BAB III

OBJEK DAN METODE PENELITIAN

3.1 OBJEK PENELITIAN

3.1.1 Sejarah Singkat Perusahaan

ABC Putra Mandiri atau biasa disebut APM merupakan perusahaan yang bergerak dibidang jasa rental mobil yang berada di Bandung. Berdiri pada tahun 1991, APM memulai bisnisnya dengan hanya memiliki 2 unit mobil. Walaupun sempat terkena dampak dari krisis ekonomi pada tahun 1997, tetapi APM dapat berbertahan dan berkembang sampai sekarang. Saat ini APM telah memiliki 22 unit kendaraan mobil berbagai type dan memiliki 15 karyawan untuk menjalankan kegiatan operasionalnya.

3.1.2 Visi dan Misi Perushaan

“Menfaatkan dunia transportasi sebagai bagian kebutuhan masyarakat”, merupakan visi utama rental mobil PT ABC Putra Mandiri Bandung.

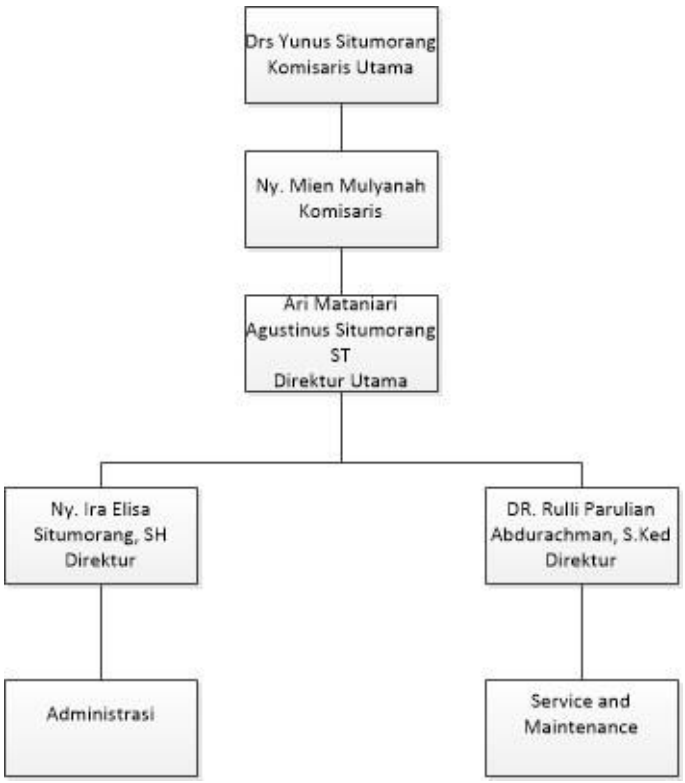
Misi PT ABC Putra Mandiri adalah “Agar tercipta suatu usaha transportasi yang representative bagi masyarakat banyak”.

3.1.3 Struktur Organisasi Perusahaan

Secara umum struktur organisasi merupakan suatu kerangka yang memperlihatkan tugas dan tanggung jawab untuk mencapai tujuan organisasi,

hubungan antar fungsi dan wewenang atas pekerjaan yang dibebankan kepadanya. Dalam suatu badan usaha baik instansi pemerintahan maupun swasta struktur organisasi sangat penting karena baiknya suatu manajemen ditunjang dari strukturorganisasi yang baik pula.

Untuk lebih jelasnya struktur organisasi PT. ABC Putra Mandiri dapat dilihat pada gambar :



Gambar 3.1. Struktur Organisasi PT. ABC Putra Mandiri

3.1.4 Deskripsi Tugas

1. Komisaris Utama

a. Memerintah organisasi dengan menetapkan kebijakan-kebijakan dan tujuan-tujuan luas dari perusahaan tersebut

b. Memastikan keberadaan dan kecukupan sumber keuangan
c. Mengesahkan anggaran tahunan
d. Menentukan gaji dan kompensasi mereka sendiri

2. Komisaris

a. Memeberikan nasihat kepada direktur utama dalam melaksanakan pengurusan perusahaan.
b. Melakukan pengawasan atas jalannya usaha pada perusahaan.
c. Bertindak sebagai wakil pemegang saham melakukan pelaksanaan dari setiap kebijaksanaan yang telah digariskan atau dikeluarkan.

3. Direktur Utama

a. Mengkoordinasikan dan mengendalikan kegiatan-kegiatan dibidang administrasi keuangan, kepegawaian dan kesekretariatan.
b. Mengkoordinasikan dan mengendalikan kegiatan pengadaan dan peralatan perlengkapan.
c. Merencanakan dan mengembangkan sumber-sumber pendapatan serta pembelanjaan dan kekayaan perusahaan.
d. Mengendalikan uang pendapatan.

4. Direktur

Mengurus dan mengelola PT untuk kepentingan PT yang sesuai dengan maksud dan tujuan PT, menjalankan kepengurusan PT sesuai dengan

kebijakan yang tepat (keahlian, peluang, dan kelaziman usaha) yang ditentukan dalam UU Perseroan Terbatas dan anggaran dasar PT.

5. Administrasi

a. Memberikan Informasi atau Tata Cara didalam penyewaan mobil.
b. Menentukan Tarif atau Harga Sewa dan Denda serta peraturan atau ketentuan yang berlaku didalam tata cara penyewaan mobil selain itu dia mempunyai tanggungg jawab terhadap tugasnya dan kepada Pimpinan.
c. Menerima pembayaran sewa mobil sesuai dengan merk atau type mobil dari pelanggan
d. Menerima pembayaran denda sewa.

6. Service and Maintenance

Service And Maintenace, bertugas melakukan pemeliharaan rutin seperti cek oli, cek mesin mobil, dan perbaikan – perbaikan bila ada kerusakan - kerusakan.

3.2 Metode Penelitian

3.2.1 Desain Penelitian

Desain penelitian ini termasuk kedalam penelitian yang bersifat deskriptif. Penelitian deskriptif adalah penelitian yang bertujuan untuk memperoleh ciri-ciri variabel, dimana dalam penelitian ini adalah untuk memperoleh gambaran tentang

kinerja program yang dirancang dan diimplementasikan kepada pengguna (user) dalam perusahaan.

Penelitian ini dengan mengambil kasus di Rental Mobil di PT. ABC Putra Mandiri unit pengamatan pada bagian Administrasi.

3.2.2 Jenis dan Metode Pengumpulan Data

Metode pengumpulan data adalah cara-cara yang dapat digunakan peneliti untuk mengumpulkan data. Pengumpulan data disini bertujuan untuk mendapatkan informasi tentang proses rental mobil di PT. ABC Putra Mandiri.

3.2.2.1 Sumber Data Primer

Data primer merupakan Data yang diperoleh langsung dari unit pengamatan / responden peneliti. Teknik / metode pengumpulan data primer meliputi :

1. Observasi

   Pengumpulan data melalui pengamatan & pencatatan terhadap obyek penelitian pada bagian Kasir di Rentalm Mobil CV. Tasya Lacaden.

2. Wawancara / Interview

   Pengumpulan data melalui tatap muka dan tanya jawab langsung dengan responden. Peneliti melakukan tanya jawab dengan Rika yang bekerja di Rental Mobil CV. Tasya Lacaden pada bagian Kasir

3.2.2.2 Sumber Data Sekunder

Data sekunder adalah data yang digunakan untuk mendukung data primer, merupakan jenis data yang sudah diolah terlebih dahulu oleh pihak pertama atau peneliti. Teknik/metode pengumpulan data sekunder meliputi sistem dan teknik perancangan basis data dari sumber-sumber yang berhubungan dengan sistem yang akan dikembangkan, seperti pengumpulan bukti pembayaran sewa mobil dan pengumpulan data pelanggan.

3.2.2 Metode Pendekatan dan Pengembangan Sistem

Dalam metode pendekatan dan pengembangan sistem akan dijelaskan mengenai metode pendekatan sistem, metode pengembangan sistem dan alat bantu analisis pengembangan sistem.

3.2.3.1 Metode Pendekatan Sistem

Dalam penelitian ini metode yang digunakan yaitu menggunakan pendekatan berorientasi objek, dengan alat bantu (tools) seperti use case diagram, class diagram, statechart diagram, activity diagram, sequence diagram, collaboration diagram, component diagram dan deployment diagram.

3.2.3.2 Metode Pengembangan Sistem

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 ketidakserasian 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.

Tahapan-tahapan dalam Prototyping adalah sebagai berikut:

1. Pengumpulan kebutuhan

   Pelanggan dan pengembang bersama-sama mendefinisikan format seluruh perangkat lunak, mengidentifikasikan semua kebutuhan, dan garis besar sistem yang akan dibuat.

2. Membangun prototyping

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

3. Evaluasi protoptyping

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

4. Mengkodekan sistem

   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 dites dahulu sebelum digunakan.

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.

3.2.3.3 Alat Bantu Analisis dan Perancangan

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.

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.

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*.

Sebuah *use case* juga dapat meng-*extend use case* lain dengan *behaviour*-nya sendiri.

Sementara hubungan generalisasi antar *use case* menunjukkan bahwa *use case* yang satu merupakan spesialisasi dari yang lain.

http://setia.staff.gunadarma.ac.id/Downloads/files/6077/Modul_UML.pdf

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 :

a. Nama (dan stereotype)
b. Atribut
c. Metoda

Class dapat merupakan implementasi dari sebuah *interface*, yaitu *class* abstrak yang hanya memiliki metoda. *Interface* tidak dapat langsung diinstansiasikan, tetapi harus diimplementasikan dahulu menjadi sebuah *class*. Dengan demikian *interface* mendukung resolusi metoda pada saat *run-time*.

Sesuai dengan perkembangan *class* model, *class* dapat dikelompokkan menjadi *package*. Kita juga dapat membuat diagram yang terdiri atas *package*.

http://setia.staff.gunadarma.ac.id/Downloads/files/6077/Modul_UML.pdf

3. Statechart diagram

Statechart diagram menggambarkan transisi dan perubahan keadaan (dari satu *state* ke *state* lainnya) suatu objek pada sistem sebagai akibat dari *stimuli* yang diterima. Pada umumnya *statechart diagram* menggambarkan *class* tertentu (satu *class* dapat memiliki lebih dari satu *statechart diagram*).

Dalam UML, *state* digambarkan berbentuk segiempat dengan sudut membulat dan memiliki nama sesuai kondisinya saat itu. Transisi antar *state* umumnya memiliki kondisi *guard* yang merupakan syarat terjadinya transisi yang bersangkutan, dituliskan dalam kurung siku. *Action* yang dilakukan sebagai akibat dari *event* tertentu dituliskan dengan diawali garis miring.

Titik awal dan akhir digambarkan berbentuk lingkaran berwarna penuh dan berwarna setengah.

http://setia.staff.gunadarma.ac.id/Downloads/files/6077/Modul_UML.pdf'

4. 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.

Sama seperti *state*, standar UML menggunakan segiempat dengan sudut membulat untuk menggambarkan aktivitas. *Decision* digunakan untuk menggambarkan behaviour pada kondisi tertentu. Untuk mengilustrasikan proses-proses paralel (*fork* dan *join*) digunakan titik sinkronisasi yang dapat berupa titik, garis horizontal atau vertikal.

Activity diagram dapat dibagi menjadi beberapa *object swimlane* untuk menggambarkan objek mana yang bertanggung jawab untuk aktivitas tertentu.

http://setia.staff.gunadarma.ac.id/Downloads/files/6077/Modul_UML.pdf

5. 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*.

http://setia.staff.gunadarma.ac.id/Downloads/files/6077/Modul_UML.pdf

6. Collaboration diagram

Collaboration diagram juga menggambarkan interaksi antar objek seperti *sequence diagram*, tetapi lebih menekankan pada peran masing-masing objek dan bukan pada waktu penyampaian *message*.

Setiap *message* memiliki *sequence number*, di mana *message* dari level tertinggi memiliki nomor 1. Messages dari level yang sama memiliki prefiks yang sama.

http://setia.staff.gunadarma.ac.id/Downloads/files/6077/Modul_UML.pdf

7. Component diagram

Component diagram menggambarkan struktur dan hubungan antar komponen piranti lunak, termasuk ketergantungan (*dependency*) di antaranya.

Komponen piranti lunak adalah modul berisi *code*, baik berisi *source code* maupun *binary code*, baik *library* maupun *executable*, baik yang muncul pada *compile time*, *link time*, maupun *run time*. Umumnya komponen terbentuk dari beberapa *class* dan/atau *package*, tapi dapat juga dari komponen-komponen yang lebih kecil.

Komponen dapat juga berupa *interface*, yaitu kumpulan layanan yang disediakan sebuah komponen untuk komponen lain.

http://setia.staff.gunadarma.ac.id/Downloads/files/6077/Modul_UML.pdf

8. Deployment gram

Deployment/physical diagram menggambarkan detail bagaimana komponen di-*deploy* dalam infrastruktur sistem, di mana komponen akan terletak (pada mesin, server atau piranti keras apa), bagaimana kemampuan jaringan pada lokasi tersebut, spesifikasi server, dan hal-hal lain yang bersifat fisikal

Sebuah *node* adalah server, *workstation*, atau piranti keras lain yang digunakan untuk men-*deploy* komponen dalam lingkungan sebenarnya. Hubungan antar *node* (misalnya TCP/IP) dan *requirement* dapat juga didefinisikan dalam diagram ini.

http://setia.staff.gunadarma.ac.id/Downloads/files/6077/Modul_UML.pdf

3.2.4 Pengujian Software

Software testing adalah investigasi yang dilakukan kepada software untuk mengetahui kualitas suatu produk. dilakukan untuk mengeliminasi defect yang terjadi setelah software diluncurkan, pengujian software sendiri tidak akan pernah secara penuh mengenali defect pada suatu software.black box testing dan white box testing adalah metode yang dikenal untuk menguji software.

Black box testing mengambil perspektif dari luar untuk melakukan testing, tes ini bisa berupa fungsional test atau non-fungsional, dilakukan dengan cara memasukkan input dengan benar maupun salah, kemudian menguji apakah outputnya sesuai dengan spesifikasi atau tidak, hal ini dilakukan tanpa harus tahu apa yang terjadi didalam mengolah input untuk mengasilkan output. black box testing popular karena kesederhanaan metodenya.

White box testing, atau biasa dikenal juga dengan nama glass box testing menguji software dari dalam, dengan melihat struktur internal software. untuk itu diperlukan kemampuan programing untuk mengidentifikasi bagian dalam suatu software.

Metode pengujian adalah cara atau teknik untuk menguji perangkat lunak, mempunyai mekanisme untuk menentukan data uji yang dapat menguji perangkat lunak secara lengkap dan mempunyai kemungkinan tinggi untuk menemukan kesalahan

Pengujian *black box* adalah pengujian aspek fundamental sistem tanpa memperhatikan struktur logika internal perangkat lunak. Metode ini digunakan untuk mengetahui apakah perangkat lunak berfungsi dengan benar. Pengujian black box merupakan metode peracangan data uji yang didasarkan pada spesifikasi perangkat lunak. Data uji dibangkitkan, dieksekusi pada perangkat lunak dan kemudian keluaran dari perangkat lunak dicek apakah telah sesuai dengan yang diharapkan.

Pengujian black box merupakan pendekatan komplementer dari teknik white box, karena pengujian *black box* diharapkan mampu mengungkap kelas kesalahan yang lebih luas dibandingkan teknik white box. Pengujian *black box* berfokus pada pengujian persyaratan fungsional perangkat lunak, untuk mendapatkan serangkaian kondisi input yang sesuai dengan persyaratan fungsional suatu program.

Pengujian *black box* berusaha menemukan kesalahan dalam kategori: fungsi-fungsi yang tidak benar atau hilang

1. kesalahan interface.
2. kesalahan dalam struktur data atau akses database eksternal.
3. kesalahan kinerja.
4. inisialisasi dan kesalahan terminasi.

http://elib.unikom.ac.id/files/disk1/585/jbptunikompp-gdl-reizamuham-29208-8-unikom_r-i.pdf