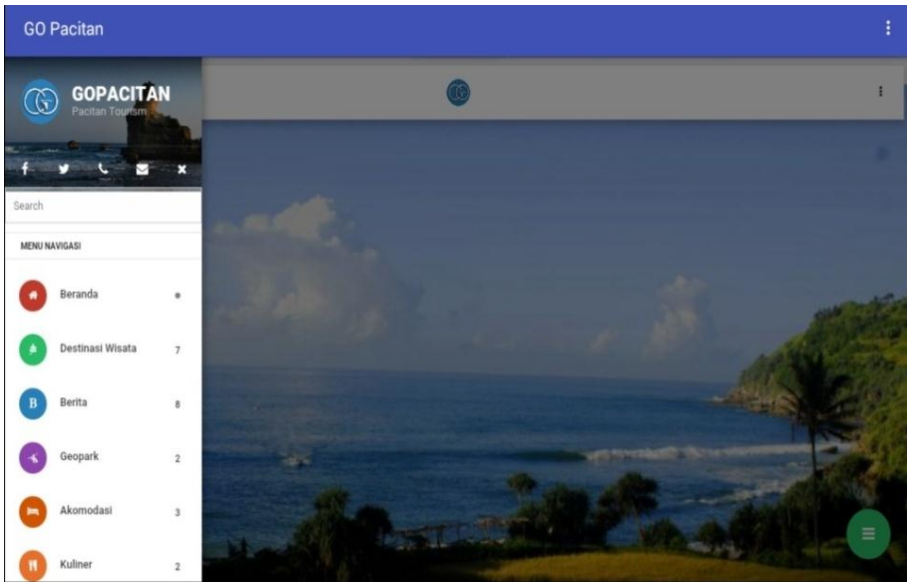
BAB II
LANDASAN TEORI

2.1. Tinjauan Pustaka

Tinjauan pustaka membandingkan aplikasi yang sudah ada dengan aplikasi yang akan dibuat. Aplikasi yang sudah ada diantaranya adalah sebagai berikut:

2.1.1. Aplikasi *Go* Pacitan

Aplikasi *Go* Pacitan adalah aplikasi wisata Kabupaten Pacitan yang dibuat dan di-*upload* di *playstore.* Aplikasi ini adalah aplikasi tentang tempat – tempat wisata di Kota Pacitan yang disertai dengan lokasi serta deskripsi dari tempat-tempat wisata tersebut sehingga pengguna akan lebih mudah untuk memilih tempat-tempat wisata sesuai dengan pertimbangannya masing-masing. Di samping itu, aplikasi *Go* Pacitan juga menyediakan informasi-informasi yang terkait dengan kebutuhan wisatawan. Aplikasi ini menyediakan 16 menu yaitu menu beranda, destinasi wisata, berita, *geopark*, akomodasi, kuliner, rumah makan, biro perjalanan, kerajinan, agenda kegiatan, galeri foto, galeri video, fasilitas umum, lain-lain, *download*, serta hubungi kami yang dapat dilihat pada Gambar 2.1.



Gambar 2.1. Tampilan Menu *Go* Pacitan

Kelemahan dari Aplikasi *Go* Pacitan adalah aplikasi ini tidak dapat dijalankan ketika tidak terkoneksi dengan internet. Disamping itu, dari segi konten aplikasi ini juga masih belum lengkap. Hal ini dapat dilihat pada menu berita, meskipun dalam menu tersebut terdapat delapan sub-menu, tetapi sub-menu tersebut hanya laman kosong, begitu juga dengan menu *download*.

2.1.2. Aplikasi Wisata Ponorogo (Reog Media)

Aplikasi Wisata Ponorogo marupakan aplikasi wisata yang dibuat dan di *upload* di *playstore* yang bertujuan untuk memberikan informasi terkait dengan wisata di Kabupaten Ponorogo. Aplikasi ini terdiri dari empat menu utama yaitu wisata, ubah bahasa, tentang dan keluar. Tampilan utama pada aplikasi ini ditunjukan pada Gambar 2.2.

Gambar 2.2. Tampilan Menu Utama Wisata Ponorogo

Kelemahan dari aplikasi Wisata Ponorogo (Reog Media) antara lain adalah, kurang lengkapnya menu dalam aplikasi wisata tersebut, karena dalam tampilan utama hanya terdapat empat menu, di samping itu aplikasi ini juga tidak mencantumkan lokasi wisatanya secara spesifik dan tidak terkoneksi dengan *Google Maps*, sehingga pengguna akan menemukan kesulitan dalam mengakses lokasi wisata tersebut. Dalam segi tampilan, aplikasi ini juga kurang menarik baik warna ataupun gambarnya. Menu dalam aplikasi ini juga ada yang tidak berfungsi, misalnya saja pada menu ubah bahasa. Selain itu untuk keluar dari aplikasi ini, pengguna harus klik tiga kali pada menu keluar.

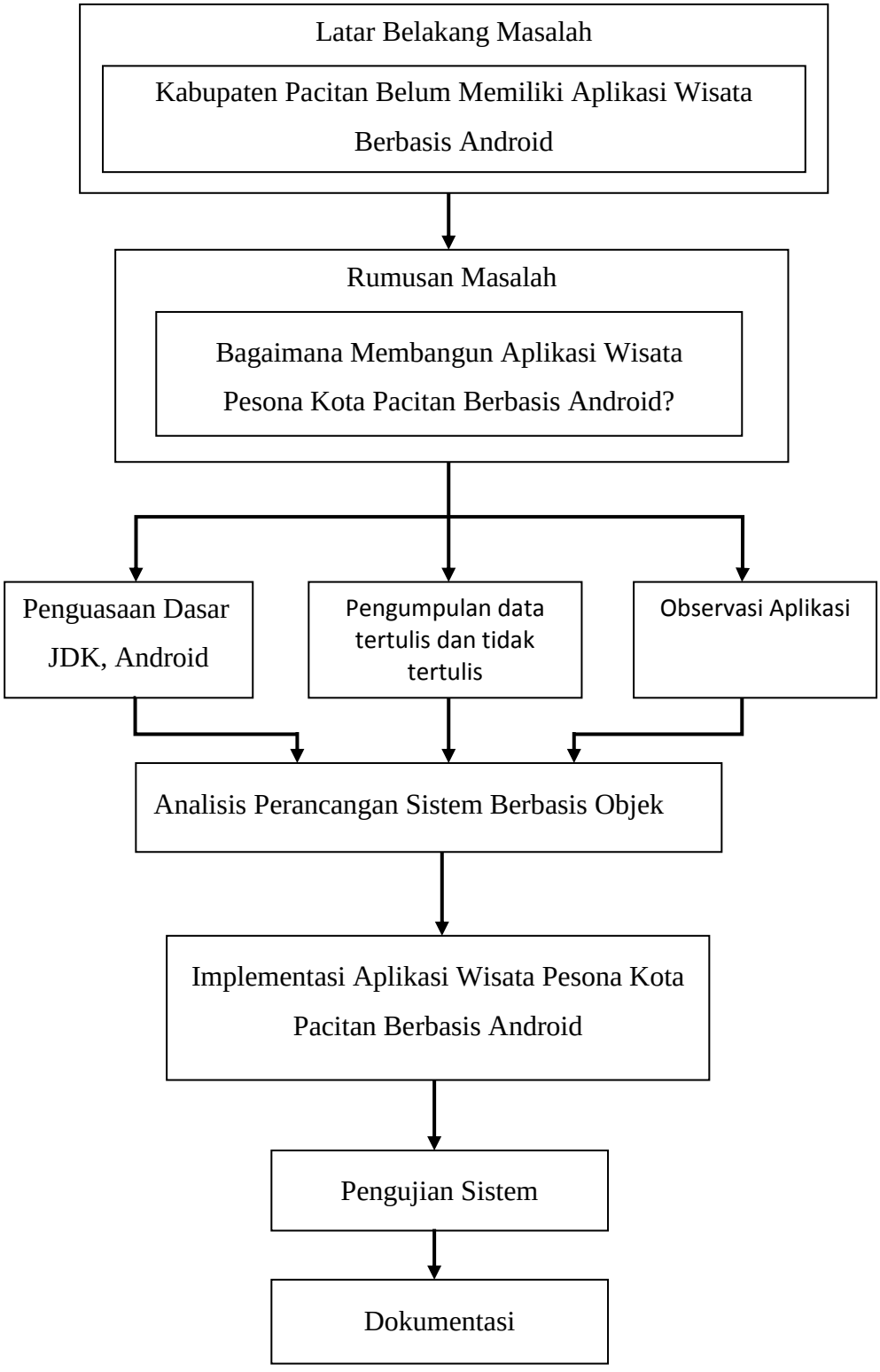
2.1.3. Aplikasi Wisata Wonogiri

Aplikasi Wisata Wonogiri merupakan aplikasi wisata yang dibuat dan di *upload* di *playstore* yang bertujuan untuk memberikan informasi terkait dengan wisata di Kabupaten Wonogiri. Aplikasi ini terdiri dari dua menu utama yaitu wisata dan area wisata. Tampilan halaman utama pada aplikasi ini ditunjukan pada Gambar 2.3.

Gambar 2.3. Tampilan Halaman Utama Wisata Wonogiri

Kelemahan dari aplikasi Wisata Wonogiri antara lain adalah, kurang lengkapnya menu dalam aplikasi wisata tersebut, karena dalam tampilan utama hanya terdapat dua menu, di samping itu aplikasi ini juga tidak mencantumkan lokasi wisatanya secara spesifik dan tidak terkoneksi dengan *Google Maps*, sehingga pengguna akan menemukan kesulitan dalam mengakses lokasi wisata tersebut. Dalam segi tampilan, aplikasi ini juga masih terlalu sederhana dari segi warna ataupun gambarnya dan kurang lengkapnya galeri gambar pada setiap informasi wisata tersebut.

2.2. Kerangka Berfikir



Gambar 2.4. Kerangka Berfikir

2.3. Teori Pendukung

2.3.1. Pengertian Aplikasi

Menurut Abdul Kadir (2003), aplikasi adalah suatu program yang dibuat oleh pemakai yang ditujukan untuk melakukan suatu tugas khusus. Berdasarkan definisi di atas dapat disimpulkan bahwa aplikasi adalah program yang dibuat untuk melakukan tugas khusus dalam perusahaan. Menurut Aji Supriyanto (2005), aplikasi adalah program yang memiliki aktivitas pemrosesan perintah yang diperlukan untuk melaksanakan permintaan pengguna dengan tujuan tertentu.

2.3.2. Pengertian Wisata

Menurut Kamus Besar Bahasa Indonesia (2017), pengertian wisata adalah bepergian secara bersama-sama dengan tujuan untuk bersenang-senang, menambah pengetahuan, dan lain-lain. Selain itu juga dapat diartikan sebagai bertamasya atau piknik.

Menurut Undang Undang Republik Indonesia No 10 Tahun 2009 pasal 1, wisata adalah kegiatan perjalanan yang dilakukan oleh seseorang atau sekelompok orang dengan mengunjungi tempat tertentu untuk tujuan rekreasi, pengembangan pribadi, atau mempelajari keunikan daya tarik wisata yang dikunjungi dalam jangka waktu sementara.

Menurut Gamal Suwantoro (2004), wisata adalah suatu proses bepergian yang bersifat sementara yang dilakukan seseorang untuk menuju tempat lain di luar tempat tinggalnya. Motif kepergiannya tersebut bisa karena ekonomi, kesehatan, agama, budaya, sosial, politik, dan kepentingan lainnya.

Dari beberapa pendapat mengenai definisi wisata di atas dapat disimpulkan bahwa wisata adalah perjalanan yang dilakukan oleh satu orang atau lebih ke suatu tempat di luar tempat tinggalnya yang bersifat sementara dengan tujuan untuk bersenang-senang ataupun untuk tujuan tertentu.

2.3.3. Pengertian *Android*

Menurut Hendra Nugraha Lengkong (2015) *Android* merupakan *subset* perangkat lunak untuk perangkat *mobile* yang meliputi sistem operasi, *middleware*¸ dan aplikasi inti yang dirilis oleh *Google. Android* adalah sistem operasi bergerak *(mobile operating system)* yang mengadopsi sistem operasi *linux*,

namun telah dimodifikasi. *Android* diambil alih oleh *Google* pada tahun 2005 dari *Android,Inc* sebagai bagian strategi untuk mengisi pasar sistem operasi bergerak. *Google* mengambil alih seluruh hasil kerja *Android* termasuk tim yang mengembangkan *Android*.

2.3.3.1.Sejarah *Android*

Menurut Abdul Kadir (2013:2), awalnya *Android* dikembangkan oleh perusahaan kecil di *Silicon Valley* yang bernama *Android Inc.* Selanjutnya, *Google* mengambil alih sistem operasi tersebut pada tahun 2005 dan mencanangkannya sebagai sistem operasi bersifat "*Open Source*". Sebagai konsekuensinya, siapapun boleh memanfaatkanya dengan gratis, termasuk dalam hal kode sumber yang digunakan untuk menyusun sistem operasi tersebut.

Perjalanan *Android* dimulai sejak Oktober 2003 ketika 4 orang pakar IT, Andi Rubin, Rich Minner, Nick Sears dan Chris White mendirikan *Android.Inc*, di California US. Visi *Android* untuk mewujudkan *mobile device* yang lebih peka dan mengerti pemiliknya, kemudian menarik raksasa dunia maya *Google*.*Google* kemudian mengakuisisi *Android* pada Agustus 2005. Sistem operasi Android dibangun berbasis *platform Linux* yang bersifat *open source*, senada dengan *Linux*, *Android* juga bersifat *Open Source.* Dengan nama besar *Google* dan konsep *open source* pada sistem operasi *Android*, tidak membutuhkan waktu lama bagi *Android* untuk bersaing dan menyisihkan *Mobile OS* lainnya seperti *Symbian, Windos Mobile, Blackberry* dan *iOS*. Kini siapa yang tak kenal *Android* yang telah menjelma menjadi penguasa *Operating System* bagi *Smartphone.*

2.3.4. Pengertian *Android Studio*

Dalam pembangunan aplikasi android diperlukan IDE (*Integrated Development Environmment*). IDE merupakan aplikasi perangkat lunak yang menyediakan fasilitas lengkap untuk *programmer* komputer untuk pengembangan perangkat lunak, salah satunya yaitu dengan menggunakan *Android Studio*.

Menurut situs resmi *Android Studio (https://developer.android.com/ studio/intro/index.html?hl=id),* Android Studio adalah Lingkungan Pengembangan Terpadu - *Integrated Development Environment* (IDE) untuk pengembangan aplikasi Android, berdasarkan *IntelliJ* IDEA. Selain merupakan

editor kode *IntelliJ* dan alat pengembang yang berdaya guna, Android Studio menawarkan fitur lebih banyak untuk meningkatkan produktivitas anda saat membuat aplikasi *Android*, misalnya:

a. Sistem pembuatan berbasis *Gradle* yang fleksibel.
b. Emulator yang cepat dan kaya fitur.
c. Lingkungan yang menyatu untuk pengembangan bagi semua perangkat Android.
d. *Instant Run* untuk mendorong perubahan ke aplikasi yang berjalan tanpa membuat *APK* baru.
e. *Template kode* dan integrasi GitHub untuk membuat fitur aplikasi yang sama dan mengimpor kode contoh.
f. Alat penguji dan kerangka kerja yang ekstensif.
g. Alat Lint untuk meningkatkan kinerja, kegunaan, kompatibilitas versi, dan masalah-masalah lain.
h. Dukungan C++ dan NDK.
i. Dukungan bawaan untuk *Google Cloud Platform*, mempermudah pengintegrasian *Google Cloud Messaging dan App Engine*.

2.3.5. Pengertian *GPS*

Menurut Hendra Nugraha Lengkong (2015) *Global Positioning System (GPS)* adalah suatu sistem radio navigasi penentuan posisi menggunakan satelit. *GPS* dapat memberikan posisi suatu objek di muka bumi dengan akurat dan cepat (koordinat tiga dimensi x, y, z) dan memberikan informasi waktu serta kecepatan bergerak secara kontinyu di seluruh dunia menurut Hendra Nugraha Lengkong (2015)

Dengan mengamati sinyal-sinyal dari satelit dalam jumlah dan waktu yang cukup, kemudian data yang diterima tersebut dapat dihitung untuk mendapatkan informasi posisi, kecepatan, dan waktu.

2.3.6. Pengertian *JDK*

Java Development Kit (JDK) adalah perangkat pengembangan aplikasi *Java* yang bisa diunduh secara gratis di www.oracle.com/technetwork/java/javase/downloads. perangkat ini mutlak diperlukan untuk membuat aplikasi *Android*,

mengingat aplikasi *Android* itu berbasis *Java*. Sebgaimana diketahui, *Java* adalah salah satu bahasa pemrograman yang biasa digunakan untuk membuat aplikasi. Namun perlu diketahui, tidak semua pustaka dalam *Java* digunakan di *Android*. Sebagai contoh, *Android* tidak menggunakan *Swing* (Abdul Kadir, 2013:4).

2.3.7. Pengertian *Java*

Java menurut definisi dari Sun adalah nama untuk sekumpulan teknologi untuk membuat dan menjalankan perangkat lunak pada komputer *standalone* ataupun pada lingkungan jaringan.

Menurut Rijalul Fikri, dkk, (2005), Java merupakan bahasa pemrograman berorientasi objek dan bebas *platform*, dikembangkan oleh SUN Micro System dengan sejumlah keunggulan yang memungkinkan Java dijadikan sebagai bahasa pengembangan *enterprise*.

Menurut Hendra Nugraha Lengkong (2015), *Java* adalah bahasa berorientasi objek yang dapat digunakan untuk pengembangan aplikasi mandiri, aplikasi berbasis internet, serta aplikasi untuk perangkat-perangkat cerdas yang dapat berkomunikasi lewat internet atau jaringan komunikasi. Di dalam *Java* ada 2 (dua) jenis program berbeda, yaitu aplikasi dan *applet.* Aplikasi adalah program yang biasanya disimpan dan eksekusi dari komputer lokal, sedangkan *applet* adalah program yang biasanya disimpan pada komputer yang jauh yang dikoneksikan pemakai lewat *web browser. Java* bukan turunan langsung dari bahasa manapun. OOP (*object oriented programming)* adalah cara yang ampuh dalam pengorganisasian dan pengembangan perangkat lunak.

2.4. Analisis dan Perancangan Sistem

2.4.1. Analisis Sistem

Analisis sistem adalah penguraian dari suatu sistem informasi yang utuh ke dalam bagian-bagian komponennya dengan maksud untuk mendefinisikan dan mengevaluasi permasalahan-permasalahan, kesempatan-kesempatan, hambatan-hambatan yang terjadi dan kebutuhan-kebutuhan yang diharapkan sehingga diusulkan perbaikan–perbaikan. Analisis sistem bertujuan untuk mengidentifikasi permasalahan-permasalahan yang ada pada sistem di mana aplikasi dibangun

yang meliputi perangkat keras *(hardware)*, perangkat lunak *(software)* dan pengguna *(User)*. Analisis ini diperlukan sebagai dasar bagi tahapan perancangan sistem. Analisis sistem meliputi spesifikasi aplikasi, spesifikasi pengguna, dan lingkungan operasi (Jogiyanto Hartanto, 2001).

Menurut Adi Nugroho (2005:154), analisis dilakukan dengan menggambarkan 3 aspek dari suatu objek: struktur statis (model objek), struktur dimnamis yang menggambarkan urutan-urutan interaksi (baik antara pengguna dengan sistem/ perangkat lunak maupun interaksi internal dalam sistem/perangkat lunak itu sendiri), serta jika memang diperlukan transformasi data yang coba digambarkan dengan model-model fungsional..

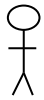
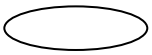
2.4.2. Perancangan Sistem

Perancangan sistem adalah tahap awal dimana pendekatan awal untuk menyelesaikan masalah dipilih. Selama perancangan sistem, struktur keseluruhan diputuskan (Adi Nugroho, 2005:204). Perancangan sistem adalah termasuk bagaimana mengorganisasi sistem ke dalam subsistem-subsistem, serta alokasi subsistem-subsistem ke komponen-komponen perangkat keras, perangkat lunak, serta prosedur-prosedur.

2.4.2.1. *Use Case Diagram*

Use Case diagram adalah teknik untuk merekam persyaratan fungsional sebuah sistem. *Use case* mendeskripsikan interaksi tipikal antara para pengguna sistem dengan sistem itu sendiri, dengan memberi sebuah narasi tentang bagaimana sistem tersebut digunakan (Martin Fowler, 2005:141). *Use Case* Diagram dibuat untuk memvisualisasikan atau menggambarkan hubungan antara *Actor* dan *Use Case*. Simbol-simbol yang digunakan pada *use case diagram* disajikan pada Tabel 2.1.

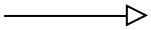
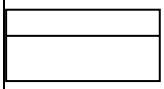
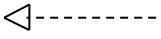
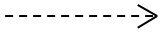
Tabel 2.1. Simbol *Use Case Diagram*

NO	GAMBAR	NAMA	KETERANGAN
1.		*Aktor*	*Idealization* orang eksternal, proses, atau hal yang berinteraksi dengan sistem, subsistem, atau kelas.
2.		*Use case*	Sebuah *use case* menggambarkan interaksi dengan *actor* sebagai urutan pesan antara sistem aktor satu atau lebih.
3.		*System Boundary*	Menspesifikasikan paket yang menampilkan sistem secara terbatas.
4.		*Generalization*	Hubungan antara *use case* umum dan *use case* yang lebih spesifik yang mewarisi dan menambahkan fitur.
5.		*Comunication Association*	Jalur komunikasi antara *actor* dan *use case* yang berpartisipasi didalam.
6.		*Extend*	Penyisipan perilaku tambahan ke dalam basis *use case* yang tidak tahu tentang hal itu.
7.		*include*	Penyisipan perilaku tambahan ke dalam basis *use case* yang secara eksplisit menggambarkan penyisipan.

2.4.2.2. *Class Diagram*

Class Diagram mendeskripsikan jenis-jenis objek dalam sistem dan berbagai macam hubungan statis yang terdapat diantara mereka. *Class Diagram* juga menunjukkan properti dan operasi sebuah *class* dan batasan-batasan yang terdapat dalam hubungan-hubungan objek tersebut (Martin Fowler, 2005:53). Simbol-simbol yang digunakan pada *class diagram* disajikan pada Tabel 2.2.

Tabel 2.2. Simbol *Class Diagram*

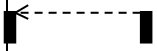
NO	GAMBAR	NAMA	KETERANGAN
1.		*Generalization*	Hubungan dimana objek anak (*descendent*) berbagai perilaku dan struktur data dari objek yang ada di atas objek induk (*ancestor*).
2.		*Nary Association*	Upaya untuk menghindari asosiasi dengan lebih dari dua objek.
3.		*Class*	Himpunan dari objek–objek yang terbagi atribut serta operasi yang sama.
4.		*Collaboration*	Deskripsi dari urutan aksi-aksi yang ditampilkan sistem yang menghasilkan suatu hasil yang terukur bagi suatu aktor.
5.		*Realization*	Operasi yang benar-benar dilakukan oleh suatu objek.
6.		*Dependency*	Hubungan yang terjadi pada suatu elemen mandiri (*independent*) akan mempengaruhi elemen yang bergantung pada elemen yan tidak mandiri.
7.		*Association*	Untuk menghubungkan objek satu dengan objek yang lainnya.

2.4.2.3. *Sequence Diagram*

Sequence diagram adalah penjabaran *behavior* sebuah skenario tunggal. *Sequence diagram* menunjukkan sejumlah objek contoh dan pesan-pesan yang melewati objek-objek ini di dalam *use case* (Martin Fowler, 2005:81). *Sequence diagram* digunakan untuk menggambarkan perilaku pada sebuah skenario.

Simbol-simbol yang digunakan pada *sequence diagram* disajikan pada Tabel 2.3.

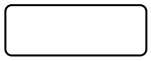
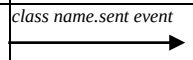
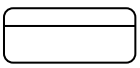
Tabel 2.3. Simbol *Sequence Diagram*

NO	GAMBAR	NAMA	KETERANGAN
1.		*LifeLine*	Objek *entity*, antarmuka yang saling berinteraksi.
2.		*Message*	*Message* ditampilkan sebagai anak panah dari lifeline dari satu objek ke objek yang lain.
3.		*Return*	Spesifikasi dari komunikasi antar objek yang memuat informasi-informasi tentang aktifitas yang terjadi.

2.4.2.4.*Statechart Diagram*

Statechart Diagram adalah teknik yang umum digunakan untuk menggambarkan behavior sebuah sistem (Martin Fowler, 2005:151), *Statechart Diagram* menggambarkan semua *state* atau kondisi yang dimiliki oleh suatu *object* dari suatu *class* dan kejadian yang menyebabkan *state* berubah. Simbol-simbol yang digunakan pada *statechart diagram* disajikan pada Tabel 2.4.

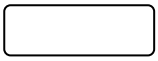
Tabel 2.4. Simbol *Statechart Diagram*

NO	GAMBAR	NAMA	KETERANGAN
1.		*State*	Meliputi seluruh pesan dari *object* yang dapat mengirim dan menerima.
2.		*Start state*	Masing-masing diagram harus mempunyai satu dan hanya satu *start state*
3.		*Stop State*	Sebuah *object* boleh mempunyai banyak *stop state*
5.		*State Transition*	Sebuah kejadian yang memicu sebuah *stateobject* dengan cara memperbarui satu atau lebih nilai atributnya.
6.		*State Detail*	*Action-action* yang mengiringi seluruh *state transition* ke sebuah *state*.

2.4.2.5. *Activity Diagram*

Activity Diagram adalah teknik untuk menggambarkan logika prosedural, proses bisnis, dan jalur kerja, dalam beberapa hal, diagram ini memainkan peran mirip sebuah diagram alir, tetapi perbedaan prinsip antara diagram ini dan notasi diagram alir adalah diagram ini mendukung *behavior* paralel (Martin Fowler, 2005:163). Simbol-simbol yang digunakan pada *use activity diagram* disajikan pada Tabel 2.5.

Tabel 2.5. Simbol *Activity Diagram*

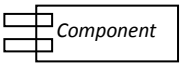
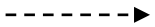
NO	GAMBAR	NAMA	KETERANGAN
1.		*Actifity State*	Aktivitas yang mewakili pelaksanaan dalam pernyataan dalam prosedur atau pelaksanaan kegiatan dalam alur kerja.
2.		*Branch/Merge*	Asosiasi percabangan dimana jika ada pilihan aktivitas lebih dari satu.
3.		*Initial State*	Status awal aktivitas sistem, sebuah diagram aktivitas memiliki sebuah status awal.
4.		*Final State*	Status akhir yang dilakukan sistem.
5.		*Fork/Join*	Sebuah control yang ditampilkan dengan cara yang sama seperti statechart, oleh beberapa panah masuk atau meninggalkan baris *heavy synchronization*

2.4.2.6.*Component Diagram*

Component merupakan bagian fisik dari sebuah sistem, karena menetap di komputer tidak berada di analis. *Component* terhubung melalui antarmuka yang digunakan dan dibutuhkan (Martin Fowler, 2005:189). *Component* merupakan implementasi *software* dari sebuah atau lebih *class*. *Component* dapat

berupa *source code*, komponen biner, atau *executable component*. Sebuah komponen berisi informasi tentang *logic class* atau *class* yang diimplementasikan sehingga membuat pemetaan dari *logical view* ke *component view*, sehingga *component diagram* merepresentasikan dunia riil yaitu *component software* yang mengandung *component*, *interface* dan *relationship*. Simbol-simbol yang digunakan pada component *diagram* disajikan pada Tabel 2.6.

Tabel 2.6. Simbol *Component Diagram*

NO	GAMBAR	NAMA	KETERANGAN
1.		*Component*	Sebuah komponen melambangkan sebuah entitas *software* dalam sebuah sistem. Sebuah komponen dinotasikan sebagai sebuah kotak segiempat dengan dua kotak kecil tambahan yang menempel di sebelah kirinya.
2.		*Dependency*	Sebuah *Dependency* digunakan untuk menotasikan relasi antara dua komponen.

2.4.2.7.*Deployment Diagram*

Deployment Diagram menujukkan susunan fisik sebuah sistem, menunjukkan bagian perangkat lunak mana yang berjalan pada perangkat keras mana Martin Fowler, (2005:137). *Deployment Diagram* juga menggambarkan tata letak sebuah sistem secara fisik, menampakkan bagian-bagian *software* yang berjalan pada bagian-bagian *hardware*, menunjukkan hubungan komputer dengan perangkat (*nodes*) satu sama lain dan jenis hubungannya.

2.5. Pengujian *Five View*

Menurut Kshirasagar dan Priyardarshi (2008:519), pengujian *fiveview* adalah pengujian yang sifatnya deskriptif dimana *software* yang diuji dinilai melalui lima sudut pandang atau kategori yang berbeda melalui penilaian dari

expertise atau orang yang berpengalaman dari masing-masing sudut pandang. Lima sudut pandang tersebut adalah sebagai berikut:

2.5.1. *Transcendental View*

Dalam *Transcendental view* kualitas merupakan sesuatu yang dapat diakui melalui pengalaman tetapi tidak digambarkan dalam beberapa bentuk pengelolaan. Kualitas dilihat sebagai sesuatu yang ideal, dimana itu terlalu kompleks untuk didefinisikan secara tepat. Namun kualitas objek yang bagus akan menonjol, dan itu sangat mudah diakui, karena sifat filosofis dari *Trancendential View* tidak ada ketentuan yang dibuat untuk mengekspresikan itu menggunakan ukuran yang konkret.

2.5.2. *User View*

User view fokus pada sejauh mana sebuah produk memenuhi kebutuhan dan ekspektasi pengguna. Kualitas tidak hanya dilihat dalam hal bagaimana sebuah produk dapat tersampaikan, tetapi itu juga dipengaruhi oleh ketentuan layanan dalam kontrak penjualan. Dalam pandangan ini, pengguna fokus dengan layak atau tidaknya sebuah produk untuk digunakan.

2.5.3. *Manufacturing View*

Manufacturing view berasal dari sektor produksi seperti sektor mobil dan elektronik. Dalam pandangan ini kualitas dilihat sebagai kesesuaian untuk pesyaratan. Setiap penyimpangan dari persyaratan tercantum dilihat sebagai kekurangan kualitas dari produk. Konsep dari proses memainkan peran kunci kedalam dalam *Manufacturing view*. Produk yang dibuat harus orisinil sehingga biaya berkurang, misal biaya pembangunan dan biaya pemeliharaan.

Kesesuaian dengan persyaratan dan spesifikasi menyebabkan keseragaman dalam produk tapi beberapa berpendapat bahwa keseragaman tersebut tidak menjamin kualitas. Kualitas produk dapat secara bertahap ditingkatkan dengan memperbaiki proses.

2.5.4. *Product view*

Jika sebuah produk diproduksi dengansifat internal (misalnya bahan dan tindakan) yang baik, maka produk akan memiliki sifat eksternal atau *output* yang baik dan dapat dieksplorasi hubungan antara sifat internal dan kualitas eksternal.

Dalam pandangan ini, tingkat kualitas saat ini dari sebuah produk menunjukkan ada atau tidaknya sifat produk yang terukur.

2.5.5. *Value-Based View*

Value-based view merupakan penggabungan dari dua konsep yaitu keunggulan dan kelayakan. Kualitas adalah ukuran dari keunggulan dan nilai adalah ukuran layak. Berapa banyak pengguna bersedia membayar untuk tingkat kualitas tertentu. Kualitas tidak berarti jika produk memenuhi nilai ekonomi. Pandangan berbasis nilai antara biaya dan kualitas.