BAB III
OBJEK DAN METODE PENELITIAN

3.1 Objek Penelitian

3.1.1 Sejarah Singkat Perusahaan

PT. Swamedia Informatika berdiri sejak tahun 1999, berkedudukan di Bandung. Dengan moto *innovative IT Solution*, kepuasan klien menjadi misi utama PT.Swamedia Informatika. PT. Swamedia Informatika menawarkan solusi teknologi *information & communication* (infocom) yang inovatif dan komprehensif untuk memberikan hasil kerja yang berkualitas handal (*reliable*), aman (*secure*), dan sesuai kebutuhan (*customized*), tapi dengan harga yang sangat kompetitif. Setelah satu dekade berjalan, banyak hasil yang telah dicapai PT. Swamedia Informatika. Kemampuan, penguasaan teknologi, dan pengalaman (*portfolios*) berkembang sangat signifikan. BUMN, lembaga pemerintah, dan sektor swasta telah menggunakan jasa PT. Swamedia Informatika , dan Industri telekomunikasi adalah klien terbesarnya.

3.1.2 Visi dan Misi Perusahaan

Visi dan misi PT. Swamedia Informatika adalah sebagai berikut :

Visi

Visi kami adalah menjadi perusahaan terkemuka yang mengedepankan kekuatan solusi dan kompetensi di bidang teknologi informasi dan komunikasi untuk memberikan kepuasan stakeholders.

Misi

- Memberikan pelayanan terbaik demi tercapainya kepuasan pelanggan dengan jaminan kualitas pekerjaan, kecepatan, ketepatan, dan harga yang kompetitif.
- Meningkatkan kemampuan dan mengoptimalkan pengelolaan sumber daya manusia yang unggul dan dapat dipercaya

Mengoptimalkan penggunaan teknologi yang handal (*reliable*), aman (*secure*), dan murah (*low cost*)

- Membangun pusat riset dan pengelolaan portfolios perusahaan (*library*)
- Membangun kemitraan strategis dan bersinergi dengan klien maupun partner dengan prinsip saling menguntungkan.

3.2 Metode Penelitian

3.2.1 Desain Penelitian

Desain Penelitian Menurut Prof. Dr. Suharsimi Arikunto (2006 : 51) adalah rencana dan struktur penyelidikan yang digunakan untuk memperoleh bukti-bukti empiris dalam menjawab pertanyaan penelitian. Desain penelitian ini merupakan kerangka atau perincian prosedur kerja yang akan dilakukan pada waktu meneliti, sehingga diharapakan dapat memberikan gambaran dan arah mana yang akan dilakukan dalam melaksanakan penelitian tersebut, serta memberikan gambaran jika peneletian itu telah jadi atau selesai penelitian tersebut diberlakukan.

Desain penelitian yang baik dapat memudahkan kita dalam melakukan penelitian.Yang dimaksud rancangan penelitian adalah pokok – pokok atau tiang – tiang dari suatu usul penelitian yang menggambarkan suatu penelitian itu dilakukan dan bagaimana hasil penelitian itu diperkirakan setelah selesai nanti.

3.2.2 Jenis Dan Metode Pengumpulan Data

3.2.2.1 Sumber Data Primer

Untuk mengumpulkan data primer penulis menggunakan metode sebagai berikut :

1. Observasi
   Penulis melakukan Observasi secara langsung terhadap proses pembuatan dari awal maupun pengembangan sebuah aplikasi di PT.Swamedia Informatika.
2. Wawancara
   Penulis melakukan wawancara secara langsung dengan pihak-pihak internal PT.Swamedia Informatika, terutama kepada *developer* yang melakukan pengembangan aplikasi, baik aplikasi yang sifatnya membangun dari awal maupun aplikasi yang di kembangkan lebih lanjut.

3.2.2.2 Sumber Data Sekunder

Pengumpulan sumber data sekunder didapatkan dari pembelajaran buku-buku terutama buku pemrograman, buku perancangnan dan analisis sistem , jurnal maupun website yang berkaitan dengan kajian penulis dan referensi penunjang lainnya.

3.2.3 Metode Pendekatan dan Pengembangan Sistem

3.2.3.1 Metode Pendekatan Sistem

Metode pendekatan sistem yang digunakan ialah metode pendekatan berorientasi objek.pendekatan berorientasi obyek memecah komponen-komponennya menjadi obyek-obyek yang saling berinteraksi. Dunia nyata yang terdiri dari obyek-obyek dapat dengan mudah dimodelkan sehingga program bisa lebih mendekati kondisi yang

sebenarnya.Terdapat beberapa keuntungan menggunakan pemrograman berorientasi obyek, antara lain:

a. Real world programming

Dunia ini disusun atas obyek obyek yang saling berinteraksi.Sebuah program yang memodelkan dunia nyata, sebisa mungkin menggambarkan kondisi yang ada dalam bentuk yang seakurat mungkin.Dalam Pemrograman berorientasi obyek, sebuah program disusun oleh obyek-obyek yang masing-masing memiliki fungsi sesuai dengan peran dan kebutuhan interaksinya.

b. Reusability of code

Kelas yang telah dibuat dalam pemrograman berorientasi obyek bisa digunakan oleh program lain. Penggunaan komponen yang telah dibuat tidak hanya mengurangi usaha pembuatan komponen tersebut, tetapi juga mengurangi kemungkinan kesalahan jika harus mengembangkan lagi dari awal. Keuntungannya adalah penghematan dari segi waktu, dan usaha yang akhirnya akan membawa kepada penghematan biaya pengembangan.

c. Resilience to change

Dunia nyata adalah sesuatu yang dinamis, perubahan akan selalu terjadi didalamnya. Program atau aplikasi yang memodelkan dunia nyata, diharapkan juga bisa bersifat dinamis pula.

d. Information hiding

Informasi dalam sebuah obyek sedapat mungkin disembunyikan dari luar kelas yang ada.Tujuannya adalah mengamankan data agar

hanya fungsi yang ada dalam kelas itu saja yang bisa membaca, mengubah dan memanipulasi data tersebut. Tetapi tetap disediakan sebuah cara (*interface*) agar obyek dari luar bisa mengakses dan mengubah data secara tidak langsung.

Konsep penyembunyian informasi ini bertujuan agar pihak luar yang membutuhkan layanan dari kelas tersebut hanya perlu untuk menerima data yang dibutuhkan saja, tanpa perlu mengetahui bagaimana sebenarnya cara kerja di dalam kelas tersebut.

e. Modularity of code

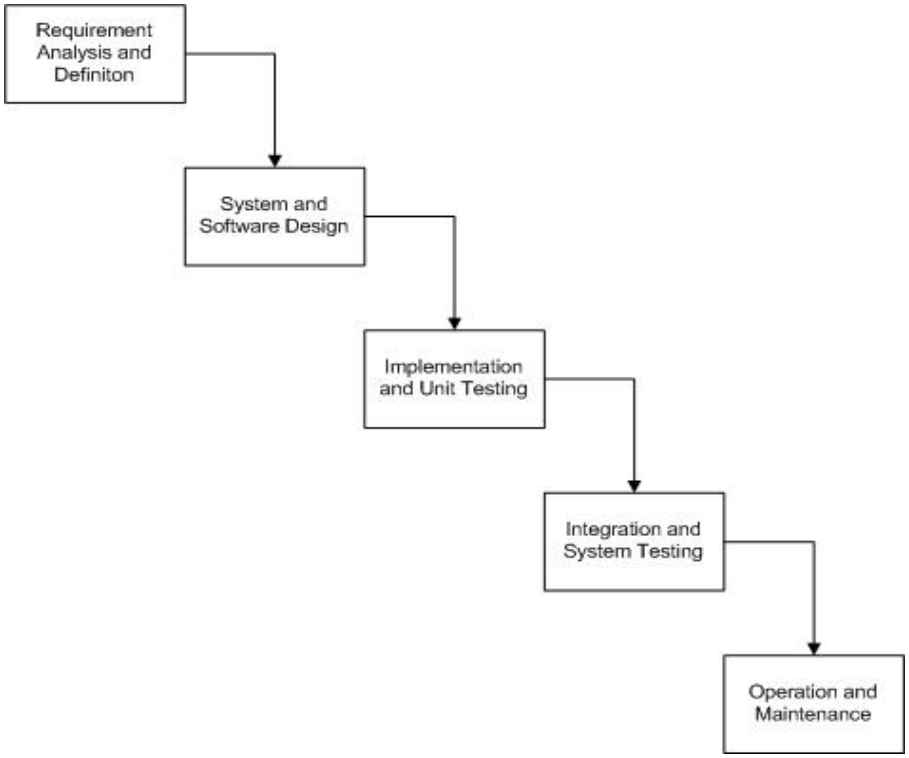
Salah satu keuntungan dari pemrograman berorientasi obyek adalah modularitas, yang berarti bahwa setiap obyek yang dibentuk dikelola secara terpisah dari obyek lainnya meskipun berasal dari sebuah kelas yang sama. Modifikasi terhadap sebuah obyek bisa dilakukan tanpa mempengaruhi fungsionalitas dari obyek yang lainnya.

3.2.3.2 Metode Pengembangan Sistem

Metode pendekatan dan pengembangan sistem yang digunakan dalam penelitian ini adalah Daur Hidup Perangkat Lunak (*System Development Life Cycle* / SDLC) Menurut Ade Hendraputra (2009 : 7), di dalam SDLC tahap-tahap yang prinsipal terhadap pemetaan model proses kedalam aktifitas pengembangan yang fundamental adalah sebagai berikut :

1. *Requirement Analysis and definition*
2. *System and Software Design*

3. *Implementation and unit testing*
4. *Integration and system Testing*
5. *Operation and maintenance*



Gambar 3.1 SDLC

Dalam metode SDLC ini, model proses pengembangan perangkat lunak yang digunakan ialah *Linear Sequential Model*. Linear sequential model (atau disebut juga "*classic life cycle*" atau "*waterfall model*") adalah metode pengembangan perangkat lunak dengan pendekatan sekuensial dengan cakupan aktivitas:

1. Pemodelan dan rekayasa sistem/informasi.

   Menetapkan kebutuhan untuk seluruh elemen sistem dan kemudian memilah mana yang untuk pengembangan perangkat lunak.

2. Analisis kebutuhan perangkat lunak

   Secara teknis pelaksanaan pekerjaan analisis kebutuhan perangkat lunak pada

   dasarnya terdiri dari urutan aktivitas:

   1) Mempelajari dan memahami persoalan

   Pada tahap ini, seorang analis mempelajari masalah yang ada pada

   perangkat lunak yang dikembangkan, sehingga dapat ditentukan

a. Siapa pemakai yang menggunakan perangkat lunak.
b. Dimana perangkat lunak akan digunakan .
c. Pekerjaan apa saja dari pemakai yang akan dibantu oleh perangkat lunak.
d. Apa saja cakupan dari pekerjaan tersebut, dan bagaimana mekanisme pelaksanaannya.
e. Apa yang menjadi kendala dilihat dari sisi teknologi yang digunakan atau dari sisi hukum dan standar.

   Cara yang digunakan oleh pengembang khususnya analis dalam memahami masalah perangkat lunak biasanya dilakukan :

a. Wawancara dengan pemakai
b. Observasi atau pengamatan lapangan
c. Kuesioner
d. Mempelajari referensi atau dokumen-dokumen yang digunakan, seperti dokumen hasil analisa dan perancangan perangkat lunak.

Hasil dari pemahaman masalah tersebut dapat digambarkan dengan model-model tertentu sesuai dengan jenis permasalahannya. Sebagai contoh jika masalah bisnis dapat digambarkan dengan *flowmap* atau *bussiness use case* untuk analisa berorientasi objek. Sedangkan untuk masalah matematika dapat digambarkan dengan graf.

2) Mengidentifikasi kebutuhan pemakai

Pada tahap identifikasi kebutuhan pemakai (*user requirement*) ini pada prakteknya menjadi satu pelaksanaannya dengan pemahaman masalah. Hanya saja substansi yang ditanyakan ada sedikit perbedaan, yaitu

a. fungsi apa yang diinginkan pada perangkat lunak.
b. data atau informasi apa saja yang akan diproses.
c. kelakuan sistem apa yang diharapkan.
d. antarmuka apa yang tersedia (*software interfaces, hardware interfaces, user interfaces,* dan *communication interfaces*)

Untuk menangkap kebutuhan dari pemakai dengan baik,terutama kesamaan persepsi. seorang analis membutuhkan

a. komunikasi dan brainstorming yang intensif dengan pelanggan.
b. pembuatan*prototype* perangkat lunak atau screenshoot.
c. Data atau dokumen yang lengkap.

3) Mendefinisikan kebutuhan perangkat lunak

Saat melakukan pengidentifikasian kebutuhan pemakai, informasi yang diperoleh masih belum terstruktur. Biasanya

pemakai akan mengungkapkan apa yang diinginkan dengan bahasa sehari-hari yang biasa mereka gunakan. Sebagi contoh, ungkapan kebutuhan pemakai dibagian akutansi. Kemudian pada tahap ini, kebutuhan pemakai yang belum terstruktur tersebut akan akan dianalisis, diklasifikasikan, dan diterjemahkan menjadi kebutuhan fungsional, antarmuka dan unjuk kerja perangkat lunak. Sebagai contoh, kebutuhan "data yang dimasukkan oleh bagian penjualan bisa langsung dijurnal" setelah dianalisis, diklasifikasikan dan diterjemahkan, mungkin akan menghasilkan pendefinisian kebutuhan sebagai berikut.

a. Kebutuhan fungsional
b. Kebutuhan antarmuka
c. Kebutuhan unjuk kerja

Kemudian kebutuhan tersebut akan dimodelkan atau digambarkan dengan teknik analisis dan alat bantu tertentu. Sebagai contoh kebutuhan fungsional dapat dimodelkan dengan menggunakan:

a. *Data flow* diagram,kamus data,dan spesifikasi proses jika menggunakan anlisis tertsruktur
b. *Use case* diagram dan skenario sistem jika menggunkan analisis berorientasi objek.

4) Membuat dokumen spesifikasi kebutuhan perangkat lunak (SKPL)

Semua kebutuhan yang telah didefinisikan selanjutnya dibuat dokumentasinya yaitu Spesifikasi Kebutuhan Perangkat Lunak (SKPL) atau *Software Requirement Specification* (SRS). Dokumen ini dibuat untuk menyatakan secara lengkap apa yang dapat dilakukan oleh perangkat lunak, termasuk deskripsi lengkap semua antarmuka yang akan digunakan.

5) Mengkaji ulang (*review*) kebutuhan

Proses untuk mengkaji ulang (validasi) kebutuhan apakah SKPL sudah konsisten, lengkap, dan sesuai dengan yang diinginkan oleh pemakai. Proses ini bisa dilakukan lebih dari satu kali. Dan sering kali akan muncul kebuthan-kebutuhan baru dari pemakai. Oleh karena itu, diperlukannya negosiasi antara pengembang dengan pelanggan sesuai dengan prinsip "*win win solution*" sampai kebutuhan tersebut disetujui oleh kedua belah pihak.

3. Perancangan

Tujuan dilakukannya perancangan oleh seorang *designer system* adalah:

a. Mendekomposisi sistem perangkat lunak menjadi komponen-komponennya.
b. Menentukan relasi antar komponen.
c. Menentukan mekanisme komunikasi antar komponen.
d. Menentukan antarmuka komponen.

e. Menjelaskan fungsionalitas masing-masing komponen.

4. Pembuatan kode

   Merupakan sekumpulan aktivitas di mana rancangan perangkat lunak yang telah dibuat pada tahap perancangan kemudian dikodekan ke dalam bentuk kode program dengan menggunakan bahasa pemrograman tertentu agar dapat dijalankan pada komputer.

   Fondasi dari aktivitas ini adalah pemrograman. Tools untuk membuat program disebut bahasa pemrograman. Programmer membuat program dengan panduan dokumentasi rancangan perangkat lunak, namun pada umumnya programmer juga memeriksa semua dokumen dari tahapan-tahapan

   sebelumnya (semisal SKPL) untuk memeriksa konsistensi dari dokumentasi-dokumentasi yang ada.

5. Pengujian

   Saat ini sudah banyak berkembang berbagai metode untuk pengujian perangkat lunak. Metode-metode tersebut memberikan pendekatan yang sistematik untuk pengujian perangkat lunak kepada pengembang. Selain itu, metode-metode tersebut memberikan mekanisme yang dapat membantu memastikan kelengkapan pengujian dan memberikan kemungkinan tertinggi untuk mengungkap kesalahan pada perangkat lunak.

Semua produk yang direkayasa dapat diuji dengan satu atau dua cara, yaitu:

a. Dengan mengetahui fungsi yang ditentukan untuk dilakukan oleh suatu produk, pengujian dapat dilakukan untuk memperlihatkan bahwa masing-masing fungsi beroperasi sepenuhnya dan pada waktu yang sama mencari kesalahan pada setiap fungsi.

b. Dengan mengetahui kerja internal suatu produk, maka pengujian dapat dilakukan untuk memastikan bahwa seluruh operasi internal bekerjasesuai spesifikasi dan semua komponen internal telah diamati dengan memadai.

Pendekatan pengujian pertama disebut sebagai pengujian *black-box* dan

pengujian kedua disebut sebagai pengujian *white-box*.

Pengujian *black-box* berkaitan dengan pengujian yang dilakukan pada

antarmuka perangkat lunak. Meskipun dirancang untuk mengungkap kesalahan, pengujian *black-box* digunakan untuk memperlihatkan bahwa fungsi-fungsi perangkat lunak dapat beroperasi, bahwa input diterima dengan baik dan output dihasilkan dengan tepat, dan integritas informasi eksternal (seperti file data) dipelihara. Pengujian *black-box* menguji beberapa aspek dasar suatu sistem dengan memperhatikan sedikit struktur logika internal perangkat lunak tersebut.

Pengujian *white-box* didasarkan pada pengamatan yang teliti terhadap detail

prosedural. Jalur-jalur logika yang melewati perangkat lunak diuji dengan memberikan kasus uji yang menguji serangkaian kondisi dan atau *loop*tertentu. Status program tersebut dapat diuji pada berbagai titik untuk menentukan apakah status yang diharapkan sesuai dengan status yang sebenarnya.

6. Pemeliharaan

   Pemeliharaan perangkat lunak merupakan proses memodifikasi sistem perangkat lunak atau komponennya setelah penggunaan oleh konsumen untuk memperbaiki kerusakan, meningkatkan kinerja, manfaat, atau kualitas lainnya atau untuk menyesuaikan sistem perangkat lunak dengan lingkungan yang berubah. Definisi ini menegaskan bahwa proses pemeliharaan perangkat lunak merupakan proses yang bersifat *post-delivery*, artinya dilakukan setelah sistem perangkat lunak digunakan oleh konsumen. Aktivitas ini dimulai sejak sistem dilepaskan ke pasaran dan oleh konsumen dan mencakup semua aktivitas yang menjaga operasional sistem dan kesesuaian dengan kebutuhan pengguna.

3.2.3.3 Alat Bantu Analisis dan perancangan

Alat bantu analisis dan perancangan yang digunakan yaitu:

1. *Entitiy Relationship* Diagram (ERD)

   ERD adalah salah satu pemodelan basis data konseptual yang menggambarkan basis data dalam bentuk entitas-entitas dan relasi yang terjadi di antara entitas-entitas yang ada. Entitas diartikan sebagai objek di dunia nyata yang bias dibedakan dengan objek yang lain. Relasi diartikan sebagai hubungan yang terjadi diantara satu entitas dengan entitas lain.

2. *Usecase* Diagram

   *Use case*menggambarkan *external view* dari sistem yang akan kita buat modelnya. (Pooley,2003:15) mengatakan bahwa model *use case* dapat di jabarkan dalam diagram *use case* , tetapi yang perlu di ingat, diagram tidak identik dengan model karena model lebih luas dari diagram.

3. *Sequence* Diagram

   (Fowler,2005:81) menjelaskan bahwa *sequence* diagram adalah diagram interaksi yang menunjukkan interaksi dengan menampilkan setiap partisipan dengan garis alir secara vertikal dan pengurutan pesan dari atas ke bawah.

4. *Activiy* Diagram

   (Fowler,2005:163) menjelaskan bahwa *Activity*diagram adalah teknik untuk menggambarkan logika procedural,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 behavioral paralel.

5. *Class* Diagram

(Fowler,2005:163) Menjelaskan bahwa *class* diagram mendeskripsikan jenis-jenis objek dalam sistem dan berbagai macam hubungan statis yang terdapat di antara mereka.*Class*diagram juga menunjukkan property dan operasi sebuah kelas dan batasan-batasan yang terdapat dalam hubungan objek tersebut.

3.2.4 Pengujian Aplikasi

Metode pengujian perangkat lunak yang digunakan adalah metode black-box testing. Tehnik black-box testing berfokus pada domain informasi dari perangkat lunak, dengan melakukan test case dengan menpartisi domain input dari suatu program dengan cara yang memberikan cakupan pengujian yang mendalam. Metode ujicoba *blackbox* memfokuskan pada keperluan fungsional dari software. Karena itu ujicobablackbox memungkinkan pengembang software untuk membuat himpunan kondisi input yang akan melatihseluruh syarat-syarat fungsional suatu program. Ujicoba blackbox bukan merupakan alternatif dari ujicoba whitebox, tetapi merupakan pendekatan yang melengkapi untuk menemukan kesalahan lainnya, selainmenggunakan metode whitebox. Ujicoba blackbox berusaha untuk menemukan kesalahan dalam beberapa kategori, diantaranya :

1. Fungsi-fungsi yang salah atau hilang
2. Kesalahan interface

3. Kesalahan dalam struktur data atau akses *database* eksternal

4. Kesalahan performa

5. kesalahan inisialisasi dan terminasi

Tidak seperti metode whitebox yang dilaksanakan diawal proses, ujicoba blackbox diaplikasikan dibeberapa tahapan berikutnya. Karena ujicoba blackbox dengan sengaja mengabaikan struktur kontrol, sehingga perhatiannya difokuskan pada informasi *domain*. Ujicoba didesain untuk dapat menjawab pertanyaan – pertanyaanberikut :

1. Bagaimana validitas fungsionalnya diuji?

2. Jenis input seperti apa yang akan menghasilkan kasus uji yang baik ?

3. Apakah sistem secara khusus sensitif terhadap nilai input tertentu ?

4. Bagaimana batasan-batasan kelas data diisolasi?

5. Berapa rasio data dan jumlah data yang dapat ditoleransi oleh sistem?

6. Apa akibat yang akan timbul dari kombinasi spesifik data pada operasi sistem?

Dengan mengaplikasikan ujicoba blackbox, diharapkan dapat menghasilkan sekumpulan kasus uji yang memenuhi kriteria berikut :

1. kasus uji yang berkurang, jika jumlahnya lebih dari 1, maka jumlah dari ujikasus tambahan harus didesain untuk mencapai ujicoba yang cukup beralasan

2. Kasus uji yang memberitahukan sesuatu tentang keberadaan atau tidaknya suatu jenis kesalahan, daripada kesalahan yang terhubung hanya dengan suatu ujicoba yang spesifik