BAB II
LANDASAN TEORI

2.1 Tinjauan Pustaka

Membangun sebuah aplikasi yang baik agar sesuai dengan kebutuhan, maka diperlukan referensi sistem informasi yang lain seperti berikut:

2.1.1 Sistem Informasi Akademik Berbasis SMS *Gateway* SMK Satria Jakarta Barat

Penelitian Yudi Wiharto (2011) Politeknik PalComTech Palembang dengan judul sistem informasi akademik berbasis SMS *Gateway* SMK Satria Jakarta Barat menjelaskan bahwa Dengan adanya aplikasi SMS *Gateway* dan *Mobile Application* (*Request Sender*), maka akan lebih memudahkan siswa atau wali siswa untuk dapat melakukan *request* dan mengetahui informasi-informasi penting dari sekolah. Dengan aplikasi SMS *Gateway*, informasi yang diinginkan siswa atau wali siswa bisa didapatkan kapanpun dan dimanapun. Siswa atau wali siswa tidak perlu repot-repot untuk mengetik *sms* untuk mendapatkan informasi yang diinginkan menggunakan *Mobile Application* (*Request Sender*).

2.1.2 Membangun Informasi Absensi Siswa SMK PGRI 3 Malang

Yusuf A.N (2013) dengan judul membangun informasi absensi siswa SMK PGRI 3 Malang menggunakan gammu, php dan mysql menjelaskan bahwa penambahan fitur SMS *Gateway* pada pengiriman Informasi Absensi Siswa SMK PGRI 3 Malang ini dibuat bertujuan untuk memudahkan orang tua untuk mendapatkan informasi anaknya yang melakukan kesalahan yang mengakibat ketidak hadiran anaknya dalam sekolah pada waktu jam pelajaran.

2.1.3 Sistem Informasi Nilai Mahasiswa Berbasis SMS *Gateway* Pada Fakultas Pertanian Universitas Bengkulu

Penelitian Prasetyo, Asnawati dan Arliando (2015) dengan judul sistem informasi nilai mahasiswa berbasis SMS *Gateway* pada fakultas pertanian universitas bengkulu menjelaskan bahwa pengujian program SMS Gateway menggunakan gammu dapat dismpulkan bahwa untuk mengimplementasikannya.

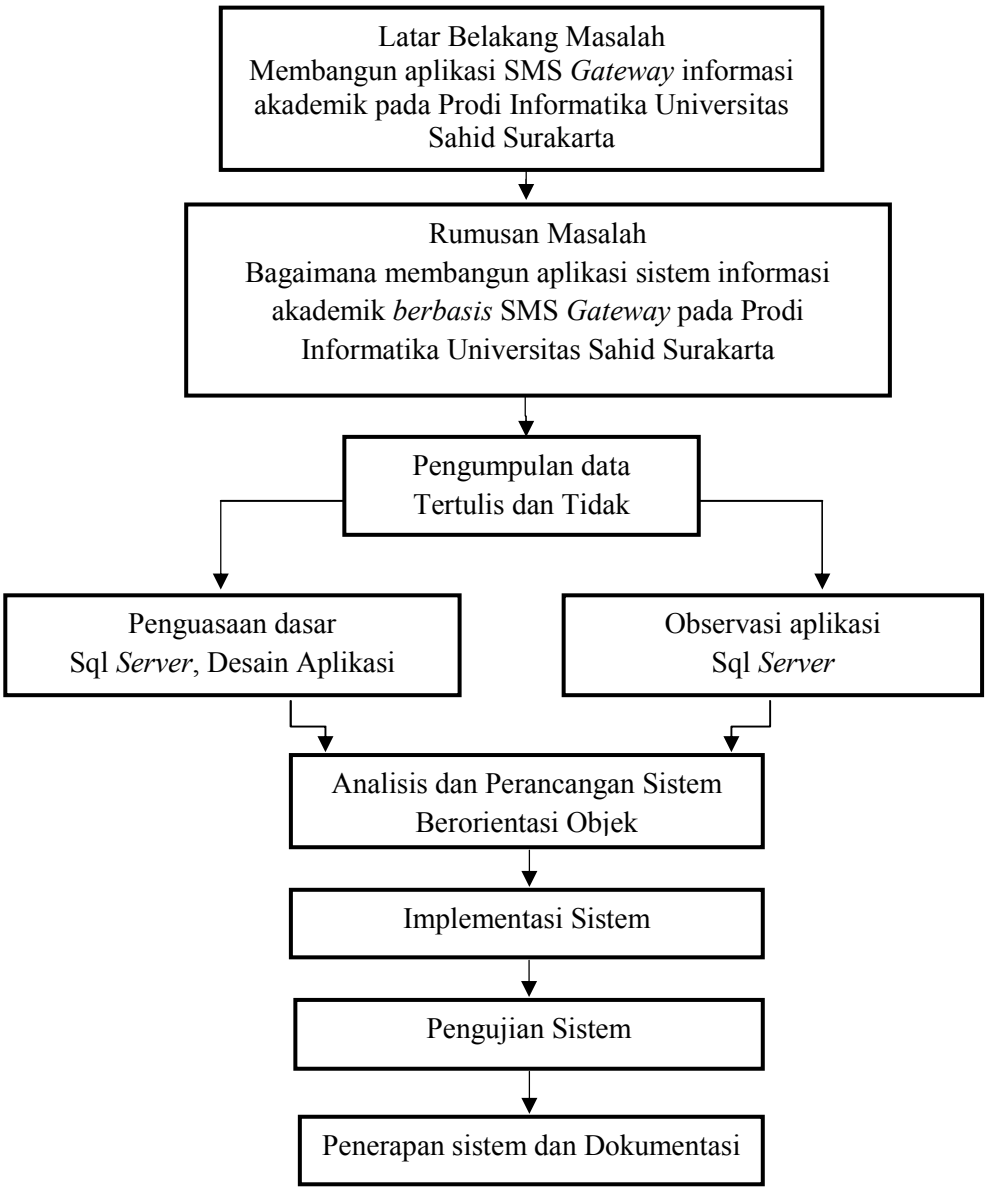
SMS *Gateway* memerlukan fitur aplikasi gammu yang dihubungkan dengan bahasa pemrograman PHP dan MySQL sehingga dapat digunakan untuk mengolah dan menangani pesan SMS berbasis web. Pemanfaatan SMS *Gateway* ini digunakan untuk informasi nilai mahasiswa dengan mengetikan format SMS yang telah disediakan sehingga dapat membantu mahasiswa dalam mendapat informasi tentang sistem akademiknya.

Penelitian pertama menggunakan java *runtime environment*, MySQL *database,* Driver *Handphone PC Suite*. Metode dari penelitian pertama yaitu Aplikasi SMS *Gateway* ini dengan nama SMK SATRIA yang dijalankan pada sebuah komputer yang terhubung dengan *database* dan menggunakan sebuah *handphone* yang dihubungkan melalui USB *port* sebagai penerima SMS (*receiver*). Aplikasi ini akan menerima semua SMS yang masuk dan meresponnya secara otomatis. Penelitian kedua menggunakan PHP dan MySQL. Metode penelitian yang digunakan untuk meneliti konsep pendataan absensi siswa SMK 3 Malang melalui cara penginputan data pada *database* melalui *interview*. Penelitian ketiga menggunakan PHP dan MySQL dengan metode untuk memberikan informasi nilai mahasiswa pada akademik program studi peternakan Fakultas Pertanian Universitas Bengkulu dengan mengetikkan format SMS yang telah disediakan.

Kesimpulan dari tiga referensi diatas dapat memberi acuan dalam penelitian tugas akhir ini untuk meningkatkan sistem informasi akademik pada Prodi Informatika Universitas Sahid Surakarta. Sehingga dalam pembuatan aplikasi tidak mendapatkan kendala yang begitu sulit karena dengan adanya referensi-referensi dari penelitian yang telah dilakukan. Perbedaan dari penelitian yang dilakukan dengan ketiga referensi diatas memiliki perbedaan *softwere* dan objek, untuk *softwere* penelitian ini menggunakan VB 6.0 dan MySQL *database*. Objeknya yaitu Prodi Informatika Universitas Sahid Surakarta.

2.2 Kerangka Pemikiran

Kerangka pemikiran dari penelitian tugas akhir ini telah tercantum pada Gambar 2.1.



Gambar 2.1 Kerangka Pemikiran

Keterangan kerangka pemikiran tugas akhir dijelaskan sebagai berikut :

a. Latar belakang masalah

   Pokok permasalahan yang mendasari perlunya dibangun sistem informasi berbasis SMS *Gateway* pada Prodi Informatika Universitas Sahid Surakarta.

b. Perumusan masalah

   Perumusan masalah merupakan inti permasalahan dan jalan keluar menyelesaikan permasalahan.

c. Pengumpulan data tertulis dan tidak tertulis

   Pada penelitian dilakukan pengumpulan data secara tertulis dan tidak tertulis pada Prodi Informatika Universitas Sahid Surakarta. Pengumpulan data penelitian ini menggunakan metode observasi, dokumentasi, dan wawancara.

d. Penguasaan dasar

   Pada penelitian dilakukan percobaan membuat sistem agar lebih menguasai.

e. Observasi sistem

   Pada penelitian dilakukan pengamatan pada sistem yang sebelumnya digunakan agar dapat menjadi referensi dalam membangun aplikasi ini.

f. Analisis dan perancangan sistem

   Pada penelitian dilakukan menganalisa dan merancang bagaimana sistem nantinya akan dibuat untuk membantu memecahkan permasalahan yang ada.

g. Implementasi sistem

   Pada penelitian dilakukan implementasi apa yang sudah dirancang untuk membangun sistem informasi pada Prodi Informatika Universitas Sahid Surakarta.

h. Pengujian sistem

   Pada penelitian dilakukan uji coba aplikasi apakah masih terjadi kesalahan ataupun kekurangan pada sistem.

i. Penerapan sistem dan Dokumentasi

Sistem yang sudah diimplementasikan dan diuji coba kemudian diterapkan pada Prodi Informatika Universitas Sahid Surakarta dan setelah itu dibuatnya dokumentasi dari keseluruhan kegiatan penyusunan Tugas Akhir.

2.3 Teori Pendukung

2.3.1 Pengertian Aplikasi

Menurut Sutabri (2012), aplikasi adalah alat terapan yang difungsikan secara khusus dan terpadu sesuai kemampuan yang dimilikinya.

Menurut Asropudin (2013), aplikasi adalah *software* yang dibuat oleh suatu perusahaan komputer untuk mengerjakan tugas-tugas tertentu, misalnya Ms.World, Ms.Excel.

Menurut pengertian diatas penulis menyimpulkan aplikasi adalah *software* atau alat terapan yang dibuat untuk mengerjakan tugas-tugas khusus.

2.3.2 Pengertian SMS *Gateway*

Menurut Riadi (2012) SMS (*Short Message Service*) merupakan layanan yang banyak diaplikasikan pada sistem komunikasi tanpa kabel (*nirkabel*), memungkinkan dilakukannya pengiriman pesan dalam bentuk *alphanumeric* antar terminal pelanggan atau antar terminal pelanggan dengan sistem eksternal. SMS berupa pesan teks, jumlah karakter pada setiap pengiriman bergantung pada operatornya. Operator selular di Indonesia umumnya membatasi 160 karakter untuk satu pengiriman dan penerimaan SMS. Selain itu SMS merupakan metode *store* dan *forward* sehingga keuntungan yang didapat adalah pada saat telepon selular penerima tidak dapat dijangkau, dalam arti tidak aktif atau diluar *service* area, penerima tetap dapat menerima SMS, apabila telepon selular tersebut sudah aktif kembali.

Menurut J.P Jumri (2012), *Short Message Service* (SMS) adalah kemampuan untuk mengirim dan menerima pesan dalam bentuk teks dari dan

kepada ponsel. Teks tersebut bisa terdiri dari huruf, angka atau kombinasi *alphanumeric*.

SMS *Gateway* adalah komunikasi menggunakan SMS yang mengandung informasi berupa nomor telepon seluler pengirim, penerima, waktu dan pesan. Informasi tersebut dapat diolah dan bisa melakukan aktivasi transaksi tergantung kode-kode yang sudah disepakati. Untuk dapat mengelola semua transaksi yang masuk dibutuhkan sebuah sistem yang mampu menerima kode SMS dengan jumlah tertentu, mengolah informasi yang terkandung dalam pesan SMS dan melakukan transaksi yang dibutuhkan. Aplikasi SMS *Gateway* adalah sebuah perangkat lunak yang menggunakan bantuan komputer dan memanfaatkan teknologi seluler yang diintegrasikan guna mendistribusikan pesan-pesan yang dipadukan lewat sistem informasi melalui media SMS yang ditangani oleh jaringan seluler. SMS *Gateway* biasanya support untuk pesan yang berupa teks, *unicode character*, dan juga *smart messaging (ringtone, picture message, logo* operator dan lain-lain).

2.3.3 Pengertian Informasi

Secara Etimologi, Informasi berasal dari bahasa Perancis kuno *informacion* yang diambil dari bahasa Latin *informationem* yang berarti "garis besar, konsep, ide". Informasi Juga dapat diartikan sebagai data yang telah di olah menjadi bentuk yang lebih berguna dan lebih berarti bagi yang menerimanya. Informasi merupakan hal yang sangat penting bagi instansi dalam pengambilan keputusan. Beberapa ahli mendefinisikan informasi sebagai berikut :

Menurut Rudy Tantra (2012) dalam bukunya yang berjudul Manajemen Proyek Sistem Informasi menerangkan informasi dapat dipahami sebagai pemrosesan *input* yang terorganisir, memiliki arti, dan berguna bagi orang yang menerimanya. Data berbeda dengan informasi. Data dapat didefinisikan sebagai fakta-fakta yang masih mentah atau acak yang menjadi *input* untuk proses yang menghasilkan informasi.

Menurut Mohamad Subhan (2012) dalam bukunya yang berjudul Analisa Perancangan Sistem mengungkapkan Sistem informasi merupakan kumpulan data yang diolah menjadi bentuk yang lebih berguna dan lebih berarti bagi yang menerima. Tanpa suatu informasi, suatu sistem tidak akan berjalan dengan lancar dan akhirnya bisa mati. Dengan kata lain sumber dari informasi adalah data. Data menggambarkan suatu kejadian yang sering terjadi, dimana data tersebut akan diolah dan akan diterapkan dalam sistem menjadi input yang berguna dalam suatu sistem. Data merupakan bentuk yang belum dapat memberikan manfaat yang besar bagi penerimanya, sehingga perlu suatu model yang nantinya akan dikelompokkan dan diproses untuk menghasilkan informasi.

2.3.4 Pengertian Gammu

Gammu merupakan sebuah aplikasi untuk membangun SMS *Gateway*. Aplikasi tersebut saat ini dikelola oleh Michal Cihar dan orang orang yang berpengalaman dalam membuat aplikasi Gnokii dan MyGnokii. Gammu memiliki kemampuan menjalankan layanan antara lain *calls, SMS, EMS, phonebook memories, filesystem, logos, pictures, ringtones*, dan lain-lain. Setiap layanan pada Gammu dijalankan melalui *command line* dari dalam *folder* Gammu *(C:/gammu_win32/win32* pada sistem operasi *Windows* dan *etc/gammu* pada sistem operasi *Linux*), di dalam *folder* ini terdapat *file-file application* untuk menjalankan gammu seperti gammu, gammurc, dan smsdrc. Selain *file application* juga terdapat *file database* yang disediakan Gammu.

Seluruh layanan pada Gammu memiliki peran masing-masing, seperti layanan *calls* memungkinkan *server* dapat melakukan *dial voice*, *answer call*, *hold call*, dan *call conference*. Selain dapat melakukan hubungan telepon, Gammu juga dapat dikondisikan untuk mengirimkan dan menerima SMS dan EMS. Dengan demikian, layanan ini dapat digunakan untuk mengirim pesan, *download ringtone*, *caller* SMS, *send picture*, *animation*, MMS, dan *VCARD*.

Gammu juga mampu menyediakan akses data pada telepon selular yang tersimpan dalam *memory* antara lain *calls*, *voice mailbox*, SIM *phonebook*, dan

phone internal phonebook. Dikarenakan kemampuan Gammu dalam mengakses memori, maka Gammu juga menyediakan layanan yang bernama *file system* dimana dengan layanan ini pengguna dapat menambah dan mengubah isi mauapun nama *folder* dan *file* yang tersimpan. Layanan terakhir yang dimiliki oleh Gammu adalah *backing up and restoring phone* / SIM data, sehingga pengguna Gammu dapat mem*backup* semua data dari telepon selular ke sebuah *text file*. *File-file* yang dapat di *backup* oleh Gammu adalah *phonebook*, *calendar notes*, SMSC *settings*, operator logo, WAP, dan *user ringtones*.

2.3.5 Pengertian Basis Data

Menurut Ladjamudin (2013), Database adalah sekumpulan data *store* (bisa dalam jumlah yang sangat besar) yang tersimpan dalam *magnetic disk*, *oftical disk*, *magnetic drum*, atau media penyimpanan sekunder lainya.

Menurut Sutarman (2012), Database sekumpulan file yang saling berhubungan dan terorganisasi atau kumpulan *record-record* yang menyimpan data dan hubungan diantaranya.

Menurut beberapa pendapat para ahli yang dikemukakan di atas dapat ditarik kesimpulan bahwa *database* adalah sekelompok data yang mempunyai ciri-ciri khusus dan dapat dikelola sedemikian rupa sehingga bisa menghasilkan sebuah format data yang baru.

2.3.6 Pengertian Mysql

MySQL dikembangkan oleh perusahaan swedia bernama MySQL AB yang pada saat ini bernama Tcx Data Konsult AB sekitar tahun 1994 - 1995. Awalnya Tcx merupakan perusahaan pengembang *software* dan konsultan *database*, dan saat ini MySQL sudah diambil alih oleh Oracle Corp.

Kepopuleran MySQL antara lain karena MySQL menggunakan SQL sebagai bahasa dasar untuk mengakses *database*, sehingga mudah untuk digunakan, kinerja *query* cepat, dan mencukupi untuk kebutuhan *database*

perusahaan-perusahaan yang berskala kecil sampai menengah, MySQL juga bersifat *open source* (tidak berbayar) .

MySQL merupakan *database* yang pertama kali didukung oleh bahasa pemrograman *script* untuk *internet* (PHP dan Perl). MySQL dan PHP dianggap sebagai pasangan *software* pembangun aplikasi web yang ideal. MySQL lebih sering digunakan untuk membangun aplikasi berbasis web, umumnya pengembangan aplikasinya menggunakan bahasa pemrograman *script* PHP.

Menurut Nugroho (2013) menjelaskan MySQL adalah *software* atau program *database server.*

2.3.7 Pengertian Sql *Server* 2000

Menurut Agus Saputra (2013) Microsoft SQL *Server* merupakan salah satu produk RDBMS. RDBMS memilik kepanjangan *Relational Database Management System* merupakan salah satu produk andalan yang dibuat oleh Microsoft yang berfungsi sebagai *reational database*. Micorsoft SQL *Server* mendukung SQL sebagai bahasa pemroses *query*, seperti SQL merupakan bahasa standar international untuk proses *query database*, ada 2 fitur yang biasa digunakan untuk mengelola *database* di dalam SQL *Server* 2000, yaitu :

1. Menggunakan *Enterprise Manager*

   Fitur ini relatif mudah digunakan karena mode pengelolaannya berbasis GUI (*Graphical User Interface*), oleh karena itu cukup dengan metode *click* dan *drag*. Membuat *database* dan tabel serta manajemen yang lain sangat mudah.

2. Menggunakan SQL *Query Analyzer*

   Fitur ini menggunakan Transact SQL (perintah-perintah SQL) untuk mengelola *database* di dalam SQL *Server* 2000. Perintah-perintah *Transact* SQL merupakan pengembangan dari perintah-perintah SQL standard yang disesuaikan dengan manajemen *database* pada SQL *Server*. Transact SQL memungkinkan untuk dapat membuat *database*, membuat tabel, mengubah

struktur tabel, menghapus *database*, menghapus tabel, menyisipkan data, mengubah data dan lain-lain.

2.3.8 Pengertian *Visual Basic* 6.0

Menurut Stefano (2014) *visual basic* merupakan sebuah bahasa pemrograman yang menawarkan *Intergrated Development Environment* (IDE) visual untuk membuat program perangkat lunak berbasis operasi *Microsoft Windows* menggunakan model pemrograman.

Bahasa pemograman *Visual Basic*, yang dikembangkan oleh Microsoft sejak tahun 1991, merupakan pengembangan dari pendahulunya yaitu bahasa pemograman BASIC (*Beginner s All Purpose Symbolic Instruction Code*) yang dikembangkan pada era 1950-an. *Visual Basic* merupakan salah satu Develovment Tool yaitu alat bantu untuk membuat berbagai macam program komputer, khususnya yang menggunakan sistem operasi windows. *Visual Basic* merupakan salah satu bahasa pemograman komputer yang mendukung *object* (*Object Oriented Programming* = OOP).

2.4 Analisis Sistem

Menurut O'Brien (2013), sistem adalah sekelompok komponen yang saling berhubungan dengan batasan yang jelas dan bekerja sama menuju tujuan tertentu dengan menerima *input* serta menghasilkan *output* yang merupakan fungsi dasar dalam proses transformasi yang teratur. Analisis sistem penulis menggunakan metode UML.

Menurut Yuni Sugiarti (2013) Unifield Modeling Language (UML) adalah sebuah bahasa yang telah menjadi standar dalam industri untuk visualisasi, merancang dan mendokumentasikan sistem perangkat lunak.

Bangunan dasar metodologi *Unified Modeling Language* (UML) menggunakan bangunan dasar untuk mendeskripsikan system atau perangkat lunak yang akan dikembangkan yaitu :

1. Sesuatu *(things)*

   Ada 4 (empat) *things* dalam *Unified Modeling Language* (UML), yaitu:

   a. *Structural things*

      Structural things merupakan bagian yang relatif statis dalam model *Unified Modeling Language* (UML). Bagian yang relatif statis dapat berupa elemen-elemen yang bersifat fisik maupun konseptual.

   b. *Behavioral things*

      Behavioral things merupakan bagian yang dinamis pada model *Unified Modeling Language* (UML), biasanya merupakan kata kerja dari model *Unified Modeling Language* (UML), yang mencerminkan perilaku sepanjang ruang dan waktu.

   c. *Grouping things*

      Grouping things merupakan bagian pengorganisasi dalam *Unified Modeling Language* (UML). Dalam penggambaran model yang rumit kadang *diperlukan* penggambaran paket yang menyederhanakan model. Paket-paket ini kemudian dapat didekomposisi lebih lanjut. Paket berguna bagi pengelompokkan sesuatu, misalnya model-model dan subsistem-subsistem.

   d. *Annotational things*

      Annotational things merupakan bagian yang memperjelas model *Unified Modeling Language* (UML) dan dapat berupa komentar-komentar yang menjelaskan fungsi serta ciri-ciri setiap elemen dalam model *Unified Modeling Language* (UML).

2. Relasi *(Relationship)*

   Ada 4 (empat) macam *relationship* dalam *Unified Modeling Language* (UML), yaitu :

   a. Kebergantungan

      Kebergantungan merupakan hubungan dimana perubahan yang terjadi pada suatu elemen mandiri *(independent)* akan mempengaruhi elemen yang bergantung padanya elemen yang tidak mandiri *(independent)*.

b. Asosiasi

Asosiasi merupakan apa yang menghubungkan antara objek satu dengan objek lainnya, bagaimana hubungan suatu objek dengan objek lainnya. Suatu bentuk asosiasi adalah agregasi yang menampilkan hubungan suatu objek dengan bagian-bagiannya.

c. Generalisasi

Generalisasi merupakan hubungan dimana objek anak (*descendent)* berbagi perilaku dan struktur data dari objek yang ada diatasnya objek induk (*ancestor)*.

Arah dari atas ke bawah dari objek induk ke objek anak dinamakan spesialisasi, sedangkan arah berlawanan sebaliknya dari arah bawah keatas dinamakan generalisasi.

d. Realisasi

Realisasi merupakan hubungan suatu operasi yang benar-benar dilakukan oleh objek tertentu.

2.4.1 Diagram UML

Menurut Rosa dan Shalahuddin (2013) pada perkembangan teknik pemrograman berorientasi objek, munculah sebuah standarisasi bahasa pemodelan untuk pembangunan perangkat lunak yang dibangun dengan menggunakan teknik pemrograman berorientasi objek, yaitu *Unified Modeling Language* (UML). UML muncul karena adanya kebutuhan pemodelan visual untuk menspesifikasikan, menggambarkan, membangun dan mendokumentasi dari sistem perangkat lunak.

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

Bahasa pemrograman berorientasi objek yang pertama dikembangkan dikenal dengan nama Simula 67 yang dikembangkan pada tahun 1967. Perkembangan aktif dari pemrograman berorientasi objek mulai menggeliat ketika berkembangnya bahasa pemrograman Smalltalk pada awal 1980-an. Pada 1996, *Object Management Group* (OMG) mengajukan proposal agar adanya standarisasi pemodelan berorientasi objek dan pada bulan September 1997 *Unified Modeling Language* (UML) diakomodasi oleh *Object Management Group* (OMG) sehingga sampai saat ini (UML) telah memberikan kontribusinya yang cukup besar dalam metodologi berorientasi objek. Macam-macam diagram dalam (UML), yaitu :

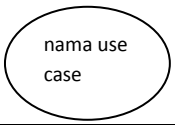
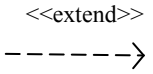
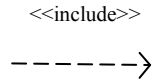
2.4.1.1 *Use Case Diagram*

Use case diagram merupakan pemodelan untuk kelakuan (*behaviour*) sistem informasi yang akan dibuat. *Use case* mendeskripsikan sebuah interaksi antara satu atau lebih aktor dengan sistem informasi yang akan dibuat. Secara kasar, *use case* digunakan untuk mengetahui fungsi apa saja yang ada didalam sebuah sistem informasi dan siapa saja yang berhak menggunakan fungsi-fungsi itu. Syarat penamaan pada *use case* adalah nama didefinisikan sederhana mungkin dan dapat dipahami. Ada dua hal utama pada *use case* yaitu pendefinisian apa yang disebut aktor dan *use case*.

1. Aktor merupakan orang, proses atau sistem lain yang berinteraksi dengan sistem informasi yang akan dibuat diluar sistem informasi yang akan dibuat itu sendiri, jadi walaupun simbol dari aktor adalah gambar orang, tapi aktor belum tentu merupakan orang.
2. *Use case* merupakan fungsionalitas yang disediakan sistem sebagai unit-unit yang saling bertukar pesan antar unit atau aktor.

Simbol-simbol *Use Case Diagram* ditunjukkan pada Tabel 2.1.

Tabel 2.1 Simbol-simbol *Use Case Diagram*

NO.	SIMBOL	NAMA	KETERANGAN
1.		*Use Case*	Fungsionalitas yang disediakan sistem sebagai unit yang saling bertukar pesan antar unit atau aktor.
2.		Aktor / *Actor*	Orang, proses, atau sistem lain yang berinteraksi dengan sistem informasi yang akan dibuat diluar sistem informasi yang akan dibuat itu sendiri.
3.		Asosiasi / *Association*	Komunikasi antara aktor dan *use case* yang berpartisipasi pada *use case* memiliki interaksi dengan aktor.
4.		Ekstensi / *Extend*	Relasi *use case* tambahan ke sebuah *use case* yang ditambahkan dapat berdiri sendiri walau tanpa *use case* tambahan itu.
5.		Generalisasi / *Generaliztion*	Hubungan generalisasi dan spesifikasi (umum-khusus) antara dua buah *use case* dimana fungsi yang satu adalah fungsi yang lebih umum dari lainnya.
6.		Menggunakan / *Include / uses*	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.

Sumber : Rosa dan Shalahuddin (2013:155 - 158)

2.4.1.2 *Class Diagram*

Class diagram menggambarkan struktur sistem dari segi pendefinisian kelas-kelas yang akan dibuat untuk membangun sistem. Kelas memiliki apa yang disebut atribut dan metode atau operasi.

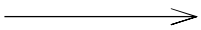
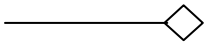
1. Atribut merupakan variabel-variabel yang dimiliki oleh suatu kelas.
2. Operasi atau metode adalah fungsi-fungsi yang di miliki oleh suatu kelas.

Kelas-kelas yang ada pada struktur sistem harus dapat melakukan fungsi-fungsi sesuai dengan kebutuhan sistem. Susunan struktur kelas yang baik pada diagram kelas sebaiknya memiliki jenis-jenis kelas berikut :

1. Kelas main, kelas yang memiliki fungsi awal di eksekusi ketika sistem dijalankan.
2. Kelas yang menangani tampilan sistem (*view*), kelas yang mendefinisikan dan mengatur tampilan ke pemakai.
3. Kelas yang di ambil dari pendefinisian *use case (controller)*, kelas yang menangani fungsi-fungsi yang harus ada di ambil dari pendefinisian *use case*, kelas ini biasanya disebut dengan kelas proses yang menangani proses bisnis pada perangkat lunak.
4. Kelas yang di ambil dari pendefinisian data, kelas yang digunakan untuk memegang atau membungkus data menjadi sebuah kesatuan yang di ambil maupun akan di simpan ke basis data. Semua tabel yang dibuat di basis data dapat dijadikan kelas, namun untuk tabel dari hasil relasi atau atribut *mutivalue* pada ERD dapat dijadikan kelas tersendiri dapat juga tidak, asalkan pengaksesannya dapat dipertanggungjawabkan atau tetap ada di dalam perancangan kelas.

Simbol-simbol *Class Diagram* ditunjukkan pada Tabel 2.2.

Tabel 2.2 Simbol-simbol *Class Diagram*

NO.	SIMBOL	NAMA	KETERANGAN
1.	Nama_Kelas + atribut + operasi()	Kelas	Kelas pada struktur sistem.
2.		Antarmuka / *Interface*	Sama dengan konsep *interface* dalam pemrograman berorientasi objek.
3.		Asosiasi / *Association*	Relasi antar kelas dengan makna umum, asosiasi biasanya juga di sertai dengan *multiplicity*.
4.		Asosiasi Berarah / *Directed Association*	Relasi antar kelas dengan makna kelas yang satu digunakan oleh kelas lain. asosiasi biasanya juga di sertai dengan *multiplicity*.
5.		Generalisasi	Relasi antar kelas dengan makna generalisasi-spesialisasi (umum khusus).
6.		Kebergantungan / *Depedency*	Relasi antar kelas dengan makna kebergantungan antar kelas.
7.		Agregasi / *Aggregation*	Relasi antar kelas dengan makna semua bagian (*whole-part*).

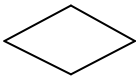
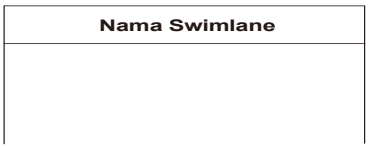
Sumber : Rosa dan Shalahuddin (2013:141 - 147)

2.4.1.3 *Activity Diagram*

Diagram aktivitas atau *activity diagram* menggambarkan *workflow* (aliran kerja) atau aktivitas dari sebuah sistem proses bisnis atau menu yang ada pada perangkat lunak, yang perlu diperhatikan di sini adalah bahwa diagram aktivitas menggambarkan aktivitas sistem bukan apa yang dilakukan aktor, jadi aktivitas yang dapat dilakukan oleh sistem.

Simbol-simbol *Activity Diagram* ditunjukkan pada Tabel 2.3.

Tabel 2.3 Simbol-simbol *Activity Diagram*

NO.	GAMBAR	NAMA	KETERANGAN
1.		Status Awal	Status awal aktivitas sistem, sebuah diagram aktivitas memiliki sebuah status awal.
2.		Aktivitas	Aktivitas yang dilakukan sistem, biasanya diawali dengan kata kerja.
3.		Percabangan / *Decision*	Asosiasi percabangan dimana jika ada pilihan aktivitas lebih dari satu.
4.		Penggabungan / *Join*	Asosiasi penggabungan lebih dari satu aktivitas digabungkan menjadi satu.
5.		Status Akhir	Status akhir dilakukan sebuah sistem.
6.		*Swinlane*	Memisahkan organisasi bisnis yang bertanggung jawab terhadap aktivitas yang terjadi.

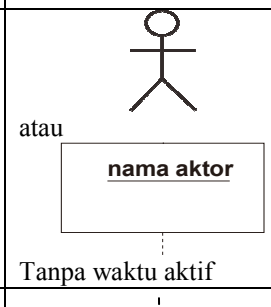
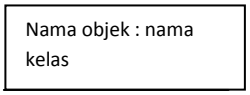
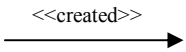
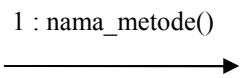
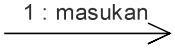
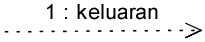
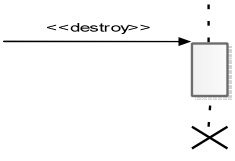
Sumber : Rosa dan Shalahuddin (2013:161 - 163)

2.4.1.4 *Sequence Diagram*

Diagram sekuen menggambarkan kelakuan objek pada *use case* dengan mendeskripsikan waktu hidup objek dan *message* yang dikirimkan dan di terima antar objek, oleh karena itu untuk menggambar diagram sekuen maka harus di ketahui objek-objek yang terlibat dalam sebuah *use case* beserta metode-metode yang dimiliki kelas yang diinstansiasi menjadi objek.

Simbol-simbol *Sequence Diagram* ditunjukkan pada Tabel 2.4.

Tabel 2.4 Simbol-simbol *Sequence Diagram*

NO.	SIMBOL	NAMA	KETERANGAN
1.		Aktor	Orang, proses, atau sistem lain yang berinteraksi dengan sistem informasi yang akan di buat di luar sistem informasi yang akan di buat sendiri.
2.		Garis Hidup	Menyatakan kehidupan suatu objek.
3.		Objek	Menyatakan suatu objek membuat objek lain, arah panah mengarah pada objek yang di buat.
4.		Waktu Aktif	Menyatakan objek dalam keadaan aktif dan berinteraksi, semua yang terhubung dengan waktu aktif ini adalah sebuah tahapan yang dilakukan didalamnya.
5.		Pesan Tipe *Create*	Menggambarkan hubungan kegiatan yang akan dilakukan.
6.		Pesan Tipe *Call*	Menyatakan suatu objek memanggil operasi/metode yang ada pada objek lain atau dirinya sendiri.
7		Pesan Tipe *Send*	Menyatakan bahwa suatu objek mengirimkan data / masukan / informasi ke objek lainnya, arah panah mengarah pada objek yang dikirimi.
8		Pesan Tipe *Return*	Menyatakan bahwa suatu objek yang telah menjalankan suatu operasi atau metode menghasilkan suatu kembalian ke objek tertentu, arah panah mengarahkan pada objek yang menerima kembalian.
9		Pesan Tipe *Destroy*	Menyatakan suatu objek mengakhiri hidup objek yang lain, arah panah mengarah pada objek yang diakhiri, sebaiknya jika ada *create* maka ada *destroy*.

Sumber : Rosa dan Shalahuddin (2013:165 - 167)

2.4.1.5 *Component Diagram*

Diagram komponen di buat untuk menunjukkan organisasi dan ketergantungan di antara kumpulan komponen dalam sebuah sistem. Diagram komponen fokus pada komponen sistem yang dibutuhkan dan ada di dalam sistem. Diagram komponen juga dapat digunakan untuk memodelkan hal-hal berikut :

1. *Source code* program perangkat lunak.
2. Komponen *executable* yang di lepas ke *user.*
3. Basis data secara fisik.
4. Sistem yang harus beradaptasi dengan sistem lain.
5. *Framework* sistem, *framework* pada perangkat lunak merupakan kerangka kerja yg di buat untuk memudahkan pengembangan dan pemeliharaan aplikasi.

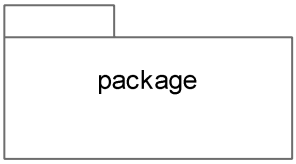
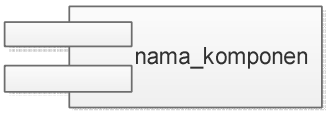
Komponen dasar yang biasanya ada dalam suatu sistem adalah sebagai berikut :

1. Komponen *user interface* yang menangani tampilan.
2. Komponen *bussiness procesiing* yang menangani fungsi-fungsi proses bisnis .
3. Komponen data yang menangani manipulasi data.
4. Komponen *security* yang menangani keamanan sistem.

Komponen lebih berfokus pada penggolongan secara umum fungsi-fungsi yang diperlukan.

Simbol-simbol *Component Diagram* ditunjukkan pada Tabel 2.5.

Tabel 2.5 Simbol-simbol *Component Diagram*

NO.	SIMBOL	NAMA	KETERANGAN
1.		*Package*	*Package* merupakan sebuah bungkusan dari satu atau lebih komponen.
2.		Komponen	Komponen sistem.
3.		Kebergantungan / *Depedency*	Kebergantungan antar komponen, arah panah mengarah pada komponen yang di pakai.
4.		Antarmuka / *Interface*	Sama dengan konsep *interface* pada pemrograman beroreintasi objek, yaitu sebagai antarmuka komponen agar tidak mengakses langsung komponen.
5.		*Link*	Relasi antar komponen.

Sumber : Rosa dan Shalahuddin (2013:148 - 150)

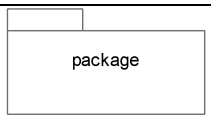
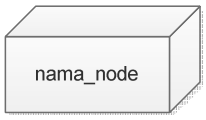
2.4.1.6 *Deployment Diagram*

Diagram *deployment* atau *deployment diagram* menunjukkan konfigurasi komponen dalam proses eksekusi aplikasi. Diagram *deployment* juga dapat digunakan untuk memodelkan hal-hal berikut :

1. Sistem tambahan (*embedded system*) yang menggambarkan rancangan *device, node* dan *hardware.*
2. Sistem *client / server*.
3. Sistem terdistribusi murni.
4. Rekaya ulang aplikasi.

Simbol-simbol *Deployment Diagram* ditunjukkan pada Tabel 2.6.

Tabel 2.6 Simbol-simbol *Deployment Diagram*

NO.	SIMBOL	NAMA	KETERANGAN
1.		*Package*	*Package* merupakan sebuah bungkusan dari satu atau lebih *node*.
2.		*Node*	Mengacu pada perangkat keras (*hardware*), perangkat lunak yang tidak di buat sendiri (*softwere*) jika di dalam *node* disertakan komponen untuk mengkonsistensikan rancangan maka komponen yang diikutsertakan harus sesuai dengan komponen yang telah didefenisikan sebelumnya pada diagram komponen.
3.		Kebergantungan / *Depedency*	Kebergantungan antar *node*, arah panah mengarah pada *node* yang di pakai.
4.		*Link*	Relasi antar *node.*

Sumber : Rosa dan Shalahuddin (2013:154 - 155)

2.4.2 Pengujian Sistem *Black Box*

Sebuah perangkat perlu dijaga kualitasnya bahwa kualitas bergantung kepada kepuasan pelanggan (*customer*). Perangkat lunak mengandung kesalahan (*error*) pada proses-proses tertentu pada saat perangkat lunak berada ditangan *user* . Kesalahan-kesalahan (*error*) pada perangkat lunak ini disebut *bug*, untuk menghindari banyaknya *bug* maka diperlukan adanya pengujian perangkat lunak sebelum perangkat lunak diberikan kepelanggan atau perangkat lunak masih terus dikembangkan.

Menurut Rosa dan Shalahuddin (2013) *Black Box Testing* yaitu menguji perangkat lunak dari segi spesifikasi fungsional tanpa menguji desain dan kode program. Pengujian dimaksudkan untuk mengetahui apakah fungsi-fungsi, masukan dan keluaran dari perangkat lunak sesuai dengan spesifikasi yang dibutuhkan. Pengujian *Black Box* dilakukan dengan membuat kasus uji yang bersifat mencoba semua fungsi dengan memakai perangkat lunak apakah sesuai

dengan spesifikasi yang dibutuhkan. Kasus uji yang dibuat untuk melakukan pengujian *Black Box* harus dibuat dengan kasus benar dan kasus salah, misalkan untk kasus proses *login* maka kasus uji yang dibuat adalah:

1. Jika *user* memasukkan nama pemakai (*username*) dan kata sandi (*password*) yang benar.
2. Jika *user* memasukkan nama pemakai (*username*) dan kata sandi (*password*) yang salah, misalnya nama pemakai benar tapi kata sandi salah, atau sebaliknya, atau keduanya salah.