BAB 2
TINJAUAN PUSTAKA

2.1. Aplikasi UNIKOM *Apps*

UNIKOM *Apps* merupakan aplikasi yang dibuat oleh salah satu divisi yang ada di kampus Universitas Komputer Indonesia yaitu Divisi CodeLabs, yang dimana divisi ini berfokus dalam bidang pembuatan perangkat lunak *(software)*. Selain menyediakan informasi diatas, aplikasi ini memiliki fitur *"Student Dashboard"* yaitu sebuah berita informasi khusus untuk mahasiswa yang masih aktif untuk melihat informasi data pribadi tentang kartu hasil studi (KHS), kartu rencana studi (KRS), status aktif, transkip nilai, absensi, dan keuangan. Lalu aplikasi UNIKOM *Apps* ini juga sudah dilengkapi dengan fitur *"Tour 360"* yang dapat digunakan untuk melihat lingkungan kampus UNIKOM dengan teknologi 360 *view* [4].

2.2. Definisi Perangkat Lunak

Menurut definisi IEEE *(Insitute of Electrical Engineers)*, perangkat lunak memiliki 4 komponen yaitu program komputer, prosedur, dan dokumentasi yang menyertai serta data yang digunakan untuk mengoprasikan sistem komputer. Definisi IEEE tersebut sangat mirip dengan definisi dari ISO (ISO, 1997, Sec. 3.11 dan ISO/IEC 9000-3 Sec. 3.14) [5] tentang perangkat lunak yakni:

1. Program computer *(code).*
2. Prosedur.
3. Dokumentasi.
4. Data yang diperlukan untuk pengoperasian sistem perangkat lunak.

2.3. Kualitas Perangkat Lunak

Menurut definisi dalam Steve McConnell's *Code Complete* membagi perangkat lunak ke dalam dua hal yaitu: internal dan *external quality characteristics*.

Karakteristik kualitas eksternal merupakan bagian-bagian dari suatu produk yang berhubungan dengan para pemakainya, sedangkan karakteristik kualitas internal tidak secara langsung berhubungan dengan pemakai. *Software Quality* didefinisikan sebagai: kesesuaian yang diharapkan pada semua *software* yang dibangun dalam hal fungsi *software* yang diutamakan dan unjuk kerja *software*, standar pembangunan *software* yang terdokumentasi dan karakteristik yang ditunjukkan oleh *software* [6]. Definisi ini menekankan pada 3 hal yaitu:

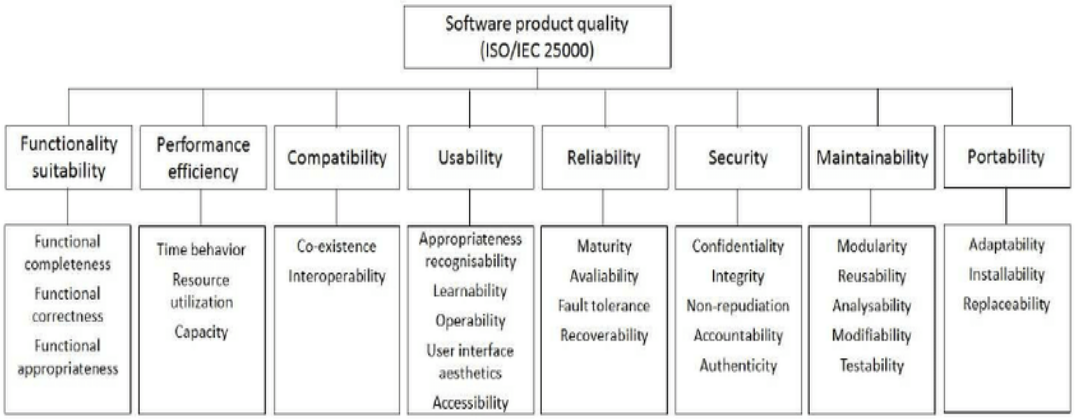
1. Kebutuhan *software* adalah fondasi ukuran kualitas *software*, jika *software* tidak sesuai dengan kebutuhan yang ditentukan maka kualitas pun akan kurang.
2. Jika menggunakan suatu standar untuk pembangunan *software* maka jika *software* tidak memenuhi standar tersebut maka dianggap kurang berkualitas.
3. Seringkali ada kualitas yang secara langsung diutarakan (tersirat) seperti kemudahan penggunaan dan pemeliharaan yang baik. Kualitas *software* dipertanyakan jika tidak memenuhi kebutuhan ini.

2.4. Standar Kualitas Perangkat Lunak ISO 25010

Menurut David (2011), pengujian untuk *mobile application* meliputi empat faktor yaitu *functional testing*, *compatibility testing*, *usability testing*, dan *performance testing*. Jika standar kualitas perangkat lunak *mobile* David dibandingkan dengan standar ISO 25010, maka pengujian sebuah aplikasi perlu dilakukan pada karakteristik *functional suitability*, *compatibility*, *usability*, dan *performance efficiency* [7]. Model ISO 25010:2011 merupakan bagian dari *Systems and software Quality Requirements and Evaluation (SQuaRE)* menggantikan ISO 91261:2001, yang telah direvisi secara teknis. Terdiri dari delapan karakteristik dan dibagi lagi menjadi sub karakteristik yang berhubungan dengan sifat-sifat statis perangkat lunak dan sifat dinamis dari sistem komputer. Kualitas yang digunakan adalah sejauh mana suatu produk atau sistem dapat digunakan oleh pengguna untuk memenuhi kebutuhan mereka dalam mencapai tujuan

tertentu dengan efektivitas, efisiensi, bebas dari resiko dan kepuasan dalam konteks penggunaan yang spesifik[3].

Standar yang digunakan untuk menganalisis kualitas perangkat lunak aplikasi ini adalah menggunakan Standar ISO 25010. ISO 25010 merupakan suatu metode standar internasional untuk digunakan dalam menilai dan menganalisis kualitas sebuah perangkat lunak yang dirilis oleh ISO/IEC. Standar ISO 25010 dapat digunakan untuk mengevaluasi kualitas produk perangkat lunak. Gambaran dari model *software products quality* dari standar ISO 25010 sebagai berikut [8]:



Gambar 2.1 Konsep ISO/IEC 25010

Dalam penelitian ini, pengujian aplikasi dilakukan sesuai standar ISO yang terdiri dari delapan aspek kualitas. Namun, peneliti hanya mengambil empat aspek kualitas menurut David (2011) bahwa pengujian untuk *mobile application* meliputi empat aspek yaitu *functional testing, compatibility testing, usability testing,*dan *performance testing* [9].

1. *Functional Testing*

   Merupakan metode pengujian tradisional yang digunakan untuk validasi fungsi aplikasi/web sesuai dengan syarat yang dibutuhkan, dalam standar ISO 25010 mewakili aspek *functional suitability*.

2. *Compability Testing*

   Pengujian aplikasi/web menggunakan berbagai macam variasi browser, OS, jenis *device*, ukuran *device*, dan kecepatan koneksi. Dalam standar ISO 25010, *compabilty testing* mewakili aspek *compability*.

3. *Usability Testing*

   Pengujian *usability* digunakan untuk menguji kepada pengguna akhir mengenai penggunaan aplikasi, dalam standar ISO 25010 mewakili aspek *usability*.

4. *Performance Testing*

   Merupakan pengujian yang melakukan pengujian mengenai penggunaan memori/CPU, konsumsi baterai, dan mengambil data dalam server dibawah kondisi tertentu. Dalam standar ISO 25010, *performance testing* mewakili aspek *performance efficiency*.

Kemudian menurut *Computer Science Corporation* (CSC) untuk pengujian *mobile application* meliputi *functional and localization testing, usability testing, platform compability and installation testing, and non-functional testing (performance and security testing)* [10]. Hal ini yang mendasari peneliti hanya mengambil empat aspek yang akan diuji.

2.4.1. Faktor Kualitas *Functional Suitability*

Functional Suitability merupakan faktor kualitas yang menunjukkan program telah mampu menjalankan fungsi sesuai dengan rancangan yang telah dikembangkan. *Functional Suitablity* dapat memastikan bahwa program telah berjalan sesuai dengan scenario dan menghasilkan keluaran yang sesuai sebagai raksi dari suatu aksi tertentu. Pengujian aspek *functional suitability* dilakukan dengan mengisi *test case* oleh responden ahli dengan kriteria responden memiliki pekerjaan sehari-hari sebagai pengembang aplikasi *mobile*.

2.4.2. Faktor Kualitaas *Compability*

Aspek *compability* merupakan salah satu aspek penting bagi pengembang untuk dapat mengembangkan produk sehingga dapat *support* ke *platform* lain (Wagner, 2013). Pengembangan aplikasi UNIKOM *Apps* ini dilakukan pada *platform android* yang sampai saat ini telah memiliki berbagai versi perangkat lunak yang terus dikembangkan. Oleh karena itu suatu aplikasi *android* haruslah dapat dijalankan pada beberapa versi *android* tersebut.

Untuk menguji aplikasi UNIKOM *Apps* menggunakan aspek kualitas *compatibility* menggunakan perangkat *compability testing* yang disediakan oleh *Google, cloud testing* dari *test droid,* dan beberapa perangkat *smartphone android* dari berbagai variasi sistem operasi dan ukuran layar kemudian dianalisis menggunakan analisis deskriptif. *Compability testing* yang disediakan oleh *Google* merupakan *tools* yang sangat berguna bagi para pengembang aplikasi *android* yang tersedia di *Google Play Developer Console,* melalui aplikasi ini pengembang dapat mengetahui seberapa banyak perangkat yang dapat mendukung aplikasi yang dikembangkan (*Google,* 2014).

2.4.3. Faktor Kualitas *Usability*

Seperti yang telah dijelaskan oleh David (2011), aspek *usability* digunakan untuk menguji kepada pengguna mengenai penggunaan aplikasi. Aspek ini berkaitan dengan sejauh mana sistem dapat digunakan oleh pengguna tertentu untuk memperoleh tujuan tertentu dengan efektif, efisien, dan memuaskan. Lebih lanjut menururt Gao & Tsai (2013) menjelaskan bahwa dalam pengujian *usability* dalam aplikasi *mobile* melibatkan pengujian *gesture, interface* konten, dan *user experience* seperti bagaimana pengguna berinteraksi dengan kamera, GPS, atau sensor sidik jari.

Pengujian aspek *usability* dalam penelitian ini menggunakan *USE Questionnaire* oleh Arnold M. Lund (2001). Instrumen *USE Questionnaire* telah digunakan dalam

berbagai penelitian salah satunya penelitian yang dilakukan oleh Rahadi (2014) dalam bidang teknologi aplikasi *mobile* sehingga teruji dalam validasinya.

2.4.4. Faktor Kualitas *Performance Efficiency*

Faktor kualitas *performance* ini meliputi tentang pengujian untuk hasil dari rata-rata perhitungan yang telah didapat terhadap instrumen yang diujikan. Dengan melihat sejauh mana karakteristik kinerja pada sebuah aplikasi terhadap sumber daya yang digunakan dalam kondisi tertentu. Karakteristik ini terbagi menjadi 2 sub karakteristik ditunjukan pada tabel berikut.

Tabel 2.1 Sub Karakteristik Aspek *Performance Efficiency*

Sub Karakteristik	Deskripsi
Time behavior	Karakteristik untuk melihat sejauh mana respon dan pengelolaan waktu produk atau sistem dapat memenuhi persyaratan ketika menjalankan suatu fungsi.
Resourceutilization	Karakteristik ini untuk mengetahui sejauh mana jumlah dan jenis dari sumber daya yang digunakan oleh suatu produk.
Capacity	Karakteristik untuk sejauh mana batas maksimum parameter dari produk atau sistem yang dapat memenuhi suatu persyaratan.

Ben David (2011) menjelaskan bahwa *performance testing* untuk mengujian pada aplikasi berbasis *mobile* adalah jenis pengujian yang menilai penggunan memori/CPU, konsumsi baterai, dan beban pada server dalam berbagai kondisi. Hal ini menentukan jenis performa apa yang diharapkan dari beban tersebut, dan tes kecepatan respon aplikasi dalam kondisi yang berbeda (kecepatan wifi, dan koneksinya).

Sebagai alat untuk pengujiannya, maka akan menggunakan *cloud testing automation tools* dari *Amazon Web Service* (AWS). Dan hasilnya dari pengujian aspek

performance efficiency akan diambil hasil yang didapatkan dari pengujian menggunakan *tools* tersebut.

2.5. Domain Perangkat Lunak

Domain perangkat lunak merupakan kategori dari setiap jenis perangkat lunak yang ada. Terdapat tujuh kategori mengenai jenis domain perangkat lunak ini sendiri, diantaranya [6]:

1. *System Software*

   System software merupakan sebuah program yang dibuat untuk mendukung program lain untuk dapat digunakan. Perangkat lunak jenis ini misalnya *compilers, editor, file management, operating system, telecommunications processors,* dan lain-lain.
2. *Application Software*

   Application software adalah sebuah program yang berdiri sendiri dan digunakan untuk mengatasi kebutuhan bisnis yang spesifik.
3. *Engineering/scientific software*

   Perangkat lunak pada domain ini biasanya ditekankan pada penggunaan algoritma. Penggunaan perangkat lunak ini terdapat pada *kebutuhan seperti astronomi, vulkanologi, pabrik, biologi, dan lain* sebagainya.
4. *Embedded software*

   Embedded software merupakan perangkat lunak yang ditanam pada suatu sistem. Perangkat lunak ini digunakan dalam mengatur fungsi untuk pengguna maupun untuk dirinya sendiri.
5. *Product-line software*

   Perangkat lunak pada domain *product-time software* dibuat untuk membantu kebutuhan pengguna yang bersifat spesifik yang dapat

digunakan oleh pengguna yang berbeda. Contoh dari perangkat lunak pada domain *product-line software* diantaranya untuk keperluan *word processing*, *multimedia*, *computer graphic*, *database management, entertainment*, dan lain sebagainya.

6. *Web application*

   Web application atau biasa disebut *webapps* adalah perangkat lunak yang berbasis *website*. Pada perangkat lunak ini bukan hanya sekedar menampilkan informasi berbentuk *teks* namun dapat juga berupa gambar.

7. *Artificial intelligence software*

   Perangkat lunak pada domain ini ditekankan pada algoritma untuk dapat menyediakan suatu masalah yang kompleks, yang tidak bisa diselesaikan dengan perhitungan ataupun analisis langsung. Perangkat lunak ini seperti untuk pengenalan pola, jaringan syaraf tiruan, robotic, dan lain-lain.

2.6. Definisi Smartphone

Menurut Williams dan Sawyer definisi *smartphone* adalah telepon selular yang memakai beberapa layanan seperti layar, mikroprosesor, memori, dan modem bawaan. Dengan begitu, *smartphone* memiliki fitur yang lebih lengkap disbanding *handphone* biasa.

Seperti halnya pada komputer atau laptop, sebuah *smartphone* membutuhkan *Operating System* (OS) agar bisa bekerja sebagaimana mestinya. Berikut ini adlah beberapa OS *smartphone*:

1. iOS
2. Android
3. Windows Phone
4. Blackberry
5. Bada
6. Firefox OS
7. MeeGo OS

8. Palm
9. Symbian
10. Ubuntu
11. Tizan

Dari sekian banyak OS *smartphone* yang digunakan, OS yang paling popular adalah Android, iOS, dan Windows Phone. Namun, secara keseluruhan *smartphone Android* lah yang paling banyak digunakan.

2.7. Definisi Mobile Application

Menurut Buyens (2001) *mobile application* berasal dari kata *application* dan *mobile*. *Application* yang artinya penerapan, lamaran, penggunaan. Secara istilah aplikasi adalah program siap pakai yang direka untuk melaksanakan suatu fungsi bagi pengguna atau aplikasi yang lain dan dapat digunakan oleh sasaran yang dituju sedamgkan *mobile* dapat diartikan sebagai perpindahan dari suatu tempat ke tempat yang lain.

Kata *mobile* mempunyai arti bergerak atau berpindah, sehingga aplikasi *mobile* menurut Rangsang Purnama (2010) adalah sebutan untuk aplikasi yang berjalan di *mobile device*. Dengan menggunakan aplikasi *mobile*, dapat dengan mudah melakukan berbagai macam aktifitas mulai dari hiburan, berjualan, belajar, mengerjakan pekerjaan kantor, *browsing* dan lain sebagainya.

Pemanfaatan aplikasi *mobile* untuk hiburan yang paling banyak digemari oleh pengguna telepon selular, karena dengan memanfaatkan adanya fitur *game*, *music player*, sampai *video player* membuat kita menjadi semakin mudah menikmati huburan kapan saja dan dimanapun.

Perangkat *mobile* memiliki banyak jenis dalam hal ukuran, desain *layout*, tetapi mereka memiliki kesamaan karakteristik yang sangat berbeda dari desktop *system*. Perangkat *mobile* memiliki memori yang kecil dibanding desktop [7].

2.8. Pemrograman Berorientasi Objek

Objek adalah kesatuan entitas yang memiliki sifat dan tingkah laku. Dalam kehidupan sehari-hari, objek adalah benda, baik benda berwujud nyata seperti manusia, hewan, mobil, komputer, handphone, pena, ataupun benda yang tidak nyata atau konsep, seperti halnya tabungan bank, sistem antrian, sistem internet banking, dan sebagainya. Jadi pengertian OOP adalah konsep yang membagi program menjadi objek-objek yang saling berinteraksi satu sama lain. Objek adalah benda, baik benda yang berwujud nyata maupun benda yang tidak nyata (konsep). Jika menggunakan OOP maka akan ada enam keuntungan yang dapat diperoleh, yaitu [8]:

1. Alami *(Natural).*
2. Dapat diandalkan *(Reliable).*
3. Dapat digunakan kembali *(Reusable).*
4. Mudah untuk dalam perawatan *(Maintainable).*
5. Dapat diperluas *(Extendable).*
6. Efisiensi waktu.

Berikut ini beberapa bahasa pemrograman yang sudah menggunakan konsep OOP, sebagai berikut:

1. *C++.*
2. *C#.*
3. *Visual Basic.*
4. *Java.*
5. *PHP.*

2.8.1. Analisis dan Desain Berorientasi Objek *(Object-Oriented Analysis and Design Process)*

Menurut Al Fatta (2007:38), "OOAD adalah metode pengembangan sistem yang lebih menekankan objek dibandingkan dengan data atau proses. Ada beberapa ciri khas dari pendekatan ini, yaitu *object*, *inheritance*, dan *object class* menurut Al Fatta (2007:38) [11] sebagai berikut:

1. *Object* adalah struktur yang mengenkapsulasi atribut dan metode yang beroperasi berdasarkan atribut-atribut. *Object* adalah abstraksi dari benda nyata dimana data dan proses diletakkan bersama untuk memodelkan struktur dan perilaku dari objek dunia nyata.
2. *Object class* adalah sekumpulan objek yang berbagai struktur yang sama dan perilakuka yang sama.
3. *Inheritance* merupakan properti yang muncul ketika tipe entitas atau *object class* disusun secara hierarki dan setiap tipe entitas atau *object class* menerima atau mewarisi atribut dan metode dari pendahulunya.

2.8.2. Orientasi Objek *(Object)*

Orientasi objek merupakan teknik dalam menyelesaikan masalah yang kerap muncul dalam pengembangan perangkat lunak. Teknik ini merupakan titik kulminasi dalam menemukan cara yang efektif dalam membangun sistem dan menjadi metode yang paling banyak dipakai oleh para pengembang perangkat lunak saat ini. Orientasi objek merupakan teknik pemodelan sistem riil yang berbasis objek. Inti dari konsep ini adalah objek yang merupakan model dari sistem nyata.

Objek adalah entitas yang memiliki atribut, karakter dan kadangkala disertai kondisi. Objek merepresentasikan sesuatu sistem nyata seperti siswa, sistem kontrol permukaan sayap pesawat, sensor atau mesin. Objek juga merepresentasikan sesuatu

dalam bentuk konsep seperti nasabah bank, merek dagang, pernikahan atau sekedar *listing*. Bahkan bisa juga mengatakan visualisasi seperti, bentuk huruf, histogram, poligon, garis atau lingkaran. Semuanya memiliki fitur atribut (untuk data), *behavior (operation* atau *method)*, keadaan (memori), identitas dan tanggung jawab. Proses menjabarkan sistem nyata menjadi objek dinamakan abstraksi *(abstraction)* [12]. Abstraksi mengeliminir aspek yang tidak perlu dalam suatu objek

2.8.3. Kelas *(Class)*

Kelas adalah penggambaran satu set objek yang memiliki atribut dan *behaviour* yang sama. Kelas mirip tipe data pada pemrograman non objek, tapi lebih komprehensif karena terdapat struktur sekaligus karakteristiknya. Programmer dapat membentuk kelas baru yang lebih spesifik dari kelas *general*-nya. Kelas dan objek merupakan jantung dari pemrograman berorientasi objek untuk menghasilkan program jenis ini sangat penting untuk selalu berfikir dalam bentuk objek.

2.8.4. Pembungkusan *(Encapsulation)*

Pembungkusan sebagai penggabungan potongan-potongan informasi dan perilaku-perilaku spesifik yang bekerja pada informasi tersebut, kemudian mengemasnya menjadi apa yang disebut sebagai objek. Dalam perbankan dikenal objek rekening yang memiliki perilaku-perilaku misalnya buka, tutup, penarikan, penyimpanan, ubah nama, ubah alamat, dan sebagainya. Akibatnya, perubahan-perubahan pada sistem perbankan yang berkaitan dengan rekening-rekening dapat secara sederhana diimplementasikan satu kali saja pada objek rekening. Keuntungan lainnya adalah membatasi efek-efek perubahan pada sistem. Misalnya, saat manajemen bank menentukan jika seseorang memiliki rekening pinjaman di bank yang bersangkutan, rekening pinjaman itu harus dapat juga digunakan sebagai sarana bagi penarikan rekening.

2.8.5. Pewarisan *(Inheritance)*

Konsep dimana metode dan atau atribut yang ditentukan di dalam sebuah objek kelas dapat diwariskan atau digunakan lagi atau digunakan lagi oleh objek kelas lainnya. Sedangkan generalisasi/spesialisasi merupakan teknik dimana atribut dan perilaku yang umum pada beberapa tipe kelas objek, dikelompokkan (atau diabstraksi) ke dalam kelasnya sendiri (dinamakan *supertype*). Atribut dan metode kelas objek *supertype* kemudian diwariskan oleh kelas objek tersebut (dinamakan *subtype*).

2.9. UML *(Unified Modeling Language)*

Menurut Sukanto dan Shalahuddin (2015:137), "UML merupakan bahasa *visual* untuk pemodelan dan komunikasi mengenai sebuah sistem dengan menggunakan diagram dan teks-teks pendukung, UML hanya berfungsi untuk melakukan pemodelan".

Menurut Mulyani (2016:48), "UML adalah sebuah teknik pengembangan sistem yang menggunakan bahasa grafis sebagai alat untuk pendokumentasian dan melakukan spesifikasi pada sistem". Sedangkan menurut Nugroho (2011:119), mengemukakan bahwa "UML adalah bahasa untuk menspesifikasi, menvisualisasikan, serta mengonstruksi bangunan dasar sistem perangkat lunak, termasuk melibatkan pemodelan aturan-aturan bisnis".

Adapun beberapa diagram yang termasuk dalam UML menurut Sukanto dan Shalahuddin (2015:155). Adalah sebagai berikut:

1. *Use Case Diagram*

   Menurut Sukamto dan Shalahuddin (2015:155) mengemukakan bahwa "*Use case* atau *diagram use case* merupakan pemodelan untuk kelakukan *(behavior)* sistem informasi yang akan dibuat. *Use case* mendeskripsikan

sebuah interaksi antara satu atau lebih aktor dengan sistem informasi yang dibuat".

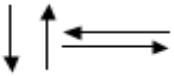
Menurut Al Fatta (2007:91), "*Use case* adalah metode berbasis teks untuk menggambarkan dan mendokumentasikan proses yang kompleks".

Menurut Mulyani (2016:49), "*Use case* diagram, yaitu diagram yang digunakan untuk menggambarkan hubungan antara sistem dengan aktor". Sedangkan menurut John Satzinger dalam Triandini dan Suardika (2012:17), "*Use case* adalah sebuah kegiatan yang dilakukan oleh sistem, biasanya dalam menanggapi permintaan dari pengguna sistem".

2. *Activity Diagram*

Menurut Sukamto dan Shalahuddin (2015:161), "Diagram aktivitas atau *activity diagram* menggambarkan *workflow* (aliran kerja) atau aktivitas dari sebuah sistem atau proses bisnis atau menu yang ada pada perangkat lunak".

Didalam *activity diagram* terdapat juga beberapa simbol. Berikut ini adalah simbol-simbol yang ada pada diagram aktivitas yaitu:

NO	GAMBAR	NAMA	KETERANGAN
1		*Activity*	Memperlihatkan bagaimana masing-masing kelas antarmuka saling berinteraksi satu sama lain
2		*Action*	State dari sistem yang mencerminkan eksekusi dari suatu aksi
3		*Initial Node*	Bagaimana objek dibentuk atau diawali.
4		*Activity Final Node*	Bagaimana objek dibentuk dan diakhiri
5		*Decision*	Diguanakan untuk menggambarkan suatu keputusan / tindakan yang harus diambil pada kondisi tertentu
6		*Line Connector*	Digunakan untuk menghubungkan satu simbol dengan simbol lainnya

Gambar 2.2 Activity Diagram

2.10. Orientasi Objek Metriks

Menurut Chidamber dan Kemerer Metric untuk mengukur dan memberikaan penilaian perangkat lunak berbasisi *object oriented* dapat menggunakan enam buah metrik [8], yaitu:

1. *Weight Methods per Class* (WMC)

   Pengukuran metrics WMC adalah menghitung jumlah *method* yang diimplementasikan dalam suatu *class*. Untuk mengetahuinya maka dihitung seluruh *method* lokal pada sebuah perangkat lunak.

   Rumus: m1 + m2 + m3 + ….. + Mh

2. *Depth of Inherientance Tree* (DIT)

   DIT merupakan perhitungan jalur *inherientance* maksimum dari sebuah *class*. Kedalaman sebuah *class* dalam hirarki pewarisan diukur oleh DIT.

   Rumus: Apabila ada pewarisan diberikan nilai 1 dan apabila tidak ada diberikan nilai 0.

3. *Nummber of Children* (NOC)

   Pengukuran NOC adalah menghitung jumlah *children* atau anak kelas.

   Rumus: Apabila ada anak diberik nilai dan apabila tidak ada, maka tidak diberi nilai.

4. *Coupling Between Object* (CBO)

   Coupling adalah ketergantungan modul satu dengan modul lainnya, *coupling* yang baik adalah *coupling* yang rendah dimana modul satu dengan modul lainnya tidak saling ketergantungan.

   Rumus: Menghitung jumlah *method* lokal yang mengakses ke *method eksternal*.

5. *Respon for a Class* (RFC)

   Pengukuran RFC yaitu menghitung jumlah *method* lokal yang diimplementasikan ditambah *method* yang dipanggil dalam objek.

   Rumus: jumlah *method* lokal + jumlah *method eksternal*.

6. *Lack of Cahesion in Method* (LCOM)

   LCOM adalah metrik untuk menghitung sebuah *Cohesi*. *Cohesi* adalah fungsi yang terkait dalam sebuah modul. *Cohesi* yang baik adalah *cohesi* yang tinggi, semakin tinggi *cohesi* semakin baik perancangannya. Pengukuran LCOM dilakukan dengan menghitung jumlah koneksi pada *method*.

   Rumus perhitungan adalah: LCOM * = (m – sum (Mu) / a) / (m – 1)

   Dimana:

   M adalah jumlah *method* dalam *class*.

   A adalah jumlah *variable* dalam *class*.

   Mu adalah jumlah *variable* yang dipanggil.