BAB III

METODOLOGI PENELITIAN

3.1 Metodologi Penelitian

Metodologi penelitian adalah langkah dan prosedur yang akan dilakukan dalam pengumpulan data atau informasi guna memecahkan permasalahan dan menguji hipotesis penelitian.

Metode yang digunakan dalam melaksanakan penelitian merupakan dasar penyusunan rancangan penelitian dan merupakan penjabaran dari metode ilmiah secara umum.

3.1.1 Desain Penelitian

Metode penelitian yang digunakan oleh penulis adalah metode penelitian deskriptif dan komparatif. Penelitian deskriptif adalah penelitian terhadap fenomena atau populasi tertentu yang diperoleh oleh peneliti dari subjek berupa individu, organisasional, industri atau perspektif yang lain. Penelitian deskriptif dilakukan untuk menjawab pertanyaan tentang: apa, siapa, kapan, di mana, dan bagaimana yang berkaitan dengan karakteristik populasi atau fenomena tersebut. Penelitian juga dilakukan dengan menggunakan metode komparatif yang dilakukan dengan membandingkan teori yang ada dengan praktik yang ditemui dan menarik kesimpulan.

Untuk memudahkan dalam melakukan penelitian yang berhubungan dengan masalah yang diteliti, maka penulis melakukan pendekatan studi kasus. Dengan menggunakan pendekatan ini, data yang dikumpulkan dapat disesuaikan dengan keadaan yang sebenarnya dan dibandingkan dengan teori yang menunjang. Dengan demikian, dapat memberikan gambaran yang cukup jelas serta dapat menarik kesimpulan dari objek yang diteliti.

3.1.2 Jenis dan Metode Pengumpulan Data

Jenis dan Metode pengumpulan data digunakan penulis untuk mendapatkan data sebagai bahan kajian dalam penulisan skripsi dengan tujuan membuat suatu perancangan Aplikasi Resep Makanan Berbasis Android . Dalam hal ini penulis menggunakan metode pengumpulan data berupa sumber data primer (observasi, wawancara dan pengamatan sistem) dan sumber data sekunder (dokumentasi).

3.1.2.1 Jenis Pengumpulan Data

1. **Sumber Data Primer**

a. Teknik observasi yaitu dengan melakukan pengamatan dan pencatatan secara langsung pada ibu rumah tangga, mahasiswa – mahasiswi dan remaja yang nantinya akan menggunakan aplikasi resep makanan ini.

b. Teknik wawancara yaitu dengan cara melakukan tanya jawab secara langsung kepada ibu rumah tangga, mahasiswa – mahasiswi dan remaja

untuk mengetahui bagaimana selama ini mendapatkan resep makanan dan apa kesulitannya.

c. Teknik kepustakaan yaitu dengan mengumpulkan data dari buku atau bahan tulisan yang ada relevansinya dengan skripsi ini.

2. Data sekunder

Merupakan data yang telah dikumpulkan lembaga pengumpul data dan dipublikasikan kepada masyarakat pengguna data . Data sekunder dalam penelitian ini adalah data – data resep makanan dan kesulitannya pengguna aplikasi.

3.1.2.2 Metode Pengumpulan Data

Pengumpulan data primer membutuhkan perancangan alat dan metode pengumpulan data.

Metode pengumpulan data penelitian:

1. Observasi
2. Wawancara

Semua butir (*item*) yang ditanyakan dalam semua metode pengumpulan data haruslah sejalan dengan rumusan masalah dan/atau hipotesis penelitian Karenanya diperlukan proses Dekomposisi variabel penelitian menjadi sub-variabel, dimensi dan butir penelitian merupakan pekerjaan yang harus dilakukan dengan hati-hati. Proses dekomposisi ini juga memudahkan proses pengukuran dan pengumpulan data. Proses dekomposisi ini dikenal sebagai proses operasionalisasi variabel penelitian

3.1.3 Metode Pendekatan dan Pengembangan Sistem

Dalam pembangunan suatu aplikasi diperlukan suatu pendekatan dan pengembangan sistem yang akan menentukan proses penyelesaian rekayasa perangkat lunak, adapun pendekatan sistem yang digunakan adalah dengan menggunakan pendekatan berorientasi objek dan pengembangan sistem dengan menggunakan model *prototype* paradigma.

3.1.3.1 Metode Pendekatan Sistem

Metode pendekatan sistem yang di gunakan adalah pendekatan dengan *Object Oriented* yang menggunakan AOO (*Analisis Object Oriented*) dan DOO yang di visualisasikan dengan UML dan di antara nya adalah sebagai berikut : *Use Case Diagram, Sequence Diagram, Class Diagram, Collaboration Diagram, Component Diagram dan Deployment Diagram.*

3.1.3.2 Metode Pengembangan Sistem

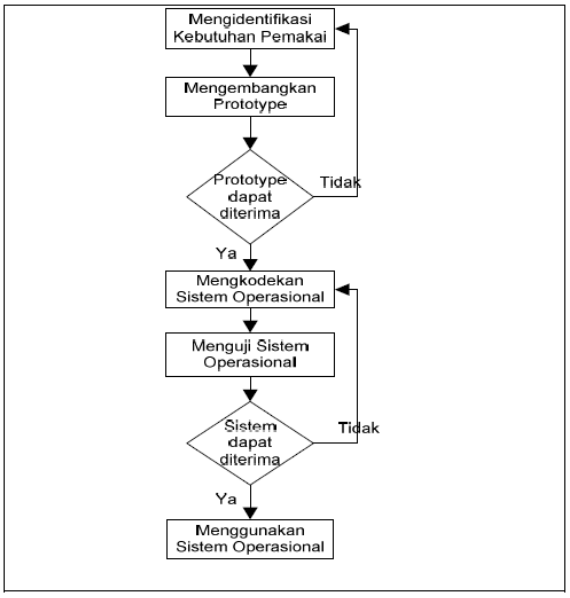
Metode pengembangan sistem yang digunakan adalah dengan menggunakan metode pendekatan *prototype*. *Prototyping* merupakan salah satu metode pengembangan perangat lunak yang banyak digunakan. *Prototype* dibangun untuk mendefinisikan kebutuhan.

Prototyping merupakan salah satu metode pengembangan perangat lunak yang banyak digunakan. Dengan metode *prototyping* ini pengembang dan pelanggan dapat saling berinteraksi selama proses pembuatan sistem.

Sering terjadi seorang pelanggan hanya mendefinisikan secara umum apa yang dikehendakinya tanpa menyebutkan secara detail output apa saja yang dibutuhkan, pemrosesan dan data-data apa saja yang dibutuhkan. Sebaliknya disisi pengembang kurang memperhatikan efesiensi algoritma, kemampuan sistem operasi dan *interface* yang menghubungkan manusia dan komputer.

Untuk mengatasi ketidak serasian antara pelanggan dan pengembang , maka harus dibutuhakan kerjasama yanga baik diantara keduanya sehingga pengembang akan mengetahui dengan benar apa yang diinginkan pelanggan dengan tidak mengesampingkan segi-segi teknis dan pelanggan akan mengetahui proses-proses dalam menyelasaikan sistem yang diinginkan. Dengan demikian akan menghasilkan sistem sesuai dengan jadwal waktu penyelesaian yang telah ditentukan.

Kunci agar model *prototype* ini berhasil dengan baik adalah dengan mendefinisikan aturan-aturan main pada saat awal, yaitu pelanggan dan pengembang harus setuju bahwa *prototype* dibangun untuk mendefinisikan kebutuhan. *Prototype* akan dihilangkan sebagian atau seluruhnya dan perangkat lunak aktual aktual direkayasa dengan kualitas dan implementasi yang sudah ditentukan .



Gambar 3.2 Metode *Prototype*

(Sumber : Mcleod Raymond.2001.*Sistem Informasi Manajemen*.

PT Prenhallindo.Jakarta)

Berikut adalah langkah-langkah dalam merancang sistem yang penulis gunakan dalam mekanisme pengembangan sistem dengan *prototype* , langkah-langkah tersebut adalah :

1. Mengidentifikasikan kebutuhan pemakai

   Analisis sistem mewarnai pemakai untuk mendapatkan gagasan dan apa yang diinginkan pemakai terhadap sistem.

2. Membangun *prototype*

   Analisis sistem mungkin bekerja sama dengan spesialis informasi lainya, menggunakan satu atau lebih peralatan *prototype* untuk mengembangkan sebuah *prototype*. Membangun *prototyping* dengan

membuat perancangan sementara yang berfokus pada penyajian pada pelanggan (misalnya dengan membuat format *input* dan *output*)

3. Evaluasi *Prototyping*

   Evaluasi ini dilakukan oleh *user* apakah *prototyping* yang sudah dibangun sudah sesuai dengan keinginan pelanggan. Jika sudah sesuai maka langkah 4 akan diambil. Jika tidak *prototyping* direvisi dengan mengulang langkah 1, 2, dan 3.

4. Pengkodean

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

5. Menguji sistem

   Setelah sistem sudah menjadi suatu perangkat lunak yang siap pakai, harus di tes dahulu sebelum digunakan. Pengujian ini dilakukan dengan *Black Box.*

6. Evaluasi Sistem

   Pelanggan mengevaluasi apakah sistem yang sudah jadi sudah sesuai dengan yang diharapkan . Jika ya, langkah 7 dilakukan; jika tidak, ulangi langkah 4 dan 5.

7. Menggunakan sistem

   Perangkat lunak yang telah diuji dan diterima pelanggan siap untuk digunakan .

3.2 Alat Bantu Analisis dan Perancangan

Dalam analisis dan perancangan penyusun menggambarkan bagaimana karakteristik sistem dengan menggunakan pemodelan yang disebut *Unifield Modelling Language* (UML).

Unified Modeling Language (UML) adalah bahasa pemodelan secara grafis untuk menspesifikasikan, menvisualisasikan, membangun, dan mendokumentasikan seluruh rancangan sistem perangkat lunak. Penggunaan model ini bertujuan untuk mengidentifikasikan bagian-bagian yang termasuk dalam lingkup sistem yang dibahas dan bagaimana hubungan antara sistem dengan subsistem maupun sistem lain di luarnya.

Selain itu UML adalah bahasa pemodelan yang menggunakan konsep orientasi objek. UML dibuat oleh Grady Booch, James Rumbaugh, dan Ivar Jacobson di bawah bendera *Rational Software Corp*. UML menyediakan notasi-notasi yang membantu memodelkan sistem dari berbagai perspektif. UML tidak hanya digunakan dalam pemodelan perangkat lunak, namun hampir dalam semua bidang yang membutuhkan pemodelan.

Berikut ini definisi *Unified Modeling Language* (UML) menurut para ahli:

1. Menurut (Adi Nugroho : 2009). "*Unified Modeling Language* (UML) adalah alat bantu analisis serta perancangan perangkat lunak berbasis objek".
2. Menurut (Joomla dari http://soetrasoft.com : 2007). "*Unified Modeling Language* (UML) merupakan standard modeling language yang terdiri dari kumpulan-kumpulan diagram, dikembangkan untuk membantu para pengembang sistem dan *software* agar bisa menyelesaikan tugas-tugas

seperti: Spesifikasi, Visualisasi, Desain Arsitektur, Konstruksi, Simulasi dan testing serta Dokumentasi".

Berdasarkan beberapa pendapat yang dikemukakan diatas dapat ditarik kesimpulan bahwa "*Unified Modeling Language* (UML) adalah sebuah bahasa yang berdasarkan grafik atau gambar untuk menvisualisasikan, menspesifikasikan, membangun dan pendokumentasian dari sebuah sistem pengembangan perangkat lunak berbasis OO (*Object Oriented*)".

Unified Modelling Language (UML) adalah sebuah bahasa untuk menentukan, visualisasi, konstruksi, dan mendokumentasikan *artifacts* dari sistem *software*, untuk memodelkan bisnis, dan sistem *nonsoftware* lainnya. *Artifacts* adalah sepotong informasi yang digunakan atau dihasilkan dalam suatu proses rekayasa *software*. *Artifacts* dapat berupa model, deskripsi, atau *software*. Untuk membuat suatu model, UML memiliki diagram grafis yang diberi nama berdasarkan sudut pandang yang berbeda-beda terhadap sistem dalam proses analisa atau rekayasa.

3.2.1 Use Case Diagram

Use Case Diagram adalah suatu kumpulan urutan interaksi di antara *user* dengan sistem untuk mencapai suatu tujuan di mana *use case* ini menggambarkan kebutuhan fungsional suatu sistem tanpa menampilkan struktur internal sistem.

3.2.2 Activity Diagram

Activity diagram adalah representasi secara grafis dari proses dari proses dan *control flow* dan berfungsi untuk memperlihatkan alur dari satu aktivitas ke aktivitas yang lain serta menggambarkan perilaku yang kompleks.

3.2.3 Sequence Diagram

Sequence diagram adalah suatu diagram yang menggambarkan interaksi antar objek dan mengindikasikan komunikasi diantara objek-objek tersebut. Diagram ini juga menunjukkan serangkaian pesan yang dipertukarkan oleh objek-objek yang melakukan suatu tugas atau aksi tertentu. Objek-objek tersebut kemudian diurutkan dari kiri ke kanan, aktor yang menginisiasi interaksi biasanya ditaruh di paling kiri dari diagram.

3.2.4 Class Diagram

Diagram kelas atau *class diagram* menggambarkan struktur sistem dari segi pendefinisian kelas - kelas yang akan dibuat untuk membangun sistem. Kelas memiliki apa yang disebut atribut dan metode atau operasi. Atribut merupakan variabel - variabel yang dimiliki oleh suatu kelas sedangkan operasi atau metode adalah fungsi - fungsi yang dimiliki oleh suatu kelas.

3.2.5 Object Diagram

Diagram objek menggambarkan struktur sistem dari segi penamaan objek dan jalannya objek dalam sistem. Pada diagram objek harus dipastikan semua

kelas yang sudah didefinisikan pada diagram kelas harus dipakai objeknya, karena jika tidak pendefinisian kelas itu tidak dapat dipertanggung jawabkan.

3.2.6 Component Diagram

Component diagram adalah diagram yang menunjukkan organisasi dan kebergantungan di antara sekumpulan komponen. Diagram ini memodelkan pandangan implementasi fisik dari sistem.

3.2.7 Deployment Diagram

Deployment diagram adalah penggambaran tugas-tugas kongkrit dari setiap *node/software* yang terlibat dalam jaringan sistem, menampilkan keseluruhan *node* dalam jaringan serta hubungan dari *node-node* tersebut termasuk proses-proses yang terlibat di dalamnya.

3.3 Pengujian *Software*

Pengujian dilakukan untuk mendapatkan suatu perangkat unak yang ‘layak‘ untuk digunakan. Suatu perangkat lunak yang telah selesai diujikan harus memiliki standard kualitas tertentu.

Blackbox testing adalah metode pengujian perangkat lunak yang tes fungsionalitas dari aplikasi yang bertentangan dengan struktur internal atau kerja (lihat pengujian *whitebox*). pengetahuan khusus dari kode aplikasi / struktur internal dan pengetahuan pemrograman pada umumnya tidak diperlukan. Uji kasus dibangun di sekitar spesifikasi dan persyaratan, yakni, aplikasi apa yang

seharusnya dilakukan. Menggunakan deskripsi eksternal perangkat lunak, termasuk spesifikasi, persyaratan, dan desain untuk menurunkan uji kasus. Tes ini dapat menjadi fungsional atau non-fungsional, meskipun biasanya fungsional. Perancang uji memilih input yang valid dan tidak valid dan menentukan *output* yang benar.Tidak ada pengetahuan tentang struktur internal benda uji itu.

Metode uji dapat diterapkan pada semua tingkat pengujian perangkat lunak: unit, integrasi, fungsional, sistem dan penerimaan. Ini biasanya terdiri dari kebanyakan jika tidak semua pengujian pada tingkat yang lebih tinggi, tetapi juga bisa mendominasi unit testing juga.

Adapun faktor-faktor pengujian *blackbox* adalah :

1. *File integrity*

   Menekankan pada data yang dimasukkan melalui aplikasi akan tidak bias diubah.

   Prosedur yang akan memastikan bahwa *file* yang digunakan benar dan data dalam *file* tersebut akan disimpan sekuensial dan benar.

2. *Service levels*

   Menekankan bahwa hasil yang diinginkan di dapat dalam waktu yang di inginkan oleh *user*. Untuk mencapai keinginan tersebut, harus dilakukan penyesuaian antara keinginan *user* dengan sumber daya yang ada.

3. *Ease of use*

Menekankan perluasan usaha yang diminta untuk belajar, mengoprasikan dan menyiapakan inputan, dan menginterpretasikan output dari sistem. Faktor ini tersangkut dengan *usability* sistem terhadap interaksi antara manusia dan sistem.

4. *Authorization*

Menjamin data diproses sesuai dengan ketentuan manajemen. *Authorization* menyangkut proses transaksi secara umum dan khusus. Fokus Pengujian *Black box testing* yaitu sebagai berikut :

a) Menguji fungsi-fungsi khusus dari aplikasi.

b) *Test input* dan *output* untuk fungsi yang ada tanpa memperhatikan prosesnya.

Beberapa jenis kesalahan yang dapat di identifikasi :

1) Fungsi tidak benar atau hilang,

2) Kesalahan antar muka,

3) Kesalahan pada struktur data (pengaksesan basis data),

4) Kesalahan inisialisasi dan akhir program.