BAB III

OBJEK DAN METODE PENELITIAN

3.1 Objek Penelitian

Adapun Pembasahasan mengenai Objek penelitian dapat dilihat pada penjelasan di bawah ini .

3.1.1 Sejarah Sejarah Singkat Hotel

Sekitar tahun 2005 tepatnya tanggal 29 maret , didirikan sebuah tempat peristirahatan berbentuk penginapan bungalow/hotel yang di beri nama Bungalow dan Restaurant Kampoeng Strawberry. Terletak di Jl. Raya Ciwidey-Rancabali Km.07 Alam Endah Ciwidey Bandung, dengan lokasi bangunan terletak di sekitar perkebunan strawberry dan pemukiman pemduduk dengan kondisi lingkungan yang masih asri. Pendirinya adalah alm.H.Panji Setia Amiarsa,bungalow ini terdiri dari 39 bungalow dengan berbagai tipe. Selain itu Kampoeng Strawberry juga memiliki ruang rapat (meeting),restoran, ruang karaoke, kolam renang dan juga tempat parkir yang luas.

3.1.2 Visi Misi

Adapun Visi Misi Hotel dan Resto Kampoeng Strawberry adalah :

Visi

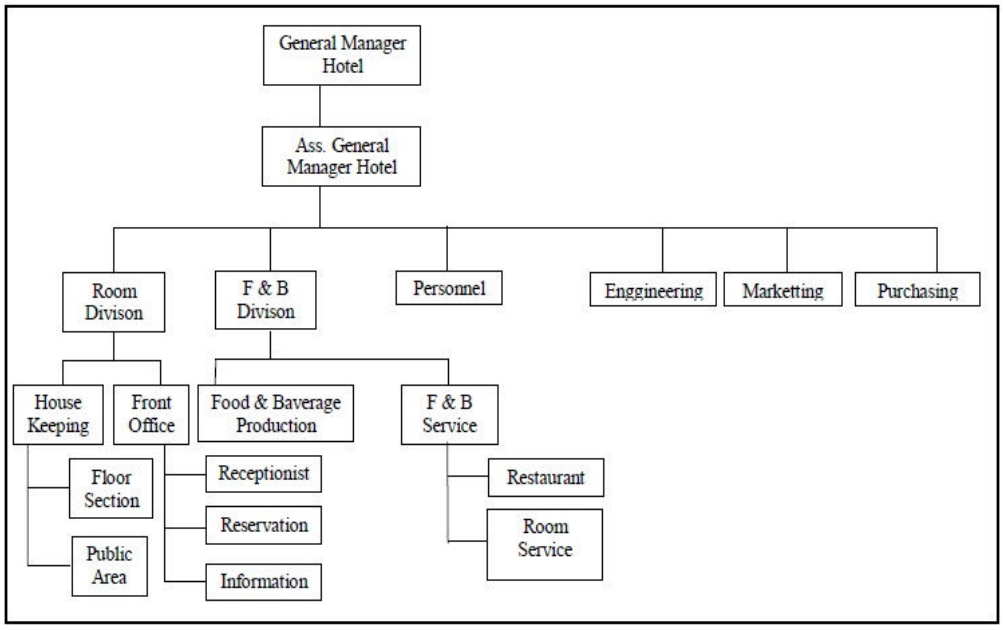
Visi Hotel dan Resto Kampoeng Strawberry : “ Menjadi Perusahaan Yang Besar dan Inovatif serta Mampu Berkembang Dengan Mengedepankan Kualitas Pelayanan Yang Memuaskan”.

Misi

1. Hotel dan Resto Kampoeng Strawberry menjalankan usaha yang berorientasi pada pelayanan terbaik kepada konsumen sehingga mengahasilkan sebuah nilai kepuasan.
2. Mengembangkan usaha yang memiliki daya saing yang kuat di pasaran.
3. Menjalankan kegiatan usaha yang berwawasan.

3.1.3 Struktur Organisasi Perusahaan

Fungsi dari struktur tingkat jabatan pegawai hotel yaitu untuk menggambarkan secara menyeluruh fungsi dari suatu jabatan beserta tingkatan tanggung jawab dan wewenang dari setiap pelaksana pekerjaan. Dimana dalam pelaksanaanya manager (*General Manager*) sebagai orang yang mempunyai garis komando, General Manager membawahi seluruh bagian yang dibantu oleh Asissten Manager (*Assistant Manager*). Adapun struktur tingkat jabatan pegawai Hotel dan Resto Kampoeng Strawberry adalah sebagai berikut :



Gambar 3.1 Struktur Organisasi

Sumber: Dokumentasi Kampoeng Strawberry Bungalow & Restoran

3.1.4 Deskripsi Tugas

1.General Manager

a. Mengkordinir bidang kerja seluruh *staff*.

b. mengawasi dan membuat perencanaan mengenai perubahan aset,gardening termasuk budgetnya.

c. Mengadakan pertemuan atau meeting rutin dengan menelaah hasil kerja yang telah dilakukan (evaluasi kerja).

2. Ass General Manager Hotel

Ass. General Manager pada dasarnya turut memastikan bahwa seluruh aspek dalam operasional Hotel dan Resto Kampoeng Strawberry sudah berjalan sesuai dengan standar yang diterapkan oleh General Manager. Kedua jabatan ini memiliki subordinat antara lain :

1. Room Division yang terdiri dari

a. Housekeeping

Yang bertugas menjaga dan/atau kebersihan,kerapihan dan kelengkapan kamar-kamar tamu. Housekeeping juga terdiri dari beberapa subordinat antara lain:

a) Floor Section

Bagian Kamar Tamu (*Room Supervisor*) yang dibantu oleh *room boy* yang bertugas melaksanakan pemebersihan,merapikan dan melengkapi kebutuhan tamu di hotel (sabun mandi,handuk,dan keperluan-keperluan lain).

b) Public Area

Bagian Ruangan Tamu (*Public Area Supervisor*) yang dibantu oleh *Houseman* yang bertugas melaksanakan pembersihan, merapikan dan melengkapi kebutuhan ruangan umum.

b. Front Office

Menjaga dan memelihara kebersihan, kerapian dan kelengkapan kebutuhan ruangan umum seperti ruangan bagian depan *receptionist* dan *restaurant.*

Front office juga terdiri dari beberapa subordinat antara lain:

a) *Receptionist*, yang bertugas sebagai penerima tamu serta menerima dan menjawab telepon serta mencatat pesan-pesan lewat telepon.

b) *Reservation,* yang bertugas menyediakan atau mempersiapkan tempat.

c) *Information,* yang bertugas membertitahukan kepada tamu hotel tentang apa saja fasilitas-fasilitas yang ada di hotel.

2. Food and Baverage Division

a) *Food Production*, yang bertugas melakukan pengendalian produksi makanan seperti : perkiraan produksi,putaran menu,catatan penjualan makanan, resep makanan dan porsi makanan.

b) *Baverage Production,* yang bertugas melakukan pengendalian produksi minuman.

c) *Food and Baverage Service*

1) *Restaurant*

Bertugas menyiapkan dan melayani pelanggan hotel maupun tamu yang berkunjung.

2) *Room Service* (Pelayanan Kamar)

Bertugas membersihkan, merawat dan mengantarkan pesanan makanan dan minuman kepada pelanggan hotel yang telah memesan di menu restaurant.

3. Personnel (Personalia)

Mengorganisasi seluruh data-data mengenai pekerja-pekerja yang ada di Hotel dan Resto Kampoeng Strawberry.

4. Enggineering (Tekhnik)

Bertugas apabila ada kerusakan bagian-bagian hotel dan menyiapkan beberapa laporan-laporan yang diperlukan bagi Manager.

5. Marketting (Pemasaran)

Yang bertugas memuaskan kebutuhan , keinginan dan tujuan-tujuan pelanggan ataupun tamu sehingga akan dapat memajukan kenyamanan Hotel dan Resto Kampoeng Strawberry.

6. Purchasing (Pembelian)

Bagian ini memilki kewajiban untuk melakukan pembelian bahan/barang yang diperlukan untuk operasionalisasi pelayanan Hotel dan Resto Kampoeng Strawberry.

3.2 Metodologi Yang Digunakan

Metode penelitian adalah suatu teknik atau cara mengumpulkan, memperoleh data, baik berupa data primer maupun sekunder yang digunakan untuk keperluan menyusun suatu karya ilmiah dan kemudian menganalisa faktor-faktor yang berhubungan dengan pokok-pokok permasalahan sehingga terdapat suatu kebenaran data-data yang akan diperoleh. Adapun pengertian metode penelitian menurut Sugiyono (2010 : 2), diantaranya adalah sebagai berikut : *" Metode penelitian pada dasarnya merupakan cara ilmiah untuk mendapatkan data dengan tujuan dan kegunaan tertentu"*.

3.2.1 Desain Penelitian

Dalam melakukan suatu penelitian sangat perlu dilakukan perencanaan dan perancangan penelitian, agar penelitian yang dilakukan dapat berjalan dengan baik dan sistematis.

Desain penelitian menurut Moh. Nazir (2003 : 84) memaparkan bahwa "Desain penelitian adalah semua proses yang diperlukan dalam perencanaan dan pelaksanaan penelitian".

Dari definisi di atas maka dapat dikatakan bahwa desain penelitian merupakan semua proses penelitian yang dilakukan oleh penulis dalam melaksanakan penelitian mulai dari perencanaan sampai dengan pelaksanaan penelitian yang dilakukan pada waktu tertentu.

3.2.2 Jenis dan Metode Pengumpulan Data

Metode pengumpulan data adalah penelitian lapangan (*field Research*), dilakukan dengan cara mengadakan peninjauan langsung pada instansi yang menjadi objek untuk mendapatkan data primer (data yang diperoleh dari Hotel dan Resto Kampoeng Strawberry) dan data sekunder.

3.2.2.1 Sumber Data Primer (Observasi)

Data primer merupakan data yang diambil secara langsung, data ini diperoleh dari kegiatan observasi yaitu pengamatan langsung pada objek penelitian dan mengadakan wawancara dengan pihak terlibat.

a. Observasi

Observasi adalah teknik pengumpulan data dengan cara pengamatan dan pencatatan terhadap obyek yang sedang diteliti. Penelitian ini menggunakan metode observasi dalam pengumpulan data dengan melihat secara detail proses-proses yang terjadi pada bagian *front office* baik itu proses reservasi maupun pemesanan kamar sehingga data yang diperoleh dapat dipertanggung jawabkan.

Observasi dilakukan pada bagian *room division* tepatnya pada bagian *front office.*

b. Wawancara

Wawancara merupakan salah satu pengumpulan data dengan cara tatap muka dan tanya jawab langsung terhadap pihak yang bersangkutan mengenai masalah-masalah yang terjadi sehingga mendapatkan masukan dalam peracancangan sistem. Wawancara

dilakukan kepada Hj. Anita Purwitasari sebagai Ass General Manager. Adapun materi pertanyaan yang diberikan kepada beliau ialah : awal mula terberntuknya Hotel dan Resto Kampoeng Strawberry, jumlah bangunan yang ada , visi dan misi perusahaan dan juga kendala yang terjadi pada saat ini yang mempengaruhi perkembangan perusahaan.

3.2.3 Metoda Pendekatan Sistem dan Pengembangan Sistem

3.2.3.1 Metode Pendekatan Sistem

Dalam melakukan penelitian ini, penulis menggunakan pendekatan sistem prototipe. Dimana di dalamnya terdapat identifikasi komponen – komponen yang akan dirancang secara rinci yang bertujuan untuk memberikan gambaran kepada pengguna atau user mengenai sistem.

Adapun tahapan perancangan adalah agar pada akhir pengembangan perangkat lunak dapat memenuhi kebutuhan *user*, efektif, efisien, mudah digunakan,

Tahapan perancangan:

1. Survey Objek Penelitian
2. Pengumpulan Data
3. Mengidentifikasi Kebutuhan Sistem
4. Perancangan Prototype Sistem
5. Pengujian Prototype Sistem
6. Implementasi Sistem

3.2.3.1.1 OOP (Object Oriented Programing)

OOP adalah konsep pemograman yang difokuskan pada penciptaan kelas yang merupkan abstraksi/blueprint/prototype dari suatu objek. Kelas ini harus mengandung sifat (data) dan tingkah laku (method) umum yang dimililiki oleh objek-objek yang kelak akan dibuat (diinstansiasi). Data dan method merupakan anggota (member) dari suatu kelas. Benny Hermawan,halaman 35,2004).

Dalam konsep OOP, objek diciptakan atau diinstansiasi dari kelas dengan mengisikan nilai parameter data dan memanfaatkan method yang telah dideskripsikan oleh kelas tersebut. Dengan demikian,kelas juga merupakan template/prototype dari suatu objek. Sama seperti ketika anda hendak membuat lingkaran ,kotak ,atau flow diagram dengan menggunakan drawing tools seperti visio. Anda hanya perlu mengakses suatu template gambar,klik dan drag objek yang diinginkan, kemudian melakukan adjustment (penyesuaian) ukuran. (Benny Hermawan,halaman 32,2004).

Empat konsep yang dimiliki oleh pemograman yang berorientasi pada objek adalah :

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

3.2.3.1.1.1 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.

3.2.3.1.1.2 Abstraction

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

3.2.3.1.1.3 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.

3.2.3.1.1.4 Polymorphism

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

3.2.3.1.2 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. *statechart diagram*
4. *activity diagram*
5. *sequence diagram*
6. *collaboration diagram*
7. *component diagram*
8. *deployment diagram*

3.2.3.1.2.1 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 bagiandari 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, sehinggaduplikasi fungsionalitas dapat dihindari dengan cara menarik keluar fungsionalitas yang common.
- Sebuah use case juga dapat meng-extend use case lain dengan behavior- nya sendiri.
- Sementara hubungan generalisasi antar use case menunjukkan bahwa use caseyang satu merupakan spesialisasi dari yang lain



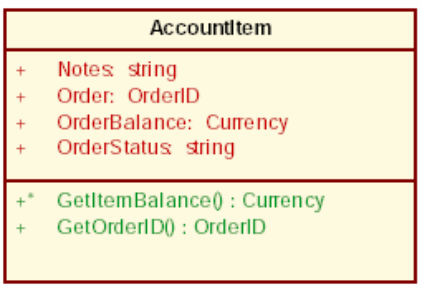
Gambar 2.1. *Use Case Diagram*

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

3.2.3.1.2.2 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 :

1. Nama (dan stereotype)
2. Atribut
3. 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.

3.2.3.1.2.3 Statechhart Diagram

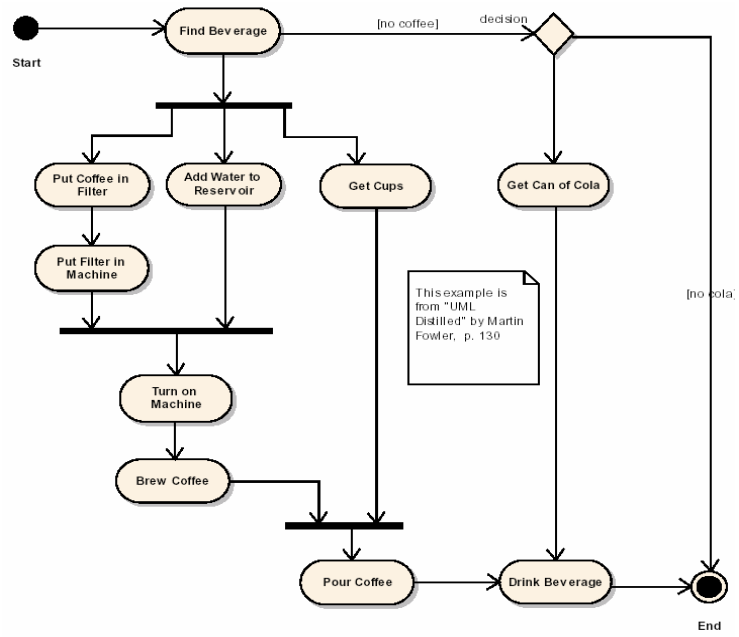
Statechart diagram menggambarkan transisi dan perubahan keadaan (dari satu *state* ke *state* lainnya) suatu objek pada sistem sebagai akibat dari *stimuli* yang diterima. Pada umumnya *statechart diagram* menggambarkan *class* tertentu (satu *class* dapat memiliki lebih dari satu *statechart diagram*).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

3.2.3.1.2.4 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.



Gambar 3.3 Contoh Activiy Diagram

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

3.2.3.1.2.5 Sequence Diagram

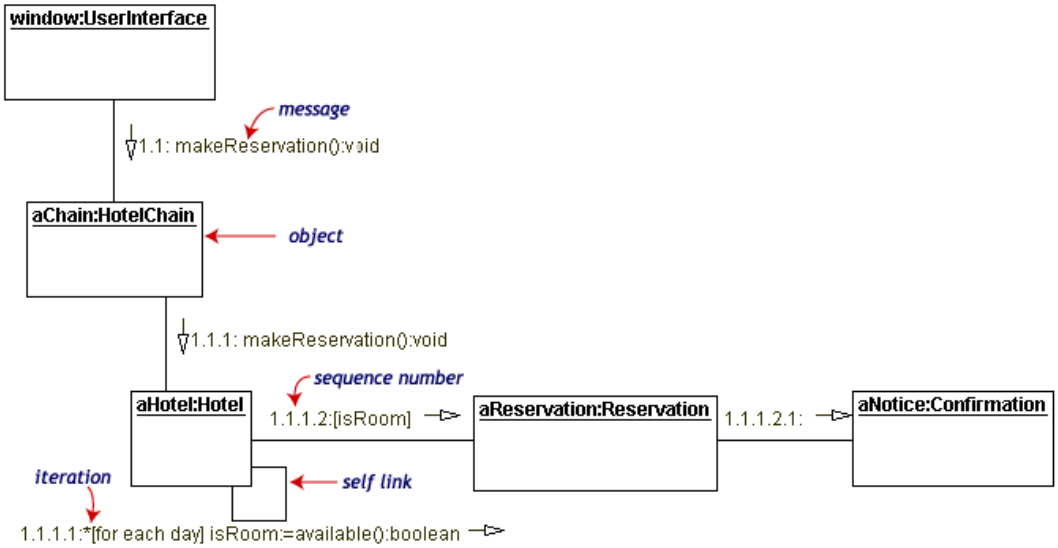
Sequence diagram menggambarkan interaksi antar objek di dalam dan di sekitar sistem (termasuk pengguna, *display*, dan sebagainya) berupa *message* yang digambarkan terhadap waktu. *Sequence diagram* terdiri atar dimensi vertikal (waktu) dan dimensi horizontal (objek-objek yang terkait).

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*.

3.2.3.1.2.6 Collaboration Diagram

Collaboration diagram j uga 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.

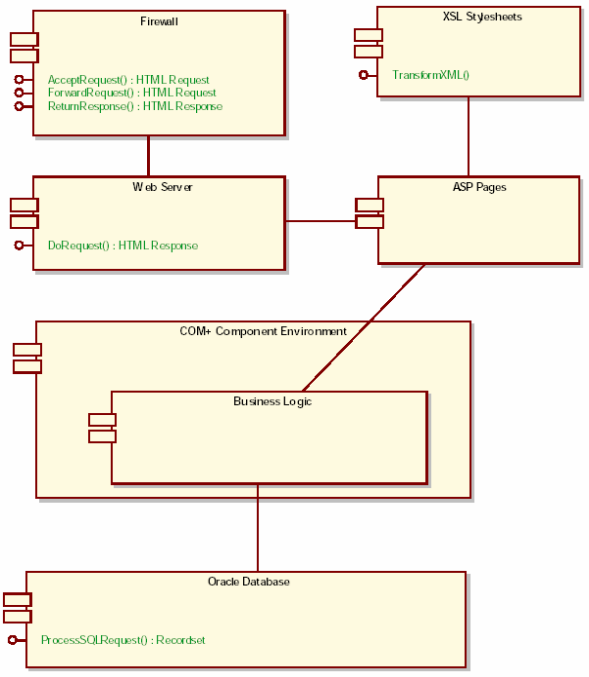


Gambar 3.4 Contoh Colaboration Diagram

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

3.2.3.1.2.7 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.

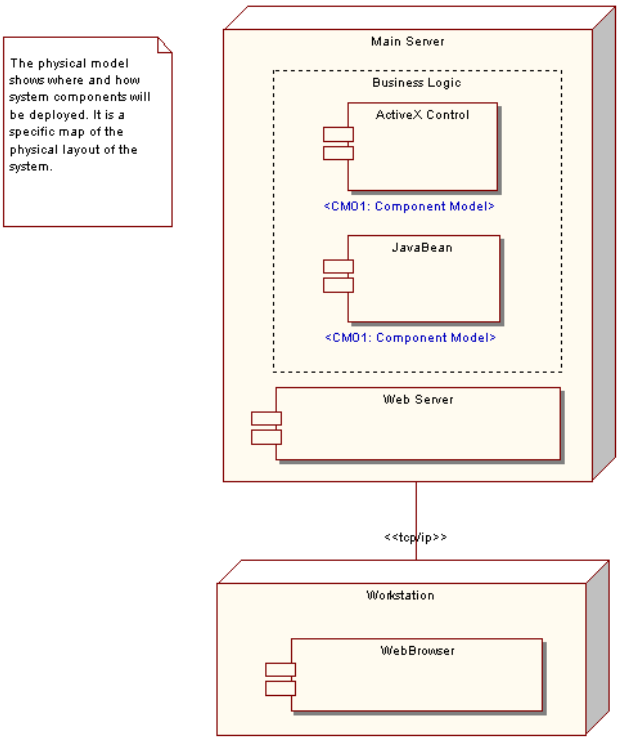


Gambar 3.5 Contoh Componen Diagram

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

3.2.3.1.2.8 Deployment Diagram

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.

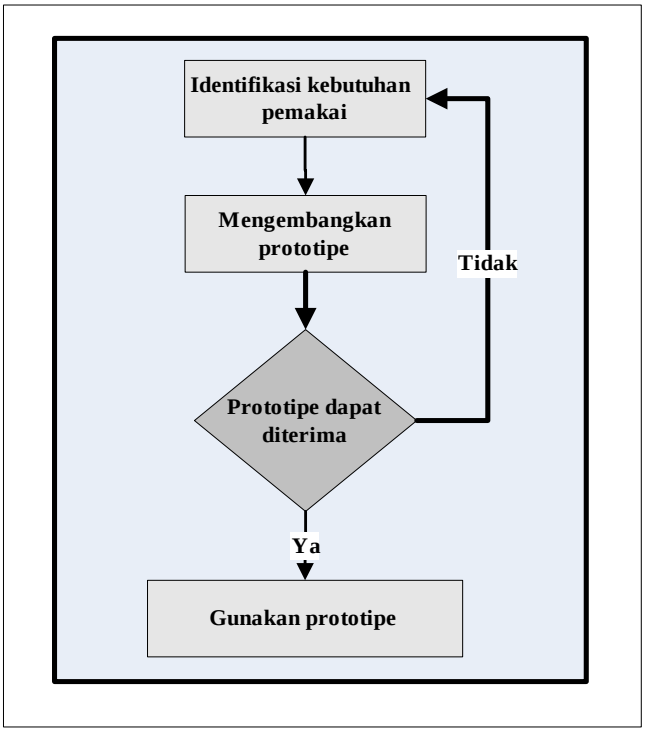


Gambar 3.6 Contoh Deployment Diagram

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

3.2.3.2 Metode Pengembangan Sistem

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



Gambar 3.7 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

3.2.3.3 Alat Bantu Analisis

OOP adalah merupakan teknik pemograman berbasis object. Pemograman dibagi dalam object-object kemudian di dalam object-object ini ada property dan

method. Property ini lebih dikenal dengan variable yang bersifat public dan method ini merupakan function. Object-object ini bisa diperluas lagi dengan inheritance atau turunan. Object-object turunan ini biasa dipakai untuk membuat object yang merupakan modifikasi dari object lain.

Dengan pengelompokan-pengelompokam ini maka progamming akan terstruktur dengan baik dan mudah dimanage. Bahasa progamming yang sudah mengimplementasikan OOP adalah C++ dan Java. Pemograman PHP akhir-akhir ini juga dikembangakan menjadi bahasa pemograman berbasis OOP (Object Oriented Programming) dengan laihirnya framework-framework berbasis PHP. (http://www.proweb.co.id)

3.2.4 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.*

3.2.4.1 Black box

Konsep kotak hitam digunakan untuk merepresrentasikan sistem yang cara kerja di dalamnya tidak tersedia untuk diinspeksi. Di dalam kotak hitam,

item-item yang diuji dianggap “gelap” karena logiknya tidak diketahui, yang diketahui hanya apa yang masuk dan apa yang keluar dari kotak hitam.

Pada pengujian kotak hitam, kasus-kasus pengujian berdasarkan pada spesifikasi sistem. Rencana pengujian dapat dimulai sedini mungkin diproses pengembangan perangkat lunak.

Teknik pengujian konvensional yang termasuk pengujian *“black box”* adalah sebagai berikut :

1. *Graph-based testing*
2. *Equivalence partitioning*
3. *Comparison testing*
4. *Orthogonal array testing*

Pada pengujian *black box*, kita mencoba beragam masukan dan memeriksa keluaran yang dihasilkan. Kita dapat mempelajari apa yang dilakukan oleh kotak, tapi tidak mengetahui sama sekali mengenai konversi dilakukan.

Teknik pengujian *black box* juga dapat digunakan untuk pengujian berbasis scenario,dimana isi dalam sistem mungkin tidak tersedia untuk diinspeksi tapi masukan dan keluaran yang didefisinikan dengan *use-case* dan informasi analisis yang lain. (Bambang Hariyanto,halaman 577,2004)