BAB 2
LANDASAN TEORI

2.1 Pemasaran (*Marketing*)

Pemasaran adalah proses dimana perusahaan menciptakan nilai bagi pelanggan dan membangun hubungan yang kuat dengan pelanggan dalam rangka untuk menangkap nilai dari pelanggan sebagai imbalan (Kotler dan Armstrong, 2015). Manajemen pemasaran sebagai seni dan ilmu memilih pasar sasaran dan mendapatkan, mempertahankan, serta meningkatkan jumlah pelanggan dengan menciptakan, menghantarkan dan mengkomunikasikan nilai pelanggan yang unggul (Kotler dan Keller, 2015).

Konsep pemasaran didasarkan pada pandangan dari luar ke dalam. Konsep ini diawali dengan mendefinisikan pasar yang jelas berfokus pada kebutuhan pelanggan, memadukan semua sistem kegiatan yang akan mempengaruhi pelanggan dan menghasilkan laba melalui pemuasan pelanggan. Konsep pemasaran bersandar pada empat pilar utama, yaitu:

- Pasar sasaran: karena tidak ada perusahan yang dapat beroperasi disemua pasar dan memuaskan semua kebutuhan dan juga tidak ada yang dapat beroperasi dengan baik dalam pasar yang luas, sehingga jika suatu perusahaan itu ingin berhasil maka ia harus dapat mendefinisikan pasar sasaran meraka dengan cermat dan menyiapkan program pemasaran.
- Kebutuhan pelanggan: memahami kebutuhan dan keinginan pelanggan tidak selalu merupakan tugas yang sederhana dikarekan beberapa pelanggan itu memiliki kebutuhan sendiri yang tidak mereka sadari atau mereka tidak dapat mengutarakan kebutuhan-kebutuhan ini.
- Pemasaran terpadu, jika semua departemen bekerja sama melayani kepentingan pelanggan maka hasilnya adalah pemasaran terpadu.

2.2 *Sales Activity*

Dalam pemasaran terdapat strategi yang disebut bauran pemasaran yaitu seperangkat alat pemasaran yang dipadukan untuk memproses tanggapan yang diinginkan target pasar (Kotler dan Keller, 2016). Bauran promosi terdiri atas lima alat-alat promosi, yaitu (Kotler dan Armstrong, 2014):

1. *Advertising*, periklanan yaitu semua bentuk presentasi dan promosi *non-personal* yang dibayar oleh *sponsor* untuk mempresentasikan gagasan, barang atau jasa.
2. *Sales Promotion,* promosi penjualan yaitu insentif-insentif jangka pendek untuk mendorong pembelian atau penjualan suatu produk atau jasa.
3. *Personal Selling*, penjualan perseorangan yaitu presentasi *personal* oleh tenaga penjualan dengan tujuan menghasilkan penjualan dan membangun hubungan dengan konsumen.
4. *Public Relations*, hubungan masyarakat yaitu membangun hubungan yang baik dengan berbagai perusahaan publik supaya memperoleh publisitas yang menguntungkan, membangun citra perusahaan yang bagus, dan menangani atau meluruskan rumor, cerita, serta *event* yang tidak menguntungkan. Bentuk promosi yang digunakan mencakup *press releases, sponsorships, special events, dan web pages.*
5. *Direct Marketing,* penjualan langsung yaitu hubungan langsung dengan sasaran konsumen dengan tujuan untuk memperoleh tanggapan segera dan membina hubungan yang abadi dengan konsumen.

Personal Selling adalah alat promosi yang terkait dengan penelitian ini, dimana *personal selling* terdiri dari interaksi antara pribadi dengan pelanggan dan calon pelanggan untuk membuat penjualan dan mempertahankan hubungan dengan pelanggan (Kotler dan Armstrong, 2014). Terdapat enam langkah yang harus dijalankan oleh *personal selling* (Kotler dan Keller, 2016), yaitu:

1. *Prospecting and Qualifiying*, langkah pertama dalam menjual adalah mengidentifikasi calon nasabah. Biasanya menyeleksi calon nasabah dilakukan dengan cara menghubungi mereka melalui pesan atau telpon untuk mengetahui tingkat ketertarikan, minat beli dan kemampuan finansial dari calon nasabah.
2. *Preaproach*, fase dimana tenaga penjualan mencari informasi mengenai calon nasabah dari perusahaan tentang kebutuhan dari calon nasabah, siapa yang memegang peranan dalam memutuskan pembelian, dan lainnya.
3. *Presentation and Demonstration*, fase dimana tenaga penjual menceritakan tentang produk kepada pembeli dengan mengunakan

pendekatan fitur (*features*), keunggulan (*advantages*), manfaat (*benefits*), dan nilai (*values*).

4. *Overcoming Objections*, fase dimana tenaga penjual mengatasi masalah yang dihadapi konsumen yang dapat mengalangi proses pembelian yang terbagi menjadi dua yaitu hambatan psikologi (preferensi merek lain, apatis, ide yang ditetapkan sebelumnya, dan lainnya) dan hambatan logis (harga, waktu pengantaran, dan karakteristik produk atau perusahaan).
5. *Closing*, fase dimana pembeli melakukan tindakan nyata, keputusan atau masukan, dan pertanyaan.
6. *Folow Up and Maintenance*, hal ini sangatlah penting utuk memastikan kepuasan pelanggan dan pembelian ulang. Secara langsung setelah *closing*, tenaga penjual harus memberitahukan semua keperluan yang *detail* seperti waktu pengantaran, ketentuan pembelian, dan hal lain yang penting bagi konsumen.

2.3 Sistem Informasi

2.3.1 Definisi Sistem

Sistem adalah sekumpulan prosedur yang saling berhubungan untuk menghasilkan atau menyelesaikan tujuan yang sama, hasil dari kegiatan yang dipicu oleh *user* (Satzinger, Jackson dan Burd, 2015). Suatu sistem memiliki beberapa karakteristik utama, yaitu:

1. Komponen (*component*), sebuah sistem terdiri dari sejumlah komponen yang saling berinteraksi. Komponen sistem biasanya dikenal dengan subsistem. Subsistem memiliki hal sistem itu sendiri dalam fungsinya dan memiliki sistem keseluruhan.
2. Batasan Sistem (*boundary*), pembatasan yang membatasi sistem merupakan daerah antara sistem dengan sistem lainnya. Batasan sistem ini memungkinkan suatu sistem dipandang sebagai satu kesatuan, menunjukkan sistem membatasi ruang lingkup sistem.
3. Lingkungan Luar Sistem (*environments*), lingkungan luar sistem di luar batas dari sistem yang mempengaruhi operasi sistem. Lingkungan luar dapat bermanfaat serta merugikan sistem. Lingkungan *eksternal* yang menguntungkan merupakan energi dari sistem dan lingkungan

luar yang merugikan harus ditahan dan dikendalikan, kalau tidak akan mengganggu kehidupan kelansungan sistem.

4. Penghubung Sistem (*interface*), sistem *link* adalah media penghubung antara subsistem lainnya. Melalui *interface* ini memungkinkan sumber daya mengalir dari satu subsistem ke subsistem lainnya.
5. Masukan Sistem (*input*), masukan sistem adalah energi yang dimasukkan ke dalam sistem. Masukan dapat pengobatan masukan (*input* pemeliharaan) dan sinyal *input* (sinyal *input*). Masukan energi pemeliharaan dimasukkan sehingga sistem tersebut dapat beroperasi. Sinyal *input* diproses untuk mendapatkan keluaran energi.
6. Keluaran Sistem (*output*), keluaran sistem adalah hasil dari energi dalam meskipun dan diklasifikasikan menjadi keluaran yang berguna dan sisa pembuangan. *Output* dapat menjadi masukan bagi subsistem lain atau suprasistem.
7. Pengolahan Sistem (*process*), suatu sistem dapat memiliki bagian pengolahan yang akan mengubah *input* menjadi *output*.
8. Sasaran Sistem (*objectives*), sebuah sasaran yang ingin dicapai untuk menentukan masukan yang diperlukan dari *output* sistem menjadi sistem yang dihasilkan.

2.3.2 Definisi Informasi

Informasi adalah data yang telah diubah menggunakan pola tertentu dan cara tertentu sehingga dapat lebih berarti dan menjadi fakta yang berguna bagi penggunanya (Satzinger, Jackson dan Burd, 2015).

2.3.3 Definisi Sistem Informasi

Sistem Informasi adalah suatu kumpulan dari komponen komputer yang saling terhubung, yang bertujuan untuk mengumpulkan, memproses, menyimpan (biasanya disimpan dalam *database*) dan menyediakan *output* berupa informasi yang dibutuhkan untuk menyelesaikan tugas bisnis (Satzinger, Jackson, dan Burd, 2015). Jenis sistem informasi (Satzinger, Jackson dan Burd, 2015), adalah sebagai berikut:

1. Sistem Pemprosesan Transaksi, yaitu sistem yang berfungsi untuk melakukan *capture* dan proses data dalam suatu transaksi bisnis yang berguna untuk memudahkan proses bisnis.
2. Sistem Informasi Manajemen, yaitu sistem informasi yang menyediakan fungsi *reporting* dari suatu proses transaksi suatu perusahaan.
3. Sistem Pendukung Keputusan, yaitu sistem informasi yang menyediakan *output* berupa data berbagai alternatif keputusan untuk menunjang pengambilan keputusan.
4. Sistem Informasi Eksekutif, yaitu sistem informasi yang digunakan *level* manajerial untuk mendukung perencanaan bisnis dan me*review* kebijakan bisnis sebelumnya.
5. Sistem Pakar, yaitu sistem yang menyediakan *alternative* pemecahan masalah dengan cara melakukan simulasi kembali pemikiran para ahli.
6. Sistem Otomatisasi Kantor, yaitu sistem informasi yang mendukung aktifitas bisnis, atau bisa disebut juga dengan menyediakan *tools* berupa sistem untuk lembar kerja karyawan.

2.4 *Object Oriented Analysis and Design Process* (OOAD)

2.4.1 Definisi *Object Oriented Analysis and Design Process* (OOAD)

Object Oriented Analysis (OOA) adalah identifikasi objek-objek yang berinteraksi langsung dengan sistem untuk menyelesaikan program dengan detail tugas masing-masing. *Object Oriented Design* (OOD) adalah semua penambahan jenis objek yang dibutuhkan untuk berkomunikasi dengan sistem yang menunjukkan detail interaksi objek dan implementasi objek dalam berbagai bahasa pemrograman atau lingkungan tertentu (Satzinger, Jackson, dan Burd, 2015).

2.4.2 Konsep *Object Oriented Analysis* dan *Design* (OOAD)

Konsep OOAD mencakup analisis dan *design* sebuah sistem dengan pendekatan *object* (Satzinger, Jackson, dan Burd, 2015), yaitu:

a. *Object Oriented Analysis* (OOA)

Object Oriented Analysis adalah metode yang melakukan analisa *requirement* (syarat) yang harus dipenuhi sebuah sistem yang dispesifikasi berdasarkan sudut pandang kelas-kelas dan *object-object* yang ditemui dalam lingkup perusahaan. Analisa berbasis objek membantu untuk menentukan objek yang menjalankan pekerjaan dan menentukan interaksi dengan pengguna apa yang diperlukan untuk menyelesaikan tugas.

b. *Object Oriented Design* (OOD)

Object Oriented Design adalah metode untuk melakukan transformasi model analisis yang dibuat dengan menggunakan OOA ke dalam model *design* yang berfungsi sebagai *blueprint design software*. Selama OOD pengembang menerapkan batasan implementasi model yang diambil dari OOA. Konsep dalam model analisis tertuju pada kelas yang diimplementasikan dan *interface* yang dihasilkan oleh model, dengan memberikan penjelasan secara rinci tentang sistem yang akan dirancang.

2.5 *System Development Life Cycle* (SDLC)

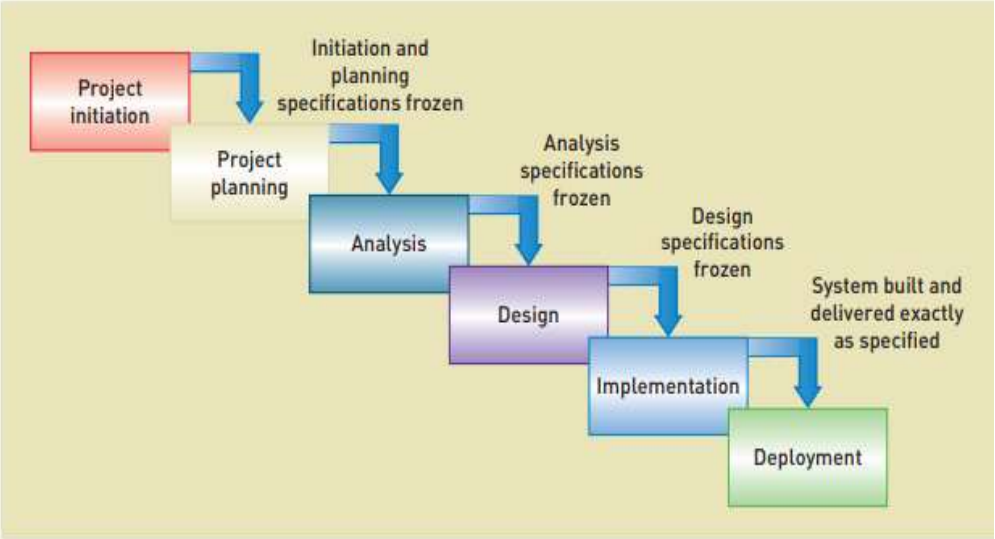
System Development Life Cycle (SDLC) adalah kerangka kerja yang mengidentifikasi semua kegiatan yang diperlukan untuk meneliti, membangun, menyebarkan, dan memelihara sistem informasi (Satzinger, Jackson, dan Burd, 2015). Setiap SDLC mencakup beberapa proses inti yang selalu dibutuhkan, sering juga disebut *six core processes* SDLC. Berikut adalah enam proses inti yang diperlukan dalam pengembangan sistem informasi apa pun (Satzinger, Jackson, dan Burd, 2015):

1. Identifikasi masalah atau kebutuhan dan dapatkan persetujuan untuk melanjutkan proyek. Kegiatan yang harus dilakukan:
   - Mengidentifikasi masalahnya
   - Memberi justifikasi mengenai faktor yang dapat memperngaruhi persetujuan proyek
   - Menganalisis resiko dan kelayakan proyek
   - Mengulas kembali dengan *client* dan mendapatkan persetujuan pelaksanaan proyek.

2. Merencanakan dan memantau proyek. Kegiatan yang harus dilakukan adalah:
   - Menentukan lingkungan proyek
   - Menetapkan jadwal kerja
   - Menempatkan sumber daya dan pekerja
   - Mengevaluasi proses kerja
   - Memantau pengerjaan dan membuat koreksi
3. Temukan dan pahami *detail* masalah atau kebutuhan. Kegiatan yang harus dilakukan adalah:
   - Mengumpulkan informasi lebih *detail*
   - Mendefinisikan keperluan yang dibutuhkan
   - Memprioritaskan keperluan
   - Mengembangkan desain tampilan
   - Mengevaluasi keperluan dengan para pengguna
4. Rancang komponen sistem yang memecahkan masalah atau memuaskan kebutuhan. Kegiatan yang harus dilakukan adalah:
   - Merancang komponen sistem
5. Membangun, menguji, dan integrasikan komponen sistem. Kegiatan yang harus dilakukan adalah:
   - Membuat program perangkat lunak
   - Menguji *unit* perangkat lunak
   - Identifikasi dan membangun uji kasus
   - Mengintegrasikan dan menguji komponen
6. Selesaikan pengujian sistem dan kemudian gunakan solusinya
   - Menunjukan uji sistem dan tekanan
   - Menunjukan uji penerimaan terhadap pengguna
   - Mengubah data yang ada
   - Membangun dan melakukan materi pelatihan
   - Mengkonfigurasikan dan menentukan lingkungan produksi
   - Menggunakan solusinya

2.6 Metode *Waterfall*

Pendekatan prediktif ke SDLC mengasumsikan bahwa proyek pengembangan dapat direncanakan dan diorganisir dan bahwa sistem informasi baru dapat dikembangkan sesuai dengan rencana. SDLC prediktif berguna untuk membangun sistem yang dipahami dan didefinisikan dengan baik. Pendekatan SDLC paling prediktif disebut model *waterfall*, dengan fase proyek mengalir turun, satu demi satu (Satzinger, Jackson, dan Burd, 2015).



Gambar 2.1 Contoh *Waterfall model of the SDLC* (Satzinger, Jackson, dan Burd, 2015)

Berdasarkan gambar 2.1 terlihat enam fase yang harus dilakukan secara sekuensial dalam pengembangan sistem, dengan urutan sebagai berikut:

1. *Project Initiation:* identifikasi permasalahan atau kebutuhan dan mendapatkan persetujuan untuk memulai *project*.
2. *Project Planning:* merencanakan dan memantau jalannya projek. Hal-hal yang perlu direncanakan terkait apa yang harus dilakukan (*what-to-do*), bagaimana melakukannya (*how-to-do-it*), dan siapa yang melakukannya (*who-does-it*).
3. *Analysis:* menemukan dan memahami detail dari permasalahan dan kebutuhan sistem.
4. *Design:* merancang komponen-komponen sistem guna menyelesaikan permasalahan atau memenuhi kebutuhan sistem.

5. *Implementation*: membangun sistem sesuai rancangan, melakukan uji coba, dan mengintegrasikan komponen-komponen sistem. Namun pada penelitian ini *implementation* tidak dilakukan tetapi hanya sampai tahap perancangan.
6. *Deployment:* tahap dimana sistem informasi dibuat tersedia bagi seluruh komunitas pengguna. Pada penelitian ini *deployment* tidak dilakukan, karena sistem informasi DSAR tidak dibuat.

2.7 *Unified Modeling Language* (UML) *Diagram*

Unified Modeling Language (UML) merupakan rangkaian standar konstruksi model dan notasi yang dikembangkan secara khusus untuk pengembangan sistem arsitektur yang bekerja dalam OOAD (*Object-Oriented Analysis/Design*) dengan satu bahasa yang konsisten.

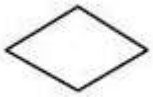
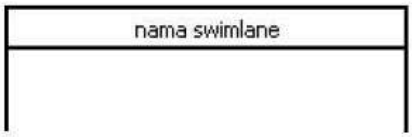
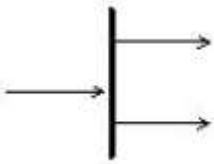
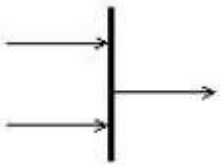
Dengan adanya UML, diharapkan dapat mengurangi kekacauan dalam bahasa permodelan yang selama ini terjadi dalam lingkungan industri. UML diharapkan juga dapat menjawab masalah penotasian dan mekanisme tukar menukar model yang terjadi selama ini. (Satzinger, Jackson, dan Burd, 2015).

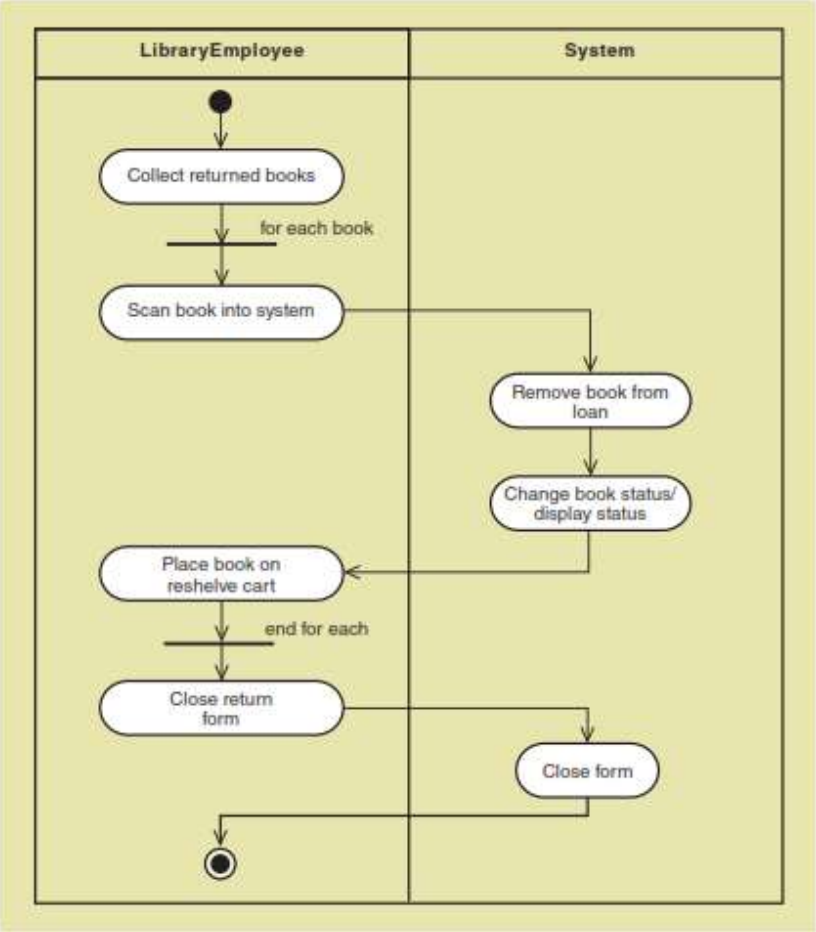
Dengan menggunakan UML, *analysts* dan *end users* dapat memilih dan memahami berbagai *diagram* yang digunakan selama projek pengembangan sistem. *Use case diagram*, *class diagram*, *sequence diagram*, *communication diagram*, dan *state machine diagram* merupakan contoh dari UML masing-masing diagram UML yang digunakan dalam penelitian ini akan dijelaskan selanjutnya pada bab ini.

2.8 *Activity Diagram*

Activity diagram adalah teknik untuk mendeskripsikan logika prosedural, proses bisnis dan aliran kerja dalam banyak kasus. *Activity Diagram* dapat menggambarkan berbagai *user* (pengguna) sistem yang biasa disebut *actor* dalam melakukan kegiatan lengkap dengan langkah-langkah pengolahan aktifitasnya (Satzinger, Jackson, dan Burd, 2015).

Tabel 2.1 Daftar *Symbol Activity Diagram* (Satzinger, Jackson, dan Burd, 2015).

Komponen	Simbol	Penjelasan
Initial Node		Menunjukkan dimana aliran kerja dimulai
Activity		Aktifitas yang dilakukan oleh sistem
Flow		Menunjukkan alur aktifitas
Decision		Digunakan untuk menggambarkan suatu keputusan atau tindakan yang harus diambil pada kondisi tertentu
Asosiasi		*Asosiasi* penggabungan dimana lebih dari satu aktifitas digabungkan menjadi satu
Swimlane		Pembagian tanggung jawab pada suatu proses
Fork		Digunakan untuk menunjukkan kegiatan yang dilakukan secara paralel
Join		Digunakan untuk menunjukkan penggabungan proses yang dilakukan secara bersamaan
Final Node		Menunjukkan dimana aliran kerja diakhiri



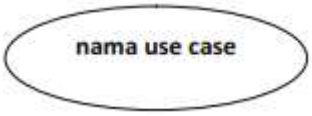
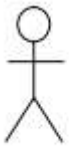
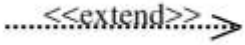
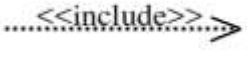
Gambar 2.2 Contoh *Activity Diagram Check Out* Sistem Perpustakaan (Satzinger, Jackson, dan Burd, 2015).

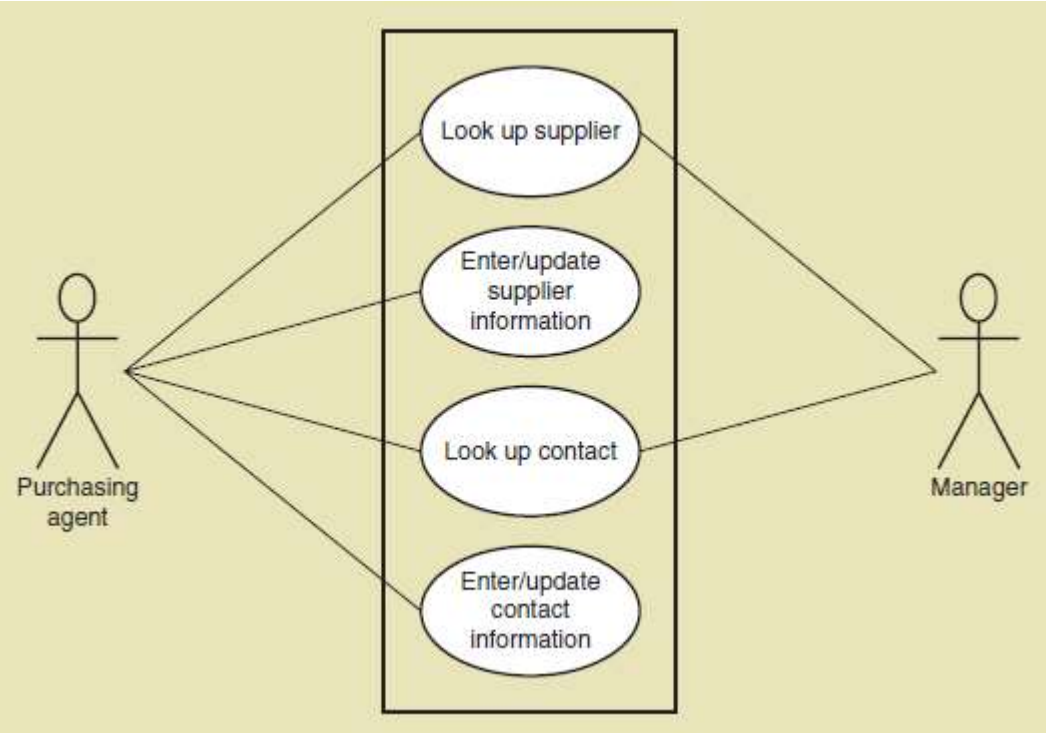
Diatas adalah contoh *activity diagram* untuk *check out* pinjaman buku di sistem perrpustakaan. Dengan alur awal adalah men*scan* identitas buku ke sistem yang terkumpul dalam kumpulan buku yang dikembalikan. Kemudian sistem menghapus dari data pinjaman dan mengubah status buku, maka proses sistem *check out* peminjaman buku selesai.

2.9 *Use Case Diagram*

Use case diagram adalah rangkaian/uraian sekelompok yang saling terkait dan membentuk sistem secara teratur yang dilakukan atau diawasi oleh sebuah aktor. *Use Case Diagram* merupakan rangkaian tindakan yang dilakukan oleh sistem, aktor mewakili *user* atau sistem lain yang berinteraksi dengan sistem yang dimodelkan (Satzinger, Jackson, dan Burd, 2015).

Tabel 2.2 Simbol-simbol *Use Case Diagram* (Satzinger, Jackson, dan Burd, 2015)

Komponen	Simbol	Penjelasan
Use Case		Fungsionalitas yang disediakan sistem sebagai *unit-unit* yang saling bertukar pesan antar *unit* atau *actor*, biasanya dinyatakan dengan menggunakan kata kerja di awal *frase* nama *use case*.
Actor		Orang, proses, atau sistem lain yang berinteraksi dengan sistem informasi yang akan dibuat, jadi walaupun simbol *actor* adalah gambar orang, tapi *actor* belum tentu merupakan orang, biasanya dinyatakan menggunakan kata benda di awal *frase* nama *actor*.
Asosiasi		Komunikasi antara *actor* dan *use case* yang berpartisipasi pada *use case* atau *use case* memiliki interaksi dengan *actor*.
Generalisasi		Hubungan *generalisasi* dan *spesialisasi* (umum-khusu) antara dua buah *use case* dimana fungsi yang satu adalah fungsi yang lebih umum dari lainnya.
Extend		Relasi *use case* tambahan ke sebuah *use case* dimana *use case* yang ditambahkan dapat berdiri sendiri walau tanpa *use case* tambahan itu, mirip dengan prinsip *inheritance* pada pemrograman berorientasi objek, biasanya *use case* tambahan memiliki nama depan yang sama dengan *use case* yang ditambahkan.
Include		Relasi use case tambahan ke sebuah use case dimana use case yang ditambahkan memerlukan use case ini untuk menjalankan fungsinya atau sebagai syarat dijalankan use case ini. Include berarti use case yang ditambahkan akan selalu dipanggil saat use case tambahan dijalankan.



Gambar 2.3 Contoh *Use Case Diagram* menampilkan data *supplier.*

Gambar diatas menggambarkan *use case diagram* sederhana untuk informasi *supplier*. Subsistem menunjukkan empat kasus penggunaan sebagai oval dan dua pengguna yaitu *Purchasing agent* dan *Manager*. Garis yang menghubungkan pengguna dan *use case* menunjukkan siapa yang menggunakan sistem. Agen pembelian melakukan keempat *use case*, sedangkan *manager* hanya mencari pemasok atau kontak.

2.10 *Use Case Description*

Use case description merupakan penjelasan terperinci mengenai proses dari suatu *use case* atau bisa disebut juga sebagai daftar kasus penggunaan *use case diagram* yang memberikan gambaran *detail* dari semua *case* pada sistem (Satzinger, Jackson, dan Burd, 2015). Informasi rinci tentang setiap kasus penggunaan digambarkan dengan menggunakan deskripsi kasus. *Use case description* dibedakan menjadi dua (Satzinger, Jackson, dan Burd, 2015), yaitu:

- *Brief Use Case Description* dapat digunakan untuk *use case* yang sederhana, khususnya sistem yang dikembangkan untuk aplikasi kecil yang mudah dimengerti.
- *Fully Developed Use Case Description* adalah *metode* paling formal

untuk mendokumentasikan *use case* karena menjelaskan secara rinci setiap proses dalam *use case diagram*.

2.10.1 Brief Use Case Description

Brief use case description merupakan deskripsi yang mencatat mengenai *detail* pemrosesan dari suatu *use case* (Satzinger, Jackson, dan Burd, 2015). *Use case* memiliki urutan yang lengkap dari tahapan-tahapan untuk menyelesaikan bisnis proses. *Scenario* atau *use case instance* merupakan suatu kumpulan unik dari aktivitas *internal* di dalam *use case* yang menggambarkan langkah unik sepanjang *use case.*

Tabel 2.3 Contoh *Brief Use Case Description* (Satzinger, Jackson, dan Burd, 2015).

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.

Tabel 2.3 menunjukkan bagaimana *use case* dijelaskan secara rinci mengenai proses apa saja yang terjadi pada suatu *use case*, misal *create customer account* berarti di dalam *use case* tersebut terjadi proses *user* memasukkan data *Customer* baru, lalu sistem memberikan *account number*, lalu membuat data *record* baru, dan kemudian *account* baru terbentuk.

2.10.2 Fully Developed Use Case Description

*Fully Developed Use Case Description m*erupakan *metode* yang paling formal mendokumentasikan sebuah *use case*. Meskipun memerlukan waktu lebih untuk mengerjakan, jenis dari *use case description* ini dapat meningkatkan kemungkinan akan pemahaman mengenai proses bisnis.

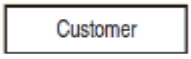
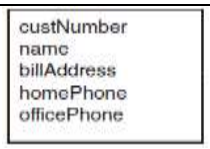
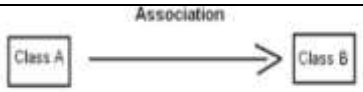
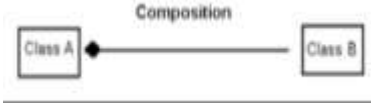
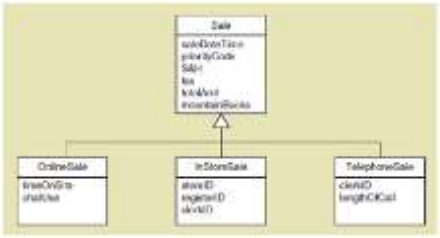
Use case name:	*Create customer account.*	
Scenario:	Create online customer account.	
Triggering event:	New customer wants to set up account online.	
Brief description:	Online customer creates customer account by entering basic information and then following up with one or more addresses and a credit 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 2.4 Contoh *Fully Developed Use Case Description Create Customer Account.*

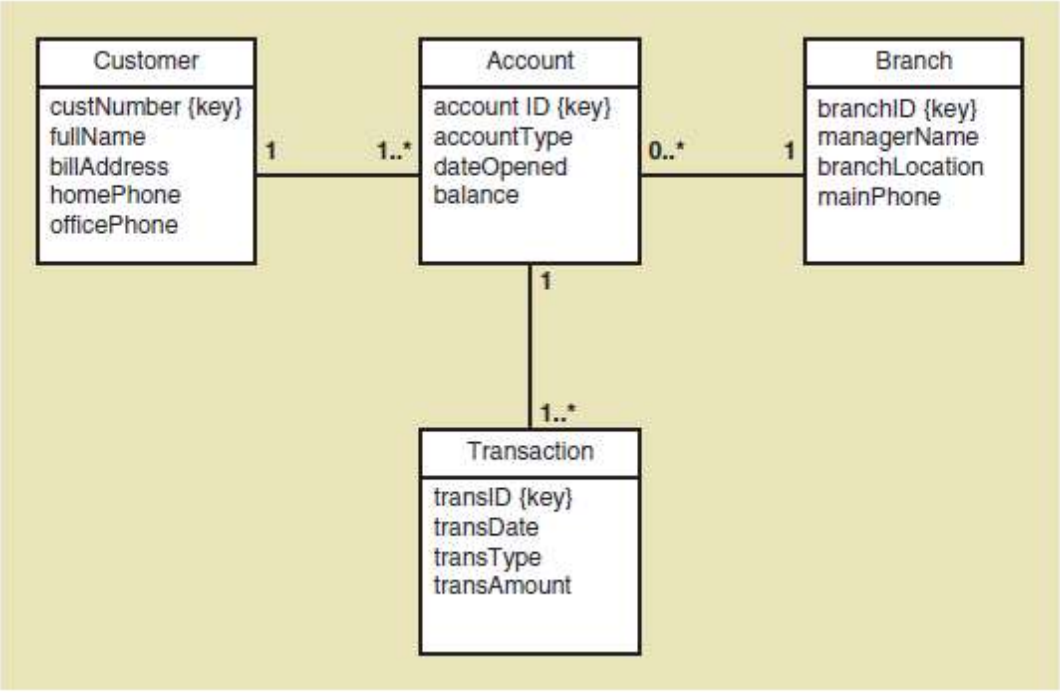
2.11 *Domain Model Class Diagram*

UML *class digram* digunakan untuk menunjukkan kelas objek untuk suatu sistem. Salah satu jenis UML *class digram* yang menunjukkan hal-hal dalam *domain* masalah pengguna disebut *domain model class diagram* (Satzinger, Jackson, dan Burd, 2015).

Tabel 2.4 Daftar Simbol *Domain Model Class Diagram* (Satzinger, Jackson, dan Burd, 2015).

Komponen	Simbol	Penjelasan
Class		*Class*, adalah kategori atau klasifikasi yang digunakan untuk menggambarkan koleksi benda.
Attributes		*Attributes*, adalah semua benda yang termasuk dalam kelas yang memiliki nilai untuk masing-masing
Association		*Association*, yaitu *class* yang merepresentasikan *many-to-many relationship* antara dua *class* lainnya.
Aggregation		*Aggregation*, yaitu hubungan antara objek dengan bagian-bagiannya dimana bagian–bagian tersebut dapat muncul terpisah.
Composition		Composition, yaitu hubungan antar *class* yang menyatakan hubungan *"part-of."*
Generalization		*Generalization/Specialization hierarchies*, yaitu hirarki bahwa struktur atau peringkat kelas dari kelas super yang lebih umum ke kelas yang lebih spesifik atau sub kelas. Generalisasi adalah penilaian bahwa kelompok sejenis hal, sedangkan spesialisasi adalah penilaian yang mengkategorikan jenis hal.
Zero to One		*Zero to one*, yaitu hubungan dari kelas ke kelas, dimana kelas induk memiliki satu kelas anak dan kelas ada bisa tidak memiliki kelas induk.

Komponen	Simbol	Penjelasan
One to One	1 ——— 1	*One to one*, yaitu hubungan dari kelas ke kelas, dimana kelas induk memiliki satu kelas anak dan sebaliknya.
One to Many	1 ——— ..*	One to Many, yaitu hubungan dari kelas ke kelas, dimana kelas induk memiliki satu atau lebih kelas anak dan kelas anak hanya memiliki satu kelas induk.
Zero to Many	0 ——— ..*	Zero to Many, yaitu hubungan dari kelas ke kelas, dimana kelas induk memiliki beberapa kelas anak dan kelas ada bisa tidak memiliki kelas.
Many to Many	..* ——— ..*	Many to Many, yaitu hubungan dari kelas ke kelas, dimana kelas induk memiliki beberapa kelas anak dan sebaliknya.



Gambar 2.5 Contoh Gambar *Domain Model Class Diagram Bank.*

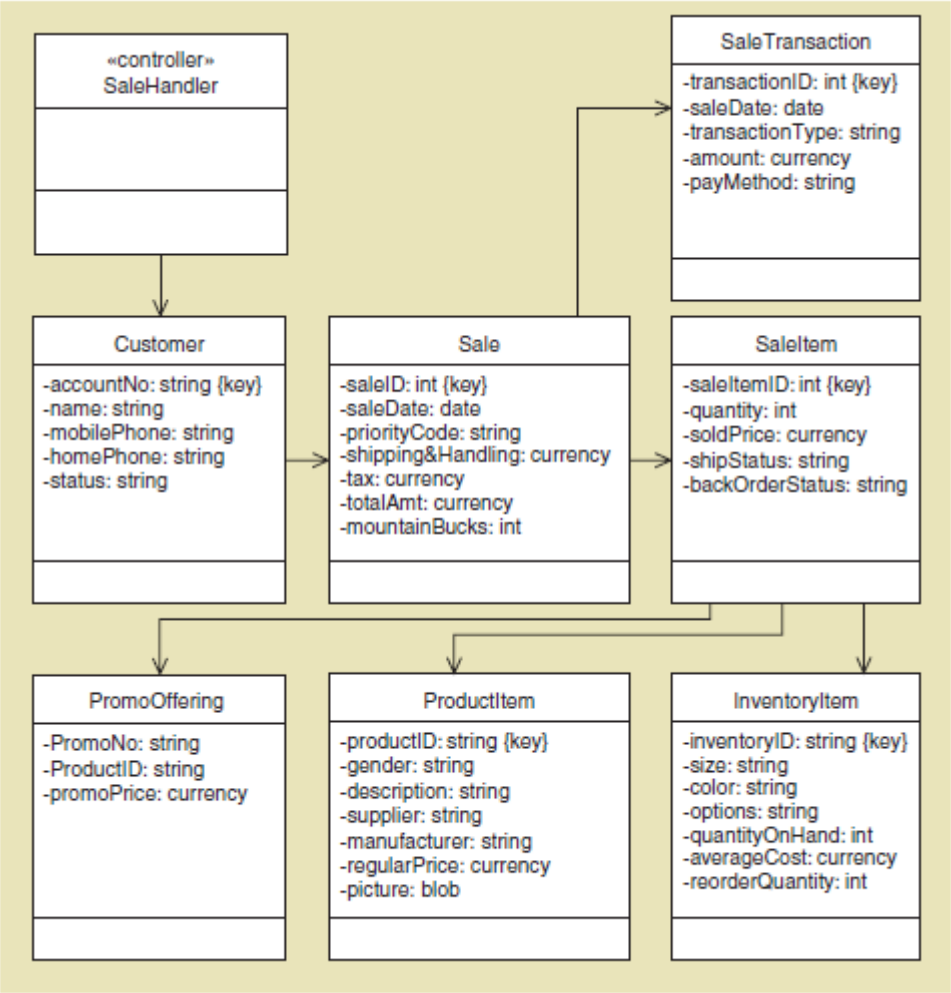
Pada Gambar 2.5 terlihat contoh *Domain Model Class Diagram* yang menggambarkan proses transaksi rekening nasabah di sebuah bank yang memiliki beberapa cabang, dimana satu *Customer* bisa memiliki beberapa *account*, dan satu cabang bisa memiliki banyak *account* dan minimal tidak ada *account* (misal untuk cabang yang baru dibuka atau kantor kas. Sedangkan satu *account* bisa memiliki banyak *record* transaksi didalamnya.

2.12 *First Cut Design Model Class Diagram*

First-cut design class diagram adalah sebuah *class* diagram yang lebih menjelaskan mengenai alur data beserta tipe datanya. Dalam penggambaran proses desain, kita dapat memulai dengan merancang *first-cut design model class diagram* berdasarkan *domain class diagram*. Pada perancangan ini, *first-cut design class diagram* tidak memerlukan *method list* (Satzinger, Jackson, dan Burd, 2015).

Pengembangan *design class diagram* dapat dilakukan pada setiap *layer*, dimana dalam *view* dan *data access layer* dilakukan penentuan beberapa class baru. Pada *domain layer*, *class* baru yang ditambahkan berfungsi sebagai *use case controller.* Penambahan *method* untuk setiap *class* dalam *updated class diagram* dapat dilakukan, dimana *method* tersebut terdiri dari 3 jenis, yaitu:

- *Constructor methods*, merupakan *method* yang membentuk *instance* dari suatu obyek.
- *Data get and set methods*, merupakan *method* yang mengambil dan mengubah nilai atribut.
- *Use case specific methods*, merupakan *method* yang mewakili *use case* yang ada.



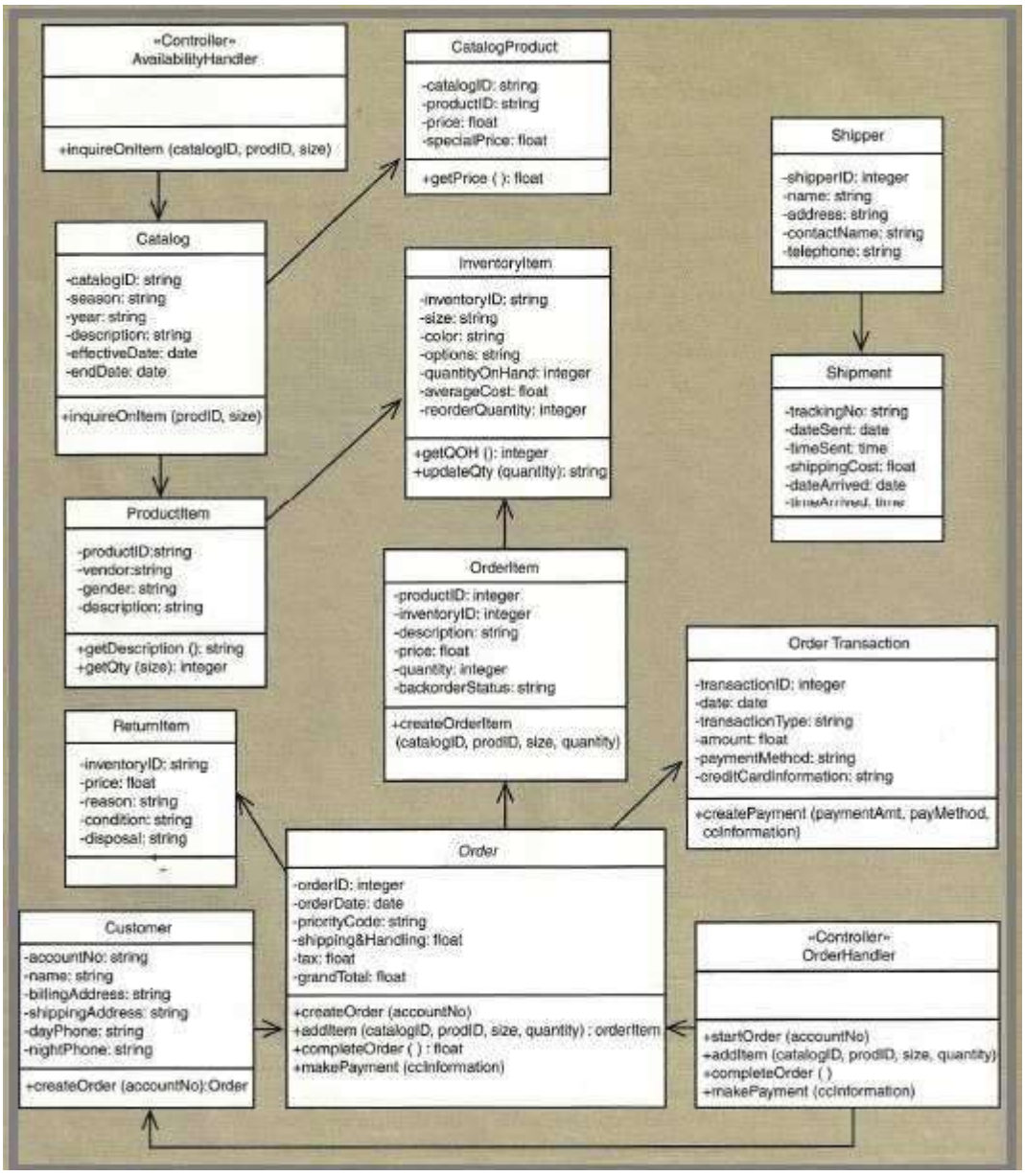
Gambar 2.6 Contoh *First Cut Design Model Class Diagram* alur data pada sistem informasi penjualan (Satzinger, Jackson, dan Burd, 2015).

Gambar 2.6 menggambarkan mengenai alur data pada sistem informasi penjualan, dimana alur data dimulai dari *data class* sales *handler* atau data penjual kemudian *data class customer* yang membeli dimana terdiri dari data no. *customer*, nama *customer*, nomor ponsel, nomor telepon rumah, status. Kemudian dilanjutkan ke alur data penjualan, lalu data *sale item* yang akan berujung pada data promo *offering, product item, inventory item*.

2.13 *Update Domain Model Class Diagram*

Updated Domain Model Class Diagram merupakan pengembangan dari *first-cut class diagram*. Pada *updated class diagram*, informasi *method* dapat ditambahkan ke dalam *class* dan dilakukan pembaharuan arah panah navigasi yang didapat dari *sequence diagram* yang telah dibuat sebelumnya. Semua *handler* akan

menjadi *class* baru (Satzinger, Jackson, dan Burd, 2015).



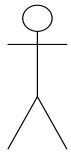
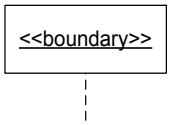
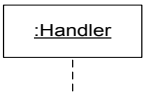
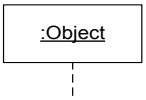
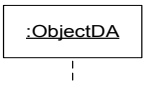
Gambar 2.7 Contoh *Update Domain Model Class Diagram* alur data pada sistem informasi penjualan (Satzinger, Jackson, dan Burd, 2015).

Dari gambar diatas disimpulkan bahwa terdapat *class* yang bernama *Catalog, CatalogProduct, InventoryItem, Shipper, Shipment, Order Transaction, Order, Customer, ReturnItem*. Serta terdapat function untuk develop di masing-masing class.

2.14 *System Sequence Diagram*

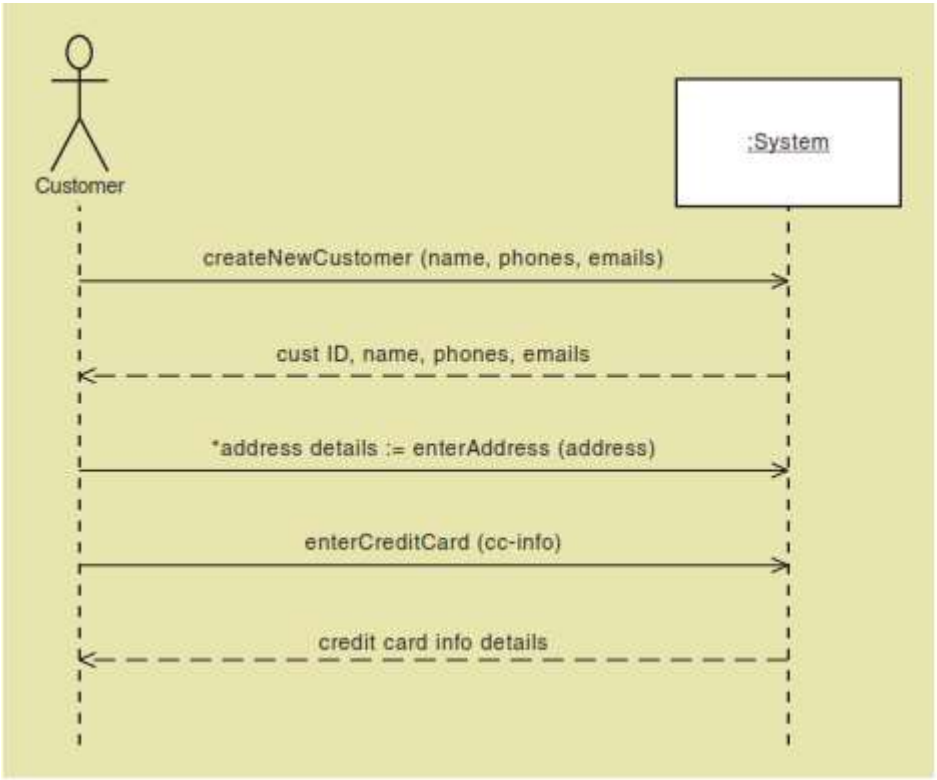
System Sequence Diagram adalah diagram yang mendokumentasikan *input-output* dan mengidentifikasi interaksi antara aktor dan sistem. Ini adalah alat yang efektif untuk membantu dalam desain awal antarmuka pengguna dengan mengidentifikasi informasi spesifik yang mengalir dari pengguna ke dalam sistem dan informasi yang mengalir keluar dari sistem kembali ke pengguna (Satzinger, Jackson, dan Burd, 2015).

Tabel 2.5 Daftar Simbol System Sequence Diagram *(Satzinger, Jackson, dan Burd, 2015)*

Simbol	Arti	Keterangan
	*Actor*	Orang yang menggunakan sistem dan juga bagaimana *actor* tersebut berinteraksi dengan sistem.
	*<<boundary>> (window)*	Berfungsi sebagai automatisai *boundary* sistem seperti *window input*.
	*Handler*	Menjadi penghubung diantara *boundary* dan *class*, serta juga berperan sebagai *switchboard* diantara *view layer* dan *domain layer.*
	*Object*	*Class-class* yang berhubungan dengan *use case* tersebut.
	*Data access*	*Database* dari sebuah *class*.

Simbol	Arti	Keterangan
	*Lifeline*	Ekstensi dari *object* maupun *actor* yang berupa garis-garis putus.
	*Activation*	Mengindikasikan sebuah *object* sedang mengeksekusi sebuah *Method.*
	*Input message*	Sebuah perintah atau pesan yang diinput *user* ke sistem.
	Output message	Pesan atau hasil yang dikembalikan atau diberikan kepada *user* dari sistem.
	*True/false condition (fragment)*	Sebuah bagian dari pesan antara *object* dan *user* yang dievaluasi sebelum pesan dikirim untuk memastikan bahwa pesan tersebut bisa dikirim.

System Sequence Diagram diatas menggambarkan data *input* dan *output* dari proses pengiriman barang pada sistem, dimana data *input* berupa data penjualan sesuai urutan, kemudian sistem akan member *output* data berupa data nama *Customer*, alamat, dan data barang yang dijual. Kemudian terdapat data *input* berupa data *id* pengirim, kemudian oleh sistem akan memberi *output* berupa *shipping confirmation*. Lalu data *input* selanjutnya adalah data ukuran barang dan beratnya, kemudian oleh sistem akan keluar *output* berupa label detail pengiriman.

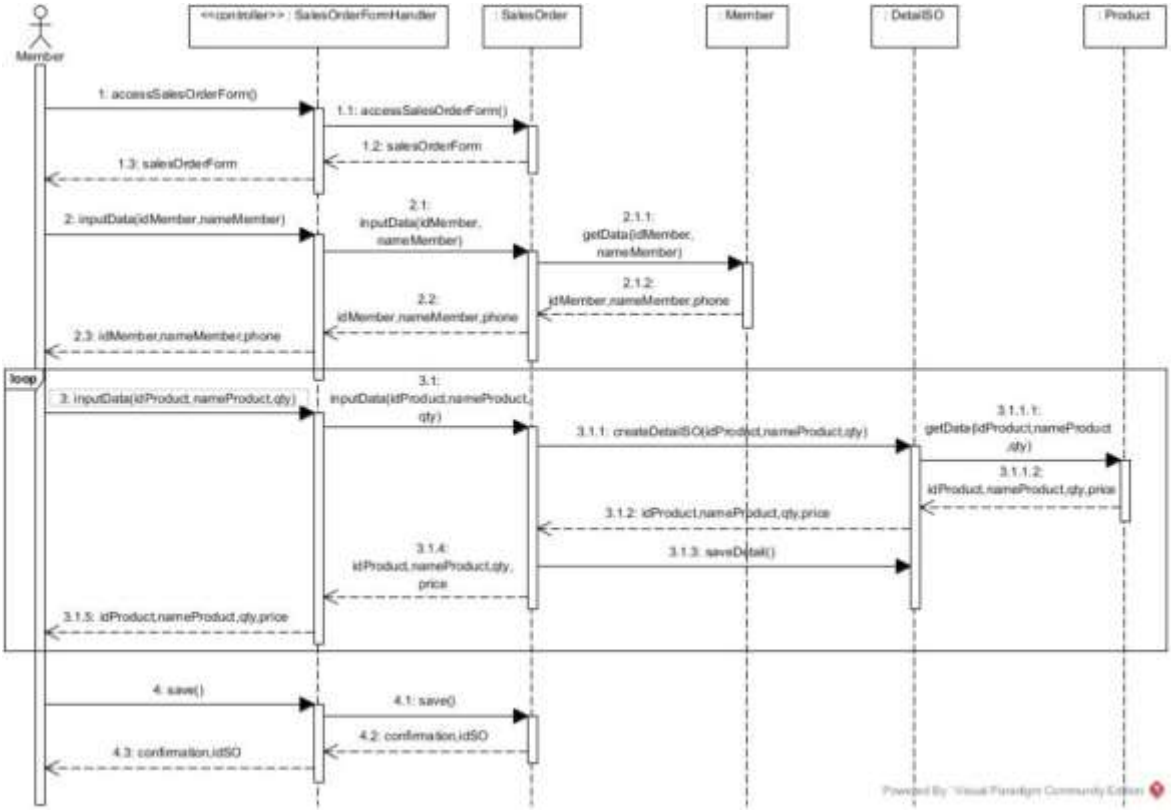


Gambar 2.8 Contoh Gambar *Sequence Diagram Create Customer Account.*

Dari gambar 2.8 adalah proses dimana *customer* membuat *account* dengan *CreateNewCustomer* yaitu mengisi identitas nama, *phone* dan *email* ke sistem. Kemudian sistem memberikan nomer id *customer* dan *customer* mengisi *Credit Card* maka sistem akan memunculkan detail *Credit Card Customer*.

2.15 *First Cut System Sequence Diagram*

First Cut System Sequence Diagram adalah diagram urutan terperinci yang menggunakan semua elemen yang menggunakan *Sistem Sequence Diagram* (SSD). Perbedaannya adalah objek sistem diganti oleh semua objek dan pesan *internal* dalam sistem.

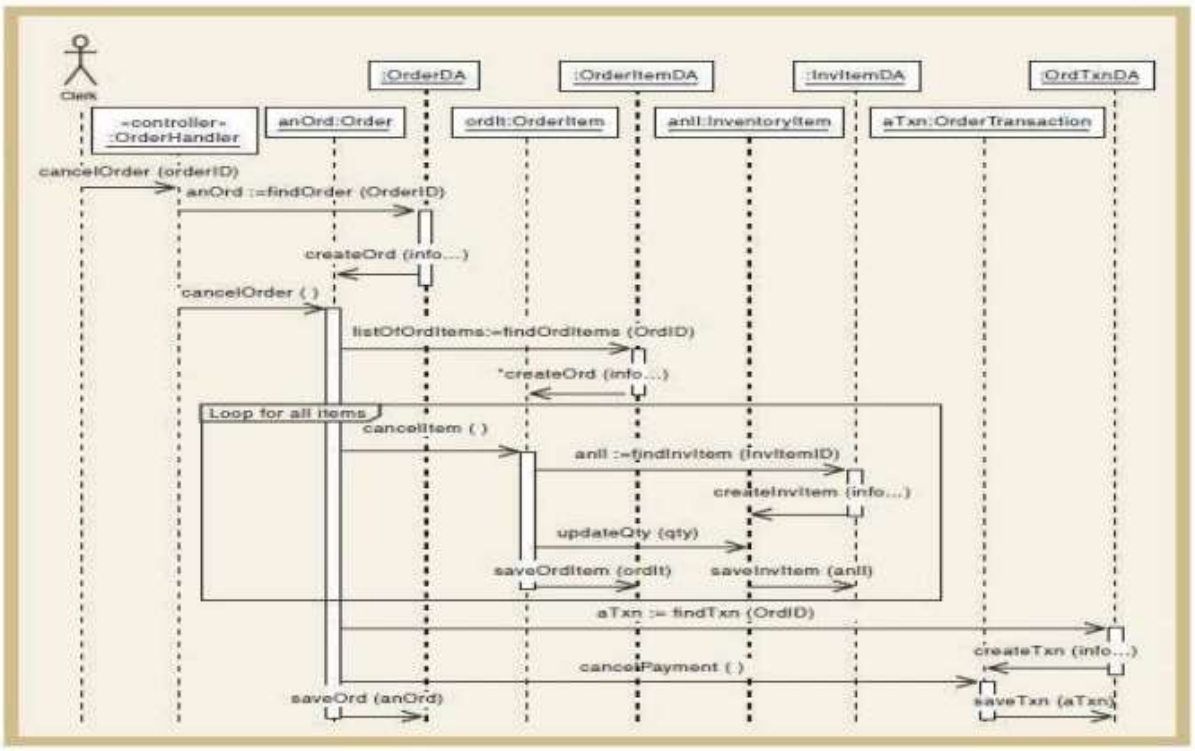


Gambar 2.9 Contoh *First-Cut System Sequence Diagram* Sistem Informasi Penjualan (Satzinger, Jackson, dan Burd, 2015).

Diagram diatas adalah contoh *First-Cut System Sequence Diagram* yang menggambarkan urutan terperinci dari semua objek dan sistem *internal* dalam rangkaian proses penjualan pada sistem informasi penjualan.

2.16 *Data Access Layer*

Prinsip pemisahan dari tanggung jawab juga diterapkan pada data *access layer*. Pada sistem yang lebih kecil terdapat perancangan 2 *layer*, dimana *statement SQL* digunakan untuk mengakses *database* tertanam *business logic layer*.

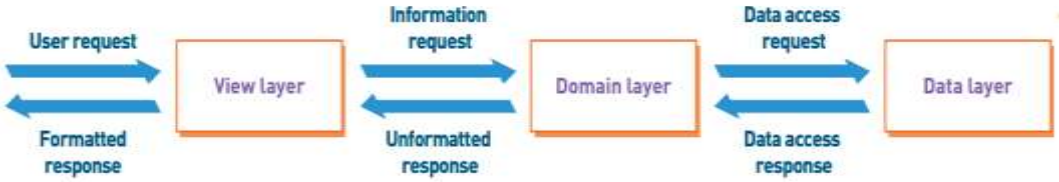


Gambar 2.10 Contoh Data *Access Layer* Sistem Informasi Penjualan
(Satzinger, Jackson, dan Burd, 2015)

2.17 *Three Layer*

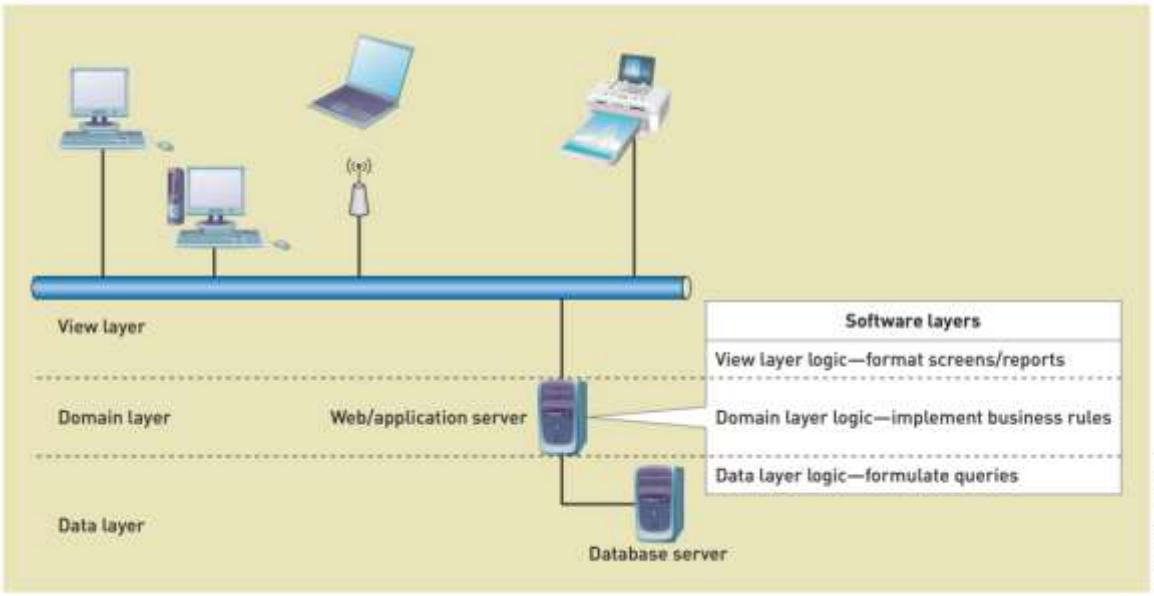
Three layer architecture adalah varian arsitektur klien/*server* yang digunakan untuk semua jenis sistem, dari aplikasi desktop yang digunakan secara *internal* hingga aplikasi berbasis *Web* yang didistribusikan secara *global*. *Three layer architecture* membagi perangkat lunak aplikasi menjadi tiga *layer* yang berinteraksi, seperti yang ditunjukkan Gambar 2.10 yaitu (Satzinger, Jackson, dan Burd, 2015):

- *User Interface* atau *view layer*, yang menerima *input* dan format pengguna dan menampilkan hasil pemrosesan
- *Business logic layer* atau *domain layer*, yang mengimplementasikan aturan dan prosedur pemprosesan bisnis
- *Data layer,* yang mengelola data yang disimpan, biasanya dalam satu atau lebih basis data



Gambar 2.11 Contoh *Three layer Architecture* (Satzinger, Jackson, dan Burd, 2015).

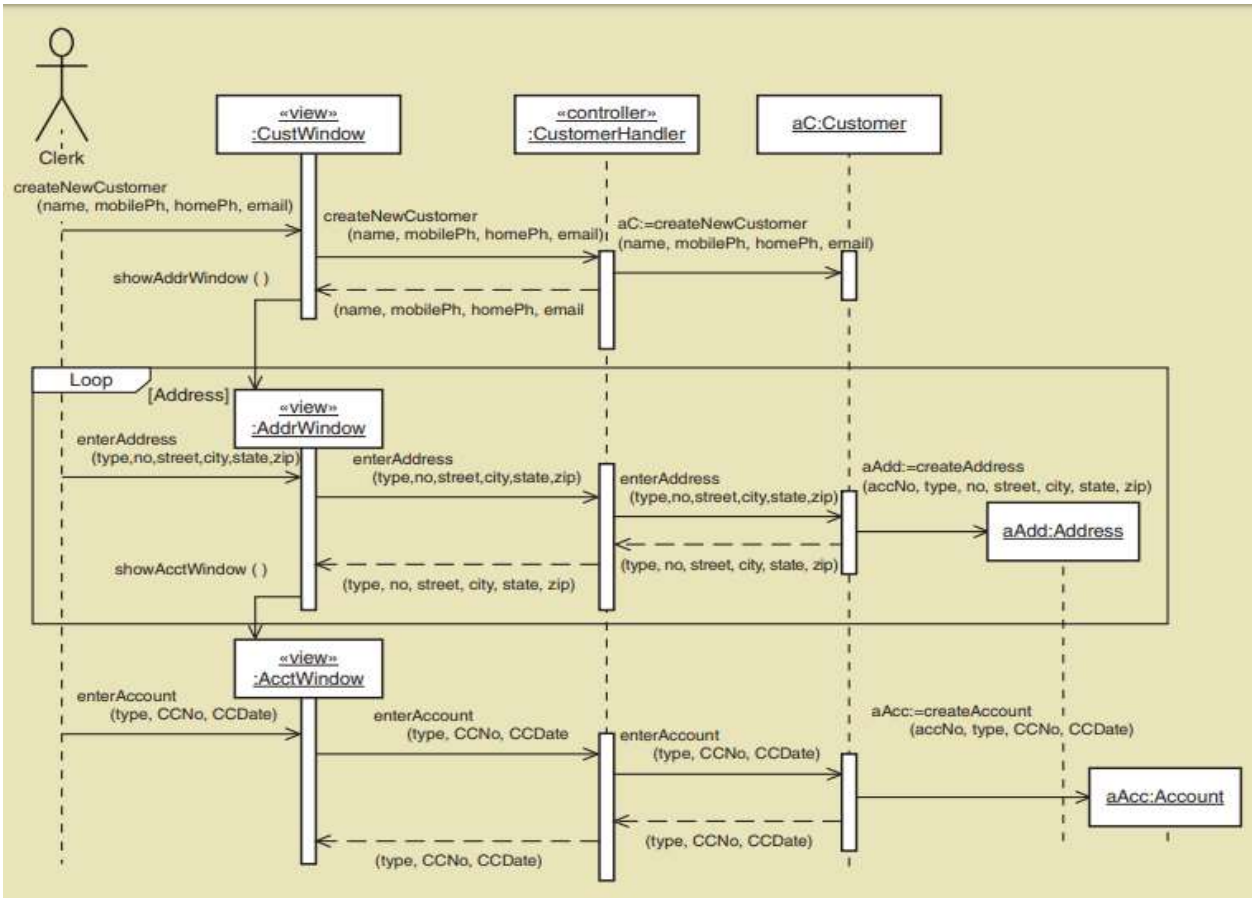
Gambar 2.11 mengilustrasikan sistem yang digunakan secara internal dengan *Three layer architecture*, dan ini menunjukkan bagaimana tiga *layer* tersebut dapat dikonfigurasi di tiga *platform* komputasi yang terpisah: klien *desktop* dan laptop, *server web* / aplikasi, dan *server database*.



Gambar 2.12 Contoh Ilustrasi *Three layer Architecture* (Satzinger, Jackson, dan Burd, 2015).

2.18 *Multi Layer*

Dalam desain *multi layer*, ada kelas *user-interface*, kelas *business logic*, dan kelas *data access*. Setiap *layer* memiliki fokus atau bidang tanggung jawab tertentu. Kelas-kelas yang memiliki fokus atau perhatian yang sama dikelompokkan bersama dalam satu *layer*. Prinsip desain ini memungkinkan fleksibilitas dalam penyebaran sistem karena *layer* yang berbeda, yaitu, pengelompokan kelas, dapat ditempatkan di komputer yang berbeda atau di lokasi yang berbeda (Satzinger, Jackson, dan Burd, 2015).

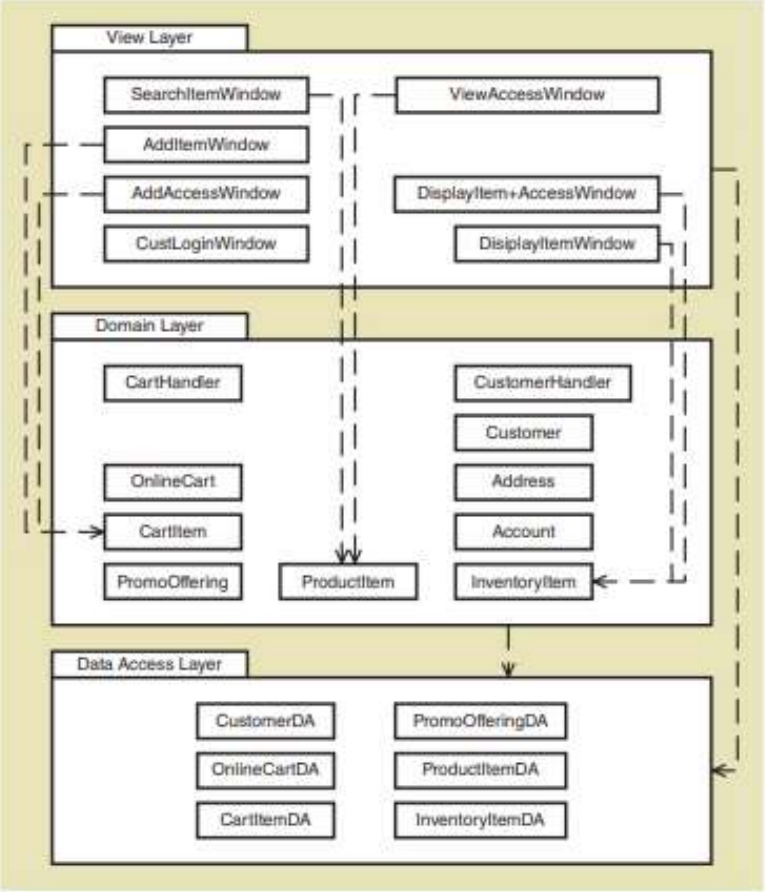


Gambar 2.13 Contoh *Multi Layer* dalam *use case diagram*
(Satzinger, Jackson, dan Burd, 2015).

Gambar 2.12 merupakan contoh *multi layer* dalam *use case* proses *input* data *Customer*, yang terdiri dari data *new Customer* berupa nama, nomor *mobile phone,* nomor telepon rumah, *email*. Kemudian juga *input* data alamat yang terdiri dari nama jalan, nomor rumah, kota, dan kode pos. Terakhir *input* data *Account* yang terdiri dari data *Type account,* nomor *account*, tanggal pembukaan.

2.19 *Packaged Diagram*

Diagram objek melengkapi notasi grafik untuk pemodelan objek, kelas dan relasinya dengan yang lain. Diagram objek bermanfaat untuk pemodelan abstrak dan membuat perancangan program. Untuk mengatur pengorganisasian diagram *Class* yang kompleks, dapat dilakukan pengelompokan kelas-kelas berupa *package* (paket-paket). *Package* adalah kumpulan elemen-elemen logika UML. Gambar di bawah ini mengenai model bisnis dengan pengelompokan kelas-kelas dalam bentuk paket-paket (Satzinger, Jackson, dan Burd, 2015).



Gambar 2.14 Contoh Ilustrasi *Package diagram* (Satzinger, Jackson, dan Burd, 2015).

2.20 *Persistent Object*

Persistent Object merupakan objek yang diingat oleh sistem dan bisa di gunakan dari waktu ke waktu (Satzinger, Jackson, dan Burd, 2015).

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

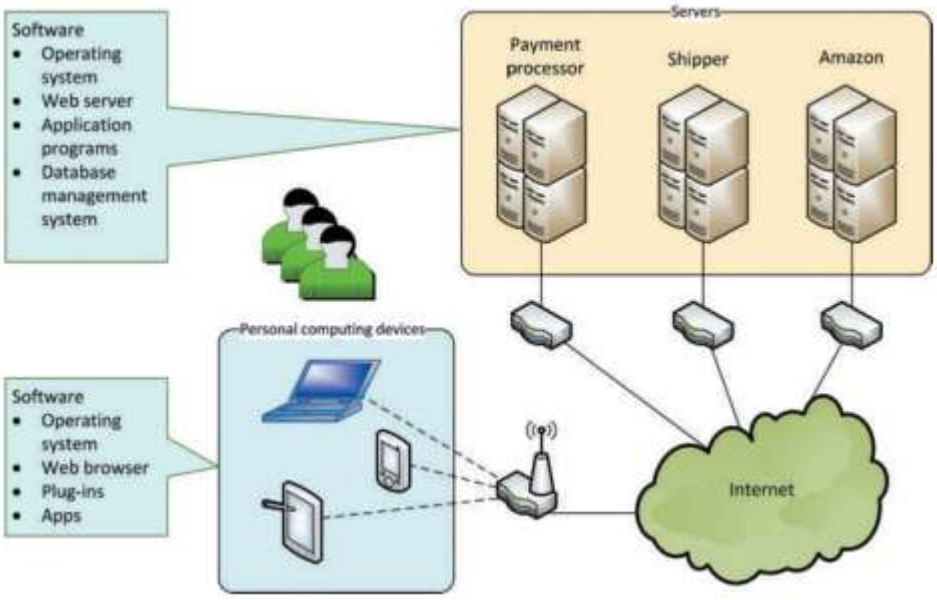
Gambar 2.15 Contoh Ilustrasi *Persistent Object* (Satzinger, Jackson, dan Burd, 2015).

Gambar 2.15 merupakan ilustrasi *Persistant Object*, dimana *object* ini merupakan data yang diingat oleh sistem dan selalu digunakan dari waktu ke waktu berupa data Katalog *ID*, *Product ID*, Harga barang, Harga *Special*. Data ini dapat diupdate, namun akan selalu data *primary* yang digunakan dalam berbagai proses dalam sistem.

2.21 *Website* (*WEB*)

Website atau disingkat *web*, dapat diartikan sekumpulan halaman yang terdiri atas beberapa laman yang berisi informasi dalam bentuk data *digital*, baik berupa teks, gambar, video, audio, dan animasi lainnya yang disediakan melalui jalur koneksi *internet* (Abdullah, 2016).

Aplikasi berbasis *web* merupakan perangkat lunak aplikasi yang menggunakan *web browser* sebagai antarmuka pengguna, memiliki *URL* untuk di akses aplikasi, menggunakan *web server* dan komponen perangkat lunak sisi *server*, dan menggunakan *web standards* untuk komunikasi antara *web browser* dan *server* (Satzinger, Jackson, dan Burd, 2015). Contoh Aplikasi bebrbasis *web* adalah seperti aplikasi belanja Amazon.



Gambar 2.16 Contoh Komponen perangkat lunak dari aplikasi belanja Amazon.com (Satzinger, Jackson, dan Burd, 2015)

2.21.1 PHP

PHP singkatan dari *Hypertext Preprocessor* yang merupakan *server-side programming*, yaitu bahasa pemograman yang diproses di sisi *server.* Fungsi utama PHP dalam membangun *website* adalah untuk melakukan pengolahan data pada *database.* Data *website* akan dimasukkan ke *database*, diedit, dihapus dan ditampilkan pada *website* yang diatur oleh PHP (Abdullah, 2016).

2.21.2 HTML

HTML singkatan dari *Hype Text Markup Language*, yaitu skrip yang berupa *tag-tag* untuk membuat dan mengatur struktur *website*. Beberapa tugas utama HTML dalam membangun *website* diantaranya sebagai berikut.

- Menentukan *layout* website
- Memformat teks dasar, seperti pengaturan paragraf dan format *font*
- Membuat *list*
- Membuat tabel
- Menyisipkan gambar, video dan audio
- Membuat *link*
- Membuat formulir

2.21.3 CSS

CSS singkatan dari *Cascading Style Sheet*, yaitu skrip yang digunakan untuk mengatur desain *website*. Walaupun HTML mempunyai kemampuan untuk mengatur tampilan *website*, namun kemampuannya terbatas. Fungsi CSS adalah memberikan pengaturan yang lebih lengkap agar struktur *website* yang dibuat dengan HTML terlihat lebih rapi dan elegan (Abdullah, 2016).

2.21.4 Javascript

Berbeda dengan PHP yang diproses di sisi *server*, *javascript* diproses pada komputer *client*. Karena pemrosesan dilakukan di komputer *client*. Membuat *javascript* lebih interaktif dibanding PHP. Peran *javascript* dalam membuat *website* adalah memberikan efek animasi yang menarik dan interaktif dalam penanganan *event* yang dilakukan oleh pengguna *website* (Abdullah, 2016).

2.22 *Database*

Database atau basis data itu sendiri merupakan tempat penyimpanan data yang besar jumlahnya terdiri dari kapasitas besar dan fungsi–fungsi yang ada di dalam *database*. Data-data yang ada di dalam perusahaan perlu disimpan dikarenakan data yang nantinya akan memberikan informasi bagi perusahaan harus terdata lengkap dan tidak boleh kurang atau hilang satupun karena data tersebut yang sudah melewati tahapan proses pengolahan data dapat memberikan nilai informasi tambah bagi perusahaan. *Database* atau basis data merupakan sekumpulan data yang terintegrasi sebagai tempat penyimpanan data, diatur dan dikontrol secara terpusat (Satzinger, Jackson, dan Burd, 2015).

2.22.1 MySQL

MySQL adalah sebuah *software database* yang merupakan tipe data relasional yang artinya *MySQL* menyimpan datanya dalam bentuk tabel-tabel yang saling berhubungan (Winarno, 2014). *MySQL* adalah *RDBMS (Relation Database Management System*) yang cepat dan mudah digunakan, serta sudah banyak digunakan untuk berbagai kebutuhan (Enterprise J.,2014).

2.22.2 Relational Database Management System (RDBMS)

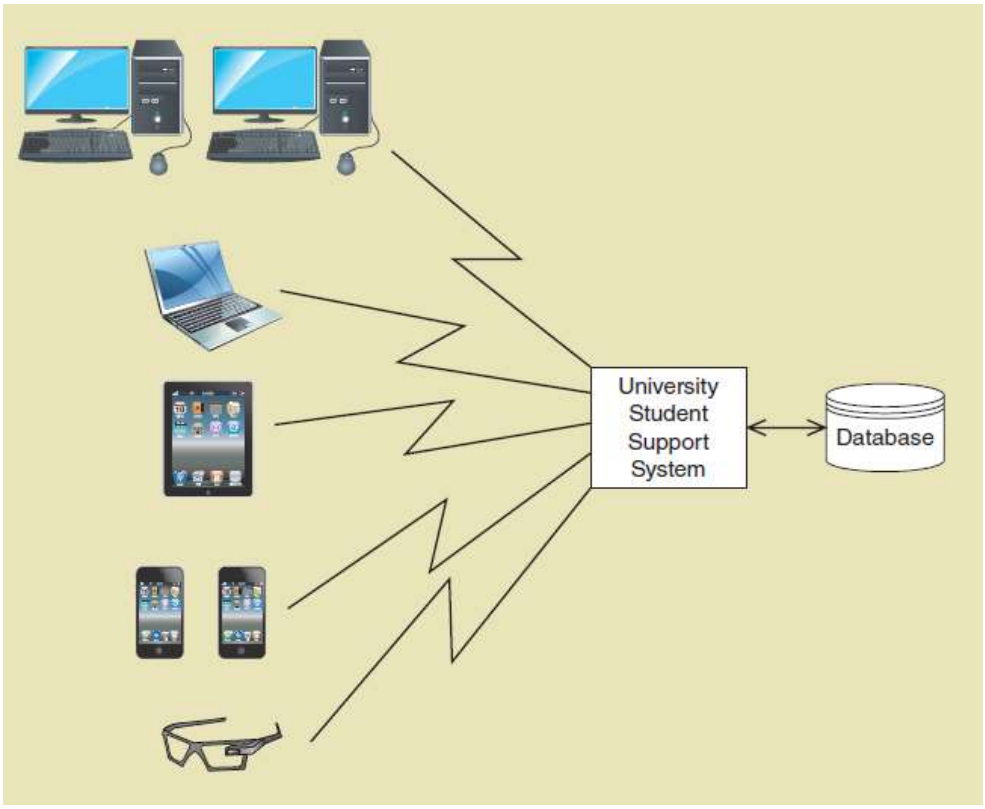
Relational Database Management System (RDBMS) adalah konsep yang digunakan untuk mengembangkan *database management system* (Priyadi, 2014). Sedangkan *Database Management System* (DBMS) adalah sistem perangkat lunak yang memungkinkan pengguna dapat mendefinisikan, membuat, merawat, dan mengatur akses ke *database* (Connolly dan Begg, 2015).

Terdapat 5 komponen utama dalam DBMS. Kelima komponen tersebut adalah *hardware, software*, data, prosedur, dan manusia (Connolly dan Begg, 2015). Contoh penggunaan DBMS ada banyak sekali dan dalam berbagai bidang kerja, misalnya akuntansi, manajemen sumber daya manusia, dan lain sebagainya.

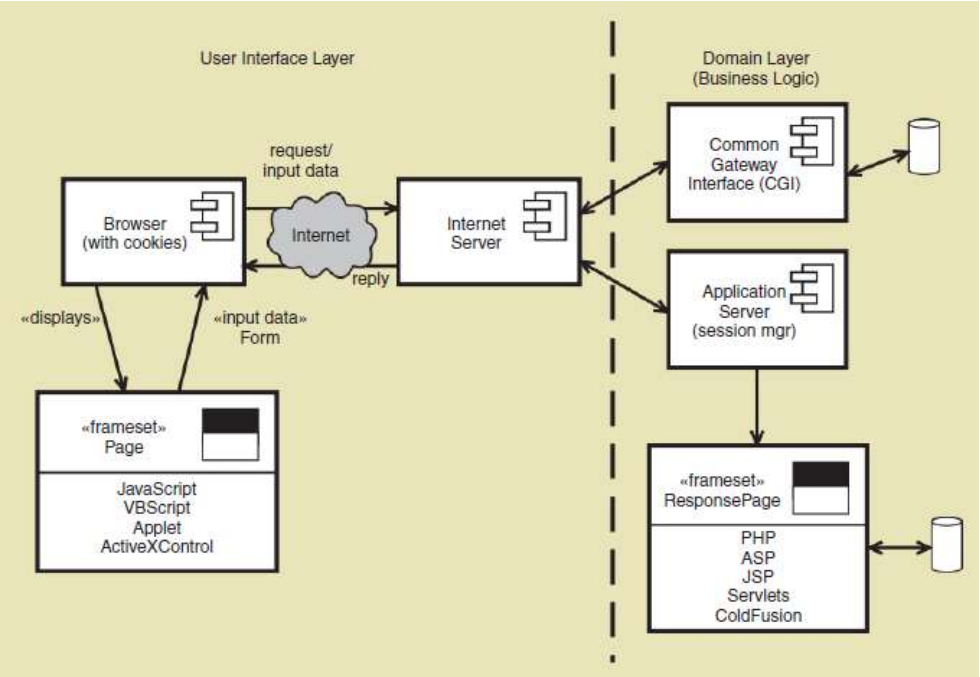
2.23 *System Interface*

System interface merupakan *input* dan *output* yang memerlukan campur tangan manusia. *System interface* memungkinkan sebuah inputan ditangkap secara

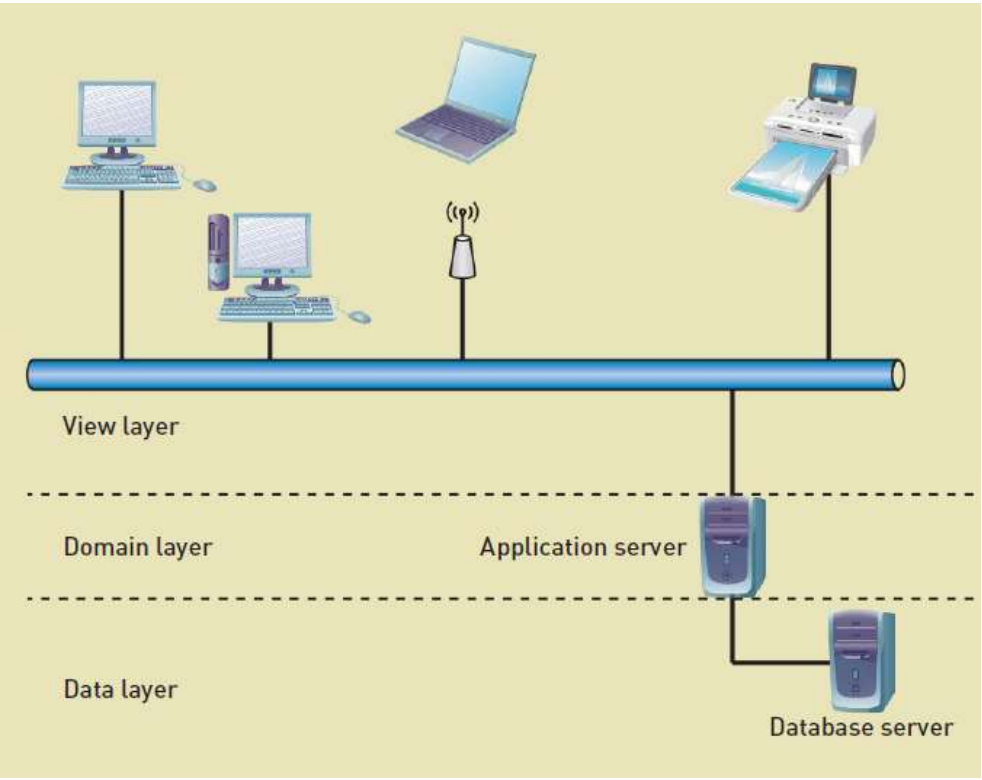
otomatis oleh perangkat *input* khusus seperti sistem *scanner*, sistem pesan elektronik ke atau dari sistem lain, atau transaksi juga bisa ditangkap oleh sistem lain (Satzinger, jakson, dan Burd, 2015).



Gambar 2.17 Contoh *System Interface* (Satzinger, jakson, dan Burd, 2015).



Gambar 2.18 Contoh *Component Diagram* (Satzinger, jakson, dan Burd, 2015).



Gambar 2.19 Contoh *Deployment Diagram* (Satzinger, jakson, dan Burd, 2015).

2.24 *User Interface*

User Interface adalah bagian dari sistem informasi yang membutuhkan interaksi dari *user* untuk membuat *input* dan *output* (Satzinger, jakson, dan Burd, 2015).

Beberapa kontrol *input* GUI pada umumnya menurut (Satzinger, Jackson, dan Burd 2015) adalah:

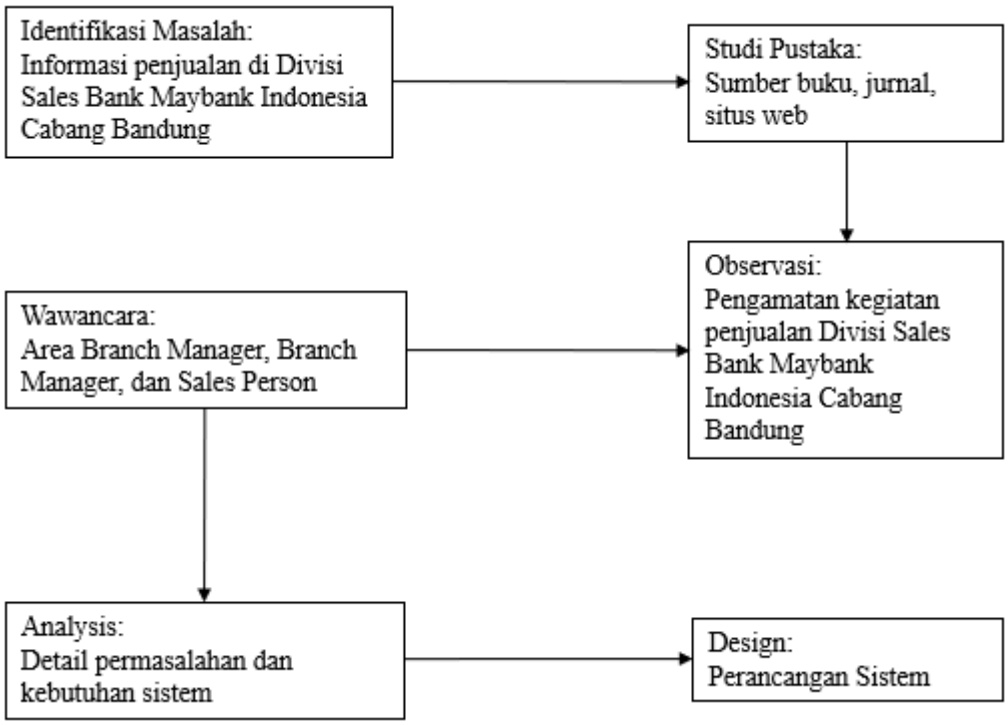
1. *Text Box*: kontrol *input* yang menerima entri data dari *keyboard*.
2. *List Box*: kontrol *input* yang berisi sebuah daftar entri yang diterima dimana pengguna dapat memilih.
3. *Spin Box*: sebuah variasi dari *list box* yang menyajikan beberapa entri dalam *text box* dimana pengguna dapat memilih.
4. *Combo Box*: variasi lain dari *list box* yang memungkinkan pengguna untuk memasukkan nilai baru atau memilih dari entri.
5. *Radio Buttons (Option Buttons)*: kontrol *input* yang memungkinkan pengguna untuk memilih satu pilihan dari suatu kelompok.
6. *Check Boxes*: kontrol *input* yang memungkinkan pengguna untuk memilih lebih dari satu pilihan dari suatu kelompok.

Gambar 2.20 Contoh *User Interface Product* Rocky Mountain Outfitter.

Gambar diatas menunjukkan contoh *user interface* untuk menu informasi produk *(product detail)* Rocky Mountain Outfitter, dimana didalam *user interface* tersebut dapat terlihat *Text box, List Box, Combo Box, Option Button,* dan *check box*.

2.25 Kerangka Pemikiran

Berikut ini merupakan tahapan penelitian dari penelitian sistem informasi *Daily Sales Activity Report* (DSAR):



Gambar 2.21 Kerangka Pemikiran.

1. Identifikasi Masalah

   Pada tahap ini, dilakukan proses pengidentifikasi informasi tentang masalah yang terjadi didalam proses bisnis pada divisi sales di PT. Bank Maybank Indonesia, Tbk. Metode yang digunakan pada penelitian ini untuk pengumpulan data sebagai berikut:

   a. Studi Pustaka

      Pengumpulan data dilakukan dengan cara mencari informasi yang dibutuhkan terkait topik dalam skripsi ini dari sumber tertulis berupa buku, jurnal, dan situs terkait.

   b. Observasi

Pengumpulan data dilakukan dengan cara mengamati langsung kegiatan proses bisnis yang terjadi di Divisi Sales Bank Maybank Indonesia cabang Bandung, terutama mengenai proses *daily activity sales* dan proses *monitoring sales activity* yang berjalan.

c. Wawancara

Pengumpulan data dilakukan dengan cara melakukan sesi tanya jawab dengan pihak yang berkaitan dengan proses *daily activity sales* dan proses *monitoring sales activity*, seperti *Area Branch manager, Branch manager, dan Sales person*

2. Metode Analisa dan Perancangan

a. Metode Analisa

Metode analisa yang digunakan pada penelitian ini adalah dengan pengembangan metode *waterfall*. Metode *waterfall* merupakan model pengembangan sistem informasi yang sistematik dan sekuensial. Metode *waterfall* memiliki tahapan – tahapan sebagai berikut:

1. *Project Initiation*

   Identifikasi permasalahan atau kebutuhan dan mendapatkan persetujuan untuk memulai project.

2. *Project Planning*

   Merencanakan dan memantau jalannya projek. Hal-hal yang perlu direncanakan terkait apa yang harus dilakukan (*what-to-do*), bagaimana melakukannya (*how-to-do-it*), dan siapa yang melakukannya (*who-does-it*).

3. *Analysis*

   Menemukan dan memahami detail dari permasalahan dan kebutuhan sistem.

4. *Design*

   Merancang komponen-komponen sistem guna menyelesaikan permasalahan atau memenuhi kebutuhan sistem.

5. *Implementation*

   Membangun sistem sesuai rancangan, melakukan uji coba, dan mengintegrasikan komponen-komponen sistem. Namun pada penelitian ini implementation tidak dilakukan tetapi hanya sampai tahap perancangan.

6. *Deployment*

   Tahap dimana sistem informasi dibuat tersedia bagi seluruh komunitas pengguna. Pada penelitian ini deployment tidak dilakukan, karena sistem informasi DSAR tidak dibuat.

b. Metode Perancangan

Metode perancangan yang digunakan dalam perancangan dan desain sistem informasi *Daily Sales Activity Report* (DSAR) menggunakan *Unified Modeling Language* (UML). UML adalah kumpulan struktur dan teknik untuk pemodelan desain program berorientasi objek (OOP). Diagram yang digunakan untuk menggambarkan perancangan sistem, yaitu *Activity Diagram, Use Case Diagram, use case description*, *brief use case description, fully developed use case description, domain model class Diagram,* dan *System Sequence Diagram*.

3. Perancangan Sistem Informasi

Perancangan sistem informasi merupakan usulan dan solusi untuk pemecahan masalah yang dihadapi PT. Bank Maybank Indonesia Area Cabang Bandung terkait masalah *monitoring Daily Sales Activity Report*. Perancangan sisitem ini meliputi *activity diagram, use case diagram*, *use case description*, *brief use case description, fully developed use case description, domain model class diagram, first cut design model class diagram, update domain model class diagram, system sequence diagram, first cut squence diagram, data access layer, three layer, multi layer, package diagram,* dan *persistent object*. Perancangan ini dibuat berdasarkan informasi yang telah didapat selama survey dan wawancara terhadap *Area Branch manager, Branch manager, Sales person* yang ada di lingkungan PT. Bank Maybank Indonesia Area Cabang Bandung.

2.26 Literature Review

1. Topik : Monitoring

   Judul : Jurnal Mantik Penusa (Volume 21 No 1 Juni 2017 ISSN 2088-3943) berjudul *Implementasi Daily Activity Monitoring System* (DAMS) pada CV. Jogja Media Telematika, hasil Penelitian Munfarida dan Astuti (2017).

Pembahasan : Sistem berbasiskan *web* yang menangani pencatatan aktifitas karyawan dan *monitoring* proyek. Sistem ini mengolah data menjadi suatu *output* laporan yang menampilkan informasi tentang aktifitas karyawan, *progress* proyek yang dikerjakan dan total waktu yang diperlukan dalam pengerjaan proyek. Topik dari jurnal ini memiliki kesamaan dengan topik penelitian yang dilakukan, yaitu sama membahas mengenai proses pelaporan dan *monitoring* aktifitas kerja. Namun perbedaanya jurnal ini berfokus pada aktifitas kerja *project*, sedangkan penelitian ini berfokus pada aktifitas kerja harian atau rutin, sehingga *output* yang dihasilkan berbeda. Dimana *output* berupa laporan data *leads* nasabah baru, data nasabah *hot leads* prospek dan *warm leads* prospek adalah fokus utama yang dibutuhkan oleh PT. Bank Maybank Indonesia.

2. Topik : M*onitoring*

   Judul : Jurnal *J-Intech*, *Journal of Information and Technology* (Volume 04 No 1 Tahun 2016 ISSN 2303-1425) berjudul Sistem Informasi Pemantau Kinerja *Sales* Memanfaatkan *Monitoring Geofencing* dan Teknologi *Cloud Message* Berbasis *Mobile*, hasil Penelitian Ari Prasetyo Suwandi (2016).

   Pembahasan : Sistem informasi pemantauan kineja *Sales* berbasis *mobile* yang di dalamnya juga termasuk sistem yang dapat mengirimkan posisi *Sales* dan data transaksi guna memantau keberadaan *Sales*, mengirimkan data transaksi, dan menerima informasi terbaru dari perusahaan secara langsung sehingga perusahaan dapat memantau aktivitas serta meberikan informasi keadaan stok suatu produk ke *Sales* dengan cepat. Topik dalam jurnal ini memiliki relevansi dengan topik penelitian yang dilakukan yaitu sama membahas mengenai sistem informasi pemantauan kinerja *Sales*. Namun terdapat beberapa perbedaan pada tujuan dan aplikasinya, dimana tujuan dari penelitian diatas lebih banyak kepada pengawasan terhadap pelanggaran wilayah kerja *marketing* (sehingga dilengkapi dengan teknologi GPRS) dan laporan yang dihasilkan akan berdampak ke arah *punishment*, sedangkan pada penelitian yang dilakukan lebih berfokus pada pengawasan strategi kerja harian (sehingga tidak memerlukan penggunaan teknologi GPRS) dan laporan yang dihasilkan bertujuan untuk memacu produktifitas.