BAB 2
LANDASAN TEORI

2.1 Sistem Informasi

Sistem informasi adalah kumpulan dari komponen yang saling berhubungan yang mengumpulkan memproses, menyimpan dan mendistribusikan informasi untuk membantu dalam pengambilan keputusan dan control dalam sebuah perusahaan (Laudon, 2013). Sistem informasi merupakan suatu sistem yang mengumpulkan, memproses, menyimpan, menganalisa dan menyebarkan informasi untuk tujuan yang spesifik (Rainer, Prince & cegielski, 2015). John W. Satzinger mendefinisikan Sistem Informasi sebagai sebuah komponen komputer yang saling berhubungan yang mengumpulkan, memproses, menyimpan (biasanya pada sebuah basis data), dan menyediakan *output* informasi yang dibutuhkan untuk menyelesaikan pekerjaan-pekerjaan dalam sebuah bisnis (Satzinger, Jackson, & Burd, 2010).

2.1.1 Komponen Sistem Informasi

Sistem merupakan kumpulan komponen yang saling berhubungan, yang secara jelas menetapaan batasan, kerjasama untuk mencapai tujuan dengan menerima *input* dan menghasilkan *output* dari proses transformasi yang terorganisir (O'Brien dan Marakas, 2014). Komponen sistem informasi terdiri dari (Rainer, Prince & Cegielski, 2015) :

1. *Hardware* (Perangkat keras)

   Berupa alat atau perlengkapan seperti *processor*, *monitor*, *keyboard*, dan *printer*. Perlengkapan ini bersama-sama menerima data dan informasi, mengolah dan menampilkannya.
2. *Software* (Perangkat lunak)

   Kumpulan program yang membantu perangkat keras mengolah data.
3. *Database* (basis data)

   Kumpulan dari tabel dan file yang berisi data dan saling berhubungan.
4. *Network* (Jaringan)

Sistem yang berhubungan dan memungkinkan beberapa komputer berbagi sumber (baik data maupun sistem informasi).

5. *Procedures* (Prosedur)

   Kumpulan interuksi mengenai cara menggabungkan komponen-komponen diatas untuk mengolah informasi dan menghasilkan *output* yang diinginkan.

6. *People* (Manusia/Orang)

   Setiap individu yang menggunakan dan berinteraksi dengan *hardware* dan *software*.

2.1.2 Karakteristik Sistem Informasi

Karateristik sistem adalah untuk mencapai tujuannya, suatu sistem harus memiliki sifat-sifat tertentu atau suatu karakteristik seperti berikut (Jogiyanto, 2005) :

1. Komponen *(Component)*

   Suatu sistem terdiri dari sejumlah komponen yang saling berinteraksi dan bekerjasama membentuk satu kesatuan.

2. Batas Sistem*(Boundary)*

   Merupakan daerah yang membatasi antara satu sistem dengan sistem lainnya atau dengan lingkungan luar.

3. Lingkungan Luar Sistem (*Environments)*

   Adalah segala sesuatu yang berada diluar batas sistem yang mempengaruhi operasi sistem baik itu yang bersifat merugikan ataupun menguntungkan.

4. Penghubung (*Interface)*

   Merupakan media penghubung antar subsistem yang memungkinkan sumber-sumber daya mengalir dari satu subsistem ke subsistem lain.

5. Masukan (*Input)*

   Adalah energi yang dimasukan kedalam sistem, yang dapat berupa masukan perawatan (*Maintenance Input)* dan masukan sinyal (*Signal Input).*

6. Keluaran (*Output)*

   Adalah hasil dari energi yang diolah dan diklasifikasikan menjadi keluaran yang berguna dari sisa pembuangan.

7. Pengolah Sistem (*Process)*

   Suatu sistem dapat mempunyai suatu bagian pengolah yang akan mengubah ma sukan menjadi keluaran

8. Sasaran (*Objective)* atau Tujuan (*Goal)*

   Suatu sistem pasti mempunyai tujuan (*goal)* , jika suatu sistem tidak mempunyai tujuan yang jelas, maka semua operasi sistem tidak ada gunanya.

2.2 Penjualan

Penjualan adalah sebuah usaha atau langkah konkrit yang dilakukan untuk memindahkan suatu produk, baik itu berupa barang atau jasa, dari produsen kepada konsumen sebagai sasarannya. Tujuan utama penjualan yaitu mendatangkan keuntungan atau laba dari produk atau barang yang dihasilkan produsennya dengan pengelolaan yang baik. Dalam pelaksanaannya, penjualan sendiri tidak dapat dilakukan tanpa adanya pelaku yang bekerja didalamnya seperti agen, pedangang, dan tenaga pemasaran.

Melakukan penjualan adalah suatu kegiatan yang ditujukan untuk mencari pembeli, mempengaruhi, dan memberi pembeli agar pembelian dapat menyesuaikan kebutuhannya dengan produksi yang ditawarkan serta mengadakan perjanjian yang ditawarkan serta mengadakan perjanjian mengenai harga yang menguntungkan kedua belah pihak (Swastha, 2001), jadi kesimpulannya bahwa penjualan adalah suatu kegiatan dan cara untuk mempengaruhi pribadi agar terjadi pembelian (penyerahan) barang atau jasa yang ditawarkan, berdasarkan harga yang telah disepakati oleh kedua belah pihak dalam kegiatan tersebut.

2.2.1 Jenis-Jenis Penjualan

Jenis-jenis penjualan antara lain (Swastha, 2001) :

a. *Trade Selling*

Trade selling dapat terjadi bilamana produsen dan pedagang besar mempersilahkan pengecer untuk berusaha memperbaiki distributor produk-produk mereka. Hal ini melibatkan para penyalur dengan kegiatan promosi, peragaan, persediaan dan produk baru.

b. *Missionary Selling*

Merupakan penjualan berusaha ditingkatkan dengan mendorong pembeli untuk membeli barang-barang dari penyalur perusahaan.

c. *Technical Selling*

Yaitu berusaha meningkatkan penjualan dengan pemberian saran dan nasehat kepada pembeli akhir dari barang dan jasanya.

d. *New Business Selling*

Merupakan berusaha membuka transaksi baru dengan merubah calon pembeli menjadi pembeli.

e. *Responsive Selling*

Ialah setiap tenaga penjualan diharapkan dapat memberikan reaksi terhadap permintaan pembeli.

2.2.2 Tahap-Tahap Penjualan

a. Persiapan sebelum penjualan

Tahap pertama dalam penjualan tatap muka adalah mengadakan persiapan-persiapan sebelum melakukan penjualan. Kegiatan yang dilakukan adalah memberikan pengertian tentang barang yang dijualnya, pasar yang dituju dan teknik-teknik penjualan yang harus dilakukan.

b. Penentuan lokasi pembeli potensial

Dengan menggunakan data pembeli yang lalu maupun sekarang, penjual dapat menentukan karakteristik calon pembeli atau pembeli potensial. Penentuan calon pembeli beserta karakteristiknya dapat dilakukan dengan segmentasi pasar. Oleh karna itu, pada tahap ini ditentukan lokasi dari segmen pasar yang menjadi sasarannya. Dari lokasi ini dapatlah dibuat sebuah daftar tentang orang-orang atau perusahaan yang secara logis merupakan pembeli potensial dari produk yang ditawarkan.

c. Pendekatan pendahuluan

Sebelum melakukan penjualan, penjual harus mempelajari semua masalah tentang individu atau perusahaan yang dapat diharapkan sebagai pembelinya.

d. Melakukan penjualan

Penjualan yang dilakukan bermula dari suatu usaha untuk memikat perhatian calon konsumen, kemudian diusahakan untuk mengetahui daya tarik minat mereka. Jika minat mereka dapat diikuti dengan munculnya keinginan untuk membeli, maka penjual tinggal merealisir penjual produknya. Pada saat ini penjualan dilakukan.

e. Pelayanan purna jual

Sebenarnya kegiatan penjualan tidak berakhir pada saat pesanan dari pembeli telah dipenuhi, tetapi masih perlu dilanjutkan dengan memberikan pelayanan pada mereka.

2.3 Sistem Informasi Penjualan

Sistem informasi penjualan adalah sistem informasi yang menyangkut pengolahan data-data terkait dengan kegiatan penjualan baik dari pembelian sampai transaksi penjualan untuk mendukung kegiatan penjualan tersebut (Furqon, 2013). Sistem informasi penjualan mempertemukan kebutuhan pengolahan transaksi, mendukung operasi, bersifat manajerial, dan kegiatan strategi dari suatu sistem penjualan dan menyediakan pihak luar tertentu denganlaporan-laporan yang diperlukan. Sistem informasi penjualan merupakan suatu sistem yang berfungsi untuk mengolah data-data terkait dengan kegiatan penjualan baik dari transaksi pembelian sampai transaksi penjualan digunakan untuk mendukung kegiatan penjualan tersebut (Yulianti, 2013).

Sistem infomasi penjualan adalah sustu sistem informasi yang mengorganisasikan serangkaian prosedur dan metode yang dirancang untuk menghasilkan, menganalisa, menyebarkan dan memperoleh informasi guna mendukung pengambilan keputusan mengenai penjualan (Nore, 2013).Penjualan merupakan faktor penting dalam kemajuan dan perkembangan perusahaan, karena pendapatan yang diperoleh dari hasil penjualan digunakan untuk membiayai kelangsungan perusahaan, terlebih dalam menghasilkan keuntungan. Berdasarkan berbagai pengertian dari para ahli, sistem informasi penjualan berfungsi untuk

mengolah data-data terkait dengan penjualan dengan menggunakan serangkaian prosedur untuk mendukung kegiatan penjualan.

2.4 *System Development Life Cycle* (SDLC)

Projek merupakan usaha yang telah dirancang, dimulai dengan fase awal perencanaan dan berakhir dengan menghasilkan produk yang diinginkan. Sistem analis ditugaskan untuk memecahkan masalah bisnis. Dalam kaitannya dengan kegiatan pemecahan masalah, perlu diatur dan difokuskan pada menghasilkan tujuan. Seorang analis mencapai hasil ini dengan mengorganisir suatu proyek, sehingga akhirnya akan menghasilkan suatu sistem informasi yang dikembangkan melalui fase-fase pengembangan. *Systems Development Life Cycle* (SDLC) adalah seluruh proses ruang lingkup sistem yang dimulai pada tahap membangun (*building*), menyebarkan (*deploying*), menggunakan (*using*), dan memperbarui (*updating*) sistem informasi (Satzinger, 2014).

System Development Life Cycle (SDLC) merupakan salah satu metodologi yang digunakan untuk mengembangkan sistem informasi, SDLC adalah salah satu metode pengembangan sistem informasi yang popular pada saat sistem informasi pertama kali dikembangkan (Susanto, 2014).

Sehingga berdasarkan definisi yang telah diuraikan diatas, dapat dijelaskan bahwa SDLC merupakan metode yang dilakukan oleh analis dan programmer dalam membangun sistem informasi melalui beberapa fase bertahap mulai dari perencanaan sampai dengan implementasi. SDLC merupakan salah satu kunci konsep dasar dalam sistem informasi.

2.4.1 Pendekatan dan Fase-Fase *System Development Life Cycle* (SDLC)

Dalam lingkungan pengembangan saat ini, banyak pendekatan yang digunakan dalam pengembangan sistem informasi melalui SDLC. SDLC merupakan *framework* yang dapat mengidentifikasi semua aktivitas yang dibutuhkan seperti riset, membangun, meluncurkan, atau menjaga sistem informasi yang normalnya terdapat aktivitas seperti perancangan, analisa, desain sistem, *programming*, pengujian, dan user *training* dalam membangun sebuah system agar dapat sukses dalam meluncurkan system informasi yang baru (Satzinger, et al, 2014).

Terdapat dua pendekatan SDLC dalam lingkungan pengembangan aplikasi dari antara lain;

1. Pendekatan Prediktif (*Predictive Approach*)

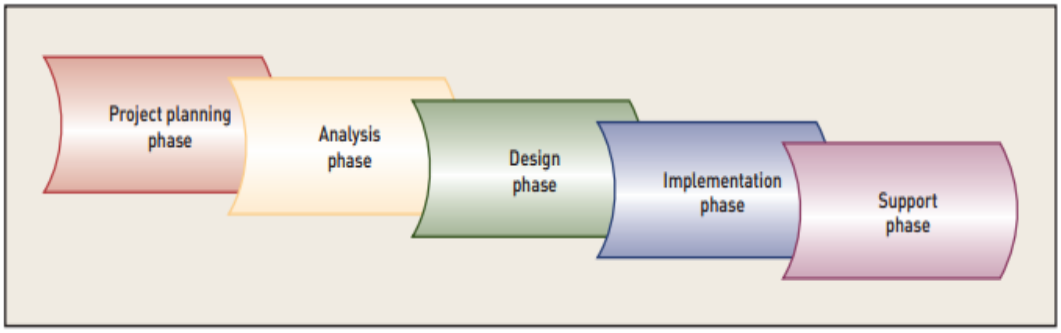
   Adalah pendekatan SDLC yang mengasumsikan bahwa pembangunan proyek dapat direncanakan dan diatur dahulu dan bahwa sistem informasi baru dapat dikembangkan sesuai dengan rencana. Metode ini berguna untuk membangun sistem yang dipahami dengan baik dan dapat diprediksikan secara jelas.

2. Pendekatan Adaptif (*Adaptive Approach*)

   Pendekatan Adaptif digunakan ketika persyaratan sistem atau kebutuhan pengguna tidak dipahami dengan baik. Dalam situasi ini, proyek tidak dapat direncanakan sepenuhnya pada awal. Beberapa persyaratan sistem mungkin belum perlu ditentukan setelah beberapa pekerjaan mulai dilakukan.

2.4.2 Fase dan Tugas dalam System Development Life Cycle (SDLC)

System Developement Life Cycle (SDLC) memiliki beberapa fase yang dibutuhkan dengan serangkaian aktivitas mulai dari fase awal hingga fase akhir. Pada umumnya SDLC memiliki 5 (lima) fase utama yaitu fase perencanaan (*Project Planning*), fase analisis (*Analysis*), fase design (*Design*), fase implementasi (*Implementation*) dan fase dukungan (*Support*) (Satzinger, et al, 2014).



Gambar 2.1 Fase-Fase SDLC (Satzinger et al, 2014)

Gambar diatas memperlihatkan fase-fase yang ada pada *system development project* dimana fase-fase tersebut memberikan masing-masing framework untuk mengatur suatu projek.

Tabel 2.1 Fase-Fase dalam SDLC (Satzinger et al, 2014)

Tahapan SDLC	Penjelasan
Project planning	Untuk mengidentifikasi ruang lingkup sistem baru, memastikan bahwa proyek ini layak dan mengembangkan jadwal, rencana sumber daya dan anggaran dari proyek
Analysis	Untuk memahami dan merincikan kebutuhan bisnis dan persyaratan pengolahan sistem baru
Design	Untuk merancang sistem yang menghasilkan solusi berdasarkan persyaratan yang ditetapkan dan keputusan yang dibuat selama analisis
Implementation	Untuk membangun, menguji dan memasang sistem informasi yang handal dengan pengguna dilatih siap untuk mendapatkan keuntungan seperti yang diharapkan dari penggunaan sistem
Support	Untuk menjaga sistem agar mampu berjalan secara produktif, baik pada awalnya dan selama bertahun-tahun hidup system

Tabel tersebut menjelaskan detail mengenai setiap fase-fase dalam SDLC dimana sebagaian besar pendekatannya menggunakan waterfall dimana fase projek mengalir kebawah secara sekuensial.

2.5 Konsep Dasar *Object-Oriented Analysis* dan *Design* (OOAD)

Desain sistem merupakan jembatan penghubung antara kebutuhan pengguna dengan pemograman sistem yang baru.Salah satu kekuatan dari pendekatan berorientasi objek adalah bahwa model desain merupakan bentuk perluasan dari persyaratan model (Satzinger, et al, 2014).

Object-Oriented Design (OOD) merupakan sebuah proses dimana sekumpulan model berbasiskan objek dibuat dan selanjutnya digunakan oleh *programmer* untuk menghasilkan suatu program untuk sistem yang baru.Tujuan dari OOD adalah untuk mengidentifikasi dan menentukan seluruh entitas yang harus bekerja sama untuk melaksanakan setiap aktivitas penggunaan dan di mana mereka berada pada simpul komputasi yang berbeda (Satzinger, et al, 2014).

2.6 Model Waterfall

Salah satu model *System Development Life Cycle* (SDLC) yang paling awal dan paling banyak digunakan adalah SDLC model air terjun (*Waterfall*). Model SDLC yang dikembangkan oleh Satzinger menggambarkan pendekatan sekuensial beberapa tahap yang biasanya disebut juga dengan model air terjun.

Gambar 2.2 *Traditional Information System Development Phases*

Dalam Model SDLC ini hal pertama yang dilakukan adalah dengan mendefinisikan perumusan permasalahan yang akan dilakukan pemecahan dari masalah yang ada *(plan)* (Satzinger, Jackson, Burd, 2014). Selanjutnya tim proyek menganalisis, mendefinisikan dan memahami secara menyeluruh masalahnya beserta kebutuhan untuk selanjutnya dicari solusi (*analysis*). Setelah masalah dipahami dan solusi ditinjau secara mendalam (*design*).

Tabel 2.1 *System Development Life Cycle Phases and Objectives*

SDLC PHASE	*OBJECTIVE*
Project Planning	Untuk mengidentifikasi ruang lingkup sistem baru.
Analysis	Untuk memahami dan mendokumentasikan secara detail kebutuhan bisnis dan pengolahan

SDLC PHASE	*OBJECTIVE*
	persyaratan sistem baru.
Design	Untuk merancang solusi sistem berdasarkan persyaratan yang ditentukan dan keputusan yang dibuat selama analisis.

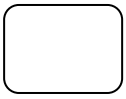
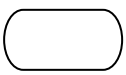
2.7 *Unifiel Modelling Leanguage (UML)*

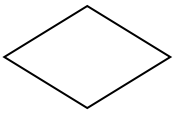
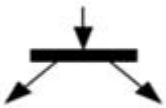
*UML*merupakan kumpulan standar model konstruksi dan notasi yang didefinisikan oleh *object management group (OMG),* sebuah organisasi standar untuk pengembangan sistem. Dengan menggunakan *UML*, analisis dan pengguna akhir dapat menggambarkan dan memahami berbagai diagram khusus yang digunakan dalam proyek pengembangan sistem (Satzinger, et al, 2014).

2.7.1 *Activity Diagram*

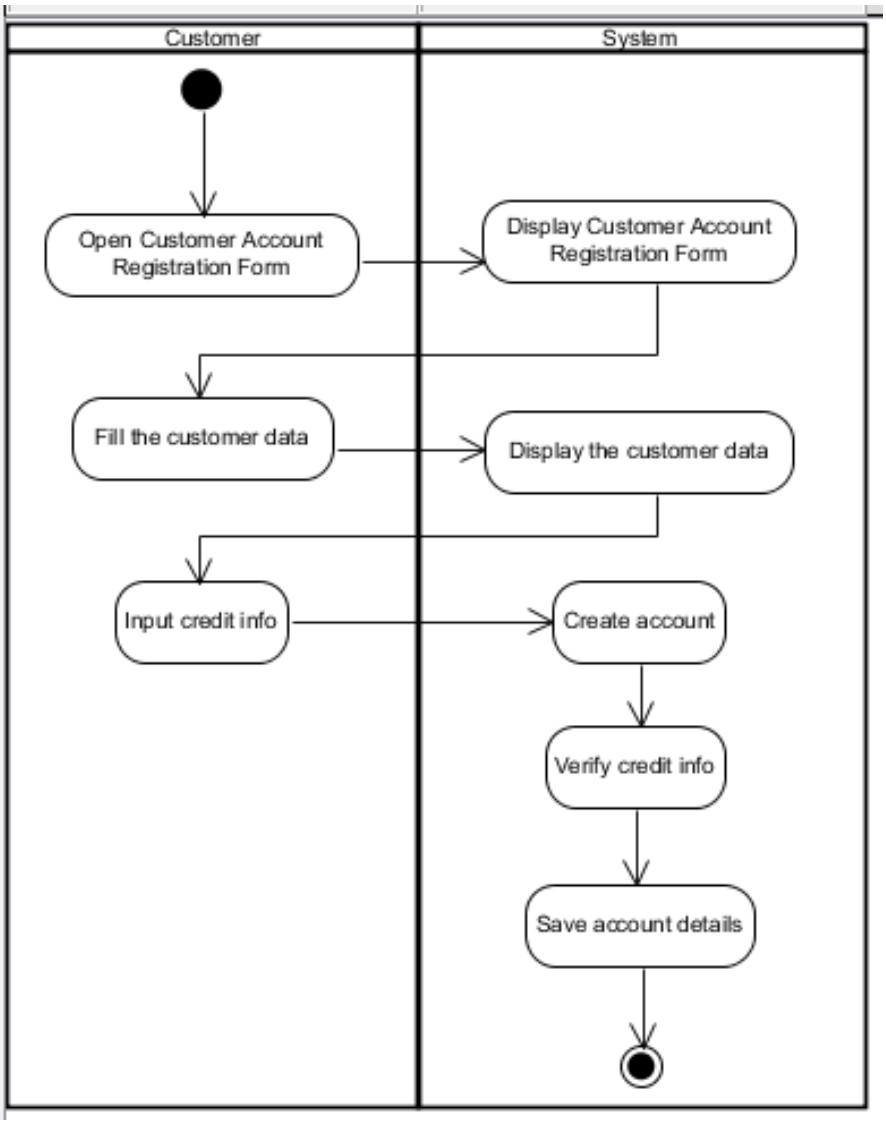
Flowcharts dirancang untuk urutan langkah-langkah pengolahan yang menanggani transaksi bisnis. Alur kerja *flowcharts* yang kompleks dapat terdiri dari puluhan atau ratusan langkah-langkah pengolahan. Banyak analisis menggunakan jenis *flowchartsdiagram* dan menyebutnya *activity diagram*(Satzinger, et al, 2014). Berikut merupakan simbol-simbol *Activity Diagram*.

Tabel 2.2 Simbol *Activity Diagram* (Satzinger et al, 2014)

No	**Simbol *Activity* Diagram**	**Nama**	**Keterangan**
1		*Activity*	Memperlihatkan bagaimana masing-masing kelas antarmuka saling berinteraksi satu sama lain
2		*Action*	State dari sistem yang mencerminkan eksekusi dari suatu aksi

No	Simbol *Activity* Diagram	Nama	Keterangan
3		*Initial Node*	Bagaimana objek dibentuk atau diawali
4		*Activity Final Node*	Bagaimana objek dibentuk dan dilakukan
5		*Decision*	Digunakan untuk menggambarkan suatu keputusan / tindakan yang harus diambil pada kondisi tertentu
6		*Action Flow*	Digunakan untuk menunjukkan arah satu simbol dengan simbol lainnya
7		*Synchronization bar (split)*	Digunakan ketika ada aktivitas yang berjalan secara parallel
8		*Synchronization bar (join)*	Digunakan untuk mempertemukan kembali kegiatan yang berjalan parallel

Suatu *activity diagram* merupakan salah satu dari *Unfield Modelling Language* (UML) diagram yang berasosiasi dengan *object-oriented approach. Work flow* merupakan *sequence* dari langkah-langkah proses dari satu bisnis transaksi atau permintaan pelanggan.Berikut merupakan contoh dari *Activity Diagram* (Satzinger, et al, 2014).



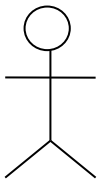
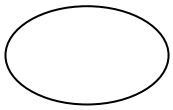
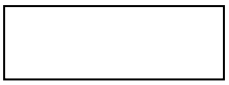
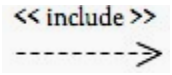
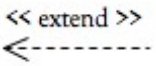
Gambar 2.3 Contoh *Activity Diagram* (Satzinger et al, 2014)

Activity diagram diatas merupakan contoh dari bagaimana seorang *customer* melakukan aktivitas pembuatan akun. Dengan menggambarkan *use case* seperti kasus diatas, fungsi yang dibangun dapat fokus pada aktivitas yang dilakukan oleh seorang aktor.

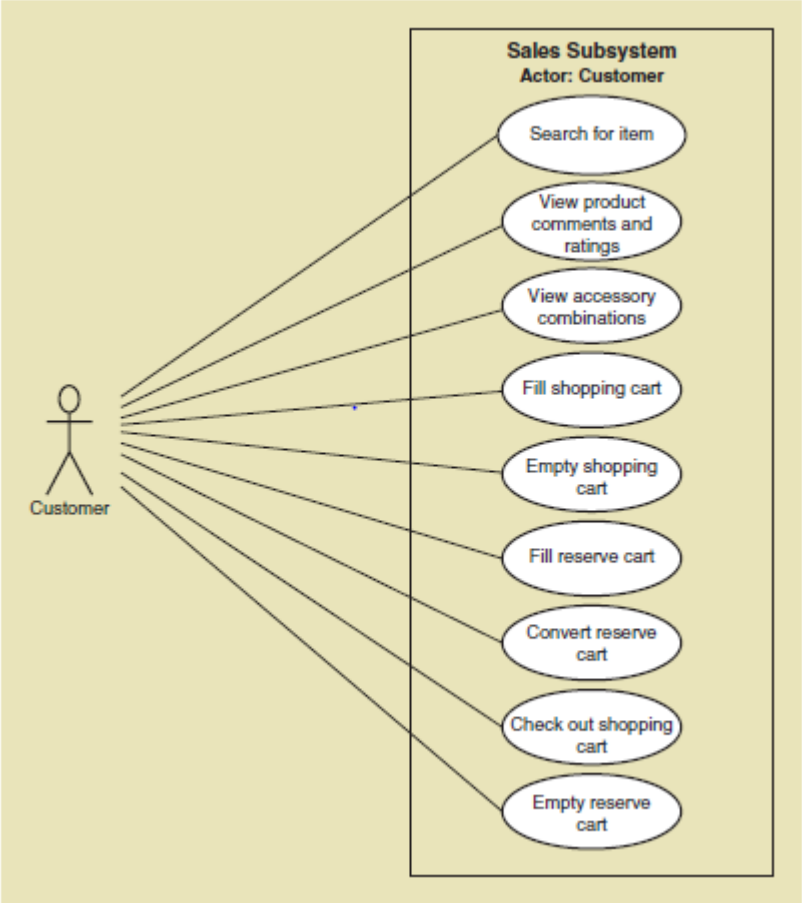
2.7.2 *Use Case Diagram*

Use Case Diagram adalah sebuah diagram yang menunjukkan berbagai peran pengguna dan cara mereka para pengguna berinteraksi dengan sistem aktor adalah orang atau hal yang benar-benar menyentuh atau berinterksi dengan sistem. Berikut merupakan beberapa simbol dalam *Use Case Diagram* (Satzinger, et al, 2014) :

Tabel 2.3 Simbol *Use Case Diagram* (Satzinger et al, 2014)

No	Gambar	Nama	Keterangan
1		*Actor*	Mempresentasikan aktor pada interaksi antara manusia dengan sistem dimana aktor merupakan manusia namun dapat juga digunakan untuk merepresentasikan perangkat keras maupun sistem eksternal
2		*Use case*	Mempresentasikan suatu pekerjaan yang dilakukan oleh aktor
3		*Association*	Mempresentasikan hubungan antara aktor dengan *use case*
4		*Boundary of System*	Mempresentasikan batasan antara *system* dari aktor yang berada di luar *system*
5		*Include*	Mempresentasikan relasi langsung antara dua *use case* ketika dibutuhkan
6		*Extend*	Mempresentasikan relasi yang biasanya berisi tambahan informasi yang mungkin dibutuhkan antara dua *use case*

Use Case Diagram menunjukan sebuah tongkat sederhana yang digunakan untuk mewakili aktor (tangan ditunjukan langsung mengakses ke sistem langsung). Kasus penggunaan sendiri dilambangkan oleh oval dengan nama *use case diagram* didalamnya. Garis yang menghubungkan aktor dengan *use case diagram* menunjukan bahwa aktor memanfaatkan penggunaan sistemnya. Pelaku juga dapat menggunakan sistem lain yang langsung menunjukan antar muka dengan sistem yang sedang dikembangkan.



Gambar 2.4 Contoh *Use Case Diagram* (Satzinger et al, 2014)

Pada *use case diagram* diatas menunjukkan bahwa aktor (*customer*) dapat mengakses *subsystem* penjualan dimana aktor dapat melakukan beberapa kasus seperti mencari barang, mengisi daftar pembelian, dan mengosongkan daftar pembelian.

2.7.3 *Use Case Description*

Use case description merupakan penjelasan terperinci mengenai proses dari suatu *use case* atau bisa disebut juga sebagai daftar kasus pengguaan diagram *use case* yang memberikan gambaran dari semua penggunaankasus untuk sistem. Informasi rinci tentang setiap kasus penggunaan digambarkan dengan menggunakan deskripsi kasus. Berikut ini merupakan jenis-jenis dari *use case description* (Satzinger, et al, 2014) :

- *Brief Use Case Description*

Merupakan deskripsi singkat dari sebuah *use case*, yang mendefinisikan gambaran umum aktivitas-aktivitas yang terjadi dari jalannya sebuah *use case*.

Use case	Brief use case description
Create customer account	User/actor enters new customer account data, and the system assigns account number, creates a customer record, and creates an account record.
Look up customer	User/actor enters customer account number, and the system retrieves and displays customer and account data.
Process account adjustment	User/actor enters order number, and the system retrieves customer and order data; actor enters adjustment amount, and the system creates a transaction record for the adjustment.

Gambar 2.5 Contoh *Brief Use Case Description* (Satzinger et al, 2014)

Brief use case description diatas menggambarkan bagaimana sebuah aktivitas umum pembuatan, pencarian, dan proses penyesuaian terhadap suatu *account*.

- *Fully Developed Use Case Description*

Merupakan deskripsi detail dari sebuah *use case* yang menjelaskan tentang (Satzinger et al, 2014) :

a) *Use case name*, harus diisi dengan nama dari use case yang akan dijelaskan di *use case description.*
b) *Scenario*, merupakan nama skenario dari *use case* yang akan dijelaskan. Pada umumnya, nama skenario dapat disamakan dengan nama dari *use case.*
c) *Triggering event*, merupakan *event* yang memicu terjadinya *use case*. Dapat dilihat dari referensi untuk *event* ini dari *event table*.
d) *Brief description*, merupakan ringkasan singkat secara umum dari *use case* yang akan dijelaskan.
e) *Actors*, merupakan para pelaku atau pengguna sistem yang terkait dengan jalannya sebuah *use case*.
f) *Related use case*, merupakan *use case* lain yang terkait atau akan dijalankan apabila *use case*, yang akan dibahas dan dieksekusi.
g) *Stakeholders*, merupakan pihak-pihak yang berkepentingan terkait dengan jalannya *use case* yang akan dijalankan.

h) *Pre conditions,* merupakan kondisi awal yang harus terpenuhi agar eksekusi dari *use case*, yang akan dijelaskan ini dapat terlaksana.

i) *Post conditions,* merupakan kondisi yang terjadi setelah *use case* selesai dieksekusi.

j) *Flow of activities,* mendeskripsikan rincian aliran aktivitas-aktivitas yang terlaksana dari *use case* yang akan dijelaskan. Selain aktivitas-aktivitas tersebut akan digambarkan secara berurutan, *flow of activities* juga akan menggambarkan interaksi aktivitas antara yang dilakukan oleh *actor* dengan sistem.

k) *Exception condition,* mendeskripsikan aktivitas-aktivitas khusus yang terjadi apabila suatu kondisi terpenuhi pada saat eksekusi dari sebuah *use case*.

Use case name:	*Create customer account.*	
Scenario:	Create online customer account.	
Triggering event:	New customer wants to set up account online.	
Brief description:	Online customer creates customer account by entering basic information and then following up with one or more addresses and a credit or debit card.	
Actors:	Customer.	
Related use cases:	Might be invoked by the *Check out shopping cart* use case.	
Stakeholders:	Accounting, Marketing, Sales.	
Preconditions:	Customer Account subsystem must be available. Credit/debit authorization services must be available.	
Postconditions:	Customer must be created and saved. One or more Addresses must be created and saved. Credit/debit card information must be validated. Account must be created and saved. Address and Account must be associated with Customer.	
Flow of activities:	**Actor**	**System**
	1. Customer indicates desire to create customer account and enters basic customer information. 2. Customer enters one or more addresses. 3. Customer enters credit/debit card information.	1.1 System creates a new customer. 1.2 System prompts for customer addresses. 2.1 System creates addresses. 2.2 System prompts for credit/debit card. 3.1 System creates account. 3.2 System verifies authorization for credit/debit card. 3.3 System associates customer, address, and account. 3.4 System returns valid customer account details.
Exception conditions:	1.1 Basic customer data are incomplete. 2.1 The address isn't valid. 3.2 Credit/debit information isn't valid.	

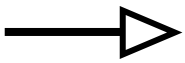
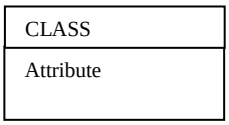
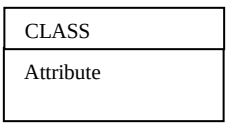
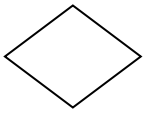
Gambar 2.6 Contoh *Fully Developed Use Case Description* (Satzinger et al, 2014)

Contoh diatas merupakan *fully developed use case description* ketika terdapat *use case*membuat *customer account* (Satzinger et al, 2014). *Fully developed use case description* dapat digunakan sebagai template dalam mendokumentasikan setiap *use case* yang ada. Kolom *scenario* berisikan informasi kejadian *use case* yang melibatkan aktor.

2.7.4 *Domain Model Class Diagram*

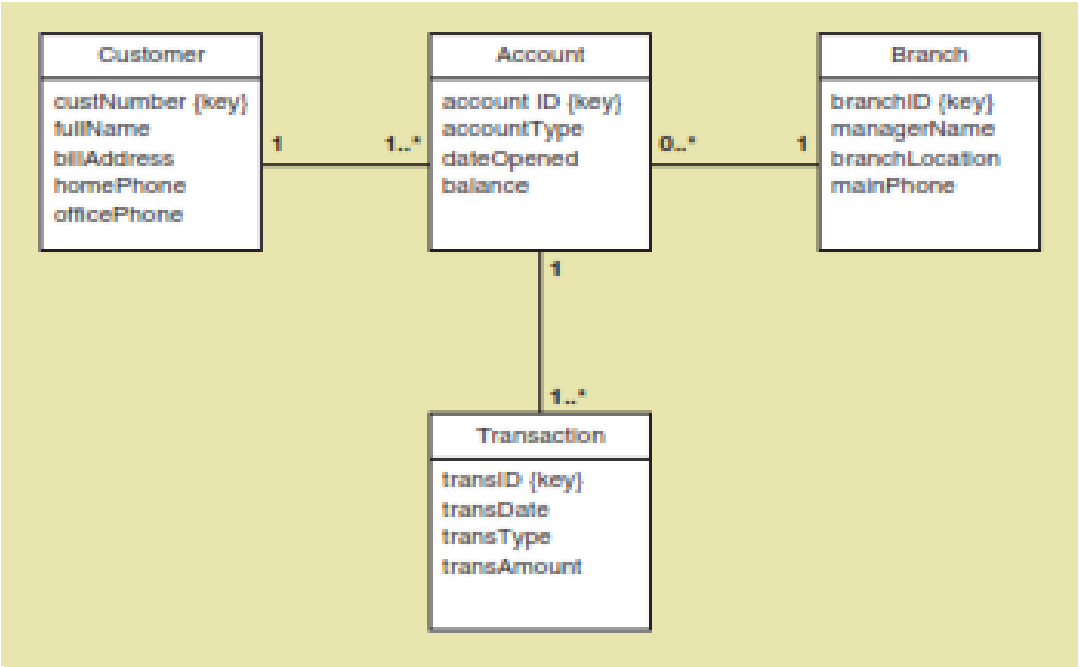
Domain Model Class Diagram adalah sebuah diagram yang terdiri dari *class, object,* serta relasi hubungan antar *class.* Dalam konteks *class diagram*, *things* dapat diasosiasikan sebagai objek-objek yang berkepentingan dan saling berhubungan di dalam eksekusi transaksi yang berjalan didalam proses bisnis. Objek-objek yang mempunyai karakteristik sama dikelompokkan kedalam suatu *class*. *Class* yang saling berhubungan dimodelkan di dalam *class diagram*. *Cardinality* didalam *class diagram* dikenal sebagai *multiplicity*. (Satzinger, et al, 2014)

Tabel 2.4 Simbol *Domain Model Class Diagram* (Satzinger et al, 2014)

No	Gambar	Nama	Keterangan
1		*Generalization*	Hubungan dimana objek anak (*descendent*) berbagi perilaku dan struktur data dari objek yang ada diatasnya objek induk (*ancestor*)
2		*Class*	Merupakan himpunan dari beberapa objek yang memiliki kesama atribut dan operasi
		*Attribute*	Attribute dalam class yang mempresentasikan karakter dari suatu class
3		*N-Ary Association*	Upaya untuk menghindari asosiasi dengan lebih dari dua objek
4	 *One to many* (1 – 1…*) *One to one* (1 – 1) *Zero to one* (0 – 1) *Zero to many* (0 – 1…*)	*Multiplicity*	Pernyataan yang menyebutkan banyaknya jumlah objek yang dapat terlibat dalam hubungan antara class

No	Gambar	Nama	Keterangan
5		*Agregation*	Merupakan kasus khusus dimana hubungan antara class bermakna "terdiri dari"
6		*Association*	Merupakan tanda yang menunjukkan relasi antara dua class, dapat berisi kata kerja yang mewakili hubungan antara dua class namun tidak wajib

Dari tabel simbol *domain model class diagram* diatas dapat dibuat sebuah contoh sebagai berikut;

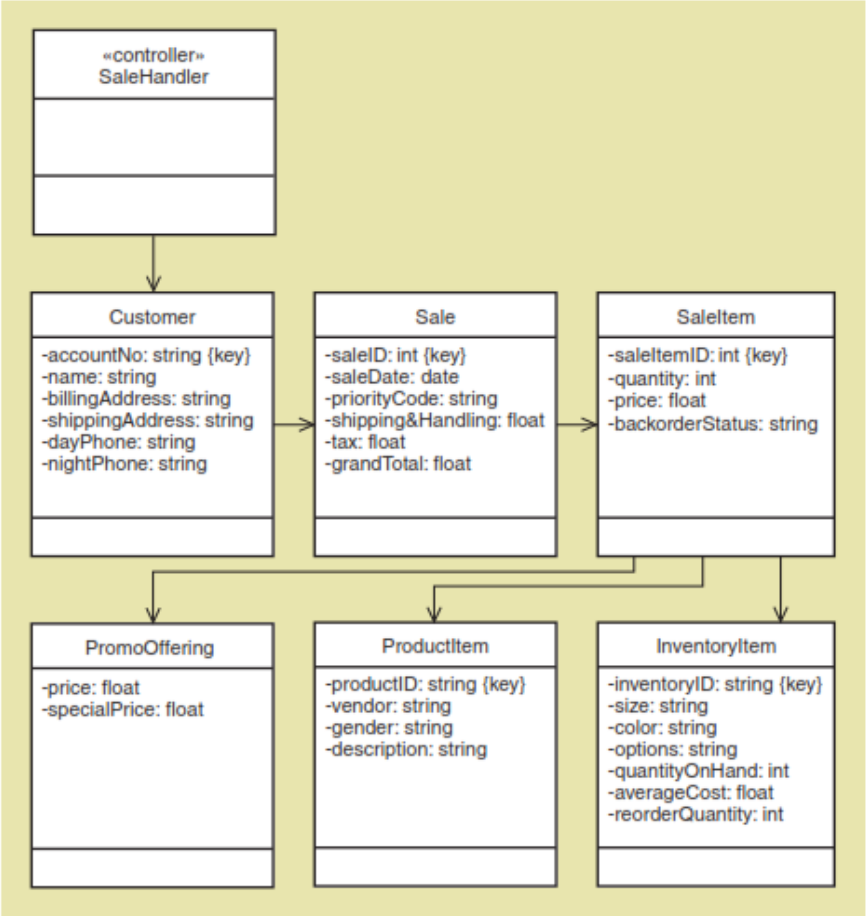


Gambar 2.7 Contoh *Domain Model Class Diagram* (Satzinger et al, 2014)

Contoh *domain model class diagram*diatas menggambarkan 4 class *customer*, *account*, *branch*, dan *transaction* yang memiliki atribut masing-masing. Dari *class customers*, *keycustomer* dapat memiliki banyak *account id* pada *class account*.*Domain model class diagram* kemudian dilanjutkan dengan desain *First Cut DesignClass Diagram* dan *UpdatedDomain Class Diagram.*

a. *First-Cut Design Class Diagram*

Desain *first-cut domain class diagram* adalah sebuah class yang menjelaskan mengenai alur data beserta tipe datanya. Hal itu memerlukan dua tahapan yaitu mengkolaborasikan atribut jenis dan informasi nilai awal kemudian menambahkan navigasi visibilitas panah. Berikut merupakan contoh dari *first cut domain class diagram.*

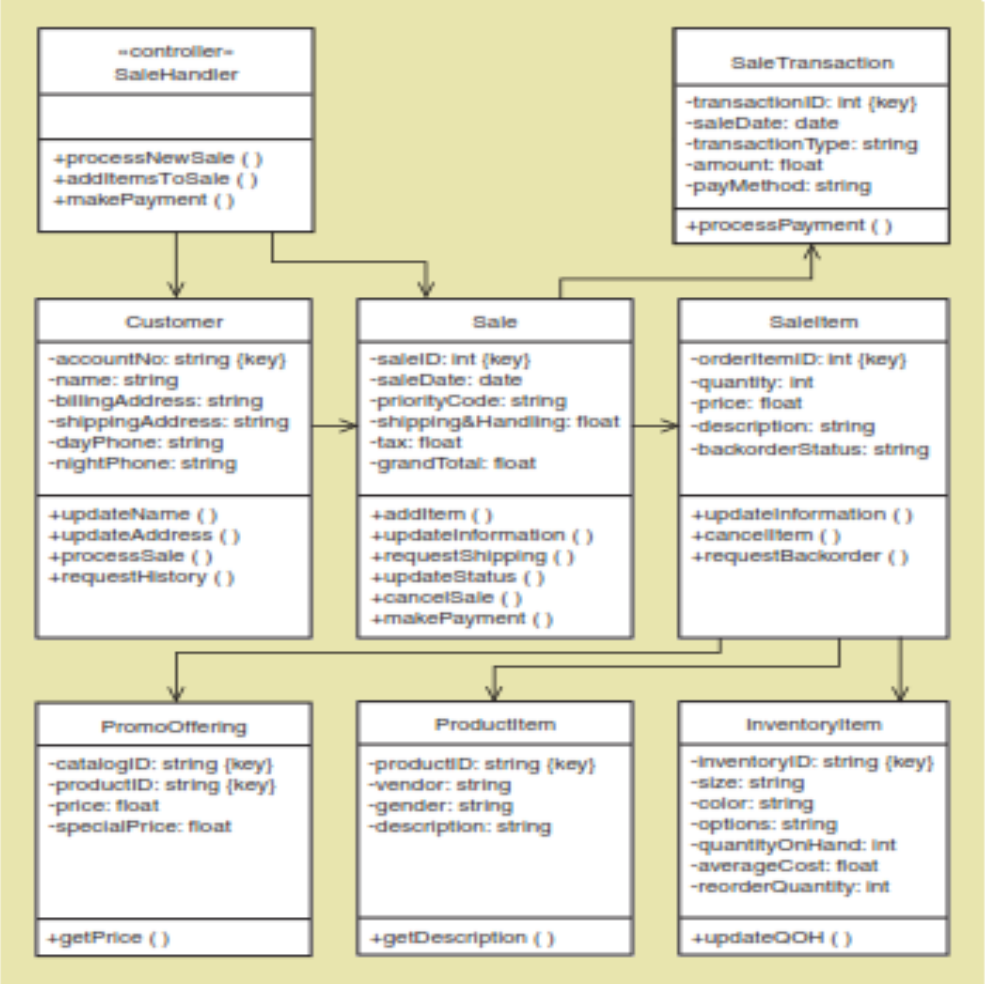


Gambar 2.8 Contoh *First-Cut Design Class Diagram* (Satzinger et al, 2014)

Pada *first-cut design class diagram* diatas memberikan informasi pada *atributt* yang ada pada setiap *class*. Anak panah *navigation visibility Customer class* memberikan data kepada *Sales class* merupakan*dependent class.* Hubungan assosiasi *one-to-many*yang dimana arah panah akan diarahkan dari *superior class order*ke *subordinateSaleItem.*

b. *Updated Design Class Diagram*

Merupakan teknik pengidentifikasian objek-objek pada kata benda yang terdapat pada daftar *requirement* yang diklasifikasikan pada area (*domain*) permasalahan yang sama untuk dijadikan *candidate class* pada *class diagram.*



Gambar 2.9 Contoh *Updated Design Class Diagram* (Satzinger et al, 2014)

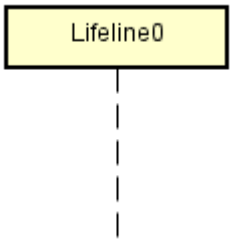
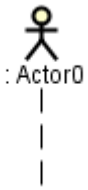
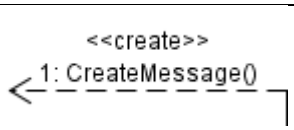
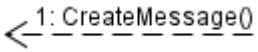
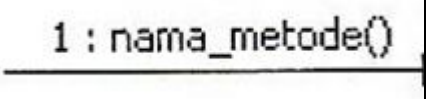
Contoh diatas merupakan*Updated design class diagram* untuk penjualan *handphone*. Terdapat beberapa *class* yaitu *customer, sales, salesitem, promooffering, productitem,* dan *inventoryitem*. Untuk *controller class* yaitu *saleshandler* diletakkan di paling atas atau pertama. Sedangkan visibilitas berupa tanda panah menunjukkan bahwa satu kelas menyimpan nilai yang mereferensikan ke *class* lain.

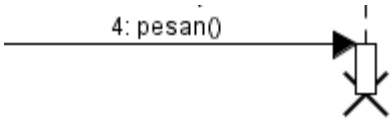
2.7.5 *System Sequence Diagram*

System sequence diagram merupakan diagram yang menunjukkan urutan pesan antara aktor eksternal dan internal sistem di dalam *use case* atau

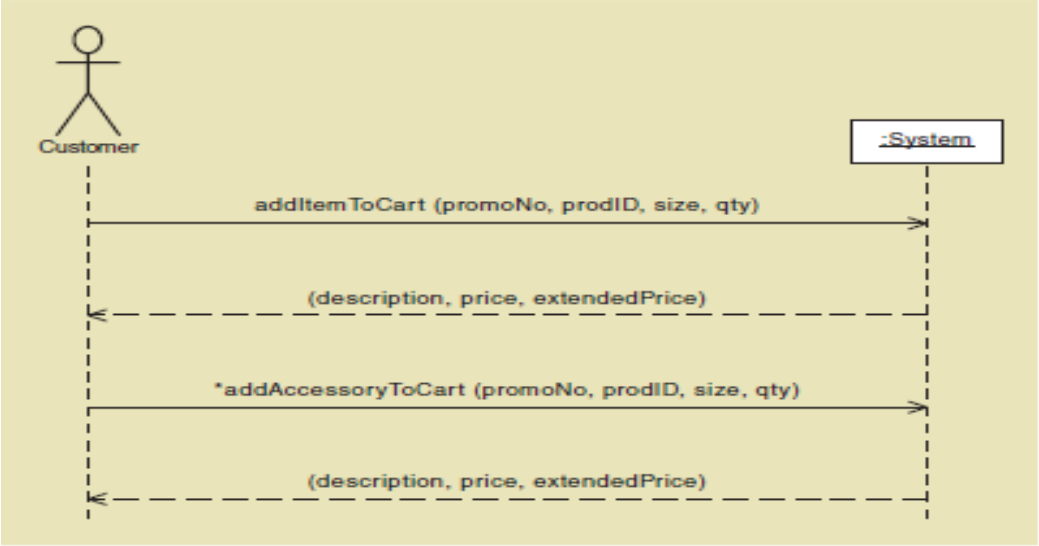
scenario yang sudah dirancang sebelumnya (Satzinger, et al, 2014). Berikut merupakan simbol-simbol *sequence diagram.*

Tabel 2.5 Simbol *System Sequence Diagram* (Satzinger et al, 2014)

No	Gambar	Nama	Keterangan
1		*Object*	Merupakan instance dari sebuah class dan dituliskan secara horizontal, digambarkan sebagai sebuah class dengan nama objek didalamnya.
2		*Actor*	Actor dapat berkomunikasi dengan objek, actor dapat diurutkan sebagai kolom
3		*Create message*	Menyatakan suatu objek membuat objek lain.
4		*Return message*	Menyatakan suatu objek yang telah menjalankan suatu operasi atau *method* menghasilkan suatu output ke objek
5		*Call message*	Menyatakan suatu objek yang memanggil method yang ada pada objek lain atau memanggil dirinya sendiri.

No	Gambar	Nama	Keterangan
6		*Destroy message*	Menyatakan bahwa suatu objek mengakhiri hidup objek lain. Atau menghentikan alur interaksi
7		*Activation*	Mengindikasikan sebuah objek yang akan melakukan aksi

Systemsequence diagram mengidentifikasi interaksi antara aktor dan sistem yang termasuk dalam *interaction diagram*, yaitu menunjukkan interaksi antar objek. Sistem *sequence diagram* dibuat setelah *activity diagram* selesai, agar memudahkan ketika mengidentifikasi kapan terjaadinya *input* dan *output*. *Input* dan *output* terjadi ketika sebuah panah dari *activity diagram* datang atau keluar dari atau ke *external actor* menuju *computer* sistem. Berikut ini merupakan bagian dari sistem *sequence diagram* : (Satzinger et al, 2014)

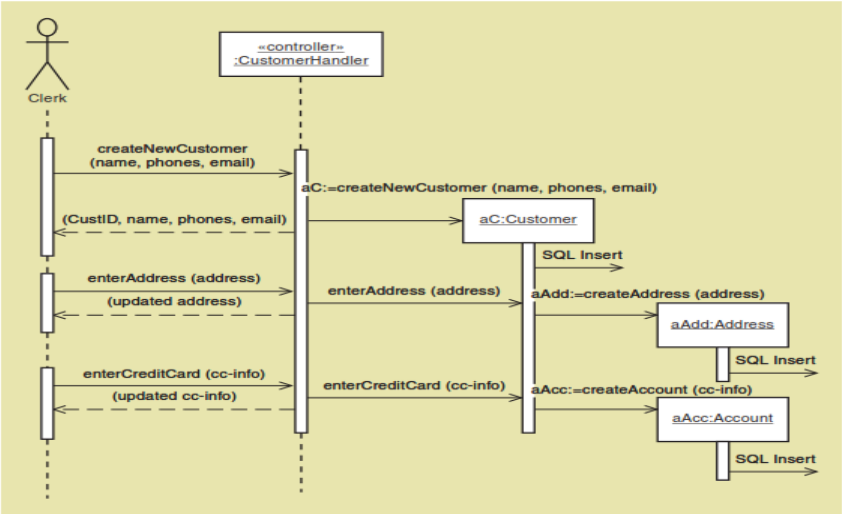


Gambar 2.10 Contoh *System Sequence Diagram* (Satzinger et al, 2014)

Pada contoh gambar *system sequence diagram* diatas merupakan sebuah diagram yang menunjukkan eksekusi *operation* pada sebuah objek yang melibatkan pemanggilan *operation* di objek lain.

a. First Cut Sequence Diagram

First Cut Sequence diagram digunakan untuk mengidentifikasi semua *class domain* dan diperlukan pesan internal antar *actor* dengan *classdomain* kemudian ada penambahan pada *view layer* dan *data access layer*. Ketika mengidentifikasi dan menciptakan suatu pesan, pertama harus menentukan asal dan tujuan objek tersebut dimana setiap objek akan diperlukan untuk memulai pesan. Setiap pesan harus mencerminkan *request* yang dikirim (Satzinger, Jackson & Burd, 2014). Berikut merupakan contoh *First Cut Sequence Diagram* berupa pembuatan pesanan penjualan.

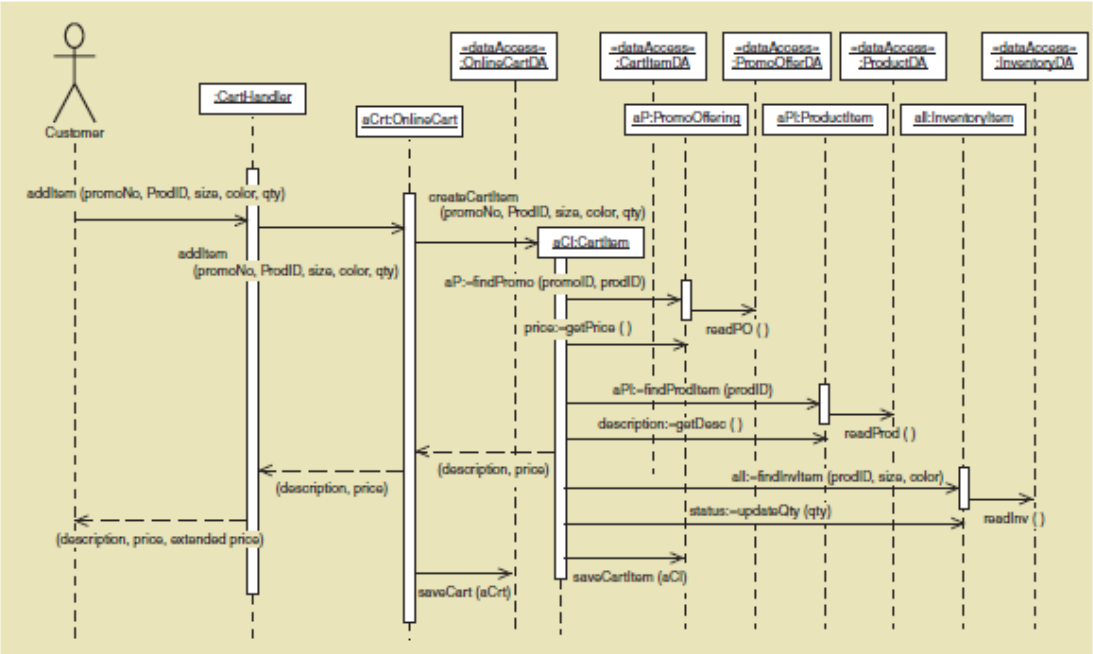


Gambar 2.11 Contoh *First Cut Sequence Diagram* (Satzinger et al, 2014)

Aktor dari contoh gambar diatas adalah member yang mengisi formulir pemesanan penjualan. Pada *first cut sequence diagram* memiliki *use case controller object* atau biasa disebut dengan *handler* untuk mengendalikan dan memasukkan pesan yang telah diberikan oleh *user*.

b. Data Access Layer Design

Data access layer design sistem *sequence diagram* menggambarkan *sequence diagram* dengan menambahkan SQL *statements* atau biasa disebut dengan *data access classes*. Tahapan dalam membuat *dataaccess layer* yaitu dengan menentukan *domain object* dengan data yang diperlukan dari *database*, kemudian melakukan *query database* dan mengirim *object reference* dan melakukan proses pengembalian informasi dalam *referensi object* (Satzinger, et al, 2014). Berikut merupakan contoh *data access layerdesign*.

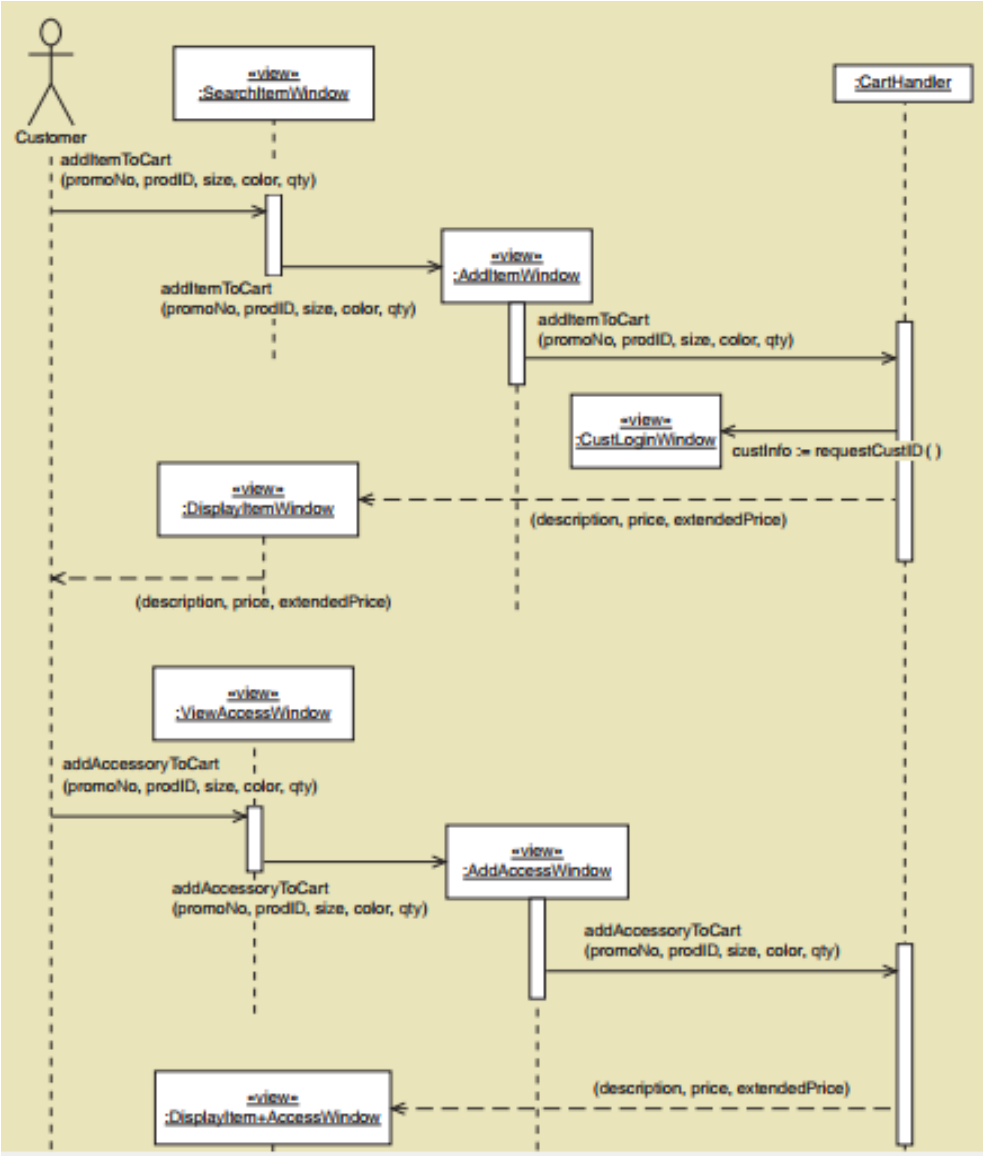


Gambar 2.12 Contoh *Data Access Layer Design* (Satzinger et al, 2014)

Gambar diatas menunjukkan bahwa setiap objek yang mengirim pesan ke objek lain harus memiliki visibilitas navigasi ke objek tersebut.

2.7.6 *View Layer*

View Layer merupakan suatu bagian dari *three-layer architecture* yang memiliki isi yaitu *user interface* dan berfungsi untuk menerima input user, dilanjutkan dengan memformatnya lalu menampilkan hasil proses (Satzinger,et al, 2014).



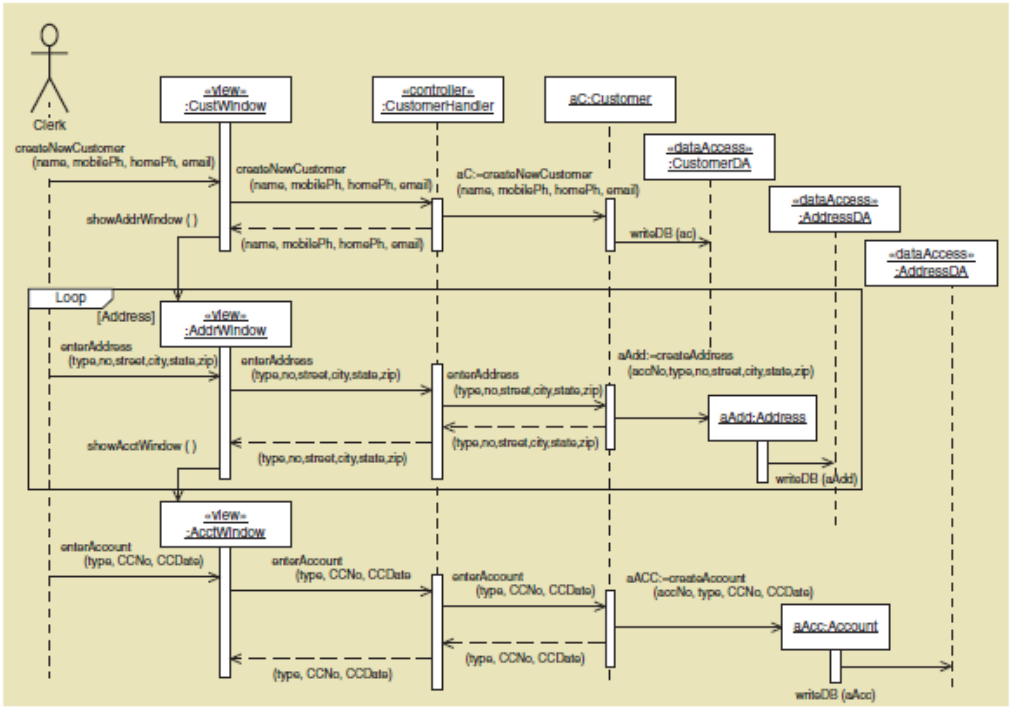
Gambar 2.13 Contoh *View Layer* (Satzinger et al, 2014)

Pada contoh gambar *view layer* diatas *class window* yang berorientasi objek dapat didefinisikan untuk setiap *form*. Setiap *message* dari *actor*eksternal juga dimasukkan ke dalam sistem dengan beberapa cara, dan *message output* harus ditampilkan. Salah satu cara yaitu melalui *class form* yang berbentuk *window* yang dapat menerima data *input* dan bila memungkinkan dapat menampilkan data keluaran.

2.7.7 *Multilayer Diagram*

*Multilayer diagram*adalah proses lanjutan dari *First-cut sequence diagram*. Terdapat dua point penting dalam *Multilayer diagram* yaitu menambahkan *View Layer* yang berfungsi sebagai *user interface*yang akan

menghantarkan messages ke *controller*dan menambahkan *Data Access layer* sebagai tempat penyimpanan akhir. Berikut ini merupakan contoh *multilayer diagram* berupa pembuatan bukti pengambilan barang berdasarkan bukti pembayaran.

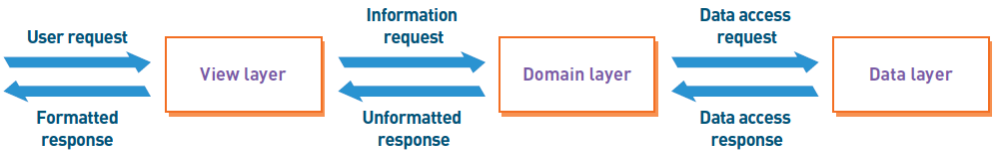


Gambar 2.14 Contoh *Multilayer Diagram* (Satzinger et al, 2014)

Pada contoh gambar diatas, saat bukti pengambilan barang membutuhkan data dari bukti pembayaran maka harus dipastikan bukti pembayaran harus ada terlebih dahulu sebelum bukti pengambilan barang bisa dibuat. Untuk membuka akses dan membaca ke bukti pembayaran digunakan method intitialization atau disingkat init() dan read().

2.7.8 *Threelayer Diagram*

Penggunaan*design class diagram*, *interaction diagram*, dan *package diagram* adalah seorang pemrogram yang dapat memulai membangun sebuah sistem. Dengan begitu penerapan sistem dalam pengertian ini membangun sistem dengan bahasa pemrograman seperti *java, php,* atau seperti *visual studio* (Satzinger et al, 2014). *Three layer architecture* digunakan untuk semua jenis sistem, termasuk aplikasi desktop dan aplikasi berbasis web, yang dibagi menjadi tiga *layer,* diaantaranya yaitu :



Gambar 2.15 Contoh *Three layer* (Satzinger et al, 2014)

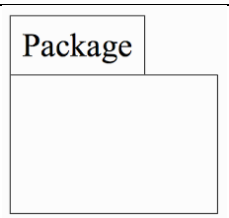
1. *User interface* atau *view layer*, yaitu menerima *input* dari pengguna dan format serta hasil tampilan pemrosesan dari *inputan* pengguna.
2. *Business logic* atau *domain layer*, yaitu implementasi aturan dan prosedur dari proses bisnis.
3. *Data layer*, yaitu mengelola penyimpanan data, biasanya satu atau lebih *database*.

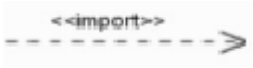
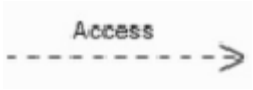
2.7.9 *Package Diagram*

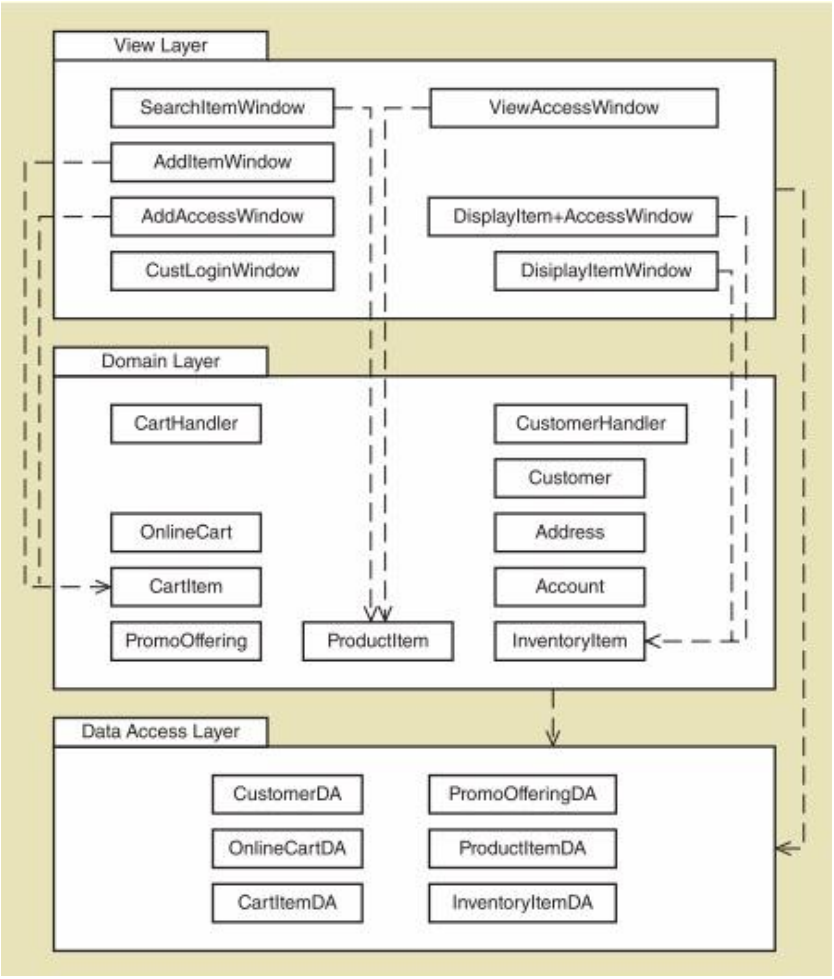
Package diagram merupakan suatu *UML diagram* yang mengizinkan perancangan untuk mengasosiasikan beberapa *class* dari grup yang saling berkaitan. Salah satu pilihan adalah dengan memisahkan *view layer*, *domain layer* dan *data access layer* ke dalam *package* yang berbeda.(Satzinger, et al, 2014)

Simbol lain yang digunakan di dalam *package diagram* adalah *dashed arrow*, dimana menggambarkan *dependency relationship*, artinya hubungan antara elemen di dalam *package diagram, class diagram* dan *interaction diagram* yang mengindikasikan dimana suatu elemen mempengaruhi elemen lainnya, sehingga perancang dapat menentukan efek perubahan yang terjadi. Kesimpulannya, *package diagram* digunakan untuk menunjukkan komponen yang saling berkaitan.

Tabel 2.6 Simbol *Package Diagram* (Satzinger et al, 2014)

No	Gambar	Nama	Keterangan
1		*Package*	Merupakan sebuah elemen atau lebih dari kelas

No	Gambar	Nama	Keterangan
2		*Import*	Suatu dependency yang mengindikasikan isi tujuan paket secara umum yang ditambahkan kedalam sumber paket
3		*Access*	Suatu dependency yang mengindikasikan isi tujuan paket secara umum yang digunakan pada nama sumber paket



Gambar 2.16 Contoh *Package Diagram* (Satzinger et al, 2014)

Diagram ini berguna untuk mendokumentasikan perbedaan atau kesamaan dalam hubungan objek pada *three layer*.tanda panah putus-putus merupakan hubungan ketergantungan seperti pada *view layer* dan *data access layer*. Ekor panah tersambung ke paket yang tergantung, dan panah dihubungkan ke *package independent*.ketergantungan hubungan digunakan pada *package diagram, class diagram,* dan *interaction diagram.*

2.7.10 *Persistent Class*

*Persistent class*es adalah nilai data yang harus disimpan oleh sistem walaupun ketika aplikasi tidak dieksekusi. Tujuan sebenarnya dari *relational database* untuk meningkatkan kemampuan dalam menghadapi masalah *domain object persistent* (Satzinger, et al, 2014). Berikut contoh *persistent object* katalok produk.

CatalogProduct : Table

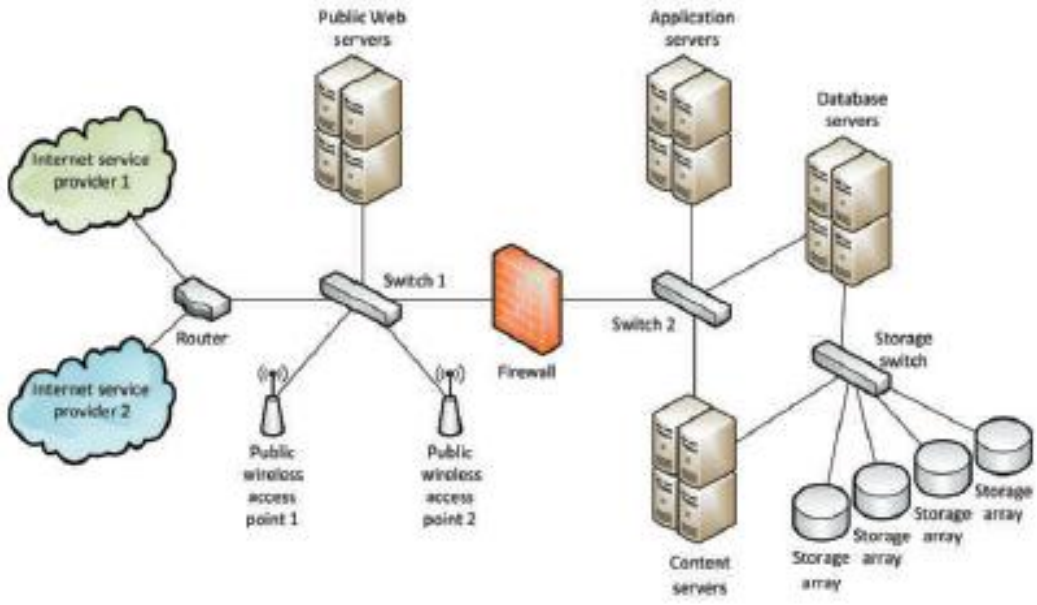
CatalogID	ProductID	Price	SpecialPrice
23	1244	$15.00	$12.00
23	1245	$15.00	$12.00
23	1246	$15.00	$13.00
23	1247	$15.00	$13.00
23	1248	$14.00	$11.20
23	1249	$14.00	$11.20
23	1252	$21.00	$16.80
23	1253	$21.00	$16.40
23	1254	$24.00	$19.20
23	1257	$19.00	$15.20

Record: 11 of 11

Gambar 2.17 Contoh *Persistent Classes* (Satzinger et al, 2014)

2.8 *Network Diagram*

Network Diagram menunjukkan bagaimana lokasi dan komponen *hardware* saling berhubungan dengan *network devices* dan *wiring*.ada banyak jenis *network diagram*, masing-masing menekankan berbagai aspek seperti *network, connected, hardware resources,* dan *user* (Satzinger et al, 2014).

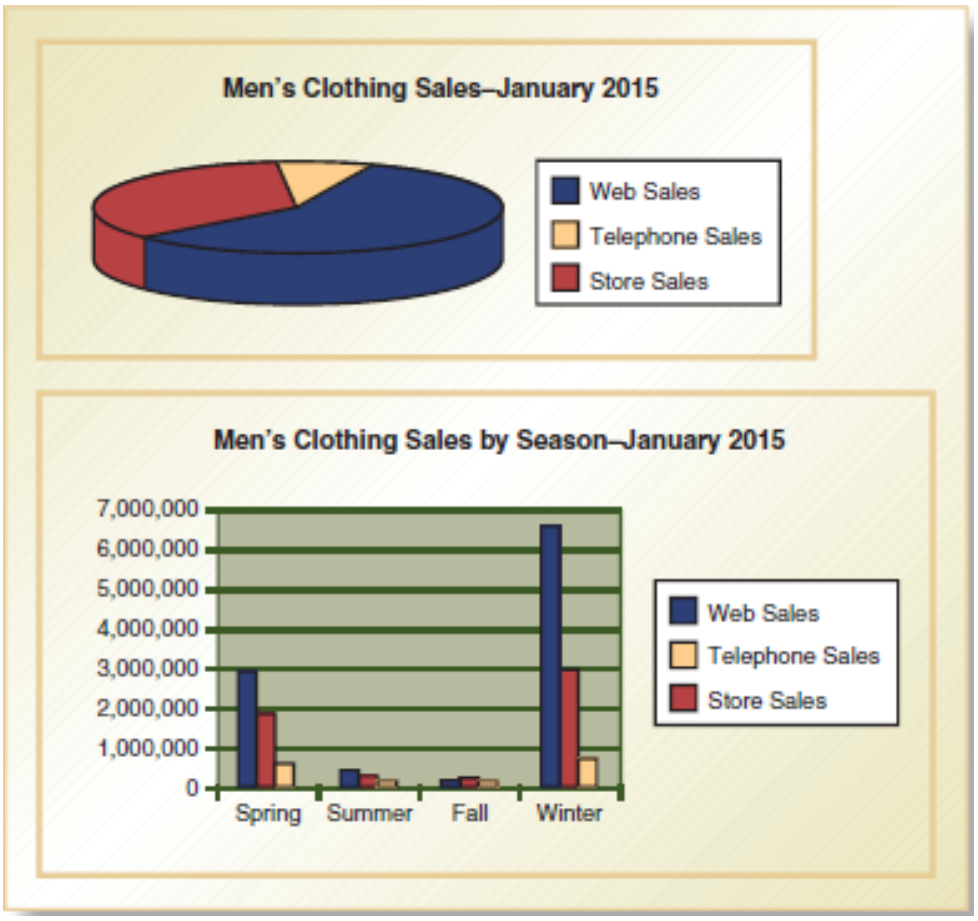


Gambar 2.18 Contoh *Network Diagram* (Satzinger et al, 2014)

Contoh gambar diatas menunjukkan *network diagram* lain yang berfokus pada *network connection* dan *network hardware*. Diagram jenis ini biasanya digunakan untuk menggambarkan *network connection* di gedung atau didalam ruang server.

2.9 *System Interface*

System Interface adalah *input* dan *output* yang membutuhkan campur tangan manusia. *Input* ditangkap secara otomatis oleh perangkat *input* khusus seperti pemindai, pesan elektronik ke atau dari sistem lain, atau transaksi yang diambil oleh sistem lain. *Output* dianggap sebagai antarmuka sistem jika mereka mengirim pesan atau informasi ke sistem lain (Satzinger, et al, 2014).



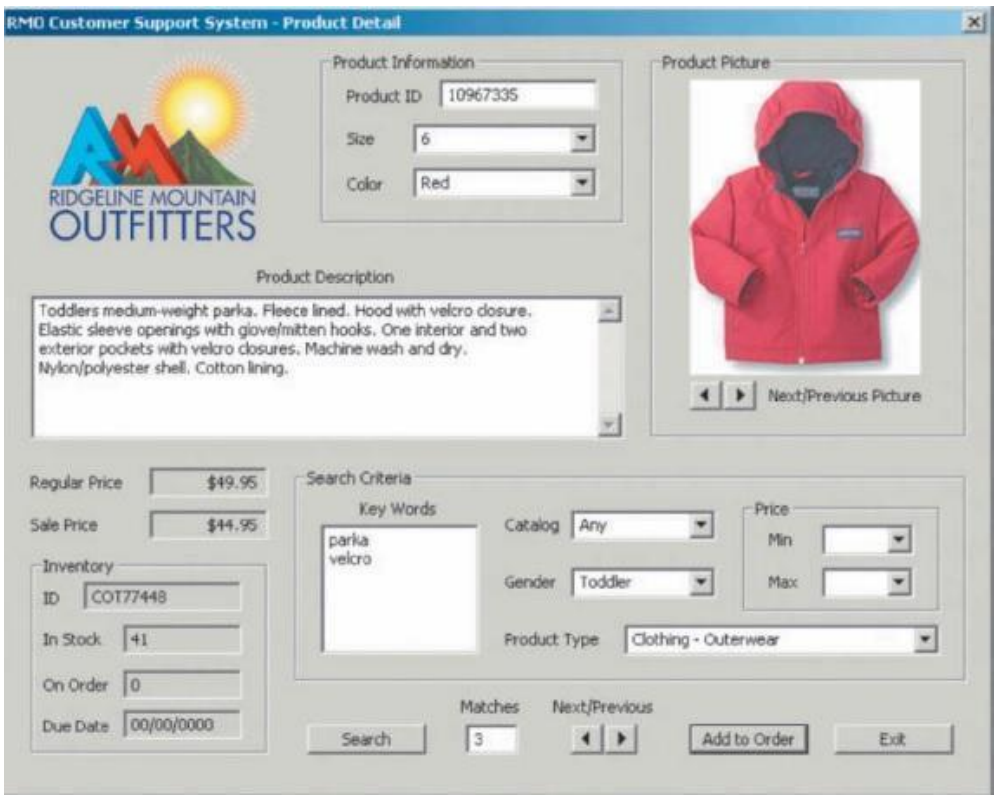
Gambar 2.19 Contoh *System Interface* (Satzinger et al, 2014)

2.10 *User Interface*

User interface merupakan *input* dan *output* yang lebih melibatkan pengguna sistem secara langsung. *User interface* dapat digunakan oleh pengguna *internal* maupun *eksternal*. Ada tiga aspek penting di dalam perancangan *user interface*, yaitu sebagai berikut : (Satzinger, et al, 2014)

- *Physical aspects*, terdiri dari perangkat yang di sentuh oleh *user*, termasuk *desk*, *chair*, *light*, *keyboard*, *mouse*, *touch screen* atau *keypad.*
- *Perceptual aspects*, terdiri dari semua hal yang dilihat, didengar atau disentuh oleh pengguna akhir (di luar *physical devices*), termasuk segala hal yang terdapat di layar monitor.
- *Conceptual aspects*, terdiri dari semua hal yang diketahui oleh pengguna di dalam menggunakan sistem, seperti *customers*, *partners*, *orders*, *shipment* dan *feedback.*

- *User-centered design* merupakan koleksi teknik yang meletakkan pengguna di tengah-tengah proses pengembangan *user interface*. Ada tiga prinsip penting *user-centered design,* yaitu sebagai berikut :
  a. Fokus awal pada pengguna dan pekerjaan mereka.
  b. Evaluasi desain untuk memastikan kegunaan.
  c. Menggunakan pengembangan yang berulang.



Gambar 2.20 Contoh *User Interface* (Satzinger et al, 2014)

Dalam merancang sebuah *User Interface* tentu tidak lepas dari kaidah-kaidah penulisan serta aturan yang baik untuk menghasilkan *User Interface* yang baik pula. Delapan Aturan Emas (*Eight golden rules*) merupakan delapan aturan untuk merancang layar antarmuka yang interaktif dan mendukung fungsi kegunaan. Berikut merupakan penjelasan delapan aturan emas sebagai berikut:

1. *Affordance and Visibility*

   Penampilan fungsi menu–menu harus jelas dan kelihatan oleh pengguna sistem akhir dan dapat digunakan secara maksimal fungsi sistem tersebut.

2. *Consistency*

   Merancang konsistensi penampilan dan antarmuka yang fungsional merupakan salah satu tujuan perancangan yang sangat penting. Pengaturan informasi yang diatur di dalam form, nama, serta pengaturan item–item

menu, ukuran dan bentuk ikon–ikon serta alur dari sistem harus konsistendan diketahui secara spesifik fungsi dari sistem secara jelas yang nantinya akan digunakan oleh pengguna sistem akhir.

3. *Shortcut*

Umumnya pengguna yang sudah sering menggunakan aplikasi lebih menginginkan kecepatan dalam mengakses informasi yang diinginkan. Jadi tingkat interaksi yang diminta lebih pendek / singkat dan langsung menunjuk fungsi tersebut tanpa melewati alur menu yang panjang dan kotak dialog yang g*and*a. *Shortut keys* berfungsi untuk mengurangi jumlah interaksi pengguna sistem dengan sistem untuk meringankan tugas pengguna sistem.

4. *Feedback*

Umpan balik harus diberikan untuk memberikan informasi kepada pengguna sesuai dengan aksi yang dilakukannya. Pengguna akan mengetahui aksi apa yang telah dan akan dilakukan dengan adanya umpan balik. Umpan balik biasanya berupa konfirmasi, informasi atau suatu aksi.

5. *Dialogs That Yield Closure*

Urutan tindakan sebaiknya diorganisir atau diatur di dalam suatu kelompok bagian awal, tengah dan akhir. Umpan balik yang diberikan akan memberitahukan pengguna sistem bahwa tindakan yang dilakukan sudah benar dan dapat melanjutkan sejumlah tindakan berikutnya.

6. *Error Handling*

Sistem dirancang untuk mencegah pengguna sistem agar tidak melakukan kesalahan fatal. Jika kesalahan fatal tersebut terjadi, maka sistem dapat langsung memberikan pencegahan kesalahan dengan cepat dan memberikan mekanisme yang simpel dan mudah dipahami oleh pengguna sistem.

7. *Easy Reversal of Actions*

Sistem dirancang bagi pengguna untuk tidak menyulitkan pengguna. Pengguna sistem dibuat untuk tidak takut akan pilihan menu-menu baru karena adanya menu *undo* atau *back* dimana memungkinan pengguna untuk melakukan tindakan kembali jika salah melakukan tindakan.

8. *Reduce Short-Term Memory*

Load Pengguna tidak disulitkan dengan menu–menu yang banyak di dalam sistem atau aplikasi sehingga pengguna dapat melakukan tindakan dengan

memilih menu yang simpel tanpa harus mengingat semua perintah atau fungsi menu–menu sistem.

2.11 *Database*

Database adalah koleksi bersama dari suatu logikal *data* dan deskripsi yang berhubungan dan dirancang untuk memenuhi kebutuhan informasi pada suatu organisasi (Conolly, et al, 2013). Dapat disimpulkan *database* adalah kumpulan *data* logikal beserta deskripsi yang dimiliki yang saling berhubungan yang dirancang dan disimpan dalam suatu *file* yang dapat digunakan oleh banyak aplikasi untuk menghasilkan suatu informasi yang dibutuhkan.

MySQL adalah *software* atau program *databaseserver.* Setiap orang bebas untuk menggunakan *MySQL*, namun tidak boleh dijadikan produk turunan yang bersifat *closed source* atau komersial. *MySQL* sebenarnya merupakan turunan salah satu konsep utama dalam *database* sejak lama, yaitu *Structured Query Languange (SQL). Structured Query Language (SQL)* adalah sebuah konsep pengoperasian *database*, terutama untuk pemilihan atau seleksi dan pemasukan *data*, yang memungkinkan pengoperasian *data* dikerjakan dengan mudah secara otomatis (Nugroho, 2013).

2.12 *Web Base*

2.12.1 *Website*

Web adalah sistem berbasis *hypermedia* yang menyediakan sarana *browsing* informasi di *internet* dengan cara non sekuensial dengan menggunakan *hyperlink* (Connolly and Begg, 2015).

2.12.2 *Hypertext Markup Language (HTML)*

Hypertext Markup Language adalah sebuah bahasa yang mencakup teks halaman, deskripsi struktur, dan *link* ke dokumen lain, gambar, atau media lainnya. HTML adalah *Markup Language* yang berarti bahasa yang ditambahkan dalam dokumen untuk memberikan arti tersendiri terhadap dokumen. HTML memisahkan konten (kata-kata, gambar, audio, video, dan lainnya) dari “penampilan” (definisi dari tipe konten dan instruksi bagaimana tipe konten tersebut harus ditampilkan) (Lemay and Coburn, 2011).

2.12.3 *Hypertext Preprocessor (PHP)*

Hypertext Preprocessor (PHP) adalah *open source* yang populer HTML-*embedded scripting language* yang didukung oleh banyak *web server* termasuk *Apache HTTP server* dan *Microsoft Internet Information Server.* PHP *code* diterjemahkan di *web-server* dan diubah menjadi HTML atau *output* lain yang akan dilihat oleh pengguna.PHP merupakan bahasa pemrograman *script* yang paling banyak digunakan saat ini. PHP banyak dipakai untuk membuat situs *web* dinamis, walaupun tidak tertutup kemungkinan digunakan untuk pemakaian lain (Connolly and Begg, 2015).

2.12.4 *Hypertext Transfer Protocol (HTTP)*

Hypertext Transfer Protocol (HTTP) merupakan sebuah protokol jaringan yang digunakan untuk sistem infromasi yang bersifat terdistribusi, kolaboratif, dan menggunakan *hypermedia* banyak dalam memanfaatkan sumber daya yang dihubungkan dengan *link* yang disebut dokumen *hypertext* yang membentuk *World Wide Web.*

2.12.5 *Cascading Style Sheet (CSS)*

Cascading Style Sheet (CSS) adalah suatu bahasa yang bekerja sama dengan HTML untuk mendefinisikan bagaimana suatu isi halaman *web* ditampilkan atau dipresentasikan. Presentasi ini meliputi *style* atau gaya teks *link*, maupun tata letak halaman (Raharjo, 2011).

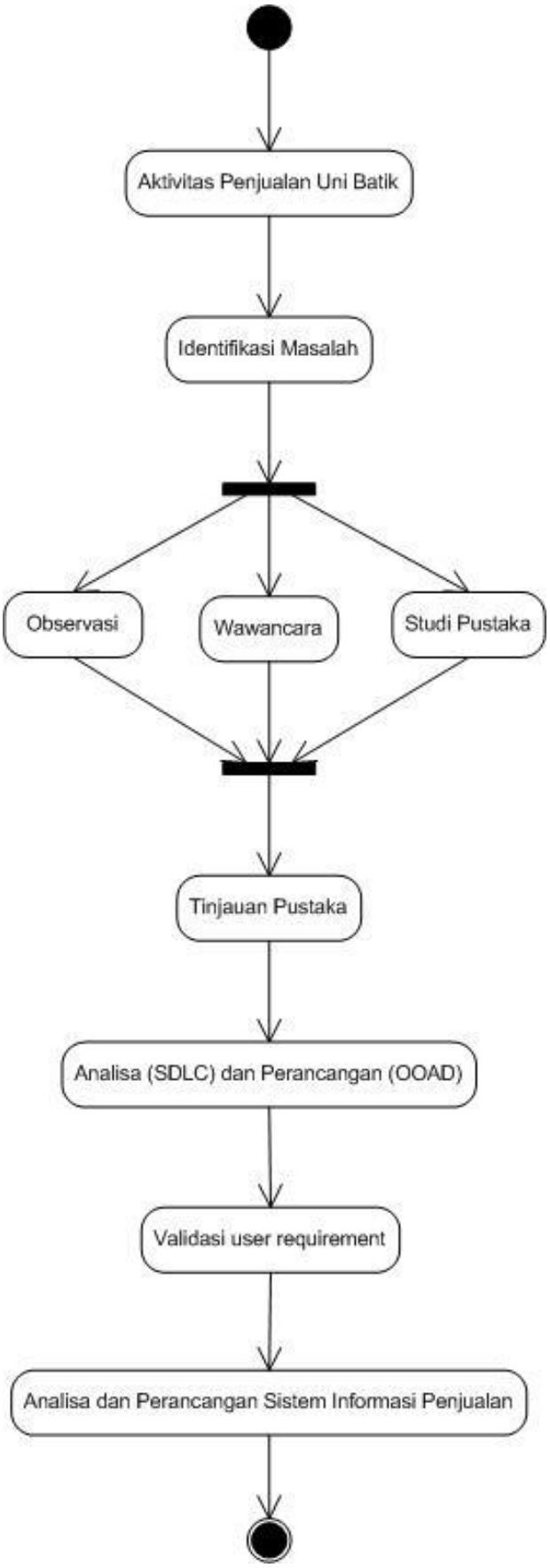
2.13 *E-Commerce*

Perdagangan elektronik atau yang disebut juga *e-commerce*, adalah penggunaan jaringan komunikasi dan komputeruntuk melaksanakan proses bisnis. Pandangan populer dari *e-commerce* adalah penggunaan internet dan komputer dengan browser Web untuk membeli dan menjual produk (McLeod, 2008). *E-commerce* atau kependekan dari *elektronik commerce* (perdagangan secara electronik), merupakan transaksi bisnis yang terjadi dalam jaringan elektronik, seperti internet. Siapapun yang dapat mengakses komputer, memiliki sambungan ke internet, dan memiliki cara untuk membayar barang-barang atau jasa yang mereka beli, dapat berpartisipasi dalam *e-commerce* (Shely, 2007).*Electronic*

*commerce*adalah pembelian, penjualan dan pemasaran barang serta jasa melalui sistem elektronik. Seperti radio, televisi dan jaringan computer atau internet (Wong, 2010). *E-Commerce* adalah suatu proses membeli dan menjual produk-produk secara elektronik oleh konsumen dan dari perusahaan ke perusahaan dengan komputer sebagai perantara transaksi bisnis. Media yang dapat digunakan dalam aktivitas *e-commerce* adalah *world wide web*atau internet (Laudon, 1998).

2.14 Kerangka Pikir

Berikut merupakan kerangka pemikiran yang merupakan kerangka konsep pemecahan masalah yang telah diidentifikasi pada proses bisnis pada Toko Uni Batik Solo



Gambar 2.21 Kerangka Pemikiran

Berikut merupakan rincian dari kerangka pemikiran penelitian ini :

1. **Aktivitas Penjualan Uni Batik**

   Hal pertama yang dilakukan dalam melaksanakan penelitian ini adalah melihat secara langsung aktivitas penjualan di toko Uni Batik Solo bersama *owner*. *Owner* memberi penjelasan ketika ada pelanggan datang untuk membeli produk. Aktivitas yang dilakukan pelanggan antara lain memilih produk, menentukan ukuran yang sesuai, melakukan pembayaran di kasir.

2. **Melakukan Identifikasi Masalah**

   Setelah melihat aktivitas penjualan, dilakukan pengidentifikasian masalah yang terjadi terhadap proses penjualan di Uni Batik Solo. Identifikasi masalah yang ditemukan adalah sebagai berikut :

   - Saat ini informasi dari semua daftar produk yang dijual tidak tedefinisi dengan detail dan jelas sehingga banyak pelanggan tidak mengetahui informasi terkait dengan produk tersebut.
   - Data pembelian masih menggunakan nota kertas. Penggunaan nota kertas memiliki resiko nota rusak, tercecer dan hilang sehingga laporan penjualan tidak sesuai dengan pembelian yang dilakukan pelanggan.
   - Saat ini Uni Batik belum melayani pengiriman produk luar kota, sehingga pelanggan harus datang ke toko untuk melihat dan membeli produk.
   - Potensi kehilangan nota kertas yang tidak dapat dihindari dapat membuat laporan penjualan perusahaan menjadi tidak sesuai dengan data pembelian.

3. **Observasi, Wawancara, dan Studi Pustaka**

   Identifikasi permasalahan pada sitem informasi penjualan pada Uni Batik Solo dilakukan dengan beberapa cara sebagai berikut :

   - Observasi

     Pengumpulan data dilakukan dengan cara peneliti ikut serta dalam aktivitas kerja staff perusahaan tersebut sehingga proses analisa lebih cepat memahami masalah yang ada dengan begitu data yang didapat akurat.

   - Wawancara

     Wawancara dilakukan kepada staff perusahaan untuk mendapatkan informasi yang akurat untuk mendukung dalam perancangan web penjualan Uni Batik.

- Studi Pustaka

  Mempelajari teknologi yang digunakan dalam pengembangan sistem dan mengimplementasiannya dengan membaca buku – buku, artikel, melihat internet dan sumber – sumber lainnya.

4. Melakukan Tinjauan Pustaka

Mempelajari teori yang digunakan dalam analisa dan perancangan. Selain itu, mempelajari tentang penelitian atau sistem yang berjalan dan membandingkan dengan penelitian yang akan dilakukan. Hal tersebut dilakukan dengan mencari buku, *e-book, e-journal* yang berkaitan dengan sistem informasi penjualan.

5. Analisa dan Perancangan

Metode yang digunakan dalam tahap analisa adalah metode *waterfall*. *Waterfall* termasuk kedalam pendekatan *System Development Life Cycle* (SDLC) (Satzinger, Jackson & Burd, 2012) dimanametode ini terdiri dari 6 tahapan, namun pada analisa dan perancangan sistem informasi penjualan ini hanya menggunakan 4 tahapan yang meliputi *project initiation, project planning, analysis* dan *design* karena penelitian ini hanya sampai pada tahap perancangan.

6. Validasi *User Requirement*

Setelah dilakukan Analisa dan Perancangan, Validasi *user requirement* telah dilakukan oleh Uni Batik Solo sesuai dengan permohonan penelitian skripsi dari bulan Maret hingga Agustus 2019.

7. Membuat Analisa dan Perancangan Sistem Informasi Penjualan

Setelah melakukan identifiasi masalah, tinjauan pustaka, dan pemilihan metode analisa perancangan. Pada tahap ini, metode perancangan menggunakan *Object Oriented Analysis and Design* (OOAD) yaitu Pemodelan perancangan sistem menggunakan *Unified Modelling Language* (UML) yang meliputi *activity diagram, use case diagram, use case description, domain model class diagram, system sequence diagram, package diagram* dan *persistent object* (Satzinger, Jackson & Burd, 2012).