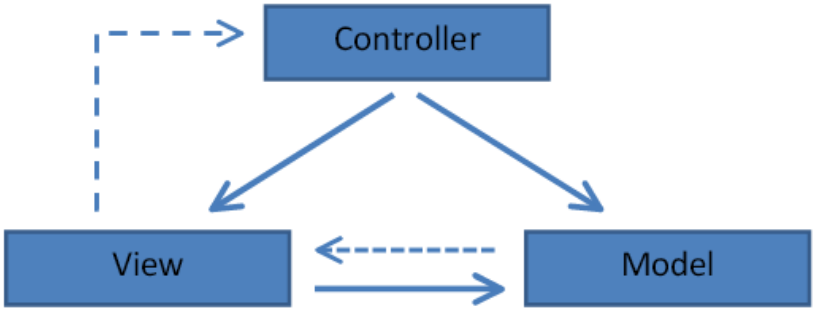
5) *Deployment*

Tahap ini difokuskan pada distribusi produk akhir (aplikasi) kepada *user*dan menyediakan pelayanan jangka panjang terhadap produk tersebut.

2.1.1.3. MVC

MVC merupakan model arsitektural yang digunakan dalam rekayasa piranti lunak (*SoftwareEngineering*). (Mihai Curteanu, 2010, p1). MVC biasa digunakan untuk mengembangkan aplikasi berbasis GUI dalam pemrogaman berbasis objek.

MVC berguna untuk merangkum data serta prosesnya dan mengisolasi data tersebut dari proses manipulasi dan presentasi sehingga dapat direpresentasikan sebagai *userinterface*. (Softwatul, Muhammad, 2010, p1).



Gambar 2.2 Struktur sistem MVC (Mihai Curteanu, 2010, p1).

Sesuai dengan namanya, MVC menggunakan tiga tipe objek untuk mengembangkan sistem mereka, yaitu *Model*, *View*, dan *Controller*.

1.*Model*

Model merupakan data akses yang berhubungan langsung dengan database untuk memanipulasi data dan biasanya digunakan untuk mengatur informasi dan memberikan notifikasi ketika ada perubahan informasi (Softwatul, Muhammad, 2010, p1).

2.*View*

View merupakan bagian yang menangani *presentationlogic* yang berfungsi untuk menerima dan mempresentasikan data kepada *user* sebagai data yang nantinya diproses oleh *Controller* (Mihai Curteanu, 2010, p1).

3.*Controller*

Controller merupakan yang mengatur hubungan antara bagian *Model* dan *View* dengan memproses *input* dari *user*. *Controller* ini berfungsi untuk menerima *request* dan data dari *user* kemudian menentukan apa yang akan diproses oleh aplikasi tersebut (Stuart, Timothy, 2005, p1).

2.1.2. *Internet*

2.1.2.1. Pengertian *Internet*

Menurut *Federal Networking Council* (*InternetMonthlyReports*, Oktober 1995), *internet* dapat didefinisikan sebagai sistem informasi global yang;

1. Dihubungkan melalui alamat yang unik secara global berdasarkan pada *Internet Protocol* (IP) atau segala sesuatu yang berhubungan dengan IP.
2. Dapat mendukung komunikasi menggunakan *TransmissionControlProtocol / InternetProtocol* (TCP/IP), termasuk protokol – protokol atau benda lain yang mendukung hubungan IP
3. Menyediakan, menggunakan, atau memungkinkan untuk digunakan, baik secara publik ataupun pribadi, jasa servis tingkat atas pada *layer* komunikasi dan infrastruktur yang terkait.

2.1.2.2. Sejarah *Internet*

Internet pertama kali dibentuk sebagai jaringan komputer di Departemen Pertahanan Amerika Serikat pada tahun 1969, melalui proyek ARPA yang disebut ARPANET (*Advanced Research Project Agency Network*).

Pada saat itu, pemerintah Amerika Serikat memanfaatkan *hardware* dan *software* komputer berbasis UNIX, yang dapat melakukan komunikasi dalam jarak yang tidak terhingga melalui saluran telepon.

Proyek ini merancang bentuk jaringan, kehandalan, serta seberapa besar informasi dapat dipindahkan.Kini, semua standar yang mereka tentukan menjadi dasar pembangunan protokol yang sekarang dikenal sebagai TCP/IP (*Transmission Control Protocol / Internet Protocol*).

2.1.2.3. WWW

World Wide Web adalah nama yang diberikan bagi sekumpulan komputer yang menyajikan informasi dalam format *hypertext* di *internet*.

Hypertext ini sendiri pertama kali ditemukan oleh Dr. Tim Berners-Lee, di *European Center for Nuclear Research* (CERN), dalam bentuk *hyperlink text document* untuk membantu proses pengiriman informasi antar peneliti di CERN.

Hyperlink text document ini kemudian disebut dengan nama*HyperText Markup Languange (HTML).*

2.1.2.4. HTTP

Hyper Text Transfer Protocol (HTTP) adalah protokol komunikasi yang digunakan untuk pengiriman / penyampaian informasi melalui *World Wide Web* (WWW).HTTP ini ditulis oleh Dr. Tim Berners-Lee.

Fungsi dari HTTP adalah sebagai media jalan untuk mem-*publish* serta mengambil halaman *Hyper Text Markup Language* (HTML) melalui alamat URL.

2.1.2.5. URL

Uniform Resource Locator merupakan referensi (berupa sebuah alamat digital) kepada suatu *resource* yang terletak di *internet*.

2.1.2.6. CSS

Menurut Duckett (2010), CSS (*Cascading Style Sheet*) digunakan untuk mengkontrol tampilan dari suatu halaman. CSS sendiri adalah sebuah *sheet* berupa *tag* (umumnya berupa *external sheet*) yang memungkinkan elemen HTML dimanipulasi sedemikian rupa sehingga bisa menampilkan desain yang menarik.

2.1.3. Basis Data

2.1.3.1. Pengertian Data

Data adalah sesuatu yang belum mempunyai arti bagi penerimanya dan memerlukan adanya suatu pengolahan agar

dapat diubah menjadi informasi. Terdapat 2 jenis data menurut Rabianski (2003), yaitu:

1. Data Primer

   Data primer adalah data yang secara langsung diambil dari objek penelitian oleh peneliti perorangan maupun organisasi.

   Contoh: Wawancara langsung terhadap calon *user* aplikasi untuk mengetahui kebutuhan aplikasi.

2. Data Sekunder

   Data sekunder adalah data yang didapat tidak secara langsung dari objek penelitian. Peneliti mendapatkan data yang sudah jadi yang dikumpulkan oleh pihak lain dengan berbagai cara atau metode.

   Contoh: Peneliti yang menggunakan data hasil riset dari organisasi lain seperti lembaga riset.

2.1.3.2. Pengertian *Database*

Menurut Connoly T. dan Begg C. (2005:15), *database* adalah sekumpulan data yang saling berhubungan dan dapat mendeskripsikan suatu data untuk dirancang menjadi suatu informasi yang dibutuhkan suatu organisasi.

Berbeda dengan penyimpanan data atau *file* biasa, *database* merupakan satu kumpulan data yang dapat digunakan secara bersamaan oleh berbagai *user* dan diatur sedemikian rupa agar memiliki tingkat duplikasi data yang minimum.

2.1.3.3. DBMS

Menurut Connoly T. dan Begg C. (2005:16), *Database Management System* (DBMS) adalah suatu *software* yang merancang, membuat, mengatur dan mengendalikan seluruh proses di dalam *database*.

DBMS memungkinkan *user* yang menggunakannya untuk mendefinisikan suatu *database* melalui proses *DataDefinitionLanguage* (DDL) dan dapat melakukan *insert*,

update, *delete* dan mendapatkan kembali data dari *database* dengan melakukan proses *DataManipulationLanguage* (DML).

Keuntungan dari penggunaan DBMS:

1. Tingkat data *redundant* yang rendah
2. Meningkatkan data yang integritas
3. Pembagian penggunaan data
4. Meningkatkan keamanan penyimpanan data
5. Meningkatkan produktifitas
6. Meningkatkan konkurensi / mengurangi kemungkinan terjadi permasalahan akibat interferensi penggunaan data
7. Mengaplikasikan penggunaan *backup* dan *recovery data*

Kerugian dari penggunaan DBMS:

1. Kompleksitas dan sulit dimengerti
2. Ukuran data cenderung besar
3. Harga untuk pengembangan DBMS cenderung mahal (tergantung ukuran data dan kompleksitasnya)
4. Diperlukan *hardware* tambahan (penyimpanan data)
5. Performa yang berkurang (pelatihan penggunaan, sistem yang mencakup seluruh proses)

2.1.3.4. SQL

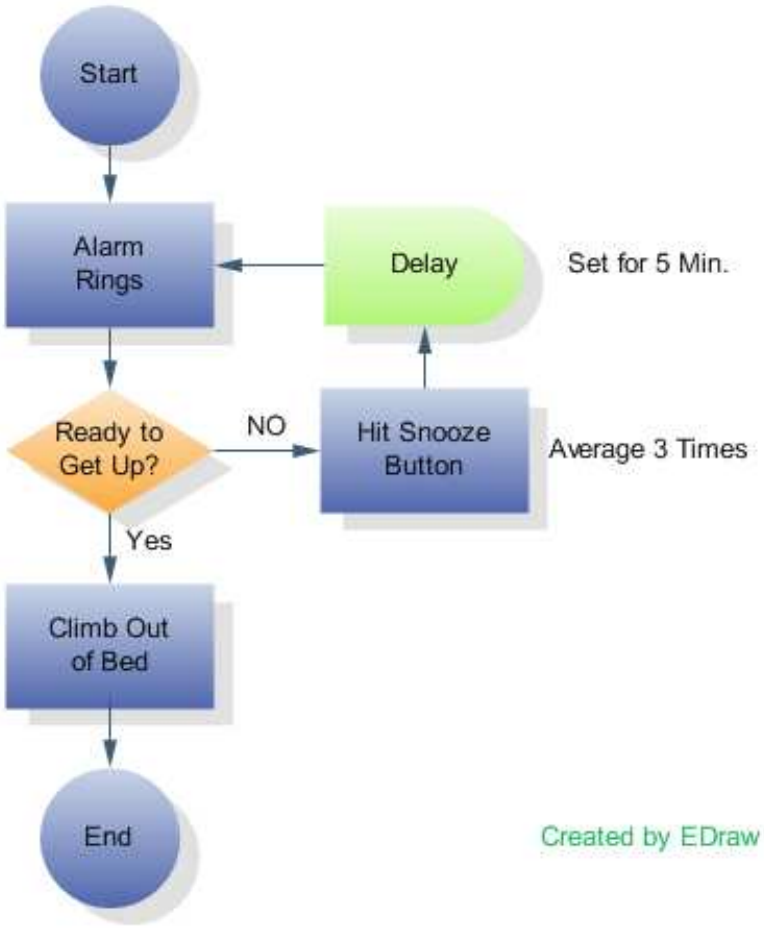
Structured Query Language (SQL) digunakan untuk berkomunikasi dengan *database*.Menurut *American National Standards Institute* (ANSI), SQL merupakan bahasa standar untuk DBMS.

Pernyataan SQL digunakan untuk memanipulasi data yang ada dalam *database*, baik itu untuk menambah data baru, melihat ataupun mengubah data yang sudah ada, serta menghapus data.

2.1.4. Perancangan

2.1.4.1. Flowchart

Sebuah *flowchart* menggambarkan aktivitas sekuensial dari sebuah proses dan menunjukkan aktor dari aktivitas tersebut. Pembuatan *flowchart* dimaksudkan untuk mempermudah desainer agar memiliki konsep yang jelas sebagai standar desain aplikasi nantinya.



Gambar 2.3 Contoh *Flowchart*

Sumber:http://3.bp.blogspot.com/_n4FdVvKSECc/SwteiUXUVuI/AAAAAAAAAB c/SNkAAAF4qNw/s1600/flow%2Bchart.png

2.1.4.2. Use Case Diagram

Menurut Whitten (2007: 246), sebuah *use case* menggambarkan fungsi sistem berdasarkan pandangan *user* dengan menggunakan terminologi yang mudah dimengerti.

Sebuah *use case diagram* merupakan gambaran lengkap hubungan antara *use case* tersebut, *actor* dan juga sistem. Dengan kata lain, *use case diagram* menggambarkan *user,* dan apa yang dilakukan oleh *user* tersebut terhadap sistem.

Terdapat 3 hal penting dalam *use case,* yaitu:

1. *Actor*

   Actor menggambarkan *user* di luar sistem yang berinteraksi dengan sistem dengan melakukan *use case*. *Selain* bisa berupa benda mati (bukan manusia), a*ctor* juga dapat memiliki satu atau lebih *use case* dalam sistem.

2. *Use case*

   Use case menggambarkan aksi atau tugas yang dilakukan oleh *actor* untuk menyelesaikan suatu masalah tertentu dengan cara berinteraksi dengan sistem.

3. *Relationship*

   Dalam *use case diagram*, terdapat relasi yang menggambarkan hubungan antara *actor* dan *use case*. Terdapat beberapa jenis *relationship,* yaitu:

   a. *Association*

      Sebuah relasi yang menggambarkan hubungan antara *usecase* dengan *actor*.

   b. *Extends*

      Use case yang menggunakan relasi ini melalui langkah – langkah yang diambil dari *use case* lain. Tujuan *use case* ini adalah untuk mempermudah dan menyederhanakan *use case* yang kompleks sehingga mudah dimengerti.

   c. *Uses* (*Include*)

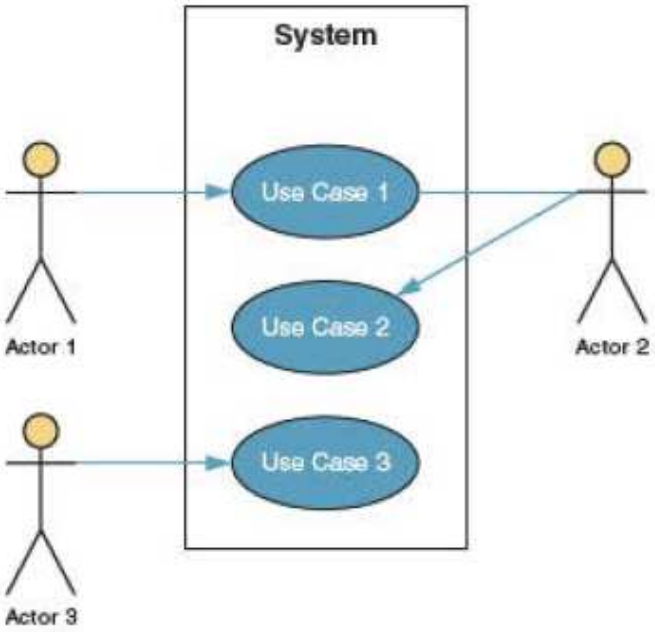
Use case yang menggunakan relasi ini merupakan *abstract use case*, yaitu *use case* dengan langkah yang mirip, yang dibuat menjadi sederhana.

d. *DependsOn*

Relasi ini menggambarkan bahwa suatu *use case* tidak dapat dijalankan apabila ada *use case* dengan relasi *depends on* belum dijalankan / belum selesai.

e. *Inheritance*

Relasi ini menggambarkan relasi antar *actor,* ketika suatu *abstract actor* meng-*inherituse case* dari*role actor*.



Gambar 2.4 Contoh *Use Case Diagram* sederhana

Sumber: Whitten J.L. (2007: 246)

2.1.4.3. Activity Diagram

Activity Diagram menurut Whitten (2007: 390) adalah suatu diagram yang digunakan untuk menggambarkan alur

proses bisnis, langkah – langkah dari *use case*, atau metode dari suatu objek.

Terdapat 8 hal penting dalam *activity diagram* yang bisa digunakan sesuai kebutuhan, yaitu:

1. *InitialNode*

   Lingkaran penuh yang menggambarkan awal suatu *activity diagram*

2. *Actions*

   Kotak yang digunakan untuk menunjukkan langkah – langkah dalam diagram tersebut

3. *Flow*

   Panah dalam diagram yang menunjukkan urutan dalam melalui suatu *activity.*Panah ini umumnya hanya diberi keterangan ketika keluar dari suatu *decision.*

4. *Decision*

   Bentuk *diamond* dalam diagram yang menunjukkan suatu pilihan. Umumnya memiliki satu panah masuk dan banyak panah keluar.

5. *Merge*

   Bentuk *diamond* yang sama seperti *decision* dalam diagram yang juga menunjukkan suatu pilihan. Namun memiliki banyak panah masuk dan satu panah keluar.

6. *Fork*

   Bentuk garis hitamdalam diagram yang menunjukkan suatu *activity* yang berjalan secara pararel.Umumnya memiliki satu panah masuk dan banyak panah keluar.

7. *Join*

   Bentuk garis hitam yang sama seperti *fork* dalam diagram yang juga menunjukkan

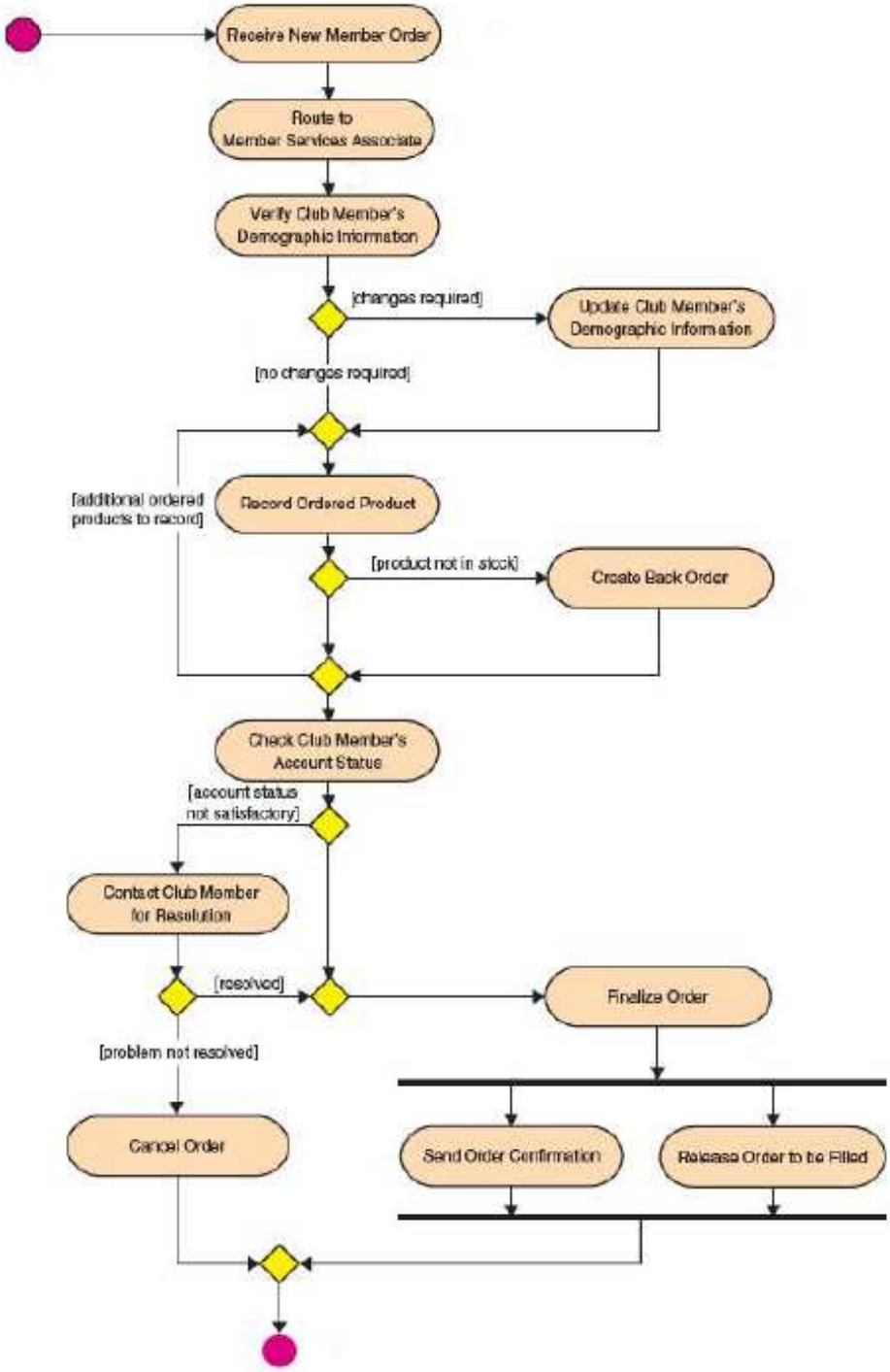
penggabungan suatu *activity* pararel. Memiliki banyak panah masuk dan satu panah keluar.

8. *ActivityFinal*

   Lingkaran penuh dalam lingkaran kosong yang menunjukkan akhir dari suatu sekuensial *activity.*Dalam 1 diagram bisa terdapat lebih dari 1 *activity final.*

9. *Swimlane*

   Batas yang menggambarkan *actor* pada suatu *activity* dengan *activity* lainnya.



Gambar 2. 5 *Activity Diagram* tanpa *swimlane*

Sumber: Whitten J.L. (2007: 392)

2.1.4.4. Sequence Diagram

MenurutWhitten (2007: 394), *sequence diagram* menggambarkan interaksi antara *actor* dan sistem dalam *use*

case. Diagram ini menunjukkan alur pengiriman pesan tersebut dalam urutan waktu berdasarkan objek yang memanggilnya.

Terdapat beberapa hal penting yang perlu diperhatikan dalam membuat *sequence diagram,* yaitu:

1. *Actor*

   Actor / objek yang melakukan pengiriman / penerimaan pesan.

2. *System*

   Kotak berisi nama yang menunjukkan bahwa objek dari proses tersebut adalah sistem.

3. *Lifelines*

   Garis yang berada di bawah objek (baik *actor* ataupun *system*) yang menunjukkan aktivasi *sequence* tersebut

4. *ActivationBar*

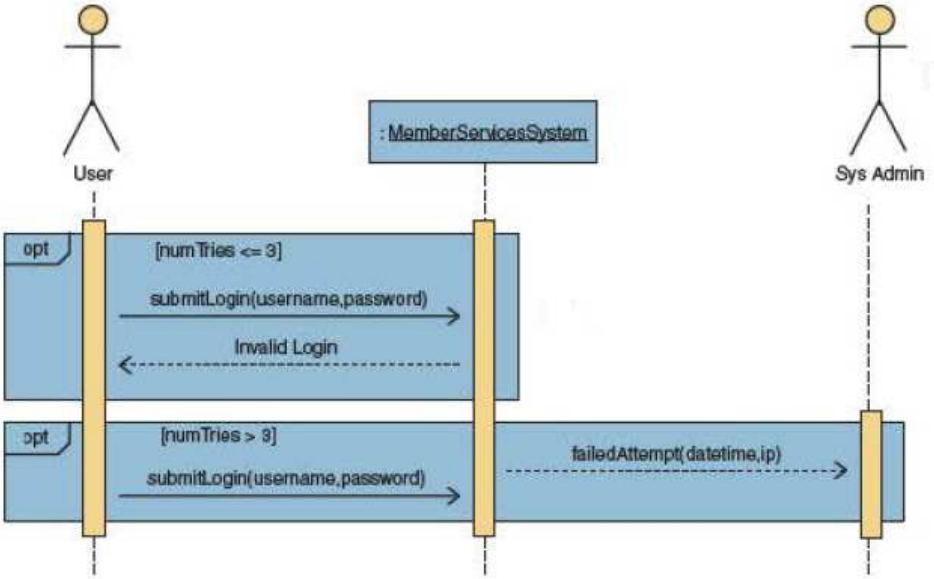
   *Bar*yang diletakkan di *lifeline* yang menunjukkan waktu ketika suatu objek aktif dalam interaksi.

5. *Message*

   Pesan yang dikirim maupun diterima oleh objek dalam interaksi tersebut

6. *Frame*

   Kotak yang menunjukkan suatu kondisi khusus dalam pengiriman atau penerimaan pesan.



Gambar 2.6 Contoh *Sequence Diagram*

Sumber: Whitten (2007: 396)

2.1.4.5. Class Diagram

Class diagram menurut Whitten (2007: 400) merupakan penggambaran grafis dari struktur objek suatu sistem, termasuk objek dalam *class* serta relasinya. Terdapat beberapa langkah penting yang perlu diperhatikan dalam membuat *class diagram,* yaitu:

1. Mengidentifikasi *association* dan *multiplicity*

   Pada tahap ini, perlu diperhatikan mengenai hubungan antara *class* dengan *class* lainnya, baik itu berupa asosiasi ataupun *multiplicity*. Contohnya adalah seperti ini: c*lass* kasir memiliki hubungan asosiasi dengan *class* pembayaran. Kemudian, sebuah kasir dapat menangani satu atau lebih pembayaran.

2. Mengidentifikasi relasi *generalization / specialization*

   Pada tahap ini, perlu diperhatikan adanya relasi *generalization / specialization* antar *class*.Salah satu cara yang disarankan untuk

mengecek relasi ini adalah apabila terdapat *multiplicity one-to-one* antara dua *class* atau ketika terdapat dua atau lebih *class* dengan *attribute* dan *behavior* yang mirip.

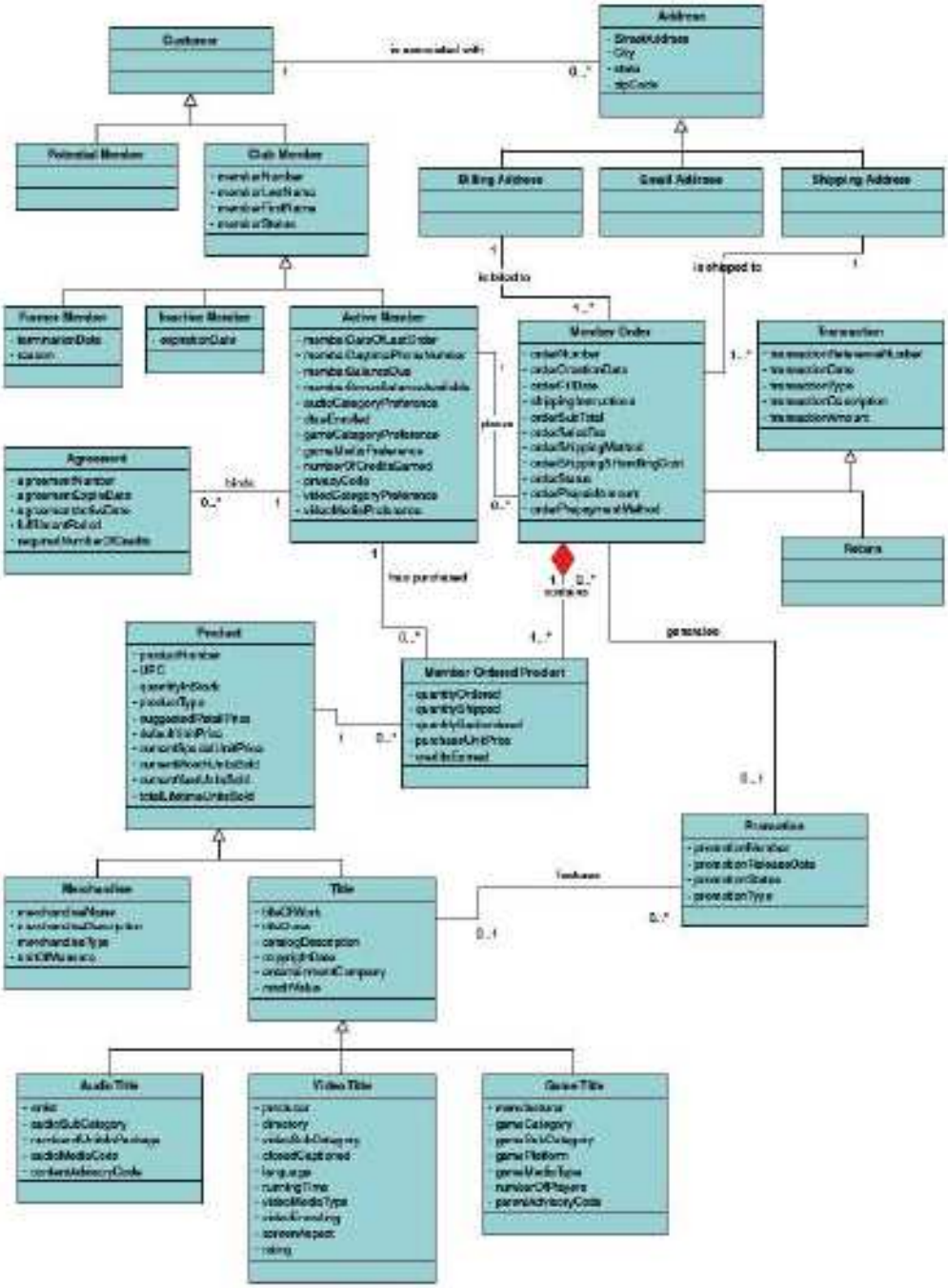
3. Mengidentifikasi relasi agregasi / komposisi

   Pada tahap ini, perlu diperhatikan mengenai relasi agregasi / komposisi. Suatu *class* dikatakan memiliki relasi agregasi ketika suatu objek A merupakan bagian dari objek B, tapi tidak berlaku sebaliknya.

4. Menyelesaikan *class diagram*

   Pada tahap ini, *class diagram* dibuat secara keseluruhan berdasarkan tahap 1 hingga 3, termasuk *attribute* dan *behavior*-nya.

Sebuah *class* dalam *Class Diagram* digambarkan dengan bentuk kotak yang terbagi menjadi 3 bagian, yaitu; (1) nama *class* tersebut, (2) attribut *class* tersebut, serta (3) operasi *(methods) class* tersebut.



Gambar 2.7 Contoh*Class Diagram*

Sumber: Whitten (2007: 396)

2.1.4.6. ERD

Menurut Whitten (2007: 271), ERD merupakan data model yang memanfaatkan beberapa notasi untuk menggambarkan entitas dan relasi dari data yang bersangkutan. Terdapat 3 hal penting yang harus diperhatikan dalam pembuatan ERD, yaitu:

1. *Entity* (Entitas)

   Entity merupakan sebuah *class* dari sekelompok objek yang digunakan untuk menyimpan data yang bersangkutan.Penggambaran ini dibuat dengan bentuk kotak dengan ujung bulat.Perlu diperhatikan bahwa terdapat *entity instance* yang merupakan salah satu bagian dari keseluruhan entitas.

   Contohnya terdapat entitas "Mahasiswa" yang mencakup keseluruhan mahasiswa.Maka, *entity instance* misalnya adalah Andrew, John, dan Jack.

2. *Attribute*

   Attribute merupakan karakteristik atau data deskriptif dari suatu entitas. Contoh *attribute* dari entitas "Mahasiswa"misalnya adalah nama, umur, alamat, nomor telepon, dan jenis kelamin. Jika dilihat pada contoh di atas, terdapat *attribute* nama pada entitas tersebut.

   Sebenarnya, "nama" merupakan *compound attribute*¸ yaitu beberapa *attribute* yang dikelompokkan menjadi satu, dalam hal ini adalah nama depan dan nama belakang menjadi satu *attribute* yaitu "nama".

   Di dalam *attribute,* terdapat beberapa hal yang perlu diperhatikan, yaitu:

a. *Domain*

Domain digunakan untuk mendeskripsikan nilai yang dapat diterima oleh *attribute* yang bersangkutan.Misalnya pada *attribute "*umur*"* terdapat *domain* hanya berupa angka.

Selain itu terdapat *default value* yang digunakan sebagai nilai *default* dari suatu *attribute* yang telah ditentukan sebelumnya.

b. *Identification*

Sebuah *attribute* atau sekelompok *attribute* dengan nilai yang unik dalam setiap entitas yang disebut *key.*Terdapat beberapa penggolongan *key,* yaitu *candidatekey* – beberapa attribute yang menjadi kandidat *primary key*, *primarykey – candidate key* yang terpilih menjadi identifikasi entitas yang utama, *alternatekey – candidate key* yang tidak terpilih menjadi *primary key.*

3. *Relationship*

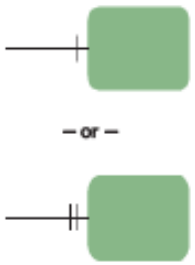
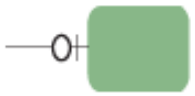
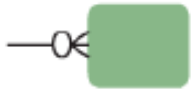
Pada kenyataannya, entitas dan *attribute* tidak berdiri sendiri, namun saling mempengaruhi satu sama lain.

Terdapat beberapa hal yang perlu diperhatikan dalam membuat *relationship,* yaitu:

a. *Cardinality*

Cardinality menggambarkan kemungkinan minimum dan maksimum dari kemunculan suatu

entitas dan hubungannya dengan entitas lain.

CARDINALITY INTERPRETATION	MINIMUM INSTANCES	MAXIMUM INSTANCES	GRAPHIC NOTATION
Exactly one (one and only one)	1	1	
Zero or one	0	1	
One or more	1	many (>1)	
Zero, one, or more	0	many (>1)	
More than one	>1	>1	

Gambar 2.8 Notasi *Cardinality*

Sumber: Whitten (2007: 276)

b. Foreign Key

Primary key pada suatu entitas yang digunakan pada entitas lain untuk menggambarkan hubungan antara entitas tersebut.

2.1.5. *Eight Golden Rules of Interface Design*

Untuk meningkatkan *usability* sebuah aplikasi, memiliki *interface* yang didesain dengan baik sangatlah penting.

Menurut Shneiderman (2010: 88), terdapat *"8 Golden Rules of Interface Design"*, yaitu:

1. *Strive for consistency*

   Penggunaan aksi yang konsisten pada situasi yang mirip; penggunaan terminologi yang konsisten dan tidak berubah – ubah; serta penggunaan perintah yang konsisten harus diutamakan.

2. *Cater to universal usability*

   Adanya kebutuhan dari beberapa tipe user yang berbeda, dan design disesuaikan dengan kebutuhan tersebut. Perbedaan ini mencakup tingkat kehandalan user (*novice* hingga *expert*), range umur, keterbatasan fisik, dan kemampuan teknologi.

3. *Offer informative feedback*

   Feedback ini berfungsi untuk menyatakan bahwa ada aksi yang telah dilakukan terhadap / oleh aplikasi. Jenis *feedback* yang digunakan bisa disesuaikan dengan jenis aksi yang dilakukan.

4. *Design dialog to yield closure*

   Aksi sekuensial dapat dibagi menjadi 3 bagian, yaitu awal, tengah, dan akhir.Nyatakan dengan jelas ketika suatu aksi yang sekuensial sudah selesai dilakukan.

5. *Prevent errors*

   Desain sistem dalam cara yang memungkinkan untuk mengurangi kemungkinan terjadinya *error*.Antisipasi terjadinya *error* dengan merancang sistem yang dapat mendeteksi *error* dan memberi solusi sederhana dari *error* tersebut.

6. *Permit easy reversal of actions*

   Sediakan cara bagi *user* untuk membatalkan aksi yang sebelumnya telah dilakukan.Sesuaikan fungsi ini dengan kebutuhan, baik itu untuk membatalkan satu aksi, *dataentry*, atau keseluruhan aksi sekuensial tersebut.

7. *Support internal locus of control.*

   Desain sistem agar *user* menjadi inisiator dari suatu aksi, bukannya merespon terhadap aksi yang diberikan oleh aplikasi.

8. *Reduce short-term memory load.*

   Keterbatasan kemampuan manusia dalam memproses informasi mengharuskan desain yang dibuat haruslah sesederhana mungkin, dan mudah dimengerti oleh *user*.

2.2 Simpulan dari Jurnal

2.2.1. Implementasi Sistem Informasi Rumah Sakit Untuk Subsistem Laboratorium Dengan Framework PRADO

Handoyo et al (2008:10) secara spesifik mengatakan bahwa "Sistem Informasi Rumah Sakit (SIRS) adalah suatu tatanan yang berurusan dengan pengumpulan data, pengelolaan data, penyajian informasi, analisis dan penyimpulan informasi serta penyampaian informasi yang dibutuhkan untuk kegiatan rumah sakit".

Penggunaan *framework*pada pengembangan *software* menggambarkan struktur pendukung agar *software* tersebut bukan hanya lebih mudah dalam pengembangannya, tapi juga terorganisir dengan baik.Namun perlu diperhatikan ketika menggunakan *framework*, pengembang berikutnya tentu membutuhkan kemampuan dalam pengembangan aplikasi dalam *framework* tersebut.Hal ini dapat menyebabkan keterbatasan dalam pengembangan aplikasi yang berikutnya.

Terdapat beberapa faktor yang perlu diperhatikan dalam proses pengembangan SIRS tersebut, dan juga pada pengembangan aplikasi *e*-klnik, yaitu:

1. Pengembangan aplikasi *e*-klinik tidak menggunakan *framework* untuk mempermudah pengembangan aplikasi ke depannya.
2. Perancangan sistem, *database,* dan layar merupakan faktor penting dalam proses pengembangan aplikasi. Hal ini dikarenakan perancangan tersebut menggambarkan fungsi sistem secara keseluruhan, agar hasil yang didapatkan sesuai dengan yang ingin dicapai.
3. Evaluasi hasil implementasi, dengan cara apapun yang dipilih, berguna untuk memastikan bahwa sistem yang telah dibuat sesuai dengan kebutuhan.

3.2.2. *Use of Electronic Health Records in U.S. Hospitals*

Dari jurnal tersebut, dapat terlihat dari data yang didapatkan oleh peneliti bahwa masih ada rumah sakit di Amerika yang tidak menerapkan sistem EHR. Menurut Jha et al (2009:1632), terdapat beberapa faktor yang penghalang dari rumah sakit yang tidak menggunakan sistem EHR, yaitu:

1. Biaya pemasangan EHR
2. Biaya *maintenance* EHR
3. Bukan merupakan investasi yang dapat mengembalikan uang yang sudah dikeluarkan
4. Kurangnya karyawan berpengalaman di bidang IT

Faktor tersebut dapat diimplementasikan dalam pengembangan aplikasi *e*-klinik, yaitu pada segi:

1. Membuat aplikasiyang sederhana, dan mudah di *maintenance* tanpa harus mengeluarkan biaya banyak, misalnya dengan mencari *hosting* yang terjangkau
2. Membuat desain yang sederhana dan mudah dimengerti sehinggan *user* tidak memerlukan keahlian khusus untuk menggunakannya.

2.2.3. Accuracy and Speed of Electronic Health Record Versus Paper-Based Ophthalmic Documentation Strategies

Dari jurnal tersebut, dapat disimpulkan bahwa sistem pendataan secara digital (EHR) memakan waktu cenderung lebih lama dari sistem tertulis, dalam segi dokumentasi pengujian *ophthalmic*.

Namun jika diperhatikan lebih lanjut, menurut Chanet al (2013:171), mempelajari sistem EHR (*Electronic Health Records*)merupakan sebuah faktor penting dalam penggunaan EHR.Ada kemungkinan terjadinya perubahan waktu dan ketepatan dalam penggunaan EHR.

Menurut Chan et al (2013:171), keuntungan utama dari teknik EHR adalah pengambilan dan analisa data digital, dan bukannya pemasukan data.

Jika diimplementasikan dalam pengembangan *aplikasi*ini, terdapat beberapa hal yang perlu diperhatikan, yaitu:

1. Penggunaan sistem EHR membutuhkan pelatihan. Jika *design*sederhana, maka akan lebih mudah dimengerti oleh *user*. Sebaliknya jika *design*yang dibuat terlalu kompleks, maka akan dibutuhkan waktu pelatihan bagi *user* agar terbiasa menggunakan aplikasitersebut.
2. Pada hasil pengujian lab dan hasil diagnostik, dapat diimplementasikan dalam kemudahan pemasukan data dan memberi akses untuk melihat data hasil pengujian lab dan hasil diagnostik secara mudah.