BAB II

LANDASAN TEORI

2.1 Konsep Dasar Sistem

2.1.1 Pengertian Sistem

Menurut Hartono (2013:9) “Sistem adalah suatu himpunan dari berbagai bagian atau elemen, yang saling berhubungan secara terorganisasi berdasarkan fungsi-fungsinya, menjadi suatu kesatuan.”

2.1.2 Karakteristik Sistem

Menurut Hartono (2013:14) sebuah sistem memiliki beberapa karakteristik sebagai berikut:

1. Komponen (*components*)

   Bagian-bagian atau elemen-elemen yang dapat berupa benda atau manusia, berbentuk nyata atau abstrak dan disebut subsistem.

2. Penghubung antarbagian (*interface*)

   Sesuatu yang bertugas menjembatani satu bagian dengan bagian lain dan memungkinkan terjadinya interaksi/komunikasi antarbagian.

3. Batas (*boundary*)

   Sesuatu yang membedakan antara satu sistem dengan sistem atau sistem-sistem lain.

4. Lingkungan (*environment*)

   Segala sesuatu yang berada di luar sistem dan dapat bersifat menguntungkan atau merugikan sistem yang berangkutan.

5. Masukan (*input*)

   Sesuatu yang merupakan bahan untuk diolah atau diproses oleh sistem.

6. Mekanisme pengolahan (*processing*)

   Perangkat dan prosedur untuk mengubah masukan menjadi keluaran dan menampilkannya.

7. Keluaran (*output*)

   Berbagai macam bentuk hasil atau produk yang dikeluarkan dari pengolahan.

8. Tujuan (*goal/objective*)

   Sesuatu atau keadaan yang ingin dicapai oleh sistem, baik dalam jangka pendek maupun jangka panjang. Sensor dan kendali (sensor & kontrol). Sesuatu yang bertugas memantau dan menginformasikan perubahan-perubahan di dalam lingkungan dan dalam diri sistem kepada sistem.

9. Umpan-balik (*feedback*)

   Informasi tentang perubahan-perubahan lingkungan dan perubahan-perubahan (penyimpangan) dalam diri sistem.

2.1.3 Klasifikasi Sistem

Menurut Sutabri (2012:15) Suatu sistem mempunyai klasifikasi yang tertentu, yaitu mempunyai :

1. Sistem Abstrak dan Sistem Fisik

Sistem abstrak adalah sistem yang berupa pemikiran atau ide-ide yang tidak tampak secara fisik dan sistem fisik adalah sistem yang ada secara fisik.

2. Sistem Alamiah dan Sistem Buatan Manusia

   Sistem alamiah adalah sistem yang terjadi karena proses alam tidak dibuat oleh manusia (ditentukan dan tunduk kepada kehendak sang pencipta alam). Sistem buatan manusia yang melibatkan interaksi manusia dengan mesin disebut dengan *human-machine system* atau ada yang menyebut dengan *man-machine system.*

3. Sistem Deterministik dan Sistem Probabilistik

   Deterministic system beroperasi dengan tingkah laku yang sudah dapat diprediksi. Interaksi diantara bagian-bagiannya dapat di deteksi dengan pasti. *Probabilistic system* adalah sistem yang kondisi masa depannya tidak dapat diprediksi karena mengandung unsur probabilitas.

4. Sistem Terbuka dan Sistem Tertutup

   Sistem tertutup merupakan sistem yang tidak berhubungan dan tidak terpengaruh oleh lingkungan luarnya. Sistem ini bekerja secara otomatis tanpa ada campur tangan dari pihak luar. Sedangkan sistem terbuka adalah sistem yang berhubungan dan dipengaruhi oleh lingkungan luarnya, yang menerima masukan dan menghasilkan keluaran untuk subsistem lainnya.

2.1.4 *System Development Life Cycle* (SDLC)

1. Pengertian SDLC

   Menurut Sukamto dan Shalahuddin (2013:26) mengemukakan bahwa SDLC atau *Software Development Life Cycle* atau sering disebut juga *System Development Life Cycle* adalah proses mengembangkan atau mengubah suatu sistem perangkat lunak dengan menggunakan model-model dan metodologi

yang digunakan orang untuk mengembangkan sistem-sistem perangkat lunak sebelumnya (berdasarkan *best practice* atau cara-cara yang sudah teruji baik).

2. Tahapan SDLC

Menurut Sukamto dan Shalahuddin (2013:26) Tahapan-tahapan yang ada pada SDLC secara global adalah:

a. Inisiasi (*initiation*)

   Tahap ini biasanya ditandai dengan pembuatan proposal proyek perangkat lunak.

b. Pengembangan konsep sistem (*system concept development*)

   Mendefinisikan lingkup konsep termasuk dokumen lingkup sistem, analisis manfaat biaya, menejemen rencana, dan pembelajaran kemudahan sistem.

c. Perencanaan (*planning*)

   Mengembangakan rencana menejemen proyek dan dokumen perencanaan lainnya. Menyediakan dasar untuk mendapatkan sumber daya (*resources*) yang dibutuhkan untuk memperoleh solusi.

d. Analisis kebutuhan (*requirements analysis*)

   Menganalisais kebutuhan pemakai sistem perangkat lunak (*user*) dan mengembangkan kebutuhan *user*. Membuat dokumen kebutuhan fungsional.

e. Desain (*design*)

   Mentransformasikan kebutuhan detail menjadi kebutuhan yang sudah lengkap, dokumen desain sistem fokus pada bagaimana dapat memenuhi fungsi-fungsi yang dibutuhkan.

f. Pengembangan (*development*)

Mengonversi desain ke sistem informasi yang lengkap termasuk bagaimana memperoleh dan melakukan instalasi lingkungan sistem yang dibutuhkan; membuat basis data dan mempersiapkan berkas atau *file* mengujian, pengkodean, pengompilasian, memperbaiki dan membersihkan program; peninjauan pengujian.

g. Integrasi dan pengujian (*integration and test*)

   Mendemonstrasikan sistem perangkat lunak bahwa telah memenuhi kebutuhan yang dispesifikasikan pada dokumen kebutuhan fungsional. Dengan diarahkan oleh staf penjamin kualitas (*quality assurance*) dan *user*. Menghasilkan laporan analisis pengujian.

h. Implementasi (*implementation*)

   Termasuk pada persiapan implementasi, implementasi perangkat lunak pada lingkungan produksi (lingkungan pada *user*) dan menjalankan resolusi dari permasalahan yang teridentifikasi dari fase integrasi dan pengujian.

i. Operasi dan pemeliharaan (*operations and maintenance*)

   Mendeskripsikan pekerjaan untuk mengoprasikan dan memelihara sistem informasi pada lingkungan produksi (lingkungan pada *user*), termasuk implementasi akhir dan masuk pada proses peninjauan.

j. Disposisi (*disposition*)

   Mendeskripsikan aktifitas ahir dari pengembangan sistem dan membangun data yang sebenarnya sesuai dengan aktifitas *user*.

3. Metode SDLC

   a. Model *waterfall*
      Menurut Sukamto dan Sahalahudin (2013:28) model SDLC air terjun (*waterfall*) sering juga disebut model sekuensial linier (*sequential linear*) atau alur hidup klasik (*classic life cycle*). Model air terjun menyediakan pendekatan alur hidup perangkat lunak secara sekuensial atau terurut mulai dari analisis, desain, pengodean, pengujian, dan tahap pendukung (*support*).

   b. Model Prototipe
      Menurut Sukamto dan Shalahuddin (2013:31) model prototipe dimulai dari mengumpulkan kebutuhan pelanggan terhadap perangkat lunak yang akan dibuat. Lalu dibuatlah program prototipe agar pelanggan lebih terbayang dengan apa yang sebenarnya diinginkan. Program ini biasanya merupakan program yang belum jadi.

   c. Model *Rapid Application Development* (RAD)

      Menurut Sukamto dan Shalahuddin (2013:34) "*Rapid Application Development* (RAD) adalah model proses pengembangan perangkat lunak yang bersifat inkremental terutama untuk waktu pengerjaan yang pendek."

   d. Model Iteratif

      Menurut Sukamto dan Shalahuddin (2013:38) menyatakan bahwa "model iteratif (*iterative model*) mengkombinasikan proses-proses pada model air terjun dan prototipe pada model iteratif."

   e. Model Spiral

      Menurut Sukamto dan Shalahuddin (2013:39) mengemukakan bahwa "model spiral (*spiral model*) memasangkan iteratif pada model prototipe dengan kontrol dan aspek sistematik yang diambil dari model air terjun."

2.1.5 Sistem Informasi Manajemen

1. Pengertian Informasi

Menurut Gordon B. Davis dalam Hartono (2013:15) memberikan definisi informasi sebagai berikut: “Informasi adalah data yang telah diolah menjadi suatu bentuk yang berguna bagi penerimanya dan memiliki nilai bagi pengambilan keputusan saat ini atau di masa yang akan datang.”

2. Pengertian Sistem Informasi

Menurut Hartono (2013:16) menyatakan bahwa “sistem informasi adalah seperangkat komponen yang saling berhubungan, yang bekerja untuk mengumpulkan dan menyimpan data serta mengolahnya menjadi informasi untuk digunakan.”

3. Pengertian Sistem Informasi Manajemen

Menurut Hartono (2013:20) menyimpulkan bahwa “sistem informasi manajemen adalah sebuah sistem, rangkaian terorganisasi dari sejumlah bagian/komponen yang secara bersama-sama berfungsi atau bergerak menghasilkan informasi untuk digunakan dalam menejemen perusahaan.”

2.1.6 Sistem Informasi Akuntansi

Menurut Wijayanto dalam Mardi (2011:4) mengemukakan bahwa “Sistem Informasi Akuntansi adalah susunan berbagai dokumen, alat komunikasi, tenaga pelaksana, dan berbagai laporan yang di desain untuk mentransformasikan data keuangan menjadi informasi keuangan.”

2.1.7 Penjualan

> Menurut Puspitawati dan Anggadini (2011:165) mengatakan bahwa penjualan merupakan aktifitas memperjual-belikan barang atau jasa kepada konsumen. Aktifitas penjualan dalam perusahaan dapat dilakukan secara tunai ataupun

kredit. Penjualan tunai merupakan penjualan dengan cara menerima uang tunai/*cash* pada saat barang diserahkan kepada pembeli. Penjualan kredit adalah aktifitas penjualan yang menimbulkan tagihan, klaim piutang kepada pembeli (*customer*) sehingga penjual tidak menerima uang pada saat barang diserahkan kepada pembeli.

2.1.8 Jurnal

Menurut Permana, dkk (2009:198) jurnal adalah alat untuk mencatat transaksi perusahaan yang dilakukan secara kronologis (berdasarkan urutan waktu) dengan menunjukan akun perkiraan yang harus di debit dan di kredit beserta jumlahnya masing-masing.

Setiap transaksi yang terjadi dalam suatu perusahaan harus langsung dicatat ke dalam jurnal sebelum dibukukan ke buku besar. Oleh karena itu, buku jurnal sering disebut sebagai buku catatan pertama di dalam sistem akutansi. Pencatatan transaksi yang bersangkutan dengan penjualan dicatat ke dalam jurnal seperti berikut atau transaksi penjualan tunai.

Kas xxx

Penjualan xxx

2.2 Peralatan Pendukung (*Tool System*)

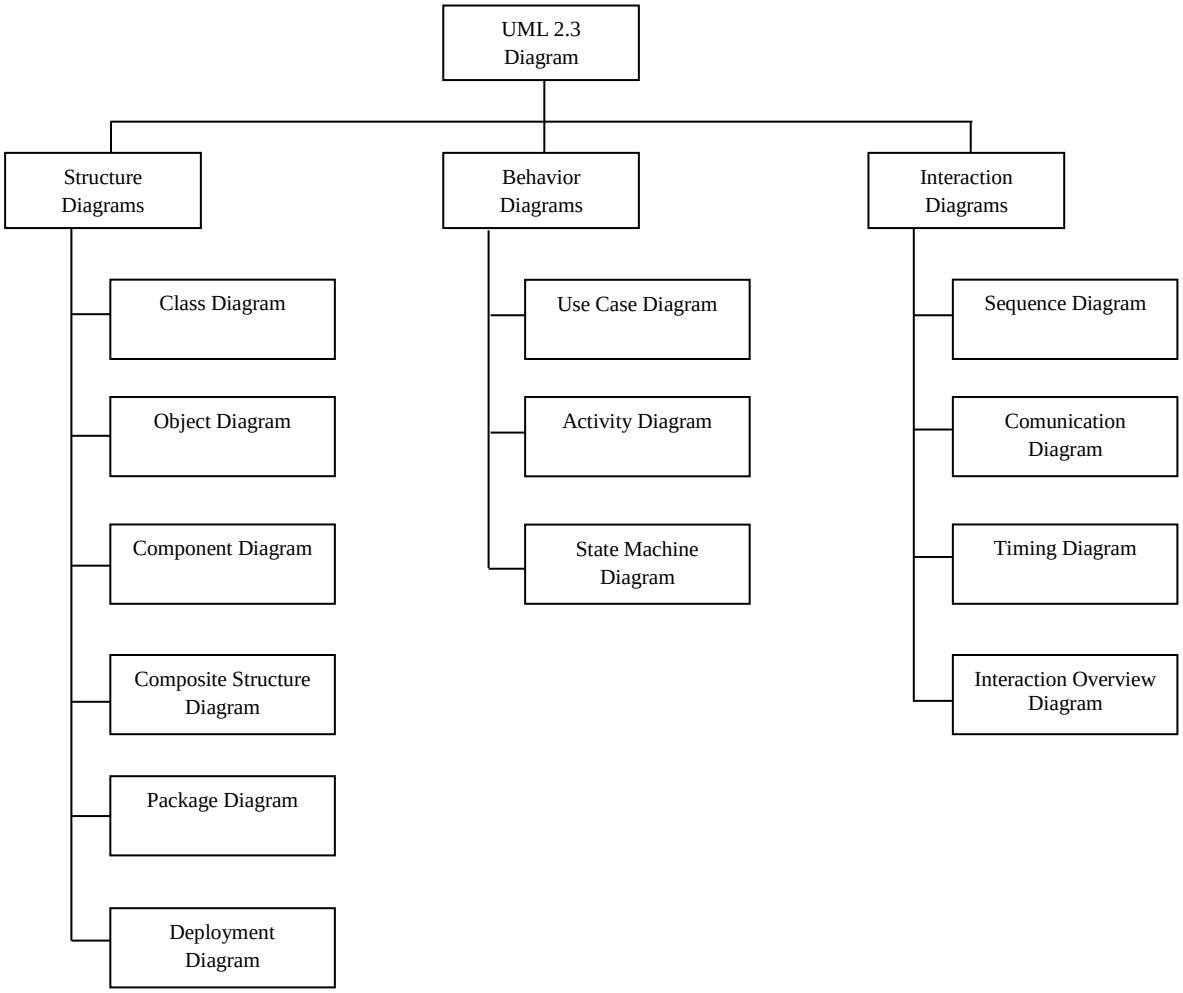
2.2.1 *Unified Modeling Language* (UML)

1. Pengertian UML

Menurut Sukamto dan Shalahuddin (2013:133) menyatakan bahwa : “UML (*Unified Modeling Language*) adalah salah satu standar bahasa yang banyak digunakan di dunia industri untuk mendefinisikan *requirement*, membuat analisis & desain, serta menggambarkan arsitektur dalam pemrograman berorientasi objek.”

2. Diagram UML

Menurut Sukamto dan Shalahuddin (2013:140) “UML terdiri dari 13 macam diagram yang dikelompokkan dalam 3 kategori.” Pembagian kategori dan macam-macam diagram tersebut dapat dilihat pada gambar di bawah ini:



Sumber : Sukamto dan Shalahuddin (2013:140)

Gambar II.1

Diagram UML

a. Activity Diagram

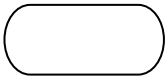
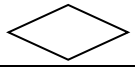
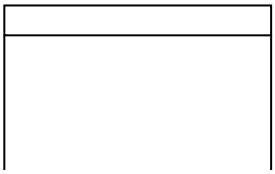
1) Pengertian *Activity* Diagram

Menurut Sukamto dan Shalahuddin (2013:161) “Diagram Aktivitas atau *Activity* Diagram menggambarkan *workflow* (aliran kerja) atau aktivitas dari sebuah sistem atau proses bisnis atau menu yang ada pada perangkat lunak.”

2) Simbol dan Fungsi *Activity* Diagram

Tabel II.1

Simbol dan Fungsi *Activity* Diagram

Status Awal 	Status awal aktivitas sistem, sebuah diagram aktivitas memiliki sebuah status awal
Aktivitas 	Aktivitas yang dilakukan sistem, aktivitas biasanya diawali dengan kata kerja
Percabangan/ *Decision* 	Asosiasi percabangan dimana jika ada pilihan aktivitas lebih dari satu
Penggabungan / Join	Asosiasi penggabungan dimana lebih dari satu aktivitas digabungkan menjadi satu
Status Akhir 	Status akhir yang dilakukan sistem, sebuah diagram aktivitas memiliki sebuah status akhir
Swimline 	Memisahkan organisasi bisnis yang bertanggungjawab terhadap aktivitas yang terjadi

Sumber : Sukamto dan Shalahuddin (2013:162)

b. *Use Case* Diagram

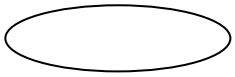
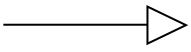
1) Pengertian *Use Case* Diagram

Mesnurut Sukamto dan Shalahuddin (2013:155) mengemukakan bahwa "*Use case* atau diagram *use case* merupakan pemodelan untuk kelakuan (*behavior*) sistem informasi yang akan dibuat."

2) Simbol dan Fungsi *Use Case* Diagram

Tabel II.2

Simbol dan Fungsi *Use Case* Diagram

Use Case 	Fungsionalitas yang disediakan sistem sebagai unit-unit yang saling bertukar pesan antar unit atau aktor.
Aktor/*Actor* 	Orang, proses, atau sistem lain yang berinteraksi dengan sistem informasi yang akan dibuat di luar sistem informasi yang akan dibuat itu sendiri.
Asosiasi/*Association*	Komunikasi antara *actor* dan *usecase* yang berpartisipasi pada *usecase* atau *usecase* memiliki interaksi dengan actor
Ekstensi/*extend*	Relasi usecase tambahan kesebuah usecase yang ditambahkan dapat berdiri sendiri walau tanpa usecase tambahan itu.
Include	Relasi *usecase* dimana proses bersangkutan akan dilanjutkan ke proses yang dituju.
Generalisasi/*eneralization* 	Hubungan Generalisasi Dan Spesialisasi (umum-khusus) antara dua buah use case dimana fungsi yang satu adlaah fungsi yang lebih umum dari lainnya

Sumber : Sukamto dan Shalahuddin (2013:156)

c. *Sequence* Diagram

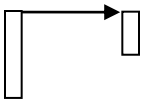
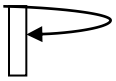
1) Pengertian *Sequence* Diagram

Menurut Sukamto dan Shalahuddin (2013:165) mengemukakan bahwa "Diagram sekuen menggambarkan kelakuan objek pada *use case* dengan mendeskripsikan waktu hidup objek dan *message* yang dikirimkan dan diterima antar objek".

2) Simbol dan Fungsi *Sequence* Diagram

Tabel II.3

Simbol dan Fungsi *Sequence* Diagram

Actor 	Orang, proses, atau sistem lain yang berinteraksi dengan sistem yang akan di buat
Lifeline	Menyatakan kehidupan suatu objek
Activation 	Menyatakan objek dalam keadaan aktif dan berinteraksi
Object message 	Menyatakan suatu objek membuat objek lain, arah panah mengarah pada objek yang dibuat
Message to self 	Menyatakan suatu objek memanggil dirinya sendiri

Sumber : Sukamto dan Shalahuddin (2013:165)

Entity Class 	Kelas entitas (*entity class*) digunaan untuk memodelkan informasi yang berumur relatif panjang dalam sistem (*persistent class*)
Control Class 	Kelas-kelas kendali (*control class*) merepresentasikan kooordinasi, urutan (*sequence*), transaksi (transaction), dan kendali ke objek lainnya dan sering diigunakan untuk membungkus (*encapsulated*) kendali yang berhubungan dengan suatu use case tertentu yang bersifat spesifik.
Interface atauBoundary Object 	Kelas pembatas (*boundary class*) digunakan untuk memodelkan interaksi antara sistem perangkat lunak dengan *actor*-nya.

Nurgroho (2010:175)

d. *Deployment* Diagram

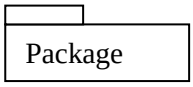
1) Pengertian *Deployment* Diagram

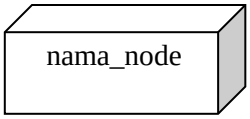
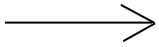
Menurut Sukamto dan Shalahuddin (2013:154) menyatakan bahwa "Diagram *deployment* atau *deployment* diagram menunjukkan konfigurasi komponen dalam proses eksekusi aplikasi."

2) Simbol dan Fungsi *Deployment* Diagram

Tabel II.4

Simbol dan Fungsi *Deployment* Diagram

Package 	Merupakan sebuah bungkusan dari satu atau lebih *node*

Node 	Biasanya mengacu pada perangkat keras (*hardware*), perangkat lunak (*software*)
Dependency 	Kebergantungan antar node, arah panah mengarah pada *node* yang dipakai
Link 	Relasi antar *node*

Sumber : Sukamto dan Shalahuddin (2013:154)

2.2.2 *Entity Relationship* Diagram (ERD)

1. Pengertian ERD

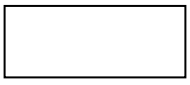
Menurut Sukamto dan Shalahuddin (2013:50) "*Entity Relationship* Diagram (ERD) dikembangkan berdasarkan teori himpunan dalam bidang matematika. ERD digunakan untuk pemodelan basis data relasional".

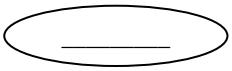
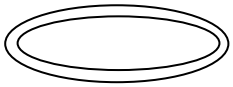
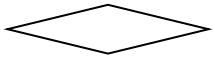
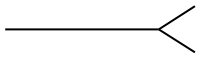
2. Simbol dan Fungsi ERD

Menurut Sukamto dan Shalahuddin (2013:51) "berikut adalah simbol dan fungsi ERD" :

Tabel II.5

Simbol *Entity Relationship Diagram* (ERD)

Entitas/*Entity* 	Entitas merupakan data inti yang akan di simpan

Atribut 	*Field* atau kolom data yang butuh disimpan dalam suatu entitas
Atribut kunci primer 	*Field* atau kolom data yang butuh disimpan dalam suatu entitas dan dgunakan sebagai kunci akses *record* yang diinginkan
Atribut multinilai 	Atribut multinilai atau *multivalue*, merupakan *field* data yang butuh disimpan dan memiliki nilai lebih dari satu
Relasi 	Relasi, relasi yang menghubungkan antar entitas; biasanya diawali dengan kata kerja
asosiasi/ *Association* N 	Asosiasi adalah penghubung antara relasi dan entitas dimana kedua ujungnya memiliki *multiplycity* kemungkinan jumlah pemakaian

Sumber : Sukamto dan Shalahuddin (2013:50)

3. *Mapping Cardinality*

Kardinalitas pemetaan atau rasio kardinalitas menunjukkan jumlah *entity* yang dihubungkan ke satu *entity* lain dengan suatu *relationship sets.* Menurut Kusrini (2007:24) kardinalitas pemetaan meliputi:

a. Hubungan satu ke satu (*one to one*)

   Yaitu satu *entity* dalam A dihubungkan dengan maksimum satu *entity* B.

b. Hubungan satu ke banyak (*one to many*)

   Yaitu satu *entity* dalam A dihubungkan dengan sejumlah *entity* B. *Entity* dalam B dihubungkan dengan maksimum satu *entity* dalam A.

c. Hubungan banyak ke satu (*many to one*)

Yaitu sejumlah *entity* dalam A dihubungkan dengan maksimum satu *entity* B. Satu *entity* dalam B dapat dihubungkan dengan sejumlah *entity* dalam A.

d. Hubungan banyak ke banyak (many to many)

Sejumlah *entity* dalam A dihubungkan dengan sejumlah *entity* dalam B. Sejumlah *entity* dalam B dihubungkan dengan sejumlah *entity* dalam A.

4. Tahapan Pembuatan ERD

Menurut Simarmata (2007:99) ada 10 langkah tahapan dalam membuat ERD atau metodologi ERD, yaitu :

a. Langkah 1, menentukan entitas. Menentukan peran, kejadian, lokasi, hal nyata, dan kosep dimana pengguna akan menyimpan data.
b. Langkah 2, menentukan relasi. Tentukan hubungan antara pasangan entitas menggunakan matriks relasi.
c. Langkah 3, gambar ERD sementara. Entitas digambarkan dengan kotak dan reasi dengan garis yang menghubungkan entitas.
d. Langkah 4, mengisi kardinalitas. Tentukan jumlah yang mengidentifikasi satu dan hanya satu kejadian pada entitas yang berhubungan.
e. Langkah 5, menetukan kunci utama. Tentukan atribut yang mengidentifikasi satu dan hanya satu kejadian pada masing-masing entitas.
f. Langkah 6, gambar ERD berdasarkan kunci. Hilangkan relasi *many-to-many* dan masukan *primary key* dan kunci tamu pada masing-masing entitas.
g. Langkah 7, menetukan atribut. Tuliskan *field-field* yang diperlukan oleh sistem.

h. Langkah 8, pemetaa atribut. Pasangkan atribut dengan satu entitas yang sesuai pada masing-masing atribut.

i. Langkah 9, gambar ERD dengan atribut. Aturlah ERD dari langkah 6 dengan menambahkan entitas atau relasi yang ditentukan pada langkah 8.

j. Langkah 10, periksa hasil. Lihatlah diagram dari sudut pandang pemilik atau pengguna sistem, periksa kardinalitasnya dan lihat daftar atribut yang berhubungan dengan masing-masing entitas untuk melihat apakah ada atribut yang hilang.

5. *Logical Relationship Structure* (LRS)

Menurut Hasugian dan Shidiq (2012:608) memberikan batasan bahwa LRS adalah "sebuah model sistem yang digambarkan dengan sebuah diagram-ER akan mengikuti pola atau aturan pemodelan tertentu dalam kaitannya dengan konvensi ke LRS". Perubahan yang terjadi yaitu mengikuti aturan-aturan sebagai berikut:

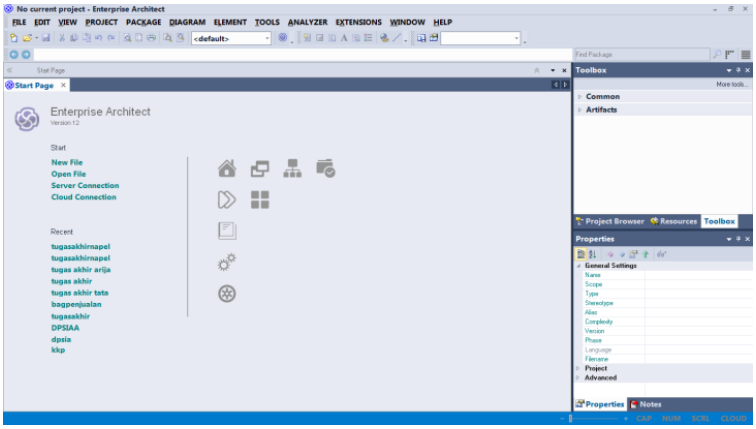
a. Setiap entitas akan diubah kebentuk kotak.

b. Semua atribut relasi disatukan dalam sebuah kotak bersama entitas jika hubungan yang terjadi pada diagram-ER 1:M (relasi bersatu dengan *cardinality* M) atau tingkat hubungan 1:1 (relasi bersatu dengan *cardinality* yang paling membutuhkan referensi).

c. Sebuah relasi dipisah dalam sebuah kotak tersendiri (menjadi entitas baru) jika tingkat hubungannya M:M (*many to many*) akan memiliki *foreign key* sebagai *primary key* yang diambil dari kedua entitas yang sebelumnya saling berhubungan.

2.2.3 *Enterprise Architect* (EA)

Menurut Parizeu dalam Murtadho (2013:1) "*enterprise architect* merupakan suatu pendekatan logis komprehensif dan *holistic* untuk merancang dan mengimplementasikan sistem dan komponen sistem secara bersama-sama yang meliputi suatu infrastruktur manajemen informasi/teknologi."



Sumber : Enterprise Architect

Gambar II.2

Enterprise Architect