BAB III

METODOLOGI PENELITIAN

Dalam suatu perancangan sistem, analisis mutlak harus dilakukan agar sistem yang dikerjakan terstruktur dengan baik, sehingga memudahkan dalam melakukan perancangan sistem, dan apabila sistem belum selesai atau ingin ditambahkan maka akan mudah dalam menyelesaikannya

3.1 Analisis Masalah

Mobile phone merupakan kebutuhan semi primer yang sangat dibutuhkan oleh setiap kalangan dan dapat memberikan kemudahan bagi pengguna untuk memperoleh suatu informasi dengan cepat. Salah satunya dapat menjadi salah satu solusi media pembelajaran selain buku, dengan menggunakan suatu aplikasi sehingga mudah untuk dipelajari. Biasanya penggunaan aplikasi harus mewajibkan pengguna memiliki akses internet yang tidak semua pengguna memiliki akses internet tersebut, jadi terasa kurang efektif.

Memanfaatkan media *mobile phone* dapat di rasakan oleh setiap orang, dari pengamatan penulis dengan fenomena yang terjadi ini maka penulis akan membuat aplikasi pembelajaran iqra untuk mempermudah dalam mendapat ilmu dengan bentuk aplikasi di *mobile phone*.

3.2 Penyelesaian Masalah

Berdasarkan permasalahan di atas, maka dibutuhkan suatu aplikasi untuk memberikan informasi tentang pembelajaran iqra. Maka hal pertama yang harus dilakukan adalah menganalisis kebutuhan dari perangkat lunaknya terlebih dahulu dengan kebutuhan fungsionalnya.

Analisis mengenai tujuan, identifikasi dan kebutuhan aplikasi yang akan dibuat dijabarkan masing-masing pada poin-poin berikut :

3.2.1 Tujuan Sistem

Tujuan perancangan yaitu mempermudah pembelajaran mengenal nama-nama rasul tanpa harus menggunakan buku. Dengan media mobile phone dapat secara mudah dan praktis digunakannya.

3.2.2 Metode Pendekatan sistem dan Pengembangan Sistem

Dalam pembangunan suatu sistem berbasis teknologi informasi diperlukan suatu pendekatan dan pengembangan sistem yang akan menentukan proses penyelesaian rekayasa perangkat lunak, adapun pendekatan sistem yang dilakukan penulis adalah dengan menggunakan pendekatan berorientasi objek (object oriented) dan pengembangan sistem dengan menggunakan model *prototype*.

3.2.2.1 Metode Pendekatan Sistem

Metode pendekatan sistem yang penulis ambil adalah paradigma berorientasi objek (*Object Oriented*), model pemrograman yang berorientasi pada obyek. Maksudnya adalah, bahwa setiap bagian dari apa yang anda program -

visual maupun non visual- adalah sebuah obyek. Obyek, dalam konteks ini adalah "sesuatu", bisa dianalogikan dengan benda, yang padanya melekat attribut-attribut. Attribut tersebut bisa dikelompokkan ke dalam 2 jenis yaitu kelompok "property" dan kelompok "method" atau "event"(Y. Wibisono).

OOP adalah paradigma pemrograman yang cukup dominan saat ini, karena mampu memberikan solusi kaidah pemrograman modern. Meskipun demikian, bukan berarti bahwa pemrograman prosedural sudah tidak layak lagi (Efendi Hariyadi) .

OOP diciptakan karena dirasakan masih adanya keterbatasan pada bahasa pemrograman tradisional. Konsep dari OOP sendiri adalah, semua pemecahan masalah dibagi ke dalam objek. Dalam OOP data dan fungsi-fungsi yang akan mengoperasikannya digabungkan menjadi satu kesatuan yang dapat disebut sebagai objek. Proses perancangan atau desain dalam suatu pemrograman merupakan proses yang tidak terpisah dari proses yang mendahului, yaitu analisis dan proses yang mengikutinya. Pembahasan mengenai orientasi objek tidak akan terlepas dari konsep objek seperti inheritance atau penurunan, encapsulation atau pembungkusan. Empat konsep yang dimiliki oleh pemograman yang berorientasi object adalah :

1. Encapsulation
2. Abstraction
3. Inheritance
4. Polymorphism

3.2.2.1.1 Encapsulation

Encapsulation adalah merupakan penyembunyian informasi melalui private dan protected. Pada dasarnya adalah kita tidak perlu mengetahui apa yang ada dalam objek itu (class), yang kita butuhkan adalah apa kegunaan, bagaimana cara memakainya dan apa yang akan terjadi.

3.2.2.1.2 Abstraction

Abstraction memiliki pengertian bahwa dalam perangkat lunak seorang programmer akan menggunakan atau menciptakan sejumlah objek dimana objek tersebut ada dan berprilaku pada sistem.

3.2.2.1.3 Inheritance

Inheritance atau pewarisan adalah merupakan suatu objek yang berasal dari turunan pertama atau lebih objek lainnya. Intinya adalah mengembangkan objek yang sudah ada. Jadi kita tidak perlu membentuk objek dari awal, objek yang sudah ada bisa dimodifikasi dan/atau ditambah fungsinya sesuai dengan kebutuhan.

3.2.2.1.4 Polymorphism

Polymorphism adalah kemampuan objek berubah menjadi objek yang lain, beberapa objek bertipe sama bereaksi secara berbeda terhadap "pesan" yang sama.

3.2.2.2 Alat Bantu Analisis dan Perancangan

Analisis sistem dapat didefinisikan sebagai penguraian dari suatu sistem informasi yang utuh ke dalam bagian-bagian komponennya dengan maksud untuk mengidentifikasikan dan mengevaluasi permasalahan-permasalahan, kesempatan-

kesempatan dan hambatan-hambatan yang terjadi dalam kebutuhan yang diharapkan sehingga dapat diusulkan perbaikan-perbaikan.

Perancangan sistem memiliki tujuan men-design sistem baru yang dapat menyelesaikan masalah-masalah yang dihadapi perusahaan yang diperoleh dari pemilihan alternatif sistem yang terbaik, kegiatan yang dilakukan dalam tahap perancangan meliputi perancangan *input, output* dan *file*. Alat permodelan perancangan sistem adalah sebagai berikut :

3.2.2.2.1 UML (Unified Modelling Language)

Menurut Bambang Hariyanto. Ir.,MT. (2004:17) UML adalah bahasa grafis untuk mendokumentasikan, menspesifikasikan, dan membangun sistem perangkat lunak. UML berorientasi objek menerapkan banyak level abstraksi, tidak bergantung proses pengembangan, tidak bergantung bahasa dan teknologi.

Tujuan UML

1. Menyediakan bahasa pemodelan visual yang ekspresif dan siap pakai untuk mengembangkan pertukaran model-model yang berarti.
2. Menyediakan mekanisme perluasan dan spesialisasi untuk memperluas konsep-konsep inti.
3. Mendukung spesifikasi independent bahasa pemprograman dan proses pengembangan tertentu.
4. Menyediakan basis formal untuk pemahaman bahasa pemodelan.

UML mendefinisikan diagram-diagram sebagai berikut:

1. use case diagram
2. class diagram
3. activity diagram
4. sequence diagram

3.2.2.2.1.1 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. Use case merupakan sebuah pekerjaan tertentu, misalnya login ke sistem, meng-create sebuah daftar belanja, dan sebagainya.

Seorang/sebuah aktor adalah sebuah entitas manusia atau mesin yang berinteraksi dengan sistem untuk melakukan pekerjaan-pekerjaan tertentu.

Use case diagram dapat sangat membantu bila kita sedang menyusun requirement sebuah sistem, mengkomunikasikan rancangan dengan klien, dan merancang test case untuk semua feature yang ada pada sistem.

1. Sebuah use case dapat meng-include fungsionalitas use case lain sebagai bagian dari proses dalam dirinya. Secara umum diasumsikan bahwa use case yang di- include akan dipanggil setiap kali use case yang meng-include dieksekusi secara normal.
2. Sebuah use case dapat di-include oleh lebih dari satu use case lain, sehingga duplikasi fungsionalitas dapat dihindari dengan cara menarik keluar fungsionalitas yang common.

3. Sebuah use case juga dapat meng-extend use case lain dengan behavior- nya sendiri.
4. Sementara hubungan generalisasi antar use case menunjukkan bahwa use case yang satu merupakan spesialisasi dari yang lain

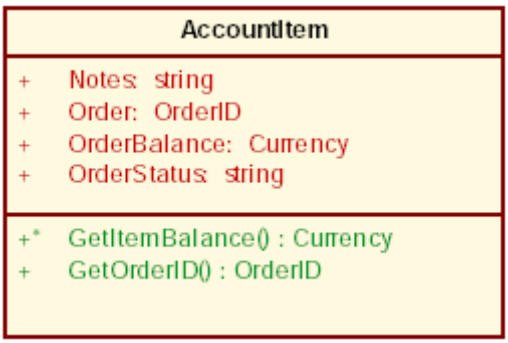


Gambar 3.1. Contoh Use Case Diagram

(Munawar. 2005. *Pemodelan visual dengan UML*. Graha Ilmu. Yogyakarta.)

3.2.2.2.1.2 Class Diagram

Class adalah sebuah spesifikasi yang jika diinstansiasi 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 (metoda/fungsi). *Class diagram* menggambarkan struktur dan deskripsi *class*, *package* dan objek beserta hubungan satu sama lain seperti *containment*, pewarisan, asosiasi, dan lain-lain. *Class* memiliki tiga area pokok yaitu : nama,atribut,metoda.



Gambar 3.2 Contoh Class Diagram

(Booch Jcobson Tumbaugh, Martin Fowler , 2005, *UML Distilled* Edisi 2, Andi)

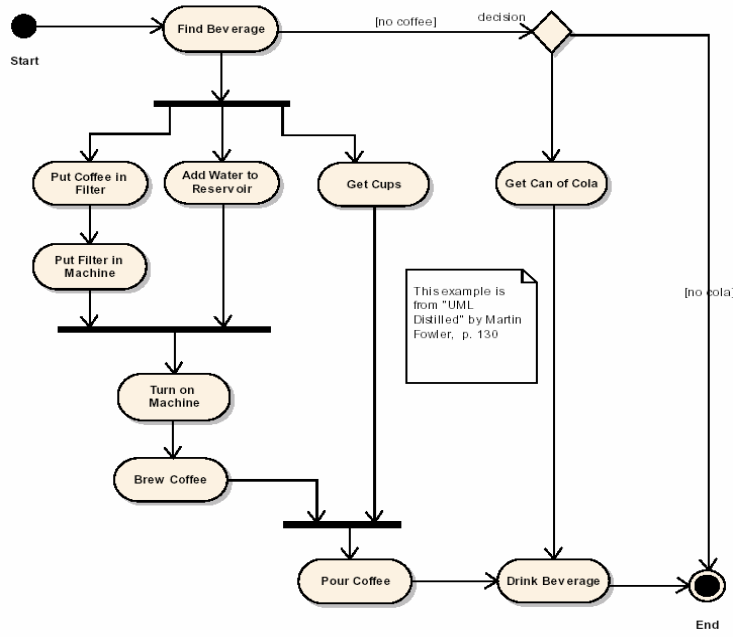
Hubungan Antar *Class*

1. Asosiasi, yaitu hubungan statis antar *class*. Umumnya menggambarkan *class* yang memiliki atribut berupa *class* lain, atau *class* yang harus mengetahui eksistensi *class* lain. Panah *navigability* menunjukkan arah *query* antar *class*.
2. Agregasi, yaitu hubungan yang menyatakan bagian ("terdiri atas..").
3. Pewarisan, yaitu hubungan hirarkis antar *class*. *Class* dapat diturunkan dari *class* lain dan mewarisi semua atribut dan metoda *class* asalnya dan menambahkan fungsionalitas baru, sehingga ia disebut anak dari *class* yang diwarisinya. Kebalikan dari pewarisan adalah generalisasi.
4. Hubungan dinamis, yaitu rangkaian pesan (*message*) yang di-*passing* dari satu *class* kepada *class* lain. Hubungan dinamis dapat digambarkan dengan menggunakan *sequence diagram* yang akan dijelaskan kemudian.

3.2.2.2.1.3 Activity Diagram

Activity diagrams 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. *Activity diagram* merupakan *state diagram* khusus, di mana sebagian besar *state* adalah *action* dan sebagian besar transisi di-*trigger* oleh selesainya *state* sebelumnya (*internal processing*).

Oleh karena itu *activity diagram* tidak menggambarkan behaviour internal sebuah sistem (dan interaksi antar subsistem) secara eksak, tetapi lebih menggambarkan proses-proses dan jalur-jalur aktivitas dari level atas secara umum. Sebuah aktivitas dapat direalisasikan oleh satu *use case* atau lebih. Aktivitas menggambarkan proses yang berjalan, sementara *use case* menggambarkan bagaimana aktor menggunakan sistem untuk melakukan aktivitas.



Gambar 3.3 Contoh Activiy Diagram

(Booch Jcobson Tumbaugh, Martin Fowler , 2005, *UML Distilled* Edisi 2, Andi)

3.2.2.2.1.4 Sequence Diagram

Sequence diagram menggambarkan interaksi antar objek di dalam dan di sekitar sistem (termasuk pengguna, *display*, dan sebagainya) berupa *message* yang digambarkan terhadap waktu. *Sequence diagram* terdiri atar dimensi vertikal (waktu) dan dimensi horizontal (objek-objek yang terkait).

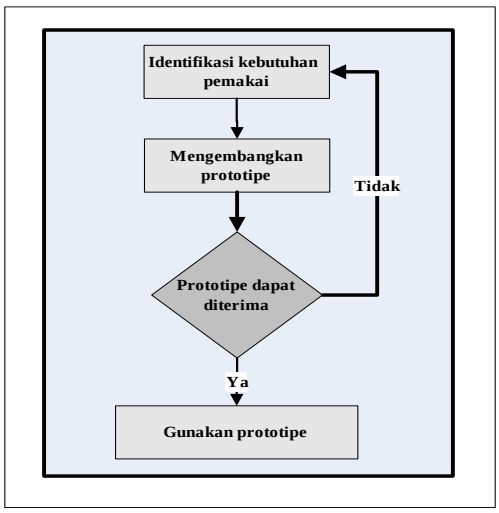
Sequence diagram biasa digunakan untuk menggambarkan skenario atau rangkaian langkah-langkah yang dilakukan sebagai respons dari sebuah *event* untuk menghasilkan *output* tertentu. Diawali dari apa yang men-*trigger* aktivitas

tersebut, proses dan perubahan apa saja yang terjadi secara internal dan *output* apa yang dihasilkan.

Masing-masing objek, termasuk aktor, memiliki *lifeline* vertikal. *Message* digambarkan sebagai garis berpanah dari satu objek ke objek lainnya. Pada fase desain berikutnya, *message* akan dipetakan menjadi operasi/metoda dari *class*. *Activation bar* menunjukkan lamanya eksekusi sebuah proses, biasanya diawali dengan diterimanya sebuah message. Untuk objek-objek yang memiliki sifat khusus, standar UML mendefinisikan *icon* khusus untuk objek *boundary*, *controller* dan *persistent entity*.

3.2.2.3 Metode Pengembangan Sistem

Metode pengembangan sistem yang digunakan adalah dengan menggunakan metode Model Prototype yang merupakan mekanisme untuk mengidentifikasi kebutuhan perangkat lunak.



Gambar 3.4 Metode Pengembangan Prototype

(Sumber : Raymond McLeod, Jr. and Goorge P. Schell **.** 2007. *Management Information Systems* .New Jersey.)

a. Keuntungan prototype

1. Evaluasi dan feedback pada rancangan interaktif.
2. Stakeholder (dalam hal ini user) dapat melihat, menyentuh, berinteraksi dengan prototype.
3. Anggota tim dapat berkomunikasi secara efektif.
4. Para perancang dapat mengeluarkan ide-idenya.
5. Memunculkan ide-ide secara visual dan mengembangkannya.
6. Dapat menjawab pertanyaan dan membantu pemilihan di antara alternatif-alternatif.
7. Mengurangi biaya pengembangan dan pembiayaan
8. Dapat memperbaikimasalah penggunaan sebelum dibuat programnya

b. Kelemahan prototype

1. Prototype hanya akan berhasil jika pemakain bersungguh-sungguh dalam menyediakan waktu dan pikiran untuk mengerjakan prototype.
2. Kemungkinan dokumentasi terabaikan karena pengembangan lebih berkonsentrasi pada pengujian dan pembuatan prototype.
3. Mengingat target waktu yang pendek, ada kemungkinan sistem yang dibuat tidak lengkap dan bahkan kurang teruji.

4. Jika terlalu banyak proses pengulangan dalam membuat prototype, ada kemungkinan pemakai merasa jenuh dan memberikan reaksi negatif
5. Apabila tidak terkelola dengan baik, prototype menjadi tidak pernah berakhir. Hal ini disebabkan permintaan perubahan terlalu mudah untuk dipenuhi

3.2.3 Pengujian Software

Pengujian merupakan bagian penting dalam siklus pengembangan perangkat lunak. Pengujian dilakukan untuk menjamin kualitas dan juga mengetahui kelemahan dar perangkat lunak. Tujuan dari pengujian ini adalah untuk menjamin bahwa perangkat lunak yang dibangun memiliki kualitas yang dapat diandalkan.

Dalam suatu program membutuhkan suatu teknik pengujian yang berguna dalam mengevaluasi program yang dibuat. Adapun teknik pengujian yang digunakan dalam perangkat lunak tersebut adalah *Black box.*

3.2.3.1 Black box

Digunakan untuk menguji fungsi-fungsi khusus dari perangkat lunak yang dirancang. Kebenaran perangkat lunak yang diuji hanya dilihat berdasarkan keluaran yang dihasilkan dari data atau kondisi masukan yang diberikan untuk fungsi yang ada tanpa melihat bagaimana proses untuk mendapatkan keluaran tersebut. Dari keluaran yang dihasilkan, kemampuan program dalam memenuhi kebutuhan pemakai dapat diukur sekaligus dapat diiketahui kesalahan-kesalahannya

Pengujian *black box* berfokus pada persyaratan fungsional perangkat lunak. Dengan demikian, pengujian *black box* memungkinkan perekayasa perangkat lunak mendapatkan serangkaian kondisi input yang sepenuhnya menggunakan semua persyaratan fungsional untuk suatu program. Pengujian *black box* berusaha menemukan kesalahan dalam kategori sebagai berikut :

1. Fungsi – fungsi yang tidak benar atau hilang.
2. Kesalahan interface.
3. Kesalahan dalam struktur data atau akses database eksternal.
4. Kesalahan kinerja.
5. Inisialisasi dan kesalahan terminasi.