BAB 2

LANDASAN TEORI

2.1 Sistem Informasi

Suatu sistem informasi (SI) (O'Brian & Marakas, 2010) dapat berupa kombinasi terorganisir orang, *hardware*, *software*, jaringan komunikasi, sumber daya data, dan kebijakan dan prosedur yang menyimpan, mengambil, mengubah, dan menyebarkan informasi dalam sebuah organisasi. Orang bergantung pada sistem informasi modern untuk berkomunikasi satu sama lain menggunakan berbagai perangkat fisik (*hardware*), instruksi pemrosesan informasi dan prosedur (*software*), saluran komunikasi (jaringan), dan data yang tersimpan (sumber data). Meskipun sistem informasi saat ini biasanya dianggap sebagai memiliki sesuatu untuk dilakukan dengan komputer, kami telah menggunakan sistem informasi sejak awal peradaban.

2.2 Internet

Menurut Shelly, Woods, & Dorin (2008:2) internet merupakan sekumpulan jaringan komputer yang ada di seluruh dunia yang menghubungkan jutaan komputer. Jaringan tersebut digunakan untuk organisasi, bisnis, pemerintahan, institusi pendidikan dan individu. Internet menghubungkan jutaan komputer melalui modem, saluran telepon, televisi, kabel, satelit, dan peralatan komunikasi lainnya.

2.3 Android

Android (Gargenta dan Nakamura, 2014:1-2) merupakan salah satu *platform open source* yang lengkap untuk perangkat *mobile*. *Android* dikembangkan oleh Google di bawah kepemilikan *Open Handset Alliance*. *Android* menyediakan seluruh peralatan dan *framework* lengkap secara gratis bagi para *developer* untuk pengembangan aplikasi berbasis *android* dengan cepat dan mudah. Para *developer* juga dapat dengan bebas memperluas dan mengembangkan

Android secara bebas serta menggunakannya untuk berbagai keperluan. *Android* merupakan *open platform* yang memisahkan antara *hardware* dan *software* yang berjalan di dalamnya. Dengan cara tersebut, *Android* memungkinkan lebih banyaknya jumlah perangkat yang menjalankan aplikasi yang sama dan menciptakan ekosistem yang lebih kaya bagi pengembang dan konsumen.

2.4 Software Development Kit (SDK)

Software Development Kit atau *SDK* (Schwarz, Duston, Steele, dan To, 2013:12) merupakan kumpulan dari *platform*, peralatan (*tools*), contoh kode, dan dokumentasi yang dibutuhkan dalam pembuatan applikasi *Android*. *SDK* merupakan *add-on* dari *Java Development Kit*(*JDK*) dan terintegrasi sebagai *plug-in* untuk *Eclipse Integrated Development Environment*(*IDE*). *Android SDK* tersedia untuk *Windows*, *Linux*, dan *Mac* dan dapat diunduh di http://developer.android.com/sdk secara gratis baik yang sudah terintegrasi dengan *Eclipse* maupun yang tidak terintegrasi dengan *Eclipse*, sehingga pembuat applikasi dapat mengunduh bagian yang dibutuhkan saja.

2.5 Android Development Tools(ADT)

Android Development Tools atau *ADT* (Deitel, Deitel, Deitel, 2013:60) merupakan extension dari *Eclipse IDE* yang memungkinkan pengembang untuk membuat, menjalankan, dan debug aplikasi-aplikasi *Android*. *ADT* juga berfungsi untuk mengubah aplikasi-aplikasi menjadi dokumen *APK* untuk kemudian didistribusikan, serta masih banyak lagi kegunaan lainnya. *ADT* menyediakan perlatan perancangan *GUI*, sehingga pengembang dapat memasukkan komponen *GUI* dengan cara drag dan drop tanpa perlu melakukan coding.

2.6 Eclipse

Eclipse (Brunette, 2005:1) merupakan *IDE* (*Integrated Development Environment*) yang dapat digunakan untuk membuat aplikasi dalam berbagai bahasa pemograman. Pada awalnya *Eclipse* digunakan untuk menggantikan *Visual Age for Java* yang dikembangkan

oleh *IBM*. Sejak November 2001, *Eclipse* menjadi *IDE open source* dan kini dikontrol oleh sebuah organisasi *non-profit Eclipse Foundation.*

2.7 eXtensible Markup Language(XML)

XML (Papilaya, 2005) adalah kumpulan aturan-aturan untuk mendefinisikan *structure* dan *semantic tags* yang akan memecah dokumen menjadi bagian-bagian kecil. Masing- masing *tag* memberikan indentitas untuk dokumen yang dilingkupinya. *XML* tidak jauh berbeda dengan *HTML*, namun memberikan kebebasan bagi pemakai untuk mendefinisikan *tags* yang dibutuhkan. *XML* tidak mendefinisikan format dari data, hanya mendefinisikan *structure* dan *semantic tags*, sedangkan *HTML* dapat digunakan untuk menentukan format.

2.8 *Hyper Text Markup Language*(*HTML5*)

HTML 5 adalah versi pertama yang memasukkan web sebagai *platform* untuk mengembangkan aplikasi web. *HTML* 5 mendefinisikan serangkaian elemen baru yang dapat digunakan untuk mengembangkan aplikasi web serta berbagai *API JavaScript* standar untuk *browser*. Contoh elemen baru *HTML* 5 adalah <video>, yang menyediakan sarana memutar konten video dalam browser tanpa memerlukan plug-in tambahan (*Crowther, Lennon, Blue, and Wanish* 2014:3).

HTML 5 juga menyediakan Media Element *Interface* yang memungkinkan untuk mengontrol pemutaran video dengan *JavaScript*. Memungkinkan untuk membuat game, membuat aplikasi mobile dan banyak lagi (*Crowther, Lennon, Blue, and Wanish* 2014:3)

2.9 Hypertext Preprocessor (PHP)

PHP bermula sebagai cara untuk mengelola situs web pribadi kecil. Pada awalnya *PHP* dijuluki *Personal Home Page Tools, PHP* dengan cepat berkembang selama bertahun-tahun dari mesin *scripting* dasar untuk website pribadi menjadi mesin yang sangat kompetitif dan dengan kode yang sangat kuat yang ditempatkan pada jutaan website di seluruh dunia.

PHP 5 diharapkan untuk mempertahankan dan bahkan meningkatkan kepemimpinan *PHP* di pasar pengembangan web. Tidak hanya itu, merevolusi berorientasi objek dukungan *PHP* tetapi juga berisi banyak fitur baru yang membuat platform pengembangan web utama. Ditulis ulang *XML* fungsi dalam *PHP* 5 menempatkan setara dengan teknologi web lainnya di beberapa daerah dan menyusul mereka pada orang lain, terutama karena perluasan *SimpleXML* baru yang membuatnya ridiculously mudah untuk memanipulasi dokumen *XML*. Selain itu, *SOAP* baru, *MySQL*, dan berbagai ekstensi lain tonggak penting dalam mendukung *PHP* untuk teknologi tambahan. (Babin, Good, Kromann, & Stephens, 2005).

2.10 Java

Java dikembangkan oleh tim yang dipimpin oleh James Gosling di *Sun Microsystems*. Sun Microsystems dibeli oleh *Oracle* pada tahun 2010. Awalnya disebut Oak, *Java* dirancang pada tahun 1991 untuk digunakan dalam chip dalam peralatan elektronik konsumen. Pada tahun 1995, berganti nama *Java*, ia dirancang untuk mengembangkan aplikasi Web. Untuk sejarah *Java*, lihat www.java.com/en/javahistory/index.jsp.

Java telah menjadi sangat populer. Ini peningkatan pesat dan penerimaan luas yang dapat ditelusuri dengan karakteristik desain, terutama janjinya yang dapat membuat Anda menulis sebuah program sekali dan menjalankannya di mana saja. Seperti yang dinyatakan oleh perancangnya, *Java* sangat sederhana, berorientasi pada objek, didistribusikan, ditafsirkan, kuat, aman, berarsitektur netral, portabel, kinerja yang tinggi, *multithreaded*, dan dinamis. Untuk anatomi karakteristik *Java*, lihat

www.cs.armstrong.edu/liang/JavaCharacteristics.pdf.

Java adalah bahasa pemrograman yang bertujuan umum berfitur lengkap yang dapat digunakan untuk mengembangkan aplikasi *mission-critical* yang kuat. Pada hari ini, *Java* digunakan tidak hanya untuk pemrograman web tetapi juga untuk mengembangkan aplikasi mandiri di *platform* pada server, komputer desktop, dan perangkat mobile. Itu digunakan untuk mengembangkan kode untuk berkomunikasi dengan

dan mengontrol robot penjelajah di Mars. Banyak perusahaan yang pernah beranggapan bahwa *Java* menjadi hype lebih dari substansi dan sekarang menggunakannya untuk membuat aplikasi terdistribusi yang dapat diakses oleh pelanggan dan mitra di Internet. Untuk setiap proyek baru yang sedang dikembangkan saat ini, perusahaan yang menanyakan bagaimana mereka dapat menggunakan *Java* untuk membuat pekerjaan mereka lebih mudah.

World Wide Web adalah repositori informasi elektronik yang dapat diakses di Internet dari mana saja di dunia. Internet, infrastruktur Web, telah ada selama lebih dari empat puluh tahun. Warna-warni *World Wide Web* dan *Web browser* canggih adalah alasan utama untuk popularitas Internet.

Java pada awalnya menjadi menarik karena program *Java* dapat dijalankan dari *browser* Web. Program tersebut disebut *applet*. *Applet* menggunakan antarmuka grafis modern dengan tombol, bidang teks, area teks, tombol radio, dan sebagainya, untuk berinteraksi dengan pengguna di Web dan memproses permintaan mereka. *Applet* membuat *Web responsif*, interaktif, dan menyenangkan untuk digunakan. *Applet* tertanam dalam file *HTML*. *HTML* (*Hypertext Markup Language*) adalah bahasa scripting sederhana untuk meletakkan dokumen, menghubungkan dokumen di Internet, dan membawa gambar, suara, dan video hidup di Web. Pada Hari ini, Anda dapat menggunakan *Java* untuk mengembangkan aplikasi Internet yang kaya. Sebuah *rich Internet application* (*RIA*) adalah aplikasi web yang dirancang untuk memberikan fitur yang sama dan fungsi biasanya terkait dengan aplikasi desktop.

Java sekarang sangat populer untuk mengembangkan aplikasi di server Web. Data-data proses aplikasi, melakukan perhitungan, dan menghasilkan halaman web yang dinamis. Banyak Website komersial dikembangkan menggunakan *Java* di *backend*.

Java adalah bahasa pemrograman serbaguna: Anda dapat menggunakannya untuk mengembangkan aplikasi untuk komputer desktop, server, dan perangkat genggam kecil. Perangkat lunak untuk ponsel *Android* dikembangkan menggunakan *Java*.

Java spesifikasi bahasa adalah definisi teknis sintaks bahasa pemrograman *Java* dan semantik. Anda dapat menemukan lengkap spesifikasi bahasa *Java* di http://docs.oracle.com/javase/specs/.

Aplikasi program interface (*API*), juga dikenal sebagai perpustakaan, berisi kelas yang telah ditetapkan dan antarmuka untuk mengembangkan program *Java*. *API* masih berkembang. Anda bisa melihat dan mengunduh versi terbaru dari *Java API* di http://download.java.net/jdk8/docs/api/.

Java adalah bahasa yang penuh dan kuat yang dapat digunakan dengan banyak cara. Muncul pada 3 edisi:

- *Java Standard Edition* (*Java* SE*)* untuk mengembangkan aplikasi *client-side*. Aplikasi dapat berjalan mandiri atau sebagai *applet* berjalan dari *browser* Web.
- *Java Enterprise Edition* (*Java EE*) untuk mengembangkan aplikasi *server-side*, seperti *servlets Java*, *JavaServer Pages* (*JSP*), dan *JavaServer Faces* (*JSF*).
- *Java Micro Edition* (*Java ME*) untuk mengembangkan aplikasi untuk perangkat mobile, seperti ponsel.

Buku ini menggunakan *Java SE* untuk memperkenalkan pemrograman *Java*. *Java SE* adalah fondasi yang semua teknologi lainnya java didasarkan. Ada banyak versi dari *Java SE*. Terbaru, *Java SE* 8, digunakan dalam buku ini. *Oracle* merilis setiap *versi* dengan *Java Development ToolKit* (*JDK*). Untuk *Java SE* 8, *Java Development ToolKit* disebut *JDK* 1.8 (juga dikenal sebagai *Java* 8 atau *JDK* 8).

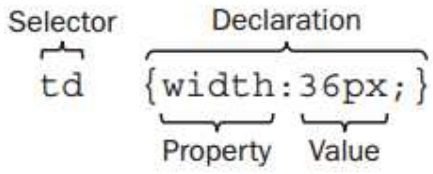
JDK terdiri dari satu set program yang terpisah, masing-masing dipanggil dari baris perintah, untuk mengembangkan dan menguji program *Java*. Alih-alih menggunakan *JDK*, Anda dapat menggunakan alat pengembangan *Java* (misalnya, *NetBeans*, *Eclipse*, dan *TextPad*) *software* aplikasi yang menyediakan *integrated development environment* (*IDE*) untuk mengembangkan program *Java* dengan cepat. *Editing*, kompilasi, membangun, *debugging*, dan bantuan online yang terintegrasi dalam satu antarmuka pengguna grafis. Anda cukup memasukkan kode sumber dalam satu jendela atau membuka file yang ada di jendela, kemudian klik tombol atau menu item atau menekan

tombol fungsi untuk mengkompilasi dan menjalankan program. (Y Daniel Liang,2015:10)

2.11 Cascading Style Sheets (CSS)

CSS (Duckett, 2010) bekerja dengan memungkinkan Anda untuk mengasosiasikan aturan dengan unsur-unsur yang muncul dalam halaman web. Aturan-aturan ini mengatur bagaimana isi dari elemen-elemen yang harus diberikan. Gambar 2-1 menunjukkan contoh aturan CSS, yang terdiri dari dua bagian:

- Pemilih, yang menunjukkan yang unsur atau elemen deklarasi berlaku untuk (jika itu berlaku untuk lebih dari satu elemen, Anda dapat memiliki koma - daftar dari beberapa elemen terpisah)
- Deklarasi, yang menetapkan bagaimana unsur-unsur sebagaimana dimaksud dalam pemilih harus ditata



(Sumber: Jon Duckett, 2010, hal.244)

Gambar 2.1 CSS

Aturan dalam Gambar 2.1 berlaku untuk semua <td> elemen dan menunjukkan bahwa mereka harus 36 pixel.

Deklarasi ini juga dibagi menjadi dua bagian, yang dipisahkan oleh titik dua:

- Sebuah properti, yang merupakan milik dari elemen yang dipilih (s) yang ingin Anda mempengaruhi, dalam hal ini properti lebar.
- Nilai, yang merupakan spesifikasi untuk properti ini; dalam hal ini adalah bahwa sel tabel harus 36 piksel lebar.

2.12 *Business Plans*

Sebuah *Business Plans* adalah dokumen yang bertujuan atau sebuah rencana tertulis untuk mencapai tujuan bisnis yang spesifik. Selain

tujuan dan struktur yang sama, *Business Plans* memiliki beberapa hal yang sama. Mereka dapat berkisar dari tiga halaman ringkasan ke ratusan halaman tujuan, rencana, dan data pendukung. (Ramsey Dan, 2009)

2.13 *Financial Data*

Finance merupakan pengelolaan dari uang dan aset lainnya. Data adalah potongan informasi. Sehingga data keuangan adalah informasi tentang uang. Anda akan membutuhkan banyak *Financial Data* untuk mengembangkan rencana bisnis Anda, dan bisnis Anda. (Ramsey Dan, 2009)

2.14 *Return on Investment*

Sebuah elemen kunci untuk *Business Plans* adalah menghitung *Return on Invesment* (ROI). ROI adalah rasio uang yang diperoleh atau kerugian pada suatu investasi relatif terhadap jumlah uang yang diinvestasikan. Misalnya, investasi sebesar $10.000 yang memperoleh keuntungan (atau bunga) sebesar $ 1.000 memiliki ROI sebesar 10 persen (1.000 / 10.000 = 0,10). Hasil tersebut adalah jumlah yang semua investor bisnis Anda cari. Mereka kemudian akan membandingkannya dengan ROI peluang dengan risiko yang sebanding. Jika mereka bisa mendapatkan ROI sebesar 10 persen pada investasi yang kurang berisiko (sedikit kesempatan loss), Mereka akan berinvestasi di sana. (Ramsey Dan, 2009)

2.15 *Income Statement*

Income Statement adalah dokumen keuangan yang menunjukkan bagaimana laba kotor (laba sebelum beban) diubah menjadi laba bersih (laba setelah biaya). Pendapatan kadang-kadang disebut garis atas dan laba bersih adalah garis bawah. Karena pernyataan itu menganggap *profits* dan *losses*, itu juga disebut sebagai *profits and losses* (P & L) *statement*. (Ramsey Dan, 2009)

2.16 *Cash-Flow Basics*

Cash flow (Ramsey Dan, 2009) adalah pelacakan pendapatan aktual dan biaya kas atau aset cair lainnya. Di masa sebelumnya, cash-flow statements disebut laporan kesempatan di posisi keuangan dan aliran pernyataan dana. Judul-judul ini membantu menjelaskan tujuan mereka sebagai catatan di mana uang itu berasal, di mana ia pergi, dan kapan. Cara lain untuk melihat pendapatan dan pengeluaran adalah:

- *Income = sources of cash*
- *Expense = uses of cash*

2.17 *Business Canvas Model*

Sebuah model bisnis menggambarkan pemikiran tentang bagaimana sebuah organisasi menciptakan, memberikan, dan menangkap nilai. (Osterwalder & Pigneur, 2009)

Model bisnis dapat digambarkan melalui sembilan blok bangunan dasar yang menunjukkan logika bagaimana sebuah perusahaan bermaksud untuk membuat uang. Sembilan blok meliputi empat bidang utama bisnis: pelanggan, penawaran, infrastruktur, dan kemampuan finansial. Model bisnis seperti cetak biru untuk strategi yang akan dilaksanakan melalui struktur organisasi, proses, dan sistem.

2.17.1 *Customer Segments*

Customer Segment Building Block mendefinisikan berbagai kelompok orang atau organisasi perusahaan bertujuan untuk menjangkau dan melayani *customer*.

Customer merupakan pusat dari setiap model bisnis. Tanpa *customer,* tidak ada perusahaan yang dapat bertahan lama. Dalam rangka untuk lebih memuaskan *customer,* perusahaan dapat mengelompokkan mereka ke dalam segmen yang berbeda dengan kebutuhan umum, perilaku umum, atau atribut lainnya.

Kelompok dari *customer* mewakili segmen terpisah jika:

- Kebutuhan mereka membutuhkan dan membenarkan tawaran yang berbeda
- Mereka dicapai melalui berbagai Saluran Distribusi
- Mereka membutuhkan berbagai jenis hubungan
- Mereka memiliki profitabilities substansial berbeda
- Mereka bersedia membayar untuk aspek yang berbeda dari tawara.

2.17.2 *Value Propositions*

Value Propositions Building Block menggambarkan bundel dari produk dan layanan yang menciptakan nilai untuk Segmen *customer* tertentu.

Value Propositions adalah alasan mengapa *customer* beralih ke satu perusahaan di atas yang lain. Ini memecahkan masalah *customer* atau memenuhi kebutuhan *customer*. Setiap *Value Propositions* terdiri dari bundel yang dipilih dari produk dan / atau jasa yang melayani kebutuhan dari segmen *customer* tertentu. Dalam pengertian ini, *Value Propositions* adalah agregasi, atau bundel, dari manfaat yang ditawarkan perusahaan untuk *customer*.

Beberapa *Value Propositions* mungkin inovatif dan mewakili tawaran *Value Propositions* baru atau mengganggu. Lainnya mungkin mirip dengan tawaran pasar yang ada, tetapi dengan fitur dan atribut yang ditambahkan.

2.17.3 *Channels*

Channels Building Block menggambarkan bagaimana sebuah perusahaan berkomunikasi dengan dan mencapai *Customer Segment* untuk memberikan *Value Propositions*.

Komunikasi, distribusi, dan penjualan *Channels* terdiri antarmuka perusahaan dengan *customer*. *Channels* adalah titik sentuh *customer* yang memainkan peran penting dalam pengalaman *customer*. *Channels* melayani beberapa fungsi, termasuk:

- Meningkatkan kesadaran di antara para *customer* tentang produk dan jasa perusahaan
- Membantu *customer* mengevaluasi perusahaan *Value Propositions*
- Membiarkan *customer* untuk membeli produk dan jasa yang
- Menyampaikan *Value Propositions* untuk pelanggan
- Memberikan pasca pembelian dukungan *customer*

2.17.4 *Customer Relationship*

Customer Relationship Building Block menjelaskan jenis hubungan perusahaan menetapkan dengan *Customer Segment* spesifik Sebuah perusahaan harus menjelaskan jenis hubungan yang ingin membangun dengan masing-masing *Customer Segment*. Hubungan dapat berkisar dari pribadi untuk otomatis. Hubungan *customer* dapat didorong oleh motivasi berikut:

- *Customer acquisition*
- *Customer retention*
- *Boosting sales (upselling)*

2.17.5 *Revenue Streams*

Revenue Stream Building Block merupakan kas perusahaan menghasilkan dari setiap *Customer Segment* (biaya harus dikurangi dari pendapatan untuk membuat laba) Jika pelanggan terdiri jantung model bisnis, *Revenue Stream* merupakan arterinya. Sebuah perusahaan harus

bertanya sendiri, Untuk apa nilai dari masing-masing *Customer Segment* yang benar-benar bersedia membayar? Berhasil menjawab pertanyaan yang memungkinkan perusahaan untuk menghasilkan satu atau lebih *Revenue Stream* dari setiap *Customer Segment*. Setiap *Revenue Stream* mungkin memiliki mekanisme harga yang berbeda, seperti harga tetap daftar, tawar, lelang, tergantung pasar, tergantung volume, atau manajemen hasil. Sebuah model bisnis dapat melibatkan dua jenis *Revenue Stream*:

- Pendapatan Transaksi yang dihasilkan dari pembayaran *customer* satu kali
- Pendapatan Berulang dihasilkan dari pembayaran berkelanjutan baik memberikan *Revenue Stream* untuk *customer* atau menyediakan pasca pembelian dukungan *customer*.

2.17.6 *Key Resources*

Key Resources Building Block menggambarkan aset yang paling penting yang dibutuhkan untuk membuat sebuah karya model bisnis.

Setiap model bisnis membutuhkan *Key Resources*. Sumber daya ini memungkinkan perusahaan untuk membuat dan menawarkan *Value Propositions*, menjangkau pasar, menjaga hubungan dengan *Customer Segment*, dan mendapatkan pendapatan. *Key Resources* berbeda diperlukan tergantung pada jenis model bisnis. Sebuah pabrik microchip membutuhkan fasilitas produksi padat modal, sedangkan desainer microchip lebih berfokus pada sumber daya manusia.

Key Resources berbentuk fisik, keuangan, intelektual, atau manusia. *Key Resources* dapat dimiliki atau disewa oleh perusahaan atau diperoleh dari *Key Partner*.

2.17.7 *Key Activities*

Key Activities Building Block menjelaskan hal yang paling penting perusahaan harus lakukan untuk membuat model bisnis kerja.

Setiap model bisnis akan mempunyai beberapa *Key Activities*. Ini adalah tindakan penting yang harus diambil bagi perusahaan untuk beroperasi dengan sukses. Seperti Key Resources, mereka diwajibkan untuk membuat dan menawarkan *Value Proposition*, menjangkau pasar, menjaga hubungan dengan pelanggan, dan mendapatkan pendapatan. Dan seperti *Key Resources*, *Key Activities* berbeda tergantung pada jenis model bisnis. Untuk pembuat perangkat lunak *Microsoft*, Aktivitas utama meliputi pengembangan perangkat lunak.

2.17.8 *Key Partnership*

Key Partnership Building Block menggambarkan jaringan pemasok dan mitra yang membuat model bisnis bekerja.

Perusahaan menjalin kemitraan untuk banyak alasan, dan kemitraan menjadi landasan banyak model bisnis. Perusahaan menciptakan aliansi untuk mengoptimalkan model bisnis mereka, mengurangi risiko, atau memperoleh sumber daya.

Kita dapat membedakan antara empat jenis kemitraan:

- Aliansi strategis antara non-pesaing
- *Coopetition*: kemitraan strategis antara pesaing
- *Joint venture* untuk mengembangkan bisnis baru
- Hubungan antara pembeli dan pemasok untuk memastikan pasokan yang dapat diandalkan

2.17.9 *Cost Structure*

Cost Structure menjelaskan semua biaya yang dikeluarkan untuk mengoperasikan model bisnis.

Blok Building ini menjelaskan biaya yang paling penting yang terjadi saat beroperasi di bawah model bisnis tertentu. Menciptakan dan memberikan nilai, menjaga hubungan dengan pelanggan, dan menghasilkan pendapatan dari semua biaya yang dikeluarkan. Biaya tersebut dapat dihitung relatif mudah setelah mendefinisikan *Key Resources*, *Key Activities*, dan *Key Partnership*. Beberapa model bisnis, meskipun, lebih *cost-driven* daripada yang lain. Disebut "*no frills*" penerbangan, misalnya, telah membangun model bisnis yang berada di sekitar Struktur Biaya rendah.

2.18 *Unified Modeling Language*(UML)

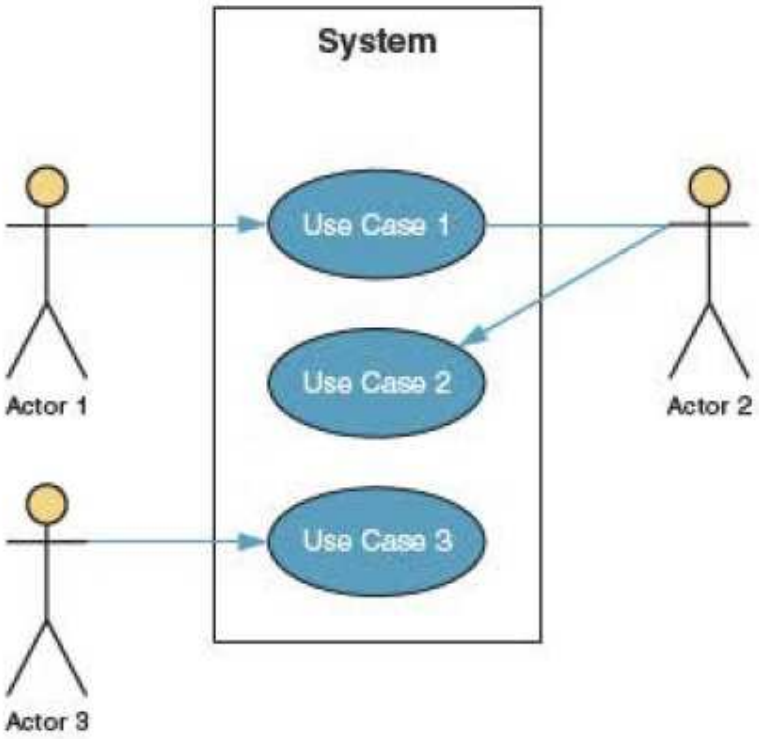
Unified Modeling Language(UML) adalah metode pemodelan yang digunakan untuk mendeskripsikan sebuah *software*(Whitten dan Bentley, 2007:371).

2.18.1 *Use Case Diagram*

Use Case Diagram merupakan diagaram yang menggambarkan sebuah sistem sebagai suatu hubungan dari *use case* dengan *actors* dan *relatioships* sebagai penghubung antar keduanya. *Use Case Diagram* merupakan metode yang cocok untuk menggambarkan interaksi yang jelas antara sistem dan pengguna (Whitten dan Bentley, 2007:246).

Komponen-komponen yang terdapat di dalam *Use Case Diagram*, antara lain:

- ***Use Case***



(Sumber: Whitten dan Bentley, 2007, Hal.247)

Gambar 2.2 Use Case

Use Case merupakan suatu komponen untuk mengidentifikasi dan mendeskrpsikan fungsi-fungsi dari sebuah sistem. Use Case digambarkan dengan bentuk elips yang dibuat secara horizontal dengan namanya yang berada di di atas/di bawah/di dalam elips tersebut. *Use Case* menjelaskan fungsi sistem dari perspektif pengguna sistem.

- ***Actors (User)***

(Sumber: Whitten dan Bentley, 2007, Hal.247)

Gambar 2.3 Actor

Actors menggambarkan pengguna yang berinteraksi dengan *use case* di dalam sistem. *Actors* digambarkan dengan sebuah figur berbentuk *stickman* dengan nama *Actors* yang berada di bawahnya.

- ***Relationship***

(Sumber: Whitten dan Bentley, 2007, hal.248)

Gambar 2.4 Relatioship

Relationship menggambarkan hubungan antara *Actors* dengan *Use Case*. *Relationship* digambarkan dengan sebuah garis yang menghubungkan figure *stickman* dengan *Use Case*.

2.18.2 *Use Case Narrative*

Member Services System

Author (s): ① **Date:** ② **Version:** ③

Use-Case Name:	Place New Order ④	**Use-Case Type** **Business Requirements:** ☐ ⑤
Use-Case ID:	MSS-BUC002.00 ⑥	
Priority:	High ⑦	
Source:	Requirement — MSS-R1.00 ⑧	
Primary Business Actor:	Club member ⑨	
Other Participating Actors:	• Warehouse (external receiver) • Accounts Receivable (external server) ⑩	
Other Interested Stakeholders: ⑪	• Marketing — Interested in sales activity in order to plan new promotions. • Procurement — Interested in sales activity in order to replenish inventory. • Management — Interested in order activity in order to evaluate company performance and customer (member) satisfaction.	
Description: ⑫	This use case describes the event of a club member submitting a new order for SoundStage products. The member's demographic information as well as his or her account standing is validated. Once the products are verified as being in stock, a packing order is sent to the warehouse for it to prepare the shipment. For any product not in stock, a back order is created. On completion, the member will be sent an order confirmation.	

hitten dan Bentley, 2007, Hal.257)

Gambar 2.5 Use Case Narative

Berdasarkan Whitten dan Bentley (2007:256-260), *Use Case Narrative* digunakan untuk mendokumentasikan detail dari setiap *Use Case* dan segala interaksi antara pengguna dengan sistem. *Use Case Narrative* mendeskripsikan interaksi yang terjadi dari cara *User* berinteraksi dengan sistem sampai kepada hasil akhir yang disebabkan oleh interaksi *User*. Detail yang terdapat di dalam *Use Case Narrative*, antara lain:

- *Author*: Nama dari Individu yang berkontribusi pada penulisan use case dan yang memberikan titik kontak bagi siapa saja yang membutuhkan informasi tambahan tentang kasus penggunaan.
- *Date*: Tanggal terakhir perubahan dari use case
- *Version*: Versi terbaru dari use case.
- *Use-case name*: nama dari use case harus mewakili tujuan yang penggunaan apakah kasus capai. Nama harus dimulai dengan kata kerja

- *Use-case Type*: dalam melakukan penggunaan use case model, kebutuhan bisnis menggunakan kasus, yang berfokus pada visi strategis dan tujuan dari berbagai stakeholders, yang dibangun pertama. Jenis kasus penggunaan adalah berorientasi bisnis dan mencerminkan pandangan tingkat tinggi dari perilaku yang diinginkan dari sistem.
- *Use-case ID*: Pengenal yang secara unik mengidentifikasi use case.
- *Priority*: Prioritas mengkomunikasikan pentingnya kasus penggunaan dalam hal tinggi, sedang, atau rendah.
- *Source*: Sumber mendefinisikan entitas yang memicu penciptaan use case. Ini bisa menjadi persyaratan, dokumen tertentu, atau stakeholder.
- *Primary business actor*: Pelaku usaha utama adalah stakeholder yang mendapat manfaat utama dari pelaksanaan use case dengan menerima sesuatu dari nilai yang terukur atau diamati.
- *Other participating actors*: Aktor lain yang berpartisipasi dalam use case untuk mencapai tujuan meliputi aktor yang memulai, memfasilitasi pelaku, pelaku server / penerima, dan aktor sekunder. Selalu sertakan cara di mana aktor berpartisipasi.
- *Interested stakeholder(s)*: Sebuah stakeholder adalah siapa saja yang memiliki saham dalam pengembangan dan pengoperasian sistem perangkat lunak. Sebuah stakeholder yang tertarik adalah orang (selain seorang aktor) yang memiliki kepentingan dalam tujuan dari use case.
- *Description*: Sebuah deskripsi ringkasan pendek yang terdiri dari beberapa kalimat yang menguraikan tujuan dari use case dan kegiatannya.
- *Precondition*: Sebuah syarat yang harus dipenuhi yang merupakan kendala pada keadaan sistem sebelum use case

dapat dieksekusi. Biasanya ini mengacu pada use case lain yang harus dijalankan sebelumnya.

- *Trigger*: merupakan pemicu dari peristiwa yang mengawali pelaksanaan use case. Hal ini sering merupakan tindakan fisik, seperti pelanggan berjalan ke counter penjualan atau memeriksa tiba surat yang datang . Waktu juga dapat memicu use case.
- *Typical course of event*: urutan normal kegiatan yang dilakukan oleh aktor dan sistem dalam rangka untuk memenuhi tujuan dari use case. Ini termasuk interaksi antara sistem dan aktor dan kegiatan yang dilakukan oleh sistem dalam menanggapi interaksi.
- *Alternate courses*: Alternate courses mendokumentasikan perilaku dari use case jika terjadi pengecualian atau variasi ketika kejadian khusus terjadi. Hal ini dapat terjadi ketika sebuah titik keputusan terjadi dalam use case atau pengecualian terjadi yang memerlukan langkah-langkah tambahan di luar lingkup kejadian khusus.
- *Post-condition*: merupakan kendala pada keadaan sistem setelah use case telah berhasil dilaksanakan. Ini bisa menjadi data yang tercatat dalam database atau tanda terima dikirim ke pelanggan.

2.18.3 *Activity Diagram*

Whitten dan Bentley (2007:390) menyatakan bahwa *activity diagram* adalah model diagram yang merepresentasikan alur dari proses bisnis, langkah dari *use case*, logika dari metode yang digunakan, atau aktivitas di dalam sistem. *Activity Diagram* digunakan untuk pengertian yang lebih baik dari alur dan urutan langkah dari *use case*. Terdapat beberapa notasi di dalam *activity diagram* yang merepresentasikan aktivitas yang terjadi di dalam sistem(Whitten & Bentley, 2007:391):

1. Initial node:

   Sebuah bulatan solid yang melambangkan permulaan dari sebuah proses

2. Actions:

   Setiap aksi diwakili oleh persegi panjang bulat yang mempunyai arti sebagai langkah individual. Aktivitas total dari diagram ditunjukkan dari diagram.

3. Flow:

   Bentuk panah pada diagram menunjukkan perkembangan melalui actions. Kebanyakan Flow tidak perlu kata-kata untuk mengidentifikasi mereka kecuali keluar dari keputusan.

4. Decision:

   Bentuk berlian dengan satu aliran masuk dan dua atau lebih flow keluar. Flow keluar ditandai untuk menunjukkan kondisi.

5. Merge

   Bentuk berlian dengan dua atau lebih arus datang, dan satu aliran keluar. Ini menggabungkan aliran yang sebelumnya dipisahkan oleh keputusan. Pengolahan berlanjut dengan salah satu aliran yang masuk ke penggabungan.

6. Fork

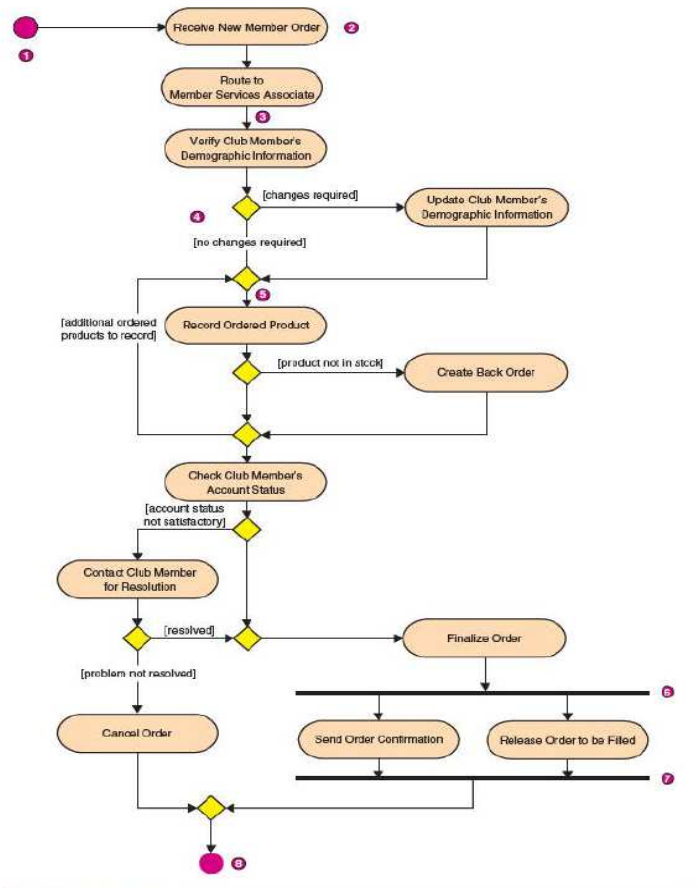
   Bar hitam dengan satu aliran masuk dan dua atau lebih arus keluar Actions arus paralel bawah garpu dapat terjadi dalam urutan atau secara bersamaan.

7. Join

   Bar hitam dengan dua atau lebih arus masuk dan satu aliran keluar, mencatat akhir pemrosesan konkuren. Semua tindakan yang masuk ke join harus diselesaikan sebelum pengolahan terus.

8. Activity Final

   Lingkaran padat di dalam lingkaran berlubang mewakili akhir proses.



(Sumber: Whitten dan Bentley, 2007, Hal.392)

Gambar 2.6 Activity Diagram

2.18.4 *Sequence Diagram*

Alat lain yang digunakan oleh beberapa *methodologists* dalam tahap desain logis adalah system sequence diagram. Seperti yang dibahas sebelumnya, *sequence diagram* menggambarkan bagaimana objek berinteraksi satu sama lain melalui pesan dalam pelaksanaan use case atau operasi. kita belum mulai menganalisis class objek individu; yang akan datang berikutnya seperti kita membangun versi pertama kami dari class diagram. Untuk saat ini kita masih berpikir tentang sistem secara keseluruhan.

Seperti yang telah dikatakan dunia berorientasi objek didorong oleh messages yang dikirim antara objek. *System sequence diagram* membantu kita untuk mulai mengidentifikasi

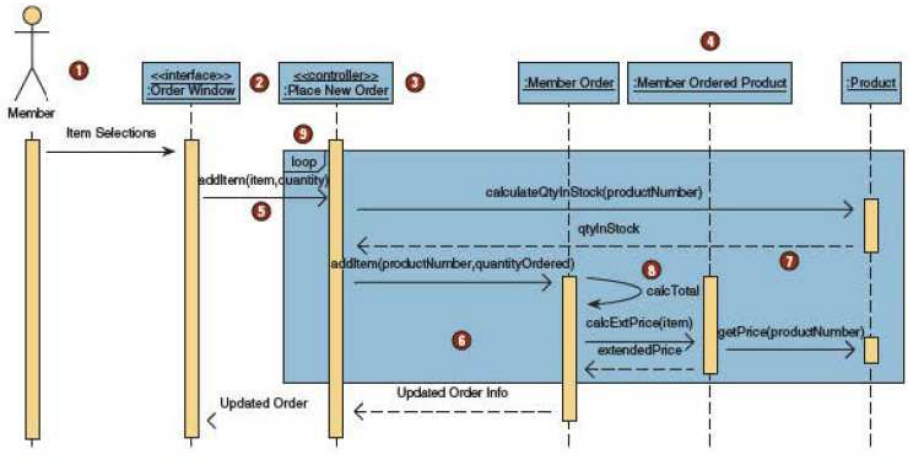
messages tingkat tinggi yang masuk dan keluar dari sistem. Kemudian *messages* tersebut akan menjadi tanggung jawab objek individu, yang akan memenuhi tanggung jawab tersebut dengan berkomunikasi dengan objek lainnya.

Gambar 2.7 menunjukkan system sequence diagram untuk use case Place New Orde. dicatat bahwa system sequence diagram tidak tidak termasuk salah satu program alternatif use case. system sequence diagram menggambarkan skenario tunggal, satu jalan melalui use case. sehingga set lengkap system sequence diagram mungkin memiliki beberapa diagram untuk kasus penggunaan tunggal.

1. *Actor* - actor berinteraksi dengan user interface yang ditampilkan dengan simbol actor use case. Kadang-kadang actor yang tinggalkan demi kesederhanaan. Kadang-kadang actor diwakili dengan kotak seperti kelas dengan notasi << actor >>. Garis vertikal putus-putus memanjang ke bawah dari aktor menunjukkan life of the sequence.
2. *Interface class*- kotak menunjukkan kode use case interface. Untuk memastikan tidak ada kebingungan mengenai apa class ini << interface >> dicatat. Seperti banyak hal dalam UML, apa pun berkomunikasi terbaik adalah titik dua (:) adalah notasi diagram urutan standar untuk menunjukkan berjalan "contoh" dari class. Garis vertikal putus-putus memanjang ke bawah dari class menunjukkan life of the sequence.
3. *Controller class* - setiap use case akan memiliki satu atau lebih Controller class digambar dengan notasi yang sama seperti interface class tercatat sebagai <<Controller >>.
4. *Entity Class* - menambahkan kotak untuk setiap entitas yang perlu berkolaborasi dalam urutan langkah. Lagi titik dua (:) menunjukkan sebuah contoh objek, dengan kata lain, urutan tertentu, produk tertentu, dan sebagainya.
5. *Messages* - panah horizontal padat menunjukkan input pesan dikirim ke class. Setiap messages menyebut perilaku

(atau metode) dari mana panah class mana panah mengarah. UML konvensi untuk message adalah mulai kata pertama dengan huruf kecil dan menambahkan kata-kata tambahan huruf besar awal dan tidak ada ruang. Dalam kurung, termasuk parameter yang perlu diteruskan, mengikuti konvensi penamaan yang sama dan memisahkan parameter individu dengan koma.

6. *Activation bars* - bar yang terbenam di jalur hidup menunjukkan periode waktu di mana setiap objek instance ada. Jika Anda sudah familiar dengan bahasa pemrograman berorientasi objek, Anda harus ingat objek menginisiasi untuk bekerja dengan mereka dalam program Anda. Activation bars menunjukkan masa sebuah contoh dalam RAM. Umumnya, objek yang dipakai dalam menanggapi pesan. Objek persisten akan, tentu saja, terus ada sebagai data yang tersimpan.
7. *Return messages* - panah horisontal putus-putus adalah Return messages. Setiap perilaku harus kembali sesuatu. Setidaknya benar / pesan palsu yang menunjukkan apakah perilaku itu berhasil. Tapi demi kesederhanaan, kembali pesan yang sering diasumsikan dan berada kiri dari diagram urutan.
8. *Self-call* - sebuah objek dapat memanggil metode sendiri.
9. *Frame* - cara menggunakan kotak Frame dalam diagram urutan sistem untuk menunjukkan bahwa satu atau lebih messages yang opsional (opt) langkah. Di sini kita menggunakan Frame untuk menunjukkan bahwa controller perlu loop melalui semua item.



(sumber: Whitten dan Bentley, 2007, hal.659)

Gambar 2.7 Sequence Diagram