BAB 2
LANDASAN TEORI

Pada bab ini dijelaskan mengenai bahan-bahan serta teori-teori pendukung penelitian. Setiap teori yang ada pada bab ini digunakan sebagai acuan dalam melakukan analisa dan pengembangan sistem untuk menjawab permasalahan yang telah disebutkan pada bab 1.

2.1 Aplikasi

Menurut Pramana (2012:17) aplikasi adalah satu unit perangkat lunak yang dibuat untuk melayani kebutuhan akan beberapa aktivitas seperti sistem perniagaan, *game*, pelayanan masyarakat, periklanan, atau semua proses yang hampir dilakukan manusia. Misalnya, termasuk perangkat lunak perusahaan, software akuntansi, perkantoran, grafis perangkat lunak dan pemutar media. Dapat disimpulkan bahwa aplikasi merupakan software yang berfungsi untuk melakukan berbagai bentuk pekerjaan atau tugas-tugas tertentu seperti penerapan, penggunaan dan penambahan data. Program aplikasi merupakan program siap pakai. Program yang direka untuk melaksanakan suatu fungsi bagi pengguna atau aplikasi yang lain. Contoh aplikasi ialah program pemproses kata dan *Web Browser*. Aplikasi akan menggunakan sistem operasi (OS) komputer dan aplikasi yang lainnya yang mendukung.

2.2 Aplikasi *Mobile*

Menurut Pressman dan Bruce (2014:9), aplikasi *mobile* adalah aplikasi yang telah dirancang khusus untuk *platform mobile* (misalnya *iOS*, *android*, atau *windows mobile*). Dalam banyak kasus, aplikasi *mobile* memiliki *user interface* dengan mekanisme interaksi unik yang disediakan oleh *platform mobile*, interoperabilitas dengan sumber daya berbasis web yang menyediakan akses ke beragam informasi yang relevan dengan aplikasi, dan kemampuan pemrosesan lokal untuk pengumpulan, analisis, dan format informasi dengan cara yang paling cocok untuk *platform mobile*. Selain itu aplikasi *mobile* menyediakan kemampuan penyimpanan *persisten* dalam *platform*. Perangkat *mobile* memiliki banyak jenis dalam hal ukuran, desain layout, tetapi mereka memiliki kesamaan karakteristik yang sangat berbeda dari *desktop system*. Perangkat *mobile* memiliki *memory* yang kecil.

2.3 Aplikasi Berbasis *Web*

Menurut O'Brien (2010:157), *Web service* merupakan komponen software yang berbasis *framework web* dan standar *object-oriented* dan teknologi untuk penggunaan web yang secara elektronik menghubungkan aplikasi *user* yang berbeda dan *platform* yang berbeda. *Web service* dapat menghubungkan fungsi bisnis untuk pertukaran data secara *real time* dalam aplikasi berbasis *web*.

Banyak dari perusahaan-perusahaan berkembang yang menggunakan Aplikasi berbasis *web* dalam merencanakan sumber daya mereka dan untuk mengelola perusahaan mereka. Aplikasi berbasis *web* ini menggunakan protokol HTTP, aplikasi di sisi server berkomunikasi dengan *client* melalui *Web server*. Aplikasi di sisi *client* umumnya berupa *web browser* jadi. Aplikasi berbasis *web (client / server-side script)* berjalan di atas aplikasi berbasis internet.

Menurut Simarmata (2010:185), aplikasi berbasis *web* adalah sistem perangkat lunak yang berdasarkan pada teknologi dan standar *World Wide Web Consortium* (W3C). Mereka menyediakan sumber daya *web* spesifik seperti konten dan layanan melalui sebuah antarmuka pengguna dan *browser web*.

2.4 *Dashboard*

Menurut Turban, et al. (2011:137), *Dashboard* adalah komponen yang umumnya memiliki *Performance Management Systems, Performance Measurement Systems, BPM Suites,* dan BI *Platforms*. *Dashboard* menyediakan tampilan visual dari informasi penting yang disatukan dan diatur dalam sebuah layar tunggal sehingga informasi dapat dipahami cukup dengan sekali lihat, serta mudah untuk dieksplorasi. Turban, et al. (2011:138) mengutip dari Eckerson (2006), yang adalah pakar yang terkenal dalam bidang BI dan *dashboard* secara umum, fitur yang paling dapat dibedakan dari sebuah *dashboard* adalah tiga lapisan informasinya, yaitu:

- *Monitoring*

  Data abstrak dan grafis untuk memonitor *key preformances metrics.*
- *Analysis*

  Dimensi data yang dirangkum untuk dianalisis akar permasalahannya.
- *Management*

  Data operasional yang terperinci yang mengidentifikasi tentang tindakan yang harus dilakukan untuk menyelesaikan masalah.

Adapun pengertian *Dashboard* menurut Johan, et al. (2004:93), *Dashboard* adalah sebuah aplikasi bagi *knowledge workers* yang berguna untuk menampilkan informasi yang sesuai dengan kebutuhan dalam satu layar. Informasi yang ditampilkan telah dipilah dan digabungkan dari berbagai sumber informasi yang ada. *Dashboard* merupakan bagian dari *Business Intelligence* yang secara langsung akan menampilkan berbagai informasi yang dibutuhkan oleh suatu organisasi atau perusahaan dengan beragam bentuk seperti grafik dan indikator warna yang dapat memudahkan pengguna dalam mengambil keputusan secara cepat dan tepat. *Dashboard* yang dibuat mampu menyediakan informasi-informasi yang dibutuhkan secara akurat bagi pihak *software maintenance*. *Dashboard* membuat suatu bagian di dalam suatu perusahaan agara lebih merespon setiap perubahan dan perkembangan yang ada. Semua informasi tersebut ditampilkan dengan tujuan agar para penggunanya mampu mengendalikan proses kerjanya dengan baik dan mampu membuat keputusan yang tepat pada saat dibutuhkan. Menurut Johan, et al. (2004:94), terdapat beberapa keuntungan dari *dashboard* antara lain:

1. Membuat pengguna dalam menentukan informasi yang utama dan penting. *Dashboard* membantu memilah informasi yang tidak diperlukan sehingga hanya menampilkan informasi yang penting.
2. Mendapatkan informasi yang telah terintegrasi. *Dashboard* dapat memberikan kemudahan dengan menampilkan informasi yang penting dalam satu layar.
3. Tetap mendapatkan informasi terkini meskipun tidak terhubung ke jaringan. Informasi tetap dapat diakses dari tempat yang berbeda baik secara *offline* maupun *online*.
4. Membantu meningkatkan kualitas dari keputusan yang diambil. Informasi yang ditampilkan secara cepat dan tepat akan membantu menghasilkan keputusan yang berkualitas.

2.5 Studi Literatur

Menurut Danial dan Warsiah (2007:80), studi literatur adalah teknik penelitian yang dilakukan oleh peneliti dengan mengumpulkan sejumlah buku, majalah, leaflet yang berkenaan dengan masalah dan tujuan penelitian. Dalam menggunakan teknik studi literatur, peneliti melakukan penambahan informasi melalui membaca dan mempelajari beberapa buku yang berhubungan dengan

penelitian yang sedang peneliti jalani, agar mendapatkan informasi lain yang dapat mendukung hasil dari penelitian tersebut.

2.6 Metode Pengumpulan Data

Menurut Gulo (2002:110), metode pengumpulan data merupakan aktivitas yang dilakukan guna mendapatkan informasi yang diperlukan dalam rangka mencapai tujuan dari suatu penelitian. Adapun tujuan penelitian adalah jawaban dari rumusan masalah ataupun hipotesis penelitian, untuk dapat menjawabnya diperlukan data atau informasi yang diperoleh melalui tahapan pengumpulan data. Informasi atau data mempunyai karakteristik yang berbeda beda sehingga membutuhkan metode yang berbeda-beda pula.

Metode pengumpulan data merupakan suatu hal yang penting dalam penelitian, karena metode ini merupakan strategi ataupun cara yang dipakai oleh peneliti guna mengumpulkan data yang dibutuhkan dalam penelitiannya. Pengumpulan data dalam penelitian dimaksudkan guna mendapatkan bahan, keterangan, kenyataan, dan informasi yang bisa dipercaya. Untuk mendapatkan data seperti yang dimaksudkan tersebut, dalam penelitian bisa dipakai berbagai macam metode, di antaranya yaitu dengan memakai angket, observasi, wawancara, tes, dan analisis dokumen.

2.6.1 Wawancara

Menurut Emzir (2010:50), wawancara ialah proses komunikasi atau interaksi untuk mengumpulkan informasi dengan cara tanya jawab antara peneliti dengan informan atau subjek penelitian. Dengan kemajuan teknologi informasi seperti saat ini, wawancara bisa saja dilakukan tanpa tatap muka, yakni melalui media telekomunikasi. Pada hakikatnya wawancara merupakan kegiatan untuk memperoleh informasi secara mendalam tentang sebuah isu atau tema yang diangkat dalam penelitian atau proses pembuktian terhadap informasi atau keterangan yang telah diperoleh lewat teknik yang lain sebelumnya.

Menurut Moleong (2012:187) ada bermacam-macam cara pembagian jenis wawancara yang dikemukakan dalam kepustakaan. Cara pembagian jenis wawancara yang dikemukakan oleh Patton (1980:197) sebagai berikut: (a) wawancara pembicaraan informal, (b) pendekatan menggunakan petunjuk umum wawancara,

dan (c) wawancara baku terbuka. Pembagian wawancara yang dilakukan oleh Patton didasarkan atas perencanaan pertanyaannya.

2.6.2 Observasi

Observasi merupakan suatu pengamatan dan pencatatan yang sistematis terhadap adanya gejala-gejala sosial yang diteliti, dimana harus sesuai dengan tujuan penelitian dan dicatat secara terstuktur sesuai yang diinginkan peneliti.

Menurut Gulo (2002:116), observasi merupakan metode pengumpulan data dimana peneliti mencatat informasi sebagaimana mereka saksikan selama penelitian. Observasi melibatkan dua komponen yaitu pelaku observasi atau *observer*, dan objek yang diobservasi atau *observed*. Observasi yang akan dilakukan oleh peneliti bertujuan untuk meninjau secara langsung kondisi nyata lapangan yang akan dijadikan pedoman awal sebelum melakukan penelitian sesungguhnya.

2.7 *Reimbursement*

Reimbursement, menurut kamus kamus *Oxford* adalah sebuah aksi pembayaran atau penggantian biaya kepada seseorang yang sudah mengeluarkan sejumlah uang. Pada sebuah perusahaan biasanya proses *reimbursement* terjadi ketika seorang karyawan melakukan sebuah kegiatan perusahaan dan mengeluarkan biaya dari uang pribadi dalam pelaksanaan kegiatan tersebut. Perusahaan nantinya akan mengganti biaya yang sudah dikeluarkan tersebut setelah menerima bukti pembayaran seperti bon, kuitansi, dll.

Dalam sebuah jurnal menurut (Guijun Liu dan Chunshan Yang, 2015) sistem manajemen *reimbursement* perusahaan terdiri dari empat modul: manajemen klaim (termasuk klaim dan audit), yang ingin membeli, yang akan mengklaim dan pusat informasi. Sistem manajemen reimbursement tersebut dikembangkan dengan *Struts*, *Spring* dan *Hibernate framework* dalam merancang sistem manajemen *reimbursement* yang digunakan oleh perusahaan. Dengan penggunaan *framework* tersebut, pengembangan sistem menjadi lebih sederhana, fleksibel serta pemeliharaan sistem juga lebih nyaman. Hal ini dapat menjadi dasar pertimbangan desain serta meningkatkan efisiensi pengembangan.

2.8 *Object Oriented Analysis and Design* (OOAD)

Menurut Satzinger, Jackson, dan Burd (2012:241), OOAD merupakan sebuah teknik pendekatan yang digunakan untuk menganalisa dan merancang sebuah aplikasi, sistem, atau bisnis dengan mendasarkan pada objek untuk mencapai komunikasi dan kualitas produk yang lebih baik.

Konsep OOAD mencakup *Object-Oriented Analysis* (OOA) dan *Object-Oriented Design* (OOD). OOA mendefinisikan objek yang bekerja dan memastikan interaksi pengguna (*use case*) dan apa saja yang dibutuhkan. Sedangkan OOD mendefiniskan semua tipe objek tambahan yang dibutuhkan untuk berkomunikasi dengan orang dan perangkat lain pada sistem, OOD menunjukan bagaimana suatu objek berinteraksi untuk menyelesaikan suatu tugas.

Unified Modeling Language adalah kumpulan dari model dan notasi yang didefinisikan *Object Management Group.* Dengan menggunakan UML, seorang analis dan *end user* dapat menggambarkan dan mengerti beragam jenis diagram yang digunakan dalam suatu proyek pengembangan sistem.

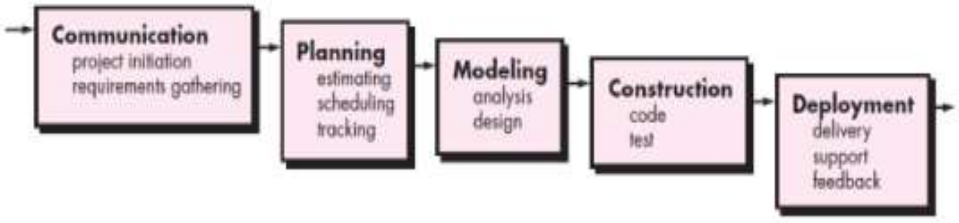
2.9 *Software Development Life Cycle* (SDLC)

Dijelaskan bahwa, "Software Development Life Cycle (SDLC) adalah pendekatan bertahap untuk melakukan analisa dan membangun rancangan sistem dengan menggunakan siklus yang spesifik terhadap kegiatan pengguna." (Kendall & Kendall, 2006).

2.9.1 Waterfall Model

Pressman (2010:39) menerangkan bahwa, "Model *waterfall*, kadang disebut *classic life cycle*, merupakan sebuah pendekatan sistematis, sekuensial untuk pengembangan *software* dimulai dengan spesifikasi kebutuhan *Customer* dan kemajuan melalui *planning*, *modeling*, *construction*, dan *deployment*, memuncak pada menjalankan *support* terhadap *software* yang telah komplit." Disebut *waterfall* (air terjun) karena fase demi fase yang dilalui harus menunggu selesainya fase sebelumnya dan harus berjalan secara urut. Model ini merupakan model yang paling tua dan paling banyak dipakai di dalam *Software Engineering* (SE). Model *waterfall* digunakan saat kebutuhan-kebutuhan yang diinginkan pada suatu produk sudah jelas. Dengan mengetahui dengan pasti teknologi dan sumber daya yang akan digunakan, maka proses perencanaan dapat dengan mudah diatur.

Terdapat 5 fase dalam model *waterfall*, Gambar 1 berikut ini menjelaskan urutan fase-fase model waterfall dari awal hingga akhir.



Gambar 1 Ilustrasi Proses *Waterfall* (Pressman, 2010:39)

1. *Communication*

Pada fase ini tugas utama yang dilakukan adalah mengumpulkan informasi untuk membantu menemukan kebutuhan pasti yang dibutuhkan oleh *Customer*. Setelah semua kebutuhan *Customer* terkumpul, langkah selanjutnya adalah *planning*/perencanaan.

2. *Planning*

Pada fase *planning*, aktivitas utama yang dilakukan adalah merencanakan jadwal, menjaga jalur proses, dan menyelesaikan estimasi-estimasi yang berhubungan dengan proyek. Fase *planning* mendeskripsikan bagaimana tugas-tugas secara teknis akan terjadi dan sumber daya apa saja yang akan dipakai juga bagaimana cara menggunakannya.

3. *Modeling*

Fase *modeling* merupakan fase yang penting karena arsitektur sistem didesain pada tahap ini. Analisis-analisis yang dibuat dikeluarkan dan dijadikan dasar untuk mendesain sebuah model *software*. Beberapa model untuk mengembangkan *software* dibentuk berdasarkan kebutuhan-kebutuhan yang dikumpulkan di fase pertama dan perancangan yang telah dilakukan sebelumnya pada fase kedua.

4. *Construction*

Penyelesaian semua *coding* dilaksanakan pada fase ini. *Coding* selesai dilakukan sesuai dengan model yang didesain pada tahap *modeling*. Jadi, pada fase *construction* ini *software* telah dikembangkan dan dicoba.

5. *Deployment*

Di fase terakhir ini, produk diluncurkan, disampaikan, dan dipasangkan pada *Customer*. Bantuan akan diberikan jika dibutuhkan. Pada fase *deployment*, umpan balik diterima dari *Customer* untuk menjamin kualitas produk agar tetap baik.

2.10 *Unified Modeling Language* (UML)

Menurut Widodo (2011:6), UML adalah bahasa pemodelan (modeling) standar yang memiliki sintak serta semantik. UML dapat berguna sebagai *blueprint*, sebab UML merupakan bahasa pemodelan yang sangat lengkap dan detail dalam perancangannya, sehingga nantinya akan diketahui informasi yang detail mengenai *coding* suatu program.

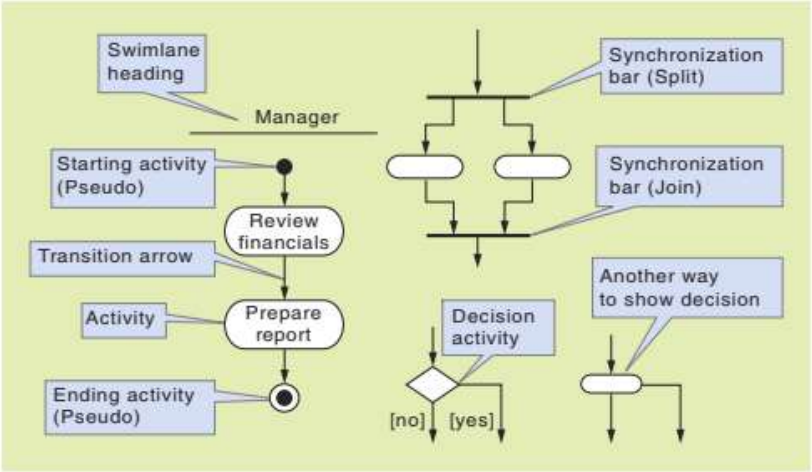
Dari pernyataan diatas, dapat ditarik kesimpulan bahwa UML merupakan sebuah bahasa yang berdasarkan grafik ataupun gambar yang bertujuan untuk menvisualisasikan, menspesifikasikan, membangun serta pendokumentasian dari sebuah sistem pengembangan perangkat lunak berbasis objek (*Object Oriented Programming*). Pemodelan ini perlu digunakan untuk menyederhanakan permasalahan-permasalahan yang kompleks sehingga lebih mudah dipelajari dan dipahami oleh semua *stakeholder*. Berikut penjelasan terkait contoh UML.

2.10.1 *Activity Diagram*

Menurut Satzinger, Jackson, dan Burd (2012:57), *activity diagram* merupakan diagram yang menunjukkan alur kerja atau aktivitas user secara berurutan. *Activity diagram* sendiri terdiri dari beberapa notasi dan fungsi kegunaan masing-masing. Berikut merupakan penjelasan notasi dan fungsi kegunaan *acitivity diagram* adalah sebagai berikut:

- *Swimlane*, merupakan suatu bentuk persegi yang merepresentasikan aktivitas-aktivitas yang diselesaikan setiap agen.
- *Synchronization Bar*, merupakan notasi yang berfungsi memisahkan (*split*) atau menyatukan (*join*) urutan jalur aktivitas.
- *Starting Activity* (*Pseudo*), merupakan notasi yang menunjukkan awal dimulainya suatu aktivitas.
- *Transition Arrow*, merupakan notasi yang berupa anak panah yang mendeskripsikan arah perpindahan suatu aktivitas.

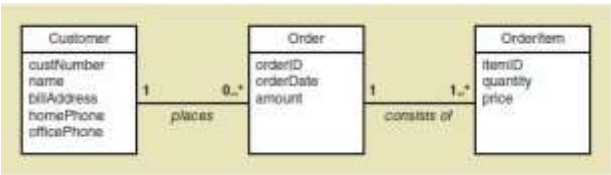
- *Activity*, merupakan notasi yang mendeskripsikan aktivitas-aktivitas.
- *Ending Activity* (*Pseudo*), merupakan notasi yang menunjukkan diakhirinya suatu aktivitas.
- *Decision Activity*, merupakan notasi yang mendeskripsikan kondisi dari suatu aktivitas.



Gambar 2 *Activity Diagram* (Burd Satzinger, Jackson & Burd, 2012)

2.10.2 *Class Diagram*

Menurut Satzinger, Jackson, dan Burd (2012:101) mengatakan bahwa *class diagram* adalah sebuah diagram UML yang terdiri dari *class* serta hubungan antar *class* tersebut. Pada *class diagram,* bentuk kotak menggambarkan *class* dan garis menunjukkan hubungan antar kelas. *Class diagram* digunakan untuk memahami hubungan antar *class* yang terdiri dari beberapa objek di dalam pengembangan dan perancangan sistem. Gambar berikut menunjukkan salah satu contoh *class diagram*.



Gambar 3 *Class Diagram* (Burd Satzinger, Jackson & Burd, 2012)

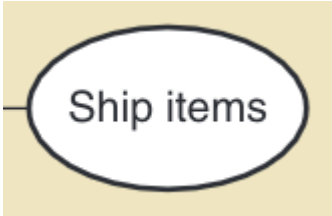
2.10.3 *Use Case Diagram*

Menurut Satzinger, Jackson, & Burd (2012:78) mengatakan bahwa *Use Case Diagram* adalah salah satu model UML yang digunakan untuk menunjukkan berbagai peran *user* dan bagaimana *user* berinteraksi dengan sistem. *Use case*

diagram digunakan untuk memodelkan konteks sistem dengan melampirkan semua kegiatan sistem berinteraksi dengan aktor di luar sistem, memodelkan *requirement* sistem dari sudut pandang di luar sistem. *Use case diagram* terdiri dari *use cases*, aktor, dan relasi.

1. *Use cases*

Mengidentifikasi dan menggambarkan fungsi-fungsi yang terdapat pada sistem. *Use case* digambarkan dalam bentuk elips horizontal dengan nama *use case* ditulis di dalam elips. *Use case* dinyatakan dengan menggunakan kata kerja.



Gambar 4 *Use Case* (Burd Satzinger, Jackson & Burd, 2012)

2. Aktor

Aktor merupakan abstraksi dari orang, sistem lain, organisasi, atau *external device* yang mengaktifkan fungsi-fungsi yang terdapat pada sistem. Mengidentifikasi aktor dapat dilakukan dengan menentukan pembagian tenaga kerja dan tugas-tugas yang berkaitan dengan tenaga kerja tersebut.

Gambar 5 Aktor (Burd Satzinger, Jackson & Burd, 2012*)*

3. Relasi

Relasi digunakan untuk menggambarkan hubungan antar komponen pada *use case* diagram. Terdapat beberapa jenis relasi pada *use case* diagram yaitu:

a. Asosiasi

Asosiasi pada relasi *use case* diagram terbagi menjadi dua yaitu:

- Asosiasi antara aktor dan *use case* digambarkan dengan garis tanpa panah mengindikasikan siapa atau apa yang meminta interaksi secara langsung.

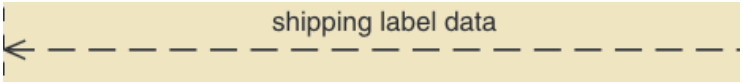
Gambar 6 Asosiasi Tanpa Panah (Burd Satzinger, Jackson & Burd, 2012)

- Asosiasi antara aktor dengan *use case* yang menggunakan panah terbuka untuk mengindikasikan bahwa aktor berinteraksi secara pasif dengan sistem.

Gambar 7 Asosiasi Dengan Panah (Burd Satzinger, Jackson & Burd, 2012)

b. Extends

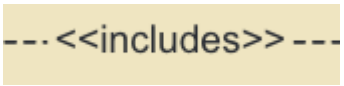
Extends merupakan perluasan dari *use case* lain jika kondisi yang sudah ditentukan atau syarat telah terpenuhi.



Gambar 8 *Extends* (Burd Satzinger, Jackson & Burd, 2012)

c. Include

Include merupakan pemanggilan *use case* oleh *use case* lain.



Gambar 9 *Include* (Burd Satzinger, Jackson & Burd, 2012)

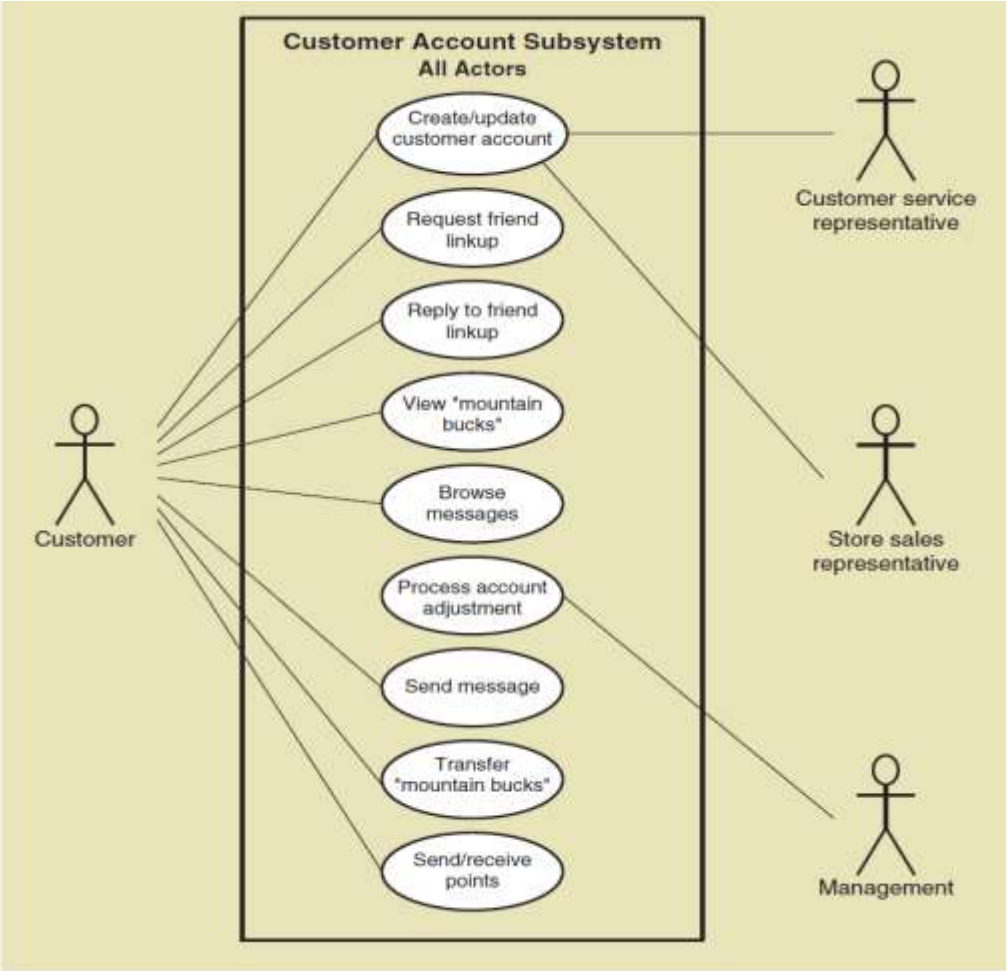
d. *System Boundary*

System Boundary adalah sebuah kotak yang mewakili batas-batas sistem. Di dalam kotak adalah semua *use cases* yang ada pada sistem dan di luar kotak adalah aktor yang berinteraksi dengan sistem. Sistem juga dapat dibagi menjadi sub-unit dengan menggunakan kotak di dalam kotak.

Menurut Satzinger, Jackson, & Burd (2012:83), langkah-langkah dalam membangun sebuah *use case diagram* adalah sebagai berikut:

1. Megidentifikasi semua *stakeholder* dan pengguna yang memiliki kebutuhan atau mendapatkan manfaat dari *use case diagram*.
2. Menentukan kebutuhan setiap *stakeholder* dan pengguna dalam meninjau kegunaan dari *use case diagram*. Biasanya, setiap *use case diagram* dibuat dari setiap subsistem, setiap tipe pengguna, *use cases* yang menggunakan relasi <<includes>>, dan dari *use cases* yang digunakan untuk stakeholder yang spesifik.
3. Untuk setiap kebutuhan komunikasi potensial, pilih *use cases* dan aktor untuk menunjukkan dan menggambar *Use Case Diagram*. Ada banyak paket perangkat lunak yang dapat digunakan untuk menggambar menggunakan *Use Case Diagram*.
4. Tentukan nama dari masing-masing diagram *Use Cases* dan kemudian perhatikan bagaimana dan kapan diagram harus digunakan untuk meninjau kasus penggunaan oleh *stakeholder* dan pengguna.

Use case diagram menunjukkan gambaran umum dari setiap *use case* pada sistem yang dapat dilihat pada gambar berikut.



Gambar 10 *Use Case Diagram* (Burd Satzinger, Jackson & Burd, 2012)

2.10.4 *Use Case Description*

Untuk melihat detail informasi yang ada pada setiap *use cases* maka digunakan *Use Case Descriptions*. *Use case descriptions* berisi informasi detail dari setiap *use case* yang ada pada sistem (Satzinger, Jackson, & Burd, 2012:121). *Use case descriptions* dibagi menjadi 3 tingkatan detail:

1. *Brief Use Case Descriptions*

   Brief Use Case Descriptions digunakan untuk mendeskripsikan *use case* secara sederhana. *Brief Use Case Descriptions* digunakan untuk aplikasi sederhana yang memiliki *scenario* sederhana (Satzinger, Jackson, & Burd, 2012:121). Gambar berikut menunjukkan contoh *Brief use case descriptions*.

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

Gambar 11 *Brief Use Case Description* (Burd Satzinger, Jackson & Burd, 2012)

2. *Intermediate Description*

Intermediate description merupakan *use case description* yang mengembangkan *brief description* untuk menyertakan aliran internal dari aktivitas sebuah *use case*. *Exception* dapat didokumentasikan jika dianggap perlu. Gambar berikut menunjukkan salah satu contoh dari *intermediate description*.

Flow of activities for scenario of *Order Clerk creates telephone order*

Main Flow:

1. Customer calls RMO and gets order clerk.
2. Order clerk verifies customer information. If a new customer, invoke *Maintain customer account information* use case to add a new customer.
3. Clerk initiates the creation of a new order.
4. Customer requests an item be added to the order.
5. Clerk verifies the item and adds it to the order.
6. Repeat steps 4 and 5 until all items are added to the order.
7. Customer indicates end of order; clerk enters end of order; system computes totals.
8. Customer submits payment; clerk enters amount; system verifies payment.
9. System finalizes order.

Exception Conditions:

1. If an item is not in stock, then customer can
   a. choose not to purchase item, or
   b. request item be added as a back-ordered item.
2. If customer payment is rejected due to bad-credit verification, then
   a. order is canceled, or
   b. order is put on hold until check is received.

Gambar 12 *Intermediate Description* (Burd Satzinger, Jackson & Burd, 2012)

3. *Fully Developed Description*

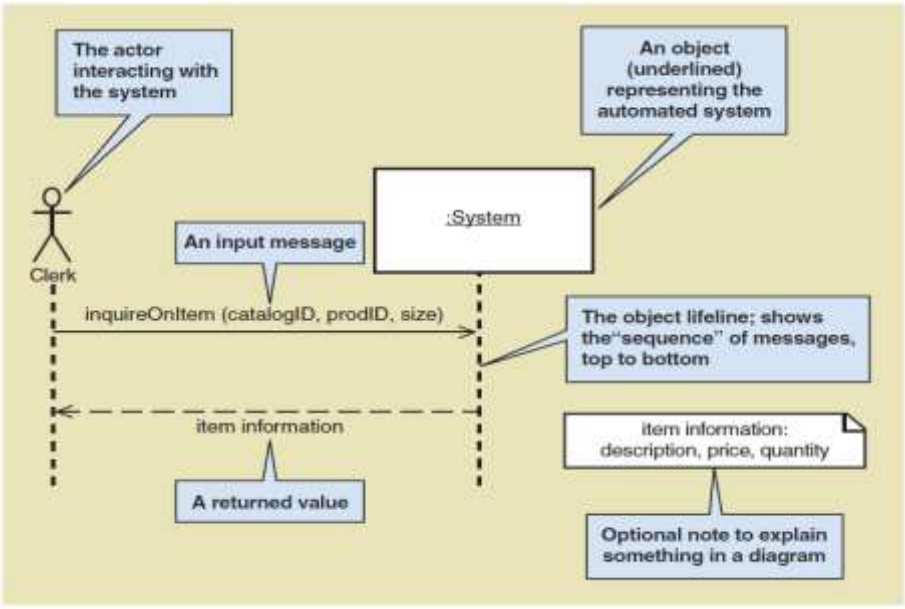
Fully Developed Use Case Descriptions adalah metode yang paling formal untuk mendeskripsikan *use case* karena dapat meningkatkan kemungkinan untuk benar-benar memahami proses bisnis dan sistem yang mendukung proses bisnis tersebut (Satzinger, Jackson, & Burd 2012:122). Gambar berikut menunjukkan salah satu contoh *fully developed use case description*.

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 13 *Fully Developed Use Case Description* (Burd Satzinger, Jackson & Burd, 2012)

2.10.5 *Sequence Diagram*

Menurut Satzinger, Jackson, & Burd (2012:126) mengatakan bahwa *System sequence diagram* (SSD) digunakan untuk menggambarkan arus informasi yang masuk dan keluar dari sistem secara otomatis. SSD digunakan untuk mendokumentasikan input dan output dan mengidentifikasi interaksi antara aktor dan sistem. Pada gambar berikut ditunjukkan contoh dari SSD dan notasi yang digunakan pada *Sequence Diagram*.



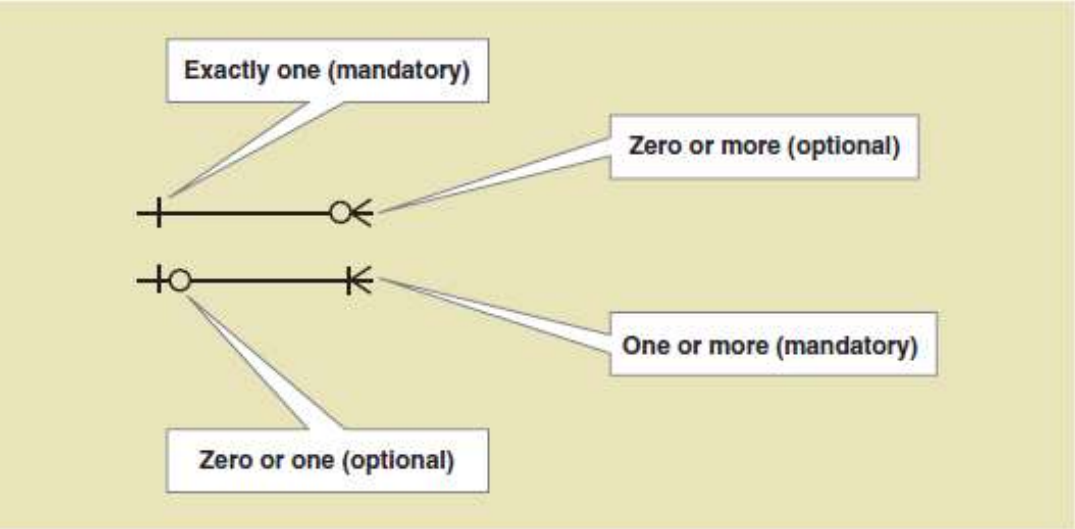
Gambar 14 *System Sequence Diagram* (Burd Satzinger, Jackson & Burd, 2012)

Menurut Satzinger, Jackson, & Burd (2012:332), Sequence Diagram digunakan untuk menjelaskan interaksi objek dan keputusan desain dokumen.

2.11 *Entity-Relationship Diagram*

Entity-Relationship Diagram (ERD) adalah diagram model yang menggambarkan analisa database berupa entitas data, atribut dan hubungan disebut "asosiasi" dalam UML antara entitas data. ERD bukan merupakan UML namun sering dikaitkan dengan UML yaitu dengan *class diagram*.

Pada diagram entity-relationship, persegi panjang mewakili entitas data dan garis yang menghubungkan persegi panjang menunjukkan hubungan antara entitas data. Notasi ini dapat mengungkapkan rincian yang tepat tentang sistem dan mencerminkan kebijakan bisnis bahwa manajemen telah ditetapkan (Satzinger, Jackson, & Burd, 2012:98).



Gambar 15 *Entity Relationship Diagram* (Burd Satzinger, Jackson & Burd, 2012)

2.12 *User Interface*

Menurut Satzinger, Jackson, & Burd, (2012:189) *user interface* merupakan bagian-bagian dari sistem informasi membutuhkan interaksi *user* untuk membuat masukan dan keluaran.*User Interface* melibatkan masukan dan keluaran yang secara langsung berhubungan dengan *user*. Terdapat Eight golden rules dalam mendesain user interface suatu sistem:

1. Konsistensi

   Konsistensi dilakukan pada urutan tindakan, perintah, dan istilah yang digunakan pada prompt, menu, serta layar bantuan.

2. Memungkinkan pengguna untuk menggunakan *shortcut*

   Ada kebutuhan dari pengguna yang sudah ahli untuk meningkatkan kecepatan interaksi, sehingga diperlukan singkatan, tombol fungsi, perintah tersembunyi, dan fasilitas makro.

3. Memberikan umpan balik yang informatif

   Untuk setiap tindakan operator, sebaiknya disertakan suatu sistem umpan balik. Untuk tindakan yang sering dilakukan dan tidak terlalu penting, dapat diberikan umpan balik yang sederhana. Tetapi ketika tindakan merupakan hal yang penting, maka umpan balik sebaiknya lebih substansial. Misalnya muncul suatu suara ketika salah menekan tombol pada waktu input data atau muncul pesan kesalahannya.

4. Merancang dialog untuk menghasilkan suatu penutupan

   Urutan tindakan sebaiknya diorganisir dalam suatu kelompok dengan bagian awal, tengah, dan akhir. Umpan balik yang informatif akan meberikan indikasi bahwa cara yang dilakukan sudah benar dan dapat mempersiapkan kelompok tindakan berikutnya.

5. Memberikan penanganan kesalahan yang sederhana

   Sedapat mungkin sistem dirancang sehingga pengguna tidak dapat melakukan kesalahan fatal. Jika kesalahan terjadi, sistem dapat mendeteksi kesalahan dengan cepat dan memberikan mekanisme yang sedehana dan mudah dipahami untuk penanganan kesalahan.

6. Mudah kembali ke tindakan sebelumnya

   Hal ini dapat mengurangi kekuatiran pengguna karena pengguna mengetahui kesalahan yang dilakukan dapat dibatalkan; sehingga pengguna tidak takut untuk mengekplorasi pilihan-pilihan lain yang belum biasa digunakan.

7. Mendukung tempat pengendali internal (*internal locus of control*)

   Pengguna ingin menjadi pengontrol sistem dan sistem akan merespon tindakan yang dilakukan pengguna daripada pengguna merasa bahwa sistem mengontrol pengguna. Sebaiknya sistem dirancang sedemikan rupa sehingga pengguna menjadi inisiator daripada responden.

8. Mengurangi beban ingatan jangka pendek

   Keterbatasan ingatan manusia membutuhkan tampilan yang sederhana atau banyak tampilan halaman yang sebaiknya disatukan, serta diberikan cukup waktu pelatihan untuk kode, *mnemonic*, dan urutan tindakan.

   Menurut Nielsen (1999:1), ada sepuluh elemen *Ten Good Deeds Web Design,* yaitu:

1. Menempatkan nama organisasi dan logo pada setiap halaman dan membuat setiap logo melink ke *homepage*.
2. Menyediakan fungsi pencarian jika halaman situs memiliki beberapa halaman.
3. Menuliskan judul halaman secara jelas sehingga dapat ditemukan jika dicari penggunakan mesin pencari.
4. Struktur halaman memfasilitasi pengguna dan membantu pengguna mendapatkan inti dari informasi tanpa harus membaca sebagian besar halaman.
5. Membuat halamanan tentang produk atau topik menggunakan *hypertext* dengan struktur yang memberikan gambaran mengenai halaman utama dan

menambahkan beberapa halaman sekuner untuk mendukung masing-masing fokus dari topik tersebut.

6. Gunakan foto produk untuk menghindari halaman utama produk menjadi berantakan. Halaman produk utama harus memuat fungsi dengan cepat.
7. Gunakan gambar yang relevan. Membuat gambar asli dengan tampilan *thumbnail* kecil sehingga dapat terlihat semuanya. Dan pada saat gambar di perbesar, gambar memiliki *size* yang relevan.
8. Gunakan judul *link* untuk memberitahu pengguna dimana mereka berada dimana setiap link akan membawa mereka kepada *link* yang dituju.
9. Memastikan bahwa semua halaman yang penting dapat diakses bagi para penyandang cacat terutama pengguna tunanetra.
10. Melakukan hal yang sama seperti yang lain, jika kebanyakan situs *web* besar melakukan sesuatu dengan cara tertentu, pengguna akan mengharapkan situs lain untuk bekerja dengan cara yang sama.

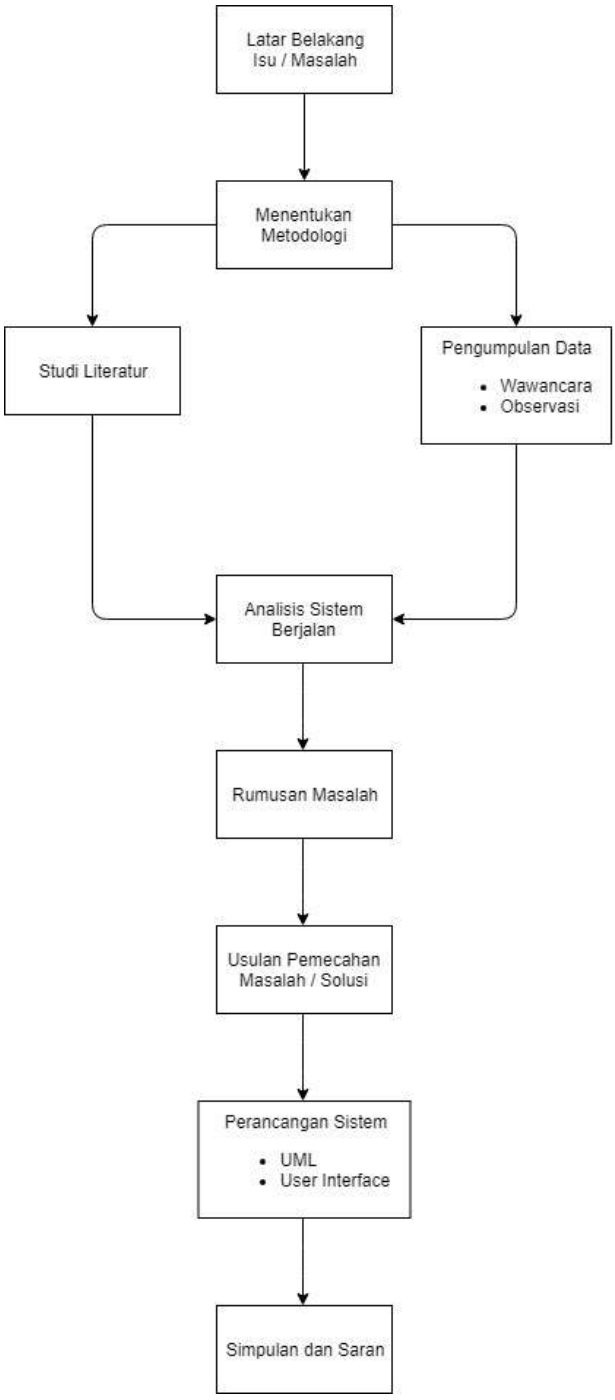
2.13 Spesifikasi Sistem

Spesifikasi sistem merupakan dokumen yang menjelaskan persyaratan penting untuk item, bahan, proses, atau jasa dari ruang solusi yang ditentukan, data yang diperlukan untuk melaksanakan persyaratan, dan metode verifikasi untuk memenuhi kriteria khusus untuk penerimaan formal. Spesifikasi ditulis tidak hanya untuk item penyampaian fisik tetapi juga untuk layanan dan komponen bertingkat, bahan yang membentuk komponen-komponen, dan proses prosedural yang diperlukan untuk mengkonversi bahan-bahan menjadi komponen yang dapat digunakan (Wasson, 2006:304).

Berdasarkan pengertian diatas dapat disimpulkan spesifikasi sistem merupakan dokumen yang berfungsi menggambarkan fungsi dan kinerja sistem berbasis komputer yang akan dikembangkan, membatasi elemen-elemen sistem yang telah dialokasikan, serta memberikan indikasi mengenai perangkat lunak dan konteks sistem keseluruhan dan informasi data dankontrol yang dimasukkan dan dikeluarkan oleh sistem yang telah digambarkan dalam diagram aliran arsitektur.

2.14 Kerangka Pemikiran

Dalam penulisan skripsi ini, dibutuhkan sebuah kerangka pemikiran yang digunakan untuk membantu proses pengerjaan. Kerangka pemikiran yang digunakan dalam penulisan skripsi ini dapat dilihat pada gambar berikut.



Gambar 16 Kerangka Pemikiran Pengerjaan Skripsi

Dari kerangka pemikiran di atas, dapat dilihat bahwa hal pertama yang dilakukan dalam pengerjaan penelitian ini adalah membuat latar belakang masalah. Setelah membuat latar belakang masalah akan ditentukan metodologi yang digunakan dalam penelitian ini. Kemudian dilakukan studi literatur dan pengumpulan data untuk mengidentifikasi proses bisnis yang sedang berjalan pada perusahaan. Metode pengumpulan data yang dilakukan terdiri dari dua metode, yaitu wawancara dan observasi. Data yang dikumpulkan akan digunakan untuk merumuskan masalah yang terjadi pada proses bisnis perusahaan. Masalah yang ditemukan akan dianalisis, kemudian dilakukan penyusunan usulan pemecahan masalah untuk menyelesaikan permasalahan dalam proses bisnis perusahaan tersebut.

Pada tahap perancangan dilakukan pemetaan usulan pemecahan masalah yang telah ditemukan menjadi sebuah sistem. Perancangan sistem ini menggunakan pendekatan dengan metodologi *Object Oriented Analysis and Design (OOAD)* yang akan digambarkan ke dalam beberapa diagram seperti *entitiy relationship diagram*, *class diagram*, *activity diagram, usecase diagram, sequence diagram* dan *user interface*. Dari sistem yang telah dirancang dapat ditentukan kesimpulan dan saran dari pengerjaan penelitian ini.