BAB II
LANDASAN TEORI

2.1 *Cloud Computing*

Menurut Peter Mell dan Timothy Grance (2012) definisi *Cloud Computing* adalah sebuah model yang memungkinkan untuk ubiquitous (Diamana pun dan kapanpun), nyaman, *On-demand* akses jaringan ke sumber daya komputasi (contoh: jaringan, server, storage, aplikasi, dan layanan) yang dapat dengan cepat dirilis atau ditambahkan. *Cloud Computing* sebagai suatu layanan teknologi informasi yang dapat dimanfaatkan oleh pengguna dengan berbasis jaringan/internet. Dimana suatu sumber daya, perangkat lunak, informasi dan aplikasi disediakan untuk digunakan oleh komputer lain yang membutuhkan. *Cloud computing* mempunyai dua kata "*Cloud*" dan "*Computing*". *Cloud* yang berarti internet itu sendiri dan *Computing* adalah proses komputasi.

Dalam pasar cloud, ada 3 pihak yang terkait di dalamnya. Tiga pihak tersebut adalah sebagai berikut:

1. End user : merupakan pengguna yang kurang paham tentang penggunaan teknologi secara keseluruhan. Pengguna hanya paham tentang hal-hal yang umum saja. Sebagai contoh, pada situs jejaring sosial Facebook mereka adalah seluruh pengguna yang terdaftar.
2. Business management : merupakan pihak yang bertanggung jawab atas keseluruhan data dan servis suatu perusahaan yang berada di cloud. Mereka adalah para pengelola sistem IT perusahaannya. Sebagai contoh, mereka adalah pemilik dan pengelola aplikasi pihak ke-3 yang ada di Facebook.
3. Cloud service provider : pemilik sekaligus pengelola dari layanan cloud.

2.1.1 Pembagian *Cloud Computing* Menurut Jenis Layanan

Pada dasarnya *cloud* memiliki 3 jenis layanan. Sebagian besar dari layanan ini digunakan ketiganya secara berkolaborasi. Akan tetapi ada juga yang hanya menggunakan salah satu dari ketiga jenis layanan ini. Ketiga jenis layanan

tersebut antara lain Infrastructure as a service, Platform as a service, dan Software as a service.

1. Infrastructure As A Service (IAAS).

   Layanan IaaS menawarkan perangkat keras komputer (server, jaringan, storage dan ruang pusat data) sebagai layanan. Adapun alasan penyewaan layanan perangkat keras ini disebabkan oleh empat hal:

   1. Harga
   2. Kumpulan resource
   3. Kecepatan penyebaran
   4. Keamanan

   Ada beberapa hal juga yang harus diperhatikan sebelum memilih provider yang menyajikan layanan ini (Harsono, 2010).

   1. Tingkat keamanan Keamanan merupakan hal yang fundamental menyangkut dengan keberadaan data, kepemilikan data, dan integritas data milik kita.
   2. Transparansi kontrol Penyediaan antarmuka untuk pelanggan dengan kontrol penuh atau sebagian, bergantung dari layanan yang diberikan oleh cloud provider.
   3. Biaya untuk resource aktual Tiap pengeluaran untuk resource yang kita gunakan harus diperhatikan. Biaya yang dikeluarkan belum tentu sepadan dengan apa yang kita dapatkan. Untuk itu perlu diteliti ulang untuk setiap rincian perangkat yang ada.
   4. 99.9% uptime Jaminan bahwa infrastruktur yang kita sewa online 24 jam merupakan hal yang utama untuk sebuah aplikasi yang berjalan pada web. Keandalan infrastruktur diukur dari sistem backup data, integritas data, dan jaminan server online 24 jam.
   5. Fitur tambahan yang diberikan Fitur tambahan memberikan nilai tambah untuk masing-masing *cloud provider*. Fitur tambahan memberikan opsi tambahan untuk pengguna dengan fungsi yang tentu berhubungan dengan apa yang kita butuhkan.

2. Platform As A Service (PAAS)

   Agar sebuah aplikasi berjalan pada cloud, dibutuhkan sebuah wadah yang berjalan disemua kondisi. Sebuah platform memberikan kemudahan bagi developer untuk membangun aplikasi serta memigrasikan aplikasinya ke dalam cloud. Kunci utama sebuah platform adalah memberikan developer sebuah dukungan full-environment dalam membuat aplikasi tanpa harus membeli tool secara terpisah.

   Adapun platform yang dibutuhkan harus memiliki elemen dasar sebagai berikut:

   1. Workflow engine
   2. Development tools
   3. Testing environment
   4. Integrating database
   5. Third-party tools and services

3. Software As A Service (SAAS)

   Layanan yang paling banyak digunakan pada cloud adalah Software as a Service (SaaS). SaaS merupakan layanan yang paling cepat berkembang setelah penggunaannya mulai menyebar dengan pesat seiring dengan laju pertumbuhan pengguna layanan internet.

   SaaS adalah area yang paling matang pada cloud. SaaS memperoleh traksi awal dengan pasar customer relationship management (CRM) dan telah diperluas ke sektor lainnya, khususnya kolaborasi pasar dan penyediaan peralatan dan manajemen lingkungan

   SaaS memiliki karakteristiknya tersendiri agar keberadaannya di pasar komersial tetap dapat berjalan. Adapun karakteristik itu adalah sebagai berikut:

   1. Sebuah aplikasi SaaS perlu disamaratakan menjadi suatu yang umum sehingga banyak pelanggan yang tertarik pada layanan tersebut. Beberapa tipe aplikasi yang pada umumnya digunakan seperti accounting, collaboration, project management, testing, analytics, content management, internet marketing, risk management, dan CRM.

2. Aplikasi SaaS harus menyediakan navigasi yang canggih dan kemudahan penggunaan.
3. Aplikasi SaaS harus berupa berorientasi modular dan berorientasi layanan.
4. Aplikasi SaaS perlu menyertakan pengukuran dan pemantauan sehingga pelanggan dapat dikenakan biaya sesuai penggunaan aktual.
5. Aplikasi SaaS harus memiliki sebuah layanan billing built-in.
6. Aplikasi SaaS harus mempublikasikan antarmuka dan ekosistem dari partner yang dapat memperluas basis pelanggan perusahaan dan tujuan pasar.
7. Aplikasi SaaS harus memastikan bahwa setiap data pelanggan dan konfigurasi khusus terpisah dan aman dari data dan konfigurasi pelanggan lainnya.
8. Aplikasi SaaS harus menyediakan sebuah konfigurasi proses bisnis yang mutakhir untuk pelanggan.
9. Aplikasi SaaS perlu menyediakan peluncuran fitur-fitur yang baru dan kemampuan baru secara konstan.
10. Aplikasi SaaS harus melindungi integritas data pelanggan.

2.2 Metode SOA (*Service Oriented Architechture*)

SOA (Service Oriented Architecture) adalah suatu metode arsitektur system yang membuat dan menggunakan proses bisnis dalam bentuk kumpulan-kumpulan layanan. SOA juga mendefinisikan dan menentukan arsitektur yang dapat menunjang berbagai aplikasi untuk saling bertukar data dan dapat mengembangkan aplikasi dengan proses bisnis yang lain. Fungsi-fungsi yang ada tidak terikat dengan system operasi dan bahasa pemrograman yang mendasari aplikasi aplikasi tersebut. (Anthony T. Velte, Toby J. Velte, Robert Elsenpeter, 2008).

Definisi Service Orientation Architecture (SOA) menurut Open Group adalah sebuah model arsitektur yang mendukung service orientation (John Erickson, Keng Siau, 2008). Definisi tersebut terfokus pada model arsitektur, service orientation, serviceserta fitur – fitur yang menonjol pada SOA. Organization for

Advancement of Structured Information Standards (OASIS) mendefinisikan SOA sebagai paradigma yang digunakan untuk mengatur dan memanfaatkan kemampuan terdistribusi yang mungkin berada di bawah kendali kepemilikan suatu domain yang berbeda (John Erickson, Keng Siau, 2008). Definisi OASIS disebut sebagai "reference model" yang selanjutnya diperluas dan diformalkan.
SOA didefinisikan oleh World Wide Web Consortium (W3C) sebagai suatu bentuk arsitektur sistem terdistribusi yang pada umumnya ditandai dengan logical view, message orientation, description orientation, granularity dan platform neutrality (John Erickson, Keng Siau, 2008). XML.com pada tahun 2007 mendefinisikan SOA sebagai sebuah gaya arsitektur yang memiliki tujuan untuk mencapai loosely couple antara agen perangkat lunak yang berinteraksi (John Erickson, Keng Siau, 2008). Service adalah satuan kerja yang dilakukan oleh penyedia service untuk mencapai hasil akhir yang diinginkan kepada consumer service. Empat karakteristik yang dimiliki oleh SOA berdasarkan Raghu Kodali antara lain :

1. Antarmuka yang disusun dengan XML yang menggunakan WSDL
2. Skema XML yang disebut dengan XSD yang harus digunakan untuk mengolah pesan
3. Registry UDDI berdasarkan pada penyimpanan daftar service yang disediakan
4. setiap service harus mempertahankan tingkat kualitas yang ditetapkan untuk melalui persyaratan keamanan QoS.

IBM mengusulkan bahwa SOA menggambarkan gaya arsitektur yang memperlakukan komponen perangkat lunak sebagai service set(UNL – IBM systemin Global Innovation Hub, 2007). Definisi tersebut ditegaskan sebagai kebutuhan bisnis yang harus mengendalikan definisi dari servicedan nilai tujuan harus terfokus dengan reusabilitydan fleksibilitas serviceyang telah didefinisikan (John Erickson, Keng Siau, 2008).

2.2.1 Prinsip – Prinsip Service Orientation

Pendekatan Service Oriented Architecture (SOA) tidak memiliki prinsip–prinsip

yang baku digunakan untuk pengembangan SOA tersebut. Beberapa prinsip yang seringkali digunakan terkait dengan pendekatan SOA terdapat pada pembahasan di bawah ini (Thomas, 2008):

1. Prinsip 1

   Service dapat digunakan kembali (Reusability) Pengembangan sistem yang menggunakan pendekatan SOA, service dirancang secara khusus untuk mendukung penggunaan kembali sesuai dengan kebutuhannya.

2. Prinsip 2

   Service merupakan formal specifation Service tidak membutuhkan suatu pembagian apa pun di dalam pengembangan yang menggunakan pendekatan SOA untuk dapat berinteraksi dengan service. Service membutuhkan sebuah kontrak yang formal yang dapat mendeskripsikan setiap service yang telah ada dan menentukan persyaratan yang dibutuhkan pada pertukaran informasi yang terjadi.

3. Prinsip 3

   Service merupakan loosely couple Service secara khusus pada pendekatan SOA dirancang untuk dapat berkomunikasi dengan antar service tanpa perlu saling ketergantungan.

4. Prinsip 4

   Inti sari service berdasarkan logika Satu – satunya bagian dari service yang terlihat di dunia luar pada penerapan pendekatan SOA merupakan hal – hal yang ditampilkan melalui kontrak service tersebut. Logika dasar yang melampui hal tersebut dinyatakan ke dalam deskripsi yang terdiri dari kontrak yang tidak nyata dan tidak relevan dengan permintaan dari service tersebut.

5. Prinsip 5

   Decomposition Service Penggunaan SOA menyebabkan service dapat menyusun service yang lain. Hal ini memungkinkan logika yang dapat digambarkan pada tingkat *granularity* yang berbeda dan mempromosikan penggunaannya kembali serta penyusunan dari inti sari yang berada pada layer.

6. Prinsip 6

   Service bersifat otonomi Logika yang menggunakan pendekatan SOA

dipengaruhi oleh sebuah service yang diletakan pada sebuah batasan yang tidak dapat dilihat. Service tersebut akan mengontrol batas tersebut dan untuk mengeksekusinya tidak perlu bergantung dengan service lain.

7. Prinsip 7

   Service bersifat *stateless Service* yang berbasiskan SOA tidak harus membutuhkan pengaturan informasi state. Hal ini dikarenakan state tersebut dapat menghalangi kemampuan service untuk bergabung atau berintegrasi.

8. Service tidak terdeteksi Service yang dirancang harus dapat memungkinkan deskripsi mengenai diri service sendiri di dalam sistem yang telah menerapkan SOA untuk dapat menemukan servicenya tersebut dan dapat dimengerti oleh manusia dan pemohon service tersebut yang dapat menggunakan logika dalam service tersebut.

2.3 Arsitektur SOA

Gambar 2.1 Arsitektur SOA

(Sumber : Barry, 2015)

Pada gambar 2.1 dapat dilihat bahwa arsitektur SOA yang paling sederhana adalah dimana ada sebuah service provider dan sebuah service consumer. Service consumer akan mengirimkan service request ke service provider dan kemudian service provider akan mengirimkan service response ke service consumer.

2.4 Tools Program/Software

2.4.1 Basis Data

Database adalah kumpulan file-file yang mempunyai kaitan antara satu file dangan file yang lain sehingga membentuk satu bangunan data untuk

menginformasikan satu perusahaan, instansi dalam batasan tertentu. (Kristanto, 2003)

Dengan memahami pengertian di atas, maka istilah basis data dapat dipahami sebagai suatu kumpulan data terhubung (*interrelated data*) yang disimpan secara bersama-sama pada suatu media, tanpa mengatap satu sama lain atau tidak perlu suatu kerangkapan data, data disimpan dengan cara-cara tertentu sehingga mudah untuk digunakan atau ditampilkan kembali, data dapat digunakan oleh satu atau lebih program-program aplikasi secara optimal, data disimpan tanpa mengalami ketergantungan dengan program yang akan menggunakanya, data disimpan sedemikian rupa sehingga proses penambahan, pengambilan dan modifikasi data dapat dilakukan dengan mudah dan terkontrol (Sutanta, 2004).

Istilah-istilah yang digunakan dalam basis data:

1) *File*

   Merupakan kumpulan dari atribut *record-record* sejenis yang mempunyai panjang elemen yang sama, atribut yang sama namun berbeda-beda dalam data *value*-nya.

2) *Record*

   merupakan kumpulan dari elemen-elemen yang saling berhubungan atau berkaitan menginformasikan tentang *entry* secara lengkap.

3) *Field*

   merupakan sekumpulan tanda-tanda yang berbentuk kesatuan tersendiri, merupakan bagian terkecil dari *record* dan bentuknya unik dijadikan *field* kunci yang dapat mewakili *record*-nya.

4) Entity

   merupakan tempat kejadian atau konsep yang informasikan direkam.

Sutanta (2004), mengatakan bahwa perancangan basis data dan model data secara umum dapat dibagi menjadi beberapa kelompok, yaitu:

1) Model data berbasis objek
2) Model data berbasis *record*

3) Model data fisik

4) Model data konseptual

Dan dari model-model data tersebut, model data yang sering digunakan pada umumnya adalah model data berbasis objek dan model data berbasis *record.*

2.4.2 MySQL

MySQL adalah sebuah perangkat lunak sistem manajemen basis data SQL (bahasa Inggris: *database management system*) atau DBMS yang *multithread*, *multi-user*, dengan sekitar 6 juta instalasi di seluruh dunia. MySQL AB membuat MySQL tersedia sebagai perangkat lunak gratis dibawah lisensi GNU General Public License (GPL), tetapi mereka juga menjual dibawah lisensi komersial untuk kasus-kasus dimana penggunaannya tidak cocok dengan penggunaan GPL. MySQL adalah Relational Database Management System (RDBMS) yang didistribusikan secara gratis dibawah lisensi GPL (General Public License). Dimana setiap orang bebas untuk menggunakan MySQL, namun tidak boleh dijadikan produk turunan yang bersifat closed source atau komersial. MySQL sebenarnya merupakan turunan salah satu konsep utama dalam database sejak lama, yaitu SQL (Structured Query Language). SQL adalah sebuah konsep pengoperasian database, terutama untuk pemilihan atau seleksi dan pemasukan data, yang memungkinkan pengoperasian data dikerjakan dengan mudah secara otomatis (Lukmanul, 2006).

2.4.3 HTML (Hyper Text Markup Language)

HTML merupakan kepanjangan dari *Hyper Text Markup Language* adalah suatu bahasa yang digunakan untuk membuat halaman-halaman hypertext (*hypertext page*) pada internet. Dengan konsep hypertext ini, untuk membaca suatu dokumen anda tidak harus melakukannya secara urut, baris demi baris, atau halaman demi halaman. Tetapi anda tidak dapat dengan mudah melompat dari satu topik ke topic lainnya yang anda sukai, seperti halnya jika anda melakukan pada online Help dari suatu aplikasi Windows.

HTML dirancang untuk digunakan tanpa tergantung pada suatu platform tertentu (*platform independent*).(Kadir, 2003).

2.4.4 PHP (Hypertext Preproccessor)

PHP diciptakan oleh Rasmus Lerdorf, seorang pemrogram C yang handal. Semula PHP hanya digunakan untuk mencatat jumlah pengunjung pada homepagenya. Rasmus adalah seorang pendukung open source. Karena itulah ia mengeluarkan Personal Home Page Tools versi 1.0 secara gratis. Setelah mempelajari YACC dan GNU Bison, Rasmus menambah kemampuan PHP 1.0 dan menerbitkan PHP 2.0. PHP mudah dibuat dan cepat dijalankan, PHP dapat berjalan dalam web server yang berbeda dan dalam sistem operasi yang berbeda pula. PHP dapat berjalan di sistem operasi UNIX, Windows 98, Windows XP, Windows NT, dan Macintosh.

Menurut Luke Welling dan Laura Thomson (2001), PHP adalah *server-side scripting language* yang didesain secara spesifik untuk web. Dalam *page HTML*, dapat dimasukkan *code* PHP yang akan dieksekusi setiap kali halaman dikunjungi. PHP *code* diterjemahkan di *web-server* dan dirubah menjadi HTML atau output lain yang akan dilihat oleh pengunjung halaman. PHP adalah bahasa pemrograman script yang paling banyak dipakai saat ini. PHP banyak dipakai untuk pemrograman situs web dinamis, walaupun tidak tertutup kemungkinan digunakan untuk pemakaian lain. Contoh terkenal dari aplikasi PHP adalah phpBB dan MediaWiki (software di belakang Wikipedia). PHP juga dapat dilihat sebagai pilihan lain dari ASP.NET/C#/VB.NET Microsoft, ColdFusion Macromedia, JSP/Java Sun Microsystems, dan CGI/Perl. Contoh aplikasi lain yang lebih kompleks berupa CMS yang dibangun menggunakan PHP adalah Mambo, Joomla, Postnuke, Xaraya, dan lain-lain.

Kelebihan PHP :

1. PHP adalah sebuah bahasa script yang tidak melakukan sebuah kompilasi dalam penggunaanya.
2. Web Server yang mendukung PHP dapat ditemukan dimana - mana dari mulai IIS sampai dengan apache, dengan konfigurasi yang relatif

mudah.

3. Dalam sisi pengembangan lebih mudah, karena banyaknya milis - milis dan developer yang siap membantu dalam pengembangan.
4. Dalam sisi pemahamanan, PHP adalah bahasa scripting yang paling mudah karena referensi yang banyak. PHP adalah bahasa open source yang dapat digunakan di berbagai mesin (linux, unix, windows) dan dapat dijalankan secara runtime melalui console serta juga dapat menjalankan perintah-perintah system.

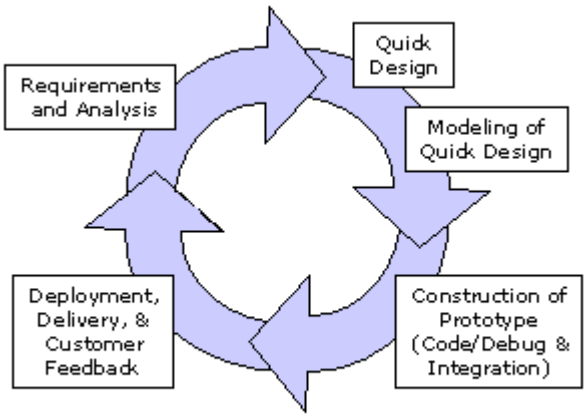
Selain berbagai kelebihan PHP, PHP memiliki beberapa masalah atau kekurangan. Pertama, dari segi bahasa ia bukanlah bahasa yang ideal untuk pengembangan berskala besar. Kekurangan utama adalah tidak adanya namespace. Namespace merupakan sebuah cara untuk mengelompokkan nama variabel atau fungsi dalam susunan hirarkis.

2.5 Metode Pengembangan Perangkat Lunak Menggunakan Metode Prototype

Pressman (2007) mengatakan Prototype Model adalah salah satu metode pengembangan perangkat lunak yang banyak digunakan. Dengan Metode Prototyping ini pengembangan dan pelanggan dapat saling berinteraksi selama proses pembuatan sistem. Sering terjadi seorang pelanggan hanya mendefinisikan secara umum apa yang dibutuhkan, Pemrosesan dan data-data apa saja yang dibutuhkan. Sebaliknya disisi pengembang Kurang memperhatikan efesiensi Algoritma. Kemampuan sistem oprasi dan interface yang menghubungkan manusia dengan komputer.

Pada Prototyping model kadang – kadang klien hanya memberikan beberapa kebutuhan umum software tanpa detail input, proses atau detail output dilain waktu mungkin tim pembangun (developer) tidak yakin terhadap efesiensi dari algoritma yang digunakan, tingkat adaptasi terhadap sistem operasi atau rancangan form user interface. Ketika situasi seperti ini terjadi model prototyping

sangat membantu proses pembangunan software. Proses pada prototyping dalam Pressman (2007) ditunjukkan pada gambar 2.1 berikut :



Gambar 2.2 Metode Pengembangan Perangkat Lunak Metode Prototype

Tahap-tahap pengembangan Prototype adalah :

1. Pengumpulan Kebutuhan

   Pada tahap ini dilakukan pengumpulan kebutuhan dari system dengan cara mendengar keluhan dari pelanggan. Untuk membuat suatu system yang sesuai kebutuhan, maka harus diketahui terlebih dahulu bagaimana system yang sedang berjalan untuk kemudian mengetahui masalah yang terjadi.

2. Pemodelan Sistem

   Pada tahap ini, dilakukan perancangan dan pembangunan prototype system. Prototype yang dibuat disesuaikan dengan kebutuhan system yang telah didefinisikan sebelumnya dari keluhan pelanggan atau pengguna.

3. Pembangunan Prototype (Pengkodean Sistem)

   Membangun prototyping dengan membuat perancangan sementara yang berfokus pada penyajian kepada pelanggan (misalnya dengan membuat input dan format output).

4. Pengkodean Sistem

   Dalam tahap ini prototyping yang sudah di sepakati diterjemahkan ke dalam bahasa pemrograman yang sesuai.

5. Evaluasi Sistem

Pada tahap ini, Prototype dari system di uji coba oleh pelanggan atau pengguna. Kemudian dilakukan evaluasi kekurangan-kekurangan dari kebutuhan pelanggan. Pengembangan kemudian kembali mendengarkan keluhan dari pelanggan untuk memperbaiki Prototype yang ada.

2.5.1 Kelebihan Metode Prototype

Kelebihan metode Prototype:

1) Adanya komunikasi yang baik antara pengembang dan pelanggan.
2) Pengembangan dapat bekerja lebih baik dalam menentukan kebutuhan pelanggan.
3) Lebih menghemat waktu dalam pengembangan system.
4) Penerapan menjadi lebih mudah karena pemakai mengetahui apa yang diharapkannya.

2.5.2 Kekurangan Metode Prototype

Kekurangan metode Prototype :

1) Resiko tinggi yaitu untuk masalah-masalah yang tidak terstruktur dengan baik, ada perubahan yang besar dari waktu ke waktu, dan adanya persyaratan data yang tidak menentu.
2) Interaksi pemakai penting. Sistem harus menyediakan dialog on-line antara pelanggan dan komputer.
3) Hubungan pelanggan dengan komputer yang disediakan mungkin tidak mencerminkan teknik perancangan yang baik.

2.6 Unified Modelling Language (UML)

UML yang merupakan singkatan dari Unified Modelling Language adalah sekumpulan pemodelan konvensi yang digunakan untuk menentukan atau menggambarkan sebuah sistem perangkat lunak dalam kaitannya dengan objek. (Rossa, 2013).

UML dapat juga diartikan sebuah bahasa grafik standar yang digunakan untuk memodelkan perangkat lunak berbasis objek. UML pertama kali dikembangkan pada pertengahaan tahun 1990an dengan kerjasama antara James Rumbaugh,

Grady Booch dan Ivar Jacobson, yang masing-masing telah mengembangkan notasi mereka sendiri di awal tahun 1990. (Rossa, 2013).

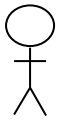
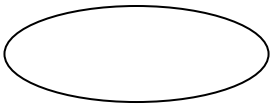
2.6.1 Komponen-komponen UML

UML mendefinisikan diagram-diagram berikut ini: (Rossa, 2013).

1) Use case Diagram

Use case diagram menggambarkan fungsionalitas yang diharapkan dari sebuah sistem. Yang ditekankan adalah "apa" yang diperbuat sistem, dan bukan "bagaimana". Sebuah use case merepresentasikan sebuah interaksi antara aktor dengan sistem.

Tabel 2.2 Simbol *Use Case*

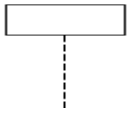
Simbol	Keterangan
Aktor 	Merupakan kesatuan *eksternal* yang berinteraksi dengan sistem.
Use Case 	Rangkaian/uraian sekelompok yang saling terkait dan membentuk sistem.
Generelation 	Menggambarkan hubungan khusus atau interaksi dalam objek.

2) Class Diagram

a) Class adalah sebuah spesifikasi akan menghasilkan sebuah objek dan merupakan inti dari pengembangan dan desain berorientasi objek. Class menggambarkan keadaan (atribut/properti) suatu sistem, sekaligus menawarkan layanan untuk memanipulasi keadaan tersebut (fungsi).

b) Class diagram menggambarkan struktur dan deskripsi class, package dan objek beserta hubungan satu sama lain seperti pewarisan, asosiasi, dan lain-lain.

Tabel 2.3. Simbol Class Diagram

GAMBAR	NAMA	KETERANGAN
	*LifeLine*	Objek *entity*, antarmuka yang saling berinteraksi.
	*Message*	Spesifikasi dari komunikasi antar objek yang memuat informasi-informasi tentang aktifitas yang terjadi
	*Message*	Spesifikasi dari komunikasi antar objek yang memuat informasi-informasi tentang aktifitas yang terjadi

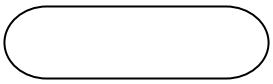
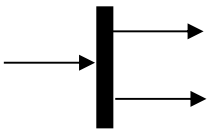
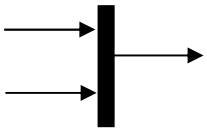
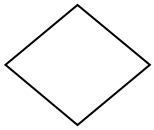
3) Activity Diagram

a) Activity diagram menggambarkan berbagai alir aktivitas dalam sistem yang sedang dirancang, bagaimana masing-masing alir berawal, decision yang mungkin terjadi, dan bagaimana mereka berakhir. Activity diagram juga dapat menggambarkan proses paralel yang mungkin terjadi pada beberapa eksekusi.

b) Activity diagram merupakan state diagram khusus, yang sebagian besar state adalah action dan sebagian besar transisi di-trigger oleh selesainya state sebelumnya (*internal processing*). Oleh karena itu, activity diagram tidak menggambarkan perilaku internal sebuah sistem dan interaksi antar subsistem, tetapi lebih menggambarkan proses-proses dan jalur-jalur aktivitas dari level atas secara umum.

Tabel 2.4 Simbol *Activity Diagram*

Simbol	Keterangan

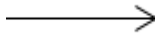
Start State 	*Start state* adalah sebuah kondisi awal sebuah *object* sebelum ada perubahan keadaan. Start state digambarkan dengan sebuah lingkaran solid.
End State 	*End state* adalah menggambarkan ketika objek berhenti memberi respon terhadap sebuah event. *End state* digambarkan dengan lingkaran solid di dalam sebuah lingkaran kosong.
State/Activities 	*State* atau *activities* menggambarkan kondisi sebuah entitas, dan digambarkan dengan segiempat yang pinggirnya.
Fork (Percabangan) 	*Fork* atau percabangan merupakan pemisalah beberapa aliran konkuren dari suatu aliran tunggal.
Join (Penggabungan) 	*Join* atau penggabungan merupakan pengabungan beberapa aliran konkuren dalam aliran tunggal.
Decision 	*Decision* merupakan suatu logika aliran konkuren yang mempunyai dua cabang aliran konkuren.

4) Sequence Diagram

Sequence diagram secara grafis menggambarkan bagaimana objek berinteraksi antara satu sama lain melalui pesan pada sebuah *use case* atau operasi.

Tabel 2.5 Simbol Sequence Diagram

GAMBAR	NAMA	KETERANGAN
	*State*	Nilai atribut dan nilai link pada suatu waktu tertentu, yang dimiliki oleh suatu objek.
	*Initial Pseudo State*	Bagaimana objek dibentuk atau diawali
	*Final State*	Bagaimana objek dibentuk dan dihancurkan

	*Transition*	Sebuah kejadian yang memicu sebuah state objek dengan cara memperbaharui satu atau lebih nilai atributnya
	Association	Apa yang menghubungkan antara objek satu dengan objek lainnya.
	*Node*	Elemen fisik yang eksis saat aplikasi dijalankan dan mencerminkan suatu sumber daya komputasi.