BAB II
LANDASAN TEORI

2.1. Penelitian Terdahulu

Penelitian ini dilakukan tidak terlepas dari hasil penelitian- penelitian terdahulu yang pernah dilakukan, sebagai bahan perbandingan dan kajian. Adapun penelitian yang dijadikan tidak terlepas dari topik penelitian yaitu mengenai deteksi, dan manajemen amarah.

Rozie (2016) dalam penelitiannya mengenai rancang bangun alat pengukur detak jantung berbasis Android dengan media Arduino. Dimana dilakukan penelitian untuk merancang sebuah *hardware* dan *software* yang dapat mengukur detak jantung dengan *micro-controller* Arduino yang akan menerima data detak jantung dan mengirimkan data detak jantung tersebut ke smartphone untuk ditampilkan nilai detak jantung (BPM) seseorang. **(Rozie, 2016)**

Penelitian yang sekarang sedang dilakukan mengadopsi tujuan dan sistem yang telah dibuat termasuk cara pengiriman data detak jantung ke Android dengan media *bluetooth*, namun dengan perbedaan, dimana sebelumnya pengukuran detak jantung dilakukan dengan *hardware* buatan sendiri berbasis *microcontroller*, yang hasilnya juga akan ditampilkan pada Android, pada penelitian ini *hardware* deteksi menggunakan *smartwatch* yang dapat melakukan pengukuran detak jantung secara *realtime*, dan hasil dikirimkan ke aplikasi sehingga lebih memudahkan pengguna untuk menggunakan aplikasi yang telah dirancang dari

penelitian ini dalam kehidupan sehari- hari, karena *smartwatch* sendiri telah dirancang untuk nyaman digunakan dalam aktivitas pengguna. **(Park, 2018)**

Ifan Kurnia Afandi (2014) dalam penelitiannya mengenai pengaruh musik dan terhadap detak jantung manusia. Dimana dilakukan penelitian untuk membuktikan pengaruh musik terhadap emosi manusia. Penelitian ini juga melakukan *review* dan pembuktian kepada penelitian serupa terdahulu. Dari Penelitian ini mendapatkan kesimpulan bahwa musik dengan tempo yang lambat cenderung menenteramkan dan memperlambat denyut jantung, sementara musik dengan tempo yang cepat cenderung membawa perasaan yang meledak-ledak dan meningkatkan denyut jantung. **(Afandi, 2014)**

Perbandingan dengan penelitian terdahulu dapat dilihat pada tabel dibawah ini:

Tabel 2.1 Tabel Perbandingan Hasil Penelitian

Penulis/ Komponen Penelitian	Penulis	
	Afandi 2014	Rozie 2016
Nama Jurnal	E-Journal Undip	Jurnal Fakultas Teknik Universitas Tanjungpura
Judul Penelitian	Musik dan Denyut Jantung pada Era Digital	Rancang Bangun Alat Monitoring Jumlah Denyut Nadi Berbasis Android

Tabel 2.1 Tabel Perbandingan Hasil Penelitian (Lanjutan)

Penulis/ Komponen Penelitian	Penulis	
	Afandi 2014	Rozie 2016
Lokasi Penelitian	Universitas Diponegoro, Semarang	Rumah Sakit Universitas Tanjungpura, Pontianak
Metode dan Alat Penelitian	Studi Literatur, Pengumpulan data, observasi	Studi Literatur, Pengumpulan data, observasi, Matlab *image processing, pulse detector circuit, microcontroller*
Kesimpulan Penelitian	Musik dengan tempo yang lambat cenderung memperlambat denyut jantung, sementara musik dengan tempo yang cepat cenderung meningkatkan denyut jantung.	Berhasil dibuat aplikasi mengukur detak jantung dengan tingkat error 2-5 BPM dibandingkan dengan ECG pada rumah sakit.

2.2. *Smartphone*

Smartphone atau yang sering disebut telepon pintar, adalah salah satu jenis telepon selular dengan mikroprosesor, memori, layar dan modem bawaan. *Smartphone* merupakan ponsel multimedia yang menggabungkan fungsionalitas

PC dan *handset* sehingga menghasilkan *gadget* yang mewah, di mana terdapat pesan teks, kamera, pemutar musik, video, *game,* akses *email,* tv digital*, search engine,* pengelola informasi pribadi, fitur GPS, jasa telepon *internet* dan bahkan terdapat telepon yang juga berfungsi sebagai kartu kredit, ataupun penyedia dompet *virtual.* (Sawyer, 2011).

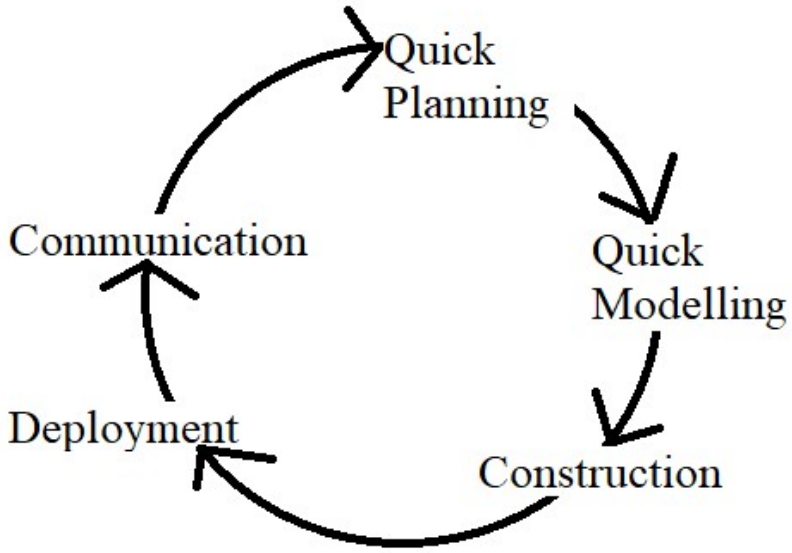
2.3. SDLC

SDLC atau yang lebih dikenal dengan istilah *System Development Life Cycle* adalah metodologi umum yang digunakan untuk mengembangkan sistem informasi. SDLC terdiri dari beberapa fase yang dimulai dari fase perencanaan, analisis, perancangan, implementasi hingga pemeliharaan sistem. Konsep SDLC ini mendasari berbagai jenis model pengembangan perangkat lunak untuk membentuk suatu kerangka kerja untuk perencanaan dan pengendalian pembuatan sistem informasi. Model-model SDLC yang sering digunakan antara lain *Waterfall* dan *Prototyping* (Susanto, 2016).

2.4. SDLC *Prototype*

SDLC model *prototyping* merupakan suatu teknik untuk mengumpulkan informasi tertentu mengenai kebutuhan-kebutuhan informasi pengguna secara cepat. Berfokus pada penyajian dari aspek-aspek perangkat lunak tersebut yang akan nampak bagi pelanggan atau pemakai. Prototipe tersebut akan dievaluasi oleh pelanggan/pemakai dan dipakai untuk menyaring kebutuhan pengembangan perangkat lunak. (Susanto, 2016).

Tahapan model pengembangan *prototyping* digambarkan pada gambar dibawah ini:



Gambar 2.1 Gambaran proses metode *prototyping*

2.5. Kuesioner

Kuisioner merupakan teknik pengumpulan data dengan cara memberi seperangkat pertanyaan atau pernyataan tertulis kepada responden untuk menjawabnya. Jika dilihat dari sisi jawaban responden, maka kuesioner dibedakan menjadi dua jenis :

1. Kuesioner Terbuka, yaitu memberi kesempatan kepada responden untuk menjawab dengan kalimat sendiri.
2. Kuesioner Tertutup, yaitu responden memilih jawaban berdasarkan pilihan yang telah disediakan. (Sugiyono, 2010).

2.6. Detak Jantung

Detak jantung adalah denyutan yang ditimbulkan jantung karena kontraksi ventrikel saat sedang melakukan pemompaan darah ke seluruh tubuh. Detak jantung sendiri dibagi menjadi dua jenis yaitu detak jantung saat istirahat dan detak jantung saat beraktivitas. Normalnya, nilai detak jantung pada setiap orang menurun seiring bertambahnya usia dimana detak jantung istirahat pada dewasa berkisar diantara 60-100 BPM, dan detak jantung aktif berkisar pada 100-200 BPM.

Beberapa kondisi abnormal pada detak jantung seringkali digambarkan sebagai gejala penyakit kronis seperti jantung koroner, angin duduk, dan stroke. Adapun kondisi abnormal seperti detak jantung terlalu lambat (dibawah 60 BPM untuk dewasa), terlalu cepat (diatas 100 BPM saat dalam kondisi istirahat), dan tidak beraturan (delta perubahan detak jantung dapat mencapai 100 BPM). Apabila hal ini detemukan, diharap seseorang secepatnya melakukan pengobatan lebih lanjut karena hal ini dapat menyebabkan kematian. **(Kokkinos, 2010)**

2.7. Sistem Aplikasi

Aplikasi adalah suatu program yang dibuat oleh pemakai yang ditujukan untuk melakukan suatu tugas khusus (Kadir, 2014). Sistem aplikasi sendiri, adalah sekelompok komponen yang saling berhubungan, bekerja bersama untuk mencapai tujuan bersama dengan menerima input serta menghasilkan ouput dalam proses transformasi yang teratur.

Sistem sendiri dapat didefinisikan dengan pendekatan prosedur dan dengan pendekatan komponen. Dengan pendekatan prosedur, sistem dapat didefinisikan

sebagai kumpulan dari prosedur- prosedur yang mempunyai tujuan tertentu. (Satzinger, 2012)

2.8. User Interface

Antarmuka pengguna (*user interface*) merupakan mekanisme yang digunakan oleh pengguna dan program untuk saling berkomunikasi. Aplikasi menerima informasi dari pengguna dan mengubahnya ke dalam bentuk yang dapat diterima oleh sistem. Selain itu, aplikasi menerima informasi yang lalu diproses, dan menyajikan hasil prosesnya ke dalam bentuk yang dapat dimengerti oleh pengguna.

Adapun untuk penyusunan tampilan antarmuka pengguna yang baik, digunakan *Pillars of UI design* sebagai pedoman, yaitu:

1. *User Interface Requirements.*

Mendata dengan jelas kebutuhan pengguna, disesuaikan dengan selera masing- masing.

2. *Guidelines documents and processes.*

Mendeskripsikan aplikasi berdasarkan proses, dan kebutuhannya dimana diantaranya meliputi: tampilan gambar, ikon, dan kata- kata, screen-layout issues, input and output devices, langkah pemakaian, penggunaan oleh user baru.

3. *User-Interface Software Tools*

Penggunaan aplikasi pembuat UI, yang dapat mempengaruhi harga, lama pembuatan, dan kompleksnya sistem.

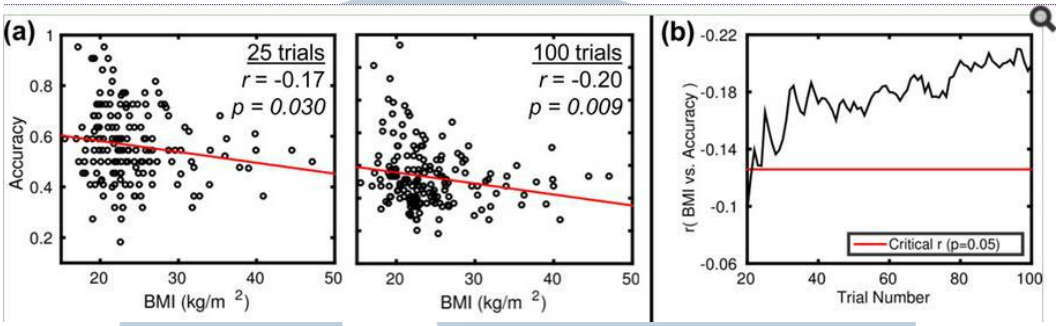
4. *Expert Reviews and Usability Testing.*

Melakukan berbagai macam review dari para ahli, pengguna yang ditarget, dan berbagai macam survei untuk kesempurnaan aplikasi. (Ilham, 2016).

2.9. Heartbeat Detection Task (HBD) Method

Heartbeat Detection Task (HBD) Method adalah sebuah metode yang paling sering digunakan untuk mendeteksi kondisi seseorang melalui detak jantung, dengan mengukur tingkat sensitivitas *interoceptive cardiac* (salah satu bagian jantung) dalam waktu tertentu secara berkala. Dengan metode ini pula, dapat dideteksi kondisi stres, emosional, dan fisik.

Untuk mendeteksi secara mendetail, HBD akan membaca tingkatan detak jantung manusia setiap menit (25-55 detik sekali), dimana ada tingkatan tertentu yang diukur berdasarkan beberapa faktor seperti berat badan, jenis kelamin, dan usia yang dapat mempengaruhi preferensi perhitungan HBD.



Gambar 2.2 Contoh grafik perhitungan skala HBD

Sumber: (Hsieh, 2011)

Dalam gambar yang ada, dapat dijelaskan skala *interoceptive cardiac* yang kurang/ melebihi standar (garis lurus) bagi seseorang dengan berat badan tertentu. Dengan *HBD method* ini, dapat dideteksi pula kondisi kesehatan seseorang dengan akurat. (Hsieh, 2011)

2.10. Android

Android merupakan sistem operasi yang *open source* dan memberikan teknologi kepada pengembang dan perusahaan untuk mengembangkan sistem operasi secara independen. Android dapat dipecah menjadi 5 bagian utama, yaitu aplikasi, *framework* aplikasi, *native libraries, Android runtime,* dan *Linux Kernel* (Krajci, Cummings, 2013).

2.11. UML

UML (*Unified Modeling Language*) adalah sebuah bahasa yang berdasarkan grafik/gambar untuk memvisualisasi, menspesifikasikan,membangun, dan pendokumentasian dari sebuah sistem pengembangan software berbasis OO (*Object-Oriented*). UML juga memberikan standar penulisan sebuah sistem *blueprint*, yang meliputi konsep bisnis proses, penulisan kelas-kelas dalam bahasa

program yang spesifik, skema *database*, dan komponen-komponen yang diperlukan dalam sistem *software*.

Adapun dalam pengklasifikasiannya, UML dapat dibagi menjadi beberapa jenis yaitu:

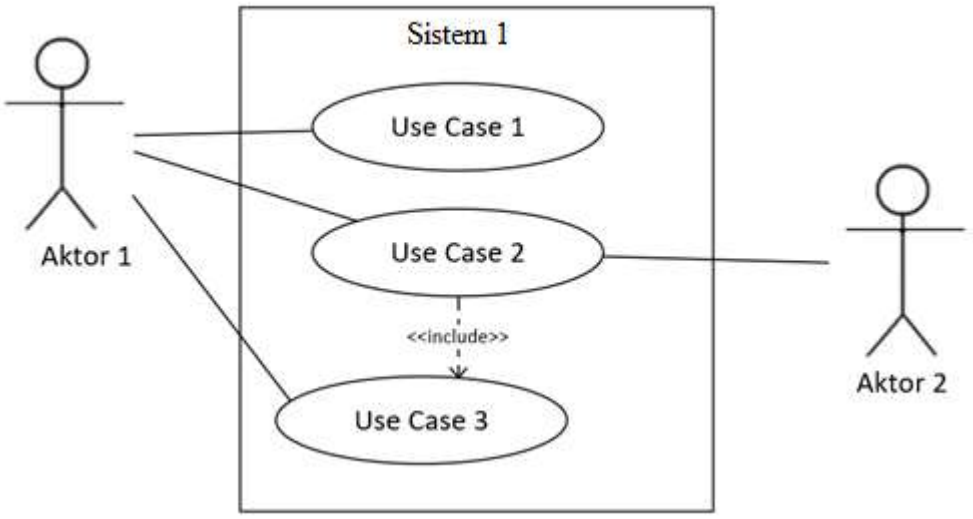
Benda / *Things*: menggambarkan abstraksi yang pertama dalam sebuah model, yang paling umum digunakan adalah *use case diagram* dan *class diagram.*

Hubungan / *Relationships*: menggambarkan hubungan dari benda-benda, yang paling umum digunakan adalah *sitemap* dan *association.*

Bagan / *Diagrams*: menggambarkan alur dari benda-benda / *things*, yang paling umum digunakan adalah *activity diagram* atau *flow chart.* (Mulyani, 2016).

2.12. Use Case Diagram

Use Case Diagram adalah salah satu dari UML berupa objek yang digunakan untuk membuat desain visualisasi suatu aplikasi dari sisi fungsionalitas yang ingin dikembangkan. Diagram ini menampilkan interaksi antara aktor dengan sistem untuk menjalankan aktivitas dalam suatu aplikasi. (Mulyani, 2016).



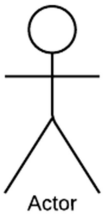
Gambar 2.4. Contoh *Use Case* beserta keterangan

Sumber: (Satzinger, 2012)

Gambar diatas menunjukkan contoh untuk suatu *use case.* Untuk penjelasan dari contoh *use case* tersebut, adalah sebagai berikut:

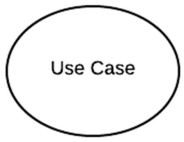
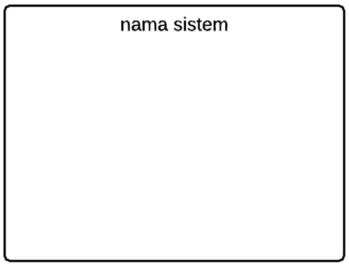
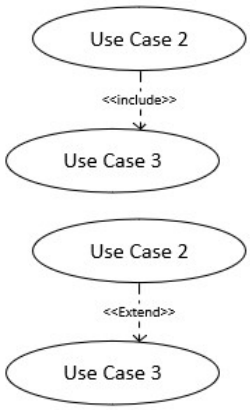
Tabel 2.2 Penjelasan Simbol *Use Case*

Sumber: (Satzinger, 2012)

Simbol	Arti	Penjelasan
	Aktor	Menunjukkan aktor atau orang yang menggunakan sistem atau menjalankan suatu *use case.*

Tabel 2.2 Penjelasan Simbol *Use Case* (Lanjutan)

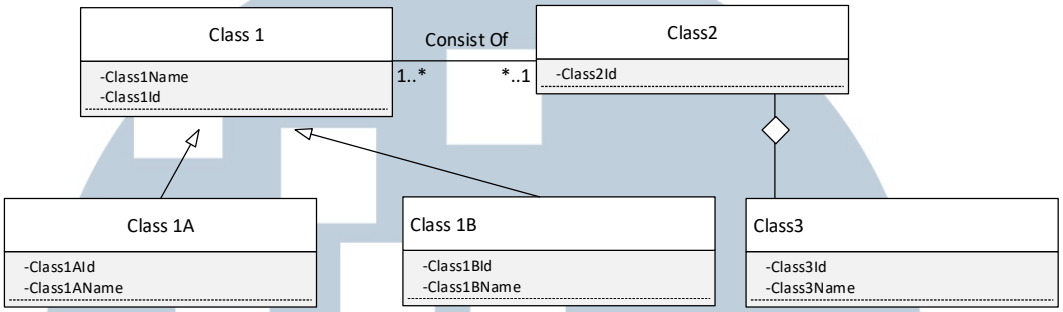
Sumber: (Satzinger, 2012)

Simbol	Arti	Penjelasan
	*Use Case*	Menunjukkan proses yang melibatkan aktor tertentu.
	*System*	Menyatakan batasan sistem dalam relasi antar aktor dan *use case*. Digambarkan dengan segi empat yang membatasi semua *use case* dalam sistem.
	*Association*	Garis yang menunjukkan hubungan suatu aktor dengan *use case* yang bersangkutan.
	*Dependency*	Garis putus- putus menggambarkan ketergantungan antar *use case*. <<include>> artinya *use case* 3 termasuk bagian dari *use case* 2 dan memerlukan *use case* 2 untuk menjalankan fungsinya , jika keterangan <<extend>> berarti *use case* 3 adalah tambahan *use case* 2, dan dapat berdiri sendiri tanpa *use case* 2.

Masing-masing relasi antara suatu aktor dan *use case* tertentu menggambarkan fungsionalitas suatu sistem dan diurutkan dari proses awal sistem sampai selesai. Inti dari diagram *use case* hanya berfokus pada konteks aktivitas apa yang dikerjakan oleh sistem, dan bukan bagaimana aktivitas tersebut dilakukan, hal ini dikarenakan diagram *use case* hanya menjelaskan dan menerangkan kebutuhan terkait fungsi yang diinginkan / dikehendaki pengguna, serta sangat berguna dalam menentukan struktur organisasi dan model daripada sebuah sistem. (Satzinger, 2012).

2.13. Class Diagram

Class diagram adalah satu dari UML berupa objek yang digunakan untuk menggambarkan desain struktur dari atribut suatu sistem beserta fungsi atau metode yang dapat digunakan dalam manipulasi atribut tersebut. (Mulyani, 2016). Struktur yang ditampilkan dalam *class diagram* terdiri dari hubungan *class*, *package* dan objek. Masing-masing dari *class* ini memiliki deskripsi yang menjadi poin penghubung, baik lewat pemanggilan, pewarisan atau asosiasi. Area dalam deskripsi *class* terdiri dari nama, metode dan atribut yang dimaksud itu sendiri. Selain itu, *class* dalam diagram dapat merupakan implementasi dari sebuah *interface* (hubungan suatu sistem dengan sistem lain). (Satzinger, 2012).



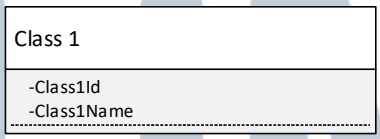
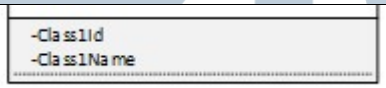
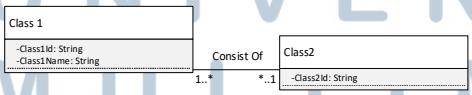
Gambar 2.5. Contoh *Class Diagram* beserta keterangan

Sumber: (Satzinger, 2012)

Gambar diatas menunjukkan contoh untuk suatu *class diagram.* Untuk penjelasan dari contoh *class diagram* tersebut, adalah sebagai berikut:

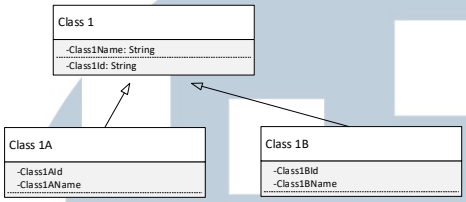
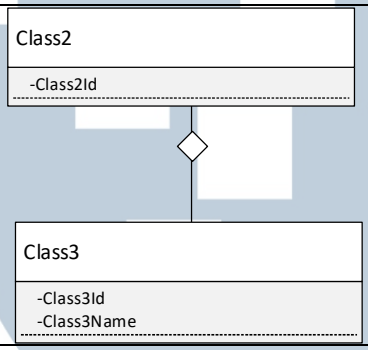
Tabel 2.3 Penjelasan Simbol *Class Diagram*

Sumber: (Satzinger, 2012)

Simbol	Arti	Penjelasan
	*Class*	Pada contoh disamping, Class 1 Menunjukkan nama dari suatu *class*, sedangkan Class1Id, dan Class1Name adalah atribut- atribut dari *class* tersebut, sedangkan Class1Operation, Class1Operation2, dan Class1Operation3 adalah operation dari *class* tersebut.
	Atribut	Atribut adalah karakteristik yang dimiliki oleh suatu *class*.
	*Association*	Menunjukkan hubungan asosiasi antar class dimana class1 1..* *consist of* *..1 *class* 2. Artinya 1 *class* 1 terdiri dari 1/ lebih *class* 2 dan sebaliknya 1/lebih *class* 2 hanya menyusun 1 *class* 1.

Tabel 2.3 Penjelasan Simbol *Class Diagram* (Lanjutan)

Sumber: (Satzinger, 2012)

Simbol	Arti	Penjelasan
	*Directed Association*	Hubungan dengan tanda panah tersebut menujukkan hubungan pewarisan bahwa *class* 1A dan 1B adalah turunan (*sub class*) dari class 1 sebagai *super class*.
	*Aggregation*	Hubungan disamping menunjukkan agregasi antara *class* 2 dan *class* 3. Berbeda dengan hubungan pewarisan pasa hubungan pewarisan *sub class* dapat berdiri sendiri.

Penjelasan hubungan antar *Class*:

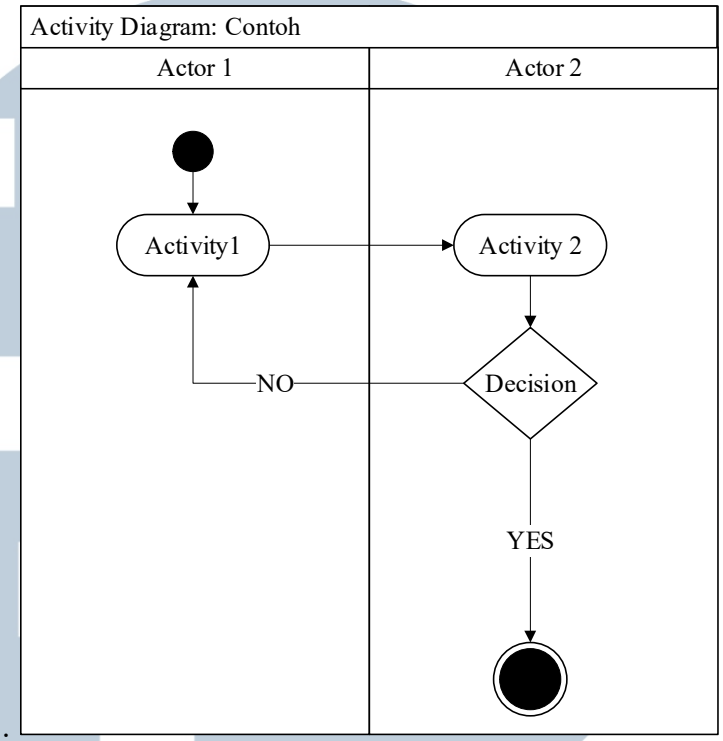
1. Pewarisan (*Directed Association)*: Adalah hubungan antar *class,* di mana satu *class* baru merupakan turunan (*sub class*) dari *class* lain sebagai *super class*. *Sub class* yang menjadi anak dalam diagram ini mewarisi atribut dan metode dari *super class* pendahulunya dan memiliki fungsionalitas baru. Sifat dari hubungan adalah hierarki. Contoh *class* pewarisan: *class* Mahasiswa SI, dan Mahasiswa TI adalah *sub class* dari *super class* Mahasiswa FTI.

2. *Association*: Adalah hubungan antar *class,* dimana satu *class* hubungan dengan *class* lain dalam suatu sistem. Contoh asosiasi adalah hubungan antara *class* mahasiswa dan *class* dosen, dimana terdapat hubungan dosen mengajar mahasiswa.

3. *Aggregation*: Adalah hubungan antar *class,* di mana satu *class* merupakan bagian dari *class* lain, namun berbeda dengan pewarisan dimana *class* tersebut dapat berdiri sendiri. Contoh agregasi adalah hubungan *class* jurusan dan *class* mahasiswa dimana *class* mahasiswa termasuk dalam bagian suatu *class* jurusan namun keduanya berdiri sendiri dan memiliki fungsionalitas yang berbeda.

2.14. Activity Diagram

Activity diagram adalah desain alur dari satu aktivitas dalam sistem yang sedang dirancang. (Mulyani, 2016). *Activity diagram* akan menampilkan aktivitas dari awal proses dimulainya suatu sistem sampai dengan proses tersebut selesai, disertai dengan cabang atau pilihan yang dapat dilakukan dalam proses tersebut. Tampilan *activity diagram* berupa kumpulan proses dan pilihan yang dapat diambil oleh sistem ketika proses terjadi. Berbeda dengan *sequence diagram, activity diagram* tidak berorientasi dari sudut pandang aktor tertentu. Setiap aktivitas diawali oleh aktivitas sebelumnya. *Activity diagram* juga digunakan untuk mendeskripsikan suatu proses yang dijalankan oleh aktor kepada sistem yang sudah dijelaskan dalam *use case*.



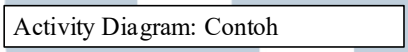
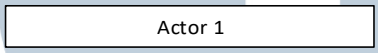
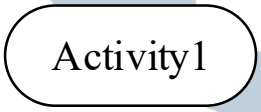
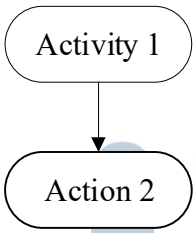
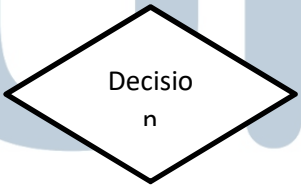
Gambar 2.4 Contoh *Activity Diagram* beserta keterangan

Sumber: (Satzinger, 2012)

Gambar diatas menunjukkan contoh untuk suatu *activity diagram*. Untuk penjelasan dari contoh *activity diagram* tersebut, adalah sebagai berikut:

Tabel 2.4 Penjelasan Simbol *Activity Diagram*

Sumber: (Satzinger, 2012)

Simbol	Arti	Penjelasan
	Nama Proses	Menunjukkan nama proses dari suatu sistem.
	*Swim Lane Heading*	Menunjukkan nama aktor dan hal yang dapat dilakukan dalam suatu *activity diagram*.
	*Starting Activity (Pseudo)*	Lingkaran hitam menunjukkan titik awal mulainya suatu proses.
	Aktivitas	Menunjukkan aktivitas dalam suatu proses.
	*Transition Arrow*	Digunakan untuk menunjukkan hubungan aktivitas, pada contoh berarti *Action 2* terjadi karena *Action*1 terjadi.
	*Decision Activity*	Menunjukkan percabangan aktivitas yang membutuhkan pilihan. Garis dengan YES berarti aktivitas yang terjadi jika pilihan bernilai ya, dan NO jika pilihan bernilai tidak.
	*Ending Activity (Pseudo)*	Menunjukkan titik akhir atau selesainya suatu proses.

2.15. Usability testing

Usability testing adalah teknik yang digunakan untuk memastikan bahwa sistem atau aplikasi yang sedang dikembangkan dapat mengerjakan fungsi yang dimiliki dengan benar, efisien, efektif, dan memuaskan. *Usability testing* dilakukan pada saat sistem memasuki tahap pra-rilis untuk mengidentifikasi masalah yang terlihat (Gaffney, 2014)

Usability testing juga digunakan untuk mengindetifikasi area atau faktor yang membuat pengguna mengalami kesulitan dalam menggunakan sebuah produk dan membantu pengembang untuk membuat rekomendasi pada tahap pengembangan. Tujuannya adalah untuk lebih memahami bagaimana pengguna berinteraksi dengan produk pengembang dan ditingkatkan dengan hasil yang diterima (Smith, 2008)

Dari 2 definisi tersebut, kesimpulan yang didapatkan adalah *Usability testing* merupakan suatu uji coba untuk mengetahui bahwa sistem yang dibuat dapat digunakan secara layak. Selain penggunaan secara layak, sistem juga harus bisa ditingkatkan berdasarkan informasi yang diberikan oleh peserta yang mengikuti *Usability testing*.

Terdapat 5 komponen utama dalam usability, yaitu:

1. *Learnablity* (Kemampuan untuk belajar)

   Kemudahan pengguna dalam menyelesaikan suatu kegiatan pada saat pertama kali menggunakan suatu *interface* aplikasi.

2. *Efficiency* (Efisiensi)

   Seberapa cepat pengguna dalam menyelesaikan kegiatan yang diberikan.

3. *Memorability* (Kemampuan untuk mengingat).

   Pengguna yang sudah lama tidak menggunakan *interface* aplikasi, masih mengingat bagaimana menggunakannya.

4. *Errors* (Kesalahan-kesalahan)

   Jumlah kesalahan yang dilakukan oleh pengguna pada saat menggunakan *interface* aplikasi.

5. *Satisfaction* (Kenyamanan)

   Senyaman apa pengguna pada saat menggunakan *interface* aplikasi. (Nielsen, 2015).

2.16. *Black Box Testing*

Black Box Testing atau sering disebut juga dengan *behavioral test*, adalah sebuah metode *testing* yang sering digunakan untuk menemukan *bug* dalam *high level operations*, pada tingkatan fitur, operasional dan skenario yang dilakukan oleh pengguna. Tester dapat membuat pengujian fungsional *black box* berdasarkan pada fungsi yang dimiliki oleh suatu sistem. *Behavioral testing* melibatkan pemahaman rinci mengenai aplikasi, masalah yang dipecahkan oleh sistem dan fungsi yang dilakukan sistem.

Behavioral test yang optimal dilakukan oleh tester yang memahami desain sistem, setidaknya pada tingkat operasional sehingga mereka dapat secara efektif menemukan *bug*, dan kekurangan umum untuk aplikasi atau sistem tersebut. Dalam melakukan *black box testing*, terlebih dahulu dibuat suatu *test case* berdasarkan informasi dari spesifikasi sistem atau aplikasi, yang nantinya akan digunakan untuk pengambilan data. *Test Case* dibuat dalam tabel yang digunakan untuk mencatat skenario dan langkah suatu fungsi sistem atau aplikasi beserta hasil yang diharapkan oleh *developer* dan hasil yang didapatkan oleh *tester*. Dari data tersebut dapat dibandingkan apakah hasil yang diharapkan sudah sesuai (*pass)* dengan hasil yang didapatkan atau tidak (*fail*). (Nidhra, 2012).

Tabel 2.5 Tabel Contoh *Black Box Testing*

No	*Summary*	*Steps*	*Actual Result*	*Expected Result*	*Pass/Fail*	*Evidence*
1	Masuk ke Aplikasi	Membuka Halaman Aplikasi	Aplikasi menampilkan halaman beranda.	Aplikasi menampilkan halaman beranda.	*Pass*	[Gambar *Result* sebagai bukti]
2	Masuk ke Halaman Fungsi	Membuka Halaman Fungsi	Aplikasi menampilkan *error*	Aplikasi menampilkan halaman fungsi	*Fail*	[Gambar *Result* sebagai bukti]