BAB III

OBJEK DAN METODE PENELITIAN

3.1. Objek Penelitian

3.1.1. Sejarah Singkat SMP Negeri 3 Subang

Lokasi SMP Negri 3 Subang merupakan salah satu sekolah menengah pertama yang berada di kecamatan Subang , dengan alamat di jalan Oto Iskandardinata no. 184 Kecamatan Subang Kabupaten Subang. Eblum menjadi nama SMP Negri 3 Subang, sekolah ini bernama SMEP yaitu Sekolah Menegah Ekonomi Pertama, setelah beberpa tahun kemudian yaitu Juli 1979 menjadi SMP Negri 3 Subang.

3.1.2. Visi dan Misi SMP Negeri 3 Subang

1. Visi SMP Negeri 3 Subang

   Adalah menjadi sekolah yang unggul dalam bidang Ilmu Pengetahuan dan Teknologi serta berlandaskan Iman dan Taqwa Kepada Allah SWT. Dengan indikator sebagai berikut:

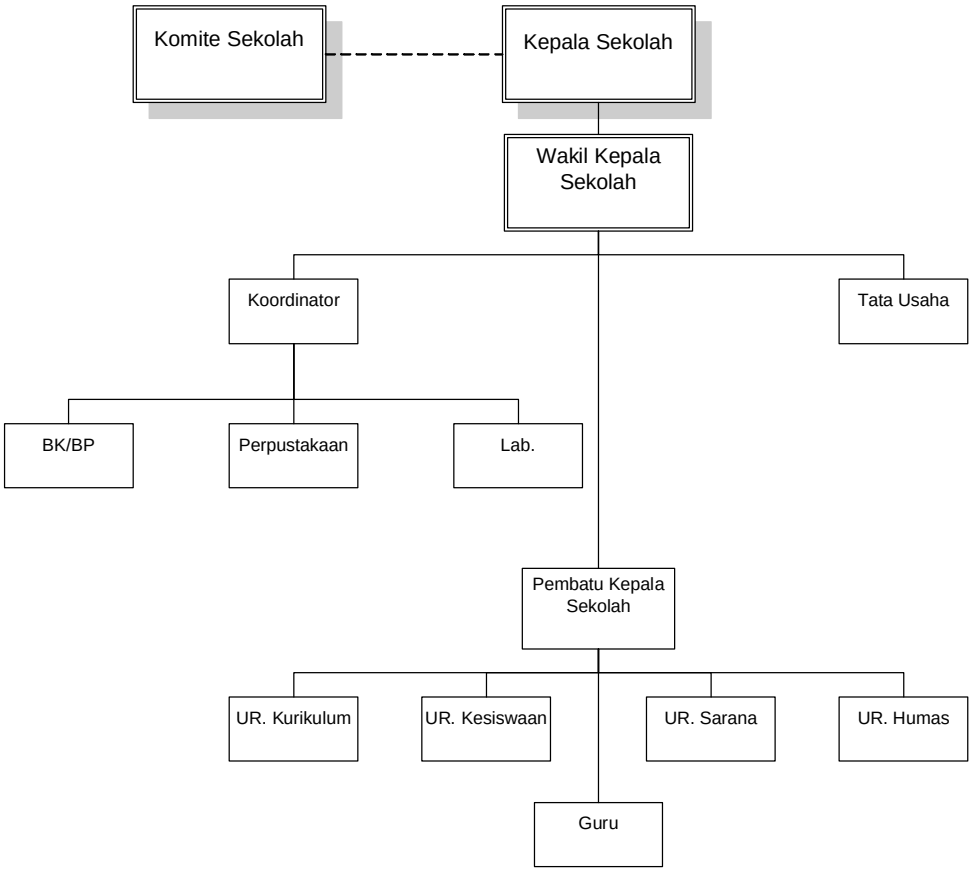
   a. Berorientasi kedepan dengan memperhatikan potensi yang ada
   b. Sesuai dengan norma dan harapan masyarakat
   c. Ingin mencapai keunggulan
   d. Mendorong semangat dan komitmen seluruh warga sekolah
   e. Mendorong adanya perubahan yang lebih baik
   f. Mengarahkan langkah-langkah yang strategis

2. Misi SMP Negeri 3 Subang

Meningkatkan disiplin dalam kerja dan pelayanan yang profesional. Menciptakan Proses Belajar Mengajar yang menyenangkan dan lingkungan sekolah yang kondusif dilandasi Iman dan Taqwa.

a. Mengembangkan rencana kerja yang rasional dan operasional
b. Mengembangkan Proses Belajar Mengajar melalui bimbingan dan praktek secara kreatif dan inovatif.
c. Mengembangkan proses pembelajaran melalui pembinaan kegiatan ekstra kulikuler yang berkesinambungan
d. Meningkatkan keimanan dan ketaqwaan
e. Menciptakan suasana lingkungan yang tertib, bersih, nyaman, dan indah.

3.1.3. Struktur Organisasi SMP Negeri 3 Subang



Gambar 3.1 Struktur Organisasi SMP Negeri 3 Subang

3.1.4. Deskripsi Tugas

1. Komite Sekolah adalah badan yang dibentuk dari perwakilan wali siswa. Tugas komite sekolah adalah memantau kebijakan-kebijakan yang dibuat oleh pihak SMP Negri 3 Subang agar kebijakan tersebut tidak terlalu memberatkan wali siswa.
2. Kepala Sekolah bertugas sebagai pimpinan sekolah dimana maju mundurnya prestasi SMPN 3 Subang merupakan tanggung jawabnya.
3. Wakil Kepala Sekolah bertugas untuk membantu Kep.Sek menyusun program, mengorganisir/ mengkoordinir pelaksanaan program yang ditetapkan Kep.Sek termasuk pembangunan fisik, menampung aspirasi warga sekolah, membantu Kep.Sek. dalam pengarahan, pengawasan dan evaluasi dan mewakili Kep.Sek. jika Kep.Sek. berhalangan.
4. Tata Usaha bertugas mengarsipkan data-data siswa dan megelola admnistrasi siswa maupun guru.
5. Urusan Kurikulum bertugas untuk menyusun program kegiatan kurikulum, mengatur pembagian tugas guru, menyusun program evaluasi pembelajaran, menyiapkan dan mengarahkan penyusunan adm pembelajaran, mengkoordinir kegiatan MGMP/MGP dan menyampaikan laporan.
6. Urusan Kesiswaan bertugas untuk menyusun program kegiatan kesiswaan, melaksanakan pembinaan siswa, mengkoordinir/mengatur kegiatan siswa, melaksanakan evaluasi kegiatan dan menyampaikan laporan.

7. Urusan Sarana bertugas untuk mengidentifikasi data dan kebutuhan, menyusun program, mengatur pengelolaan sarapras, mengawasi dan mengevaluasi dan menyampaikan laporan.
8. Urusan Humas bertugas untuk menyusun program kerja Kom.Kerja., mewujudkan hubungan dan kerja sama intern, menjalin komunikasi dan kerjasama ekstern, mengatur komunikasi dan informasi dan mengevaluasi dan menyampaikan laporan.
9. Koordinator BK/BP bertugas untuk menyusun Program BK/BP, mengatur pembagian tugas guru BK/BP, mengatur pelaksanaan kegiatan BK/BP, mengevaluasi kegiatan BK/BP, mengatur Koordinasi dg Guru/Wali. Kelas, mengatur penanganan siswa bermasalah dan melaksanakan tidak lanjut. dan mengatur tindak lanjut kegiatan BK/BP.
10. Koordinator perpustakaan bertugas sebagai pengelola ruang perpustakaan beserta isinya dan mendata peminjaman buku oleh siswa maupun guru.
11. Koordinator lab. bertugas mengurus fasilitas dan kelengkapan barang yang berada di lab.

3.2. Metode Penelitian

Metode yang di gunakan penulis dalam pengembangan sistem adalah metode *prototype*. Seringkali pemesan mendefinisikan kumpulan sasaran untuk diakomodasikan oleh perangkat lunak tanpa dapat mengidentifikasikan rincian-rincian masukan, pengolahan dan keluaran yang diharapkannya. Oleh Karen itu penulis memilih model *prototype*, karena cocok untuk situasi ini.

Pada model *prototype* ini dimulai dengan pengumpulan informasi mengenai kebutuhan dimana:

a. Pengembangan dan pemesanan bertemu dan mendefinisikan saran-saran umum,
b. Mengidentifikasi kebutuhan yang diketahui.
c. Mencari bidang-bidang yang masih memerlukan pendefinisian.

3.2.1. Desain Penelitian

Dalam penelitian ini, peneliti menggunakan rancangan penelitian berdasarkan metode deskriptif yaitu suatu metode dalam meneliti status kelompok manusia, suatu obyek, suatu set kondisi, suatu sistem pemikiran ataupun suatu kelas peristiwa pada masa sekarang.

3.2.2. Jenis dan Metode Pengumpulan Data

3.2.2.1. Jenis Data

Dalam penelitian ini, terdapat dua jenis data yang digunakan yaitu sebagai berikut :

1. Sumber Data Primer

   Data primer adalah data yang diperoleh langsung dari unit pengamatan atau responden penelitian.

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.

3.2.2.2. Metode Pengumpulan Data

A. Sumber Data Primer

Dalam penelitian ini penulis mengumpulkan data primer dengan menggunakan teknik. Metode pengumpulan data yang di gunakan adalah:

1. Penelitian Kepustakaan (*Library Research*)

   Yaitu penelitian dengan membaca literature yang berhubungan dengan masalah yang akan di bahas.

2. Penelitian Lapangan (*Field Research*)

   Yaitu penelitian langsung ke perusahaan dan instansi yang bersangkutan, teknik yang di lakukan yaitu dengan cara :

   a. Wawancara

      Data diperoleh dengan melakukan wawancara langsung dengan cara tanya jawab dengan pihak-pihak yang di anggap atau di harapkan dapat mengetahui permasalahan yang terjadi di SMP Negeri 3 Subang.

   b. Observasi Lapangan

      Dengan teknis ini, maka penulis mengharapkan dapat mengetahui keadaan sistem yang sedang berjalan saat ini, sehingga dapat ditarik kesimpulan mengenai permasalahan yang timbul atau kebutuhan-kebutuhan apa saja yang dapat dipenuhi dalam pengembangan sistem yang dilakukan.

c. Metode Pengamatan

Dengan cara melakukan pengamatan langsung terhadap sistem yang kerja secara langsung terhadap objek yang sedang diteliti.

B. Sumber Data Sekunder

Adapun data yang berasal dari sumber data sekunder diperoleh dengan teknik dokumentasi. Dokumentasi adalah teknik pengumpulan data dengan cara mengumpulkan dokumen-dokumen yang berhubungan dengan obyek penelitian. Dalam hal ini, dokumen-dokumen yang diperoleh dianalisis sehingga diperoleh data-data yang sesuai untuk kegiatan pengembangan sistem.

3.2.3. Metode Pendekatan dan Pengembangan

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 paradigma*.

3.2.3.1. Metode Pendekatan

Metode pendekatan yang di pakai oleh penulis adalah dengan menggunakan pendekatan berorientasi objek (*object oriented*) yang di visualisasikan dengan UML.

Menurut Bambang Hariyanto (2004:259) UML adalah bahasa grafis untuk mendokumentasi, menspesifikasikan dan membangun sistem perangkat lunak. UML berorientasi objek menerapkan banyak level abstraksi, tidak bergantung proses pengembangan, tidak bergantung bahasa dan teknologi, pemaduan beberapa notasi di beragam metodologi, usaha bersama dari banyak pihak, didukung oleh kakas-kakas yang diintegerasikan lewat XML (XMI). Standar UML dikelola oleh OMG (*Object Managment Group).*

3.2.3.2. Metode Pengembangan

(http://rifkirespati.blogspot.com/2008/08/*paradigma-life-cycle-prototype-paradigm_30.html*/ 11 Mei 2010)

Metode pendekatan yang penulis pakai adalah metode prototipe metode ini dimulai dengan pengumpulan kebutuhan. Pengembang dan pelanggan bertemu dan mendefinisikan obyektif keseluruhan dari software, mengidentifikasi segala kebutuhan yang diketahui, dan area garis besar dimana definisi lebih jauh merupakan keharusan kemudian dilakukan “perancangan kilat”. Perancangan kilat berfokus pada penyajian dari aspek – aspek software tersebut yang akan nampak bagi pelanggan atau pemakai (contohnya pendekatan input dan format

output). Perancangan kilat membawa kepada konstruksi sebuah prototipe. Prototipe tersebut dievaluasi oleh pelanggan/pemakai dan dipakai untuk menyaring kebutuhan pengembangan software. Iterasi terjadi pada saat prototipe disetel untuk memenuhi kebutuhan pelanggan, dan pada saat yang sama memungkinkan pengembang untuk secara lebih baik memahami apa yang harus dilakukannya.



Gambar 3.2 Model Prototyping

Sumber : (http://www.albany.edu/acc/courses/fall97/acc681/prtp.gif).

Secara ideal prototipe berfungsi sebagai sebuah mekanisme untuk mengidentifikasi kebutuhan software. Bila prototipe yang sedang bekerja dibangun, pengembang harus mempergunakan fragmen – fragmen program yang ada atau mengaplikasikan alat –alat bantu (contohnya report generator, window manager, dll) yang memungkinkan program yang bekerja untuk dimunculkan secara cepat.

Model proses *prototyping* terdiri atas tahapan - tahapan sebagai berikut:

1. *Communication*

   Pada tahap ini *software engineer* dan *costumer* bertemu dan mendefinisikan tujuan perangkat lunak yang diinginkan secara *overall.*

2. *Quick plan, modelling*, dan *quick design*

   Ketiga tahap ini dilakukan setelah gambaran perangkat lunak secara umum diketahui. *Quick design* difokuskan pada perancangan bagaimana "bentuk" perangkat lunak nantinya, yaitu bagaimana *layout* antarmukanya atau bagaimana tampilan keluarannya.

3. *Construction of prototype*

   Setelah tujuan umum dan rancangan perangkat lunak diketahui, maka prototipe perangkat lunak mulai dikerjakan.

4. *Deployment, delivery* dan *feedback*

   Begitu prototipe perangkat lunak selesai maka prototipe tersebut diserahkan kepada *costumer* untuk dievaluasi. Pada tahap inilah *costumer* dapat mengetahui apa yang sebenarnya ia inginkan dari perangkat lunak dengan mengetahui "apa yang kurang" dari perangkat lunak tersebut dengan memberikan *feedback* kepada *developer*. *Feedback* inilah yang membantu *developer* untuk lebih mengerti apa-apa saja yang harus diperbaiki dari prototipe yang telah dibuat. Seiring dengan dievaluasinya prototipe perangkat lunak oleh *costumer*, tahap *communication* pun kembali terulang diiringi dengan tahap-tahap selanjutnya hingga kepuasan *costumer* terhadap perangkat lunak yang ia inginkan tercapai.

A. Kelebihan Model *Prototyping:*

1. Metode ini cukup efektif sebagai paradigma dalam rekayasa perangkat lunak. Kuncinya adalah mendapatkan kebutuhan dan aturan yang jelas yang disetujui pelanggan dan pembuat perangkat lunak. Walaupun pada umumnya prototype akan dihilangkan dan dibuat perangkatyang sebenarnya.
2. Ketika diperlihatkan working version, pelanggan bisa merasakan seakan akan itu adalah sistem yang sebenarnya.

B. Kekurangan Model *Prototyping:*

1. Pelanggan melihat apa yang tampak sebagai versi software yang bekerja tanpa melihat bahwa prototipe itu dijalin bersama – sama "dengan permen karet dan baling wire", tanpa melihat bahwa di dalam untuk membuatnya bekerja, kita belum menyantumkan kualitas software secara keseluruhan atau kemampuan pemeliharaan untuk jangka waktu yang panjang. Ketika diberi informasi bahwa produk harus dibangun lagi agar tingkat kualitas yang tinggi bisa dijaga, pelanggan akan meneriakan kecurangan dan permintaan agar dipakai "beberapa campuran" untuk membuat prototipe menjadi sebuah produk yang bekerja yang lebih sering terjadi, sehingga manajemen pengembangan software menjadi penuh dengan belas kasihan.
2. Pengembang sering membuat kompromi – kompromi implementasi untuk membuat prototipe bekerja dengan cepat. Sistem operasi atau bahasa pemrograman yang tidak sesuai bisa dipakai secara sederhana

karena mungkin diperoleh dan dikenal; algoritma yang tidak efisien secara sederhana bisa diimplementasikan untuk mendemonstrasikan kemampuan. Setelah selang waktu tertentu, pengembang mungkin mengenali pilihan – pilihan tersebut dan melupakan semua alasan mengapa mereka tidak cocok. Pilihan yang kurang ideal telah menjadi bagian integral dari sebuah sistem.

Meskipun berbagai masalah bisa terjadi, prototipe bisa menjadi paradigma yang efektif bagi Software Engineering. Kuncinya adalah mendefinisikan aturan main pada saat awal; yaitu pelanggan dan pengembang keduanya harus setuju bahwa prototipe dibangun untuk berfungsi sebagai mekanisme pendefinisian kebutuhan. Prototipe kemudian disingkirkan (paling tidak sebagian), dan software aktual direkayasa dengan tertuju kepada kualitas dan kemampuan pemeliharaan.

3.2.3.3. Alat Bantu Analisis dan Perancangan

1. ***Actor***

   Menurut Bambang Hariyanto (2004:269) *actor* adalah pemakai sistem, dapat berupa manusia atau sistem terotomatisasi lain. *Actor* adalah sesuatu atau seseorang yang berinteraksi dengan sistem, yaitu siapa atu apa yang menggunakan sistem. Yang dimaksud dengan berinteraksi adalah *actor* menrima atau mengirim pesan ke atau dari

sistem. Dalam prspektif *actor, use-case* melakukan sesuatu yang berharga bagi *actor*.

Actor adalah tipe (kelas), bukan instan. *Actor* merepresentasikan peran bukan pemakai individu dari sistem. *Actor* mempunyai nama. Nama yang dipilih seharusnya menyatakan peran *actor*.

Actor berkomunikasi dengan sistem lewat pengiriman dan penerimaan pesan, *use-case* selalu diawali oleh aktor yang mengirim pesan, disebut stimulus. Ketika satu *use-case* dilakukan, *use-case* dapat mengirim pesan ke satu *actor* atau lebih.

2. *Use-case Diagrams*

Use-case Diagrams digunakan untuk mendeskripsikan apa yang seharusnya dilakukan oleh sistem.

Menurut Martin Fowler (2005 : 141), Use Case *adalah "teknik untuk merekam persyaratan fungsional sebuah sistem"*.

Use-case Diagrams menyediakan cara medeskripsikan pandangan eksternal terhadap sistem dan interaksi-interaksinya dengan dunia luar. Dengan cara ini diagram *use-case* menggantikan diagram konteks pada pendekatan konvensional. Untuk penciptaan model *use-case* melibatkan pendefinisian

sistem, pencarian aktor-aktor dan *use-case*, mendeskripsikan *use-case* dan mendefinisikan hubungan antar *use-case* dan terakhir adalah melakukan validasi model.

Use-case bertindak sebagai mekanisme terstuktur untuk diagram-diagram interaksi. Umumnya, satu diagram interaksi digambarkan untk masing-masing *use-case* diagram. Salah satu bahaya *use-case* merupakan bagian vital dari pengembangan berorientasi objek. Kita seharusnya menggunakannya setiap ingin memahami kebutuhan-kebutuhan sistem.

3. *Class Diagrams*

Class menggambarkan keadaan (atribut / properti) suatu sistem, sekaligus menawarkan layanan untuk memanipulasi keadaan tersebut (metoda / fungsi).

Class Diagram mendeskripsikan jenis-jenis objek dalam sistem dan berbagai macam hubungan statis yang terdapat diantara mereka. *Class Diagram* juga menunjukkan property dan operasi sebuah class dan batasan-batasan yang terdapat dalam hubungan-hubungan objek tersebut. UML menggunkan istilah fitur sebagai istilah umum yang meliputi property dan operasi sebuah *class*.

Class diagrams menggambarkan struktur dan deskripsi *class*, package dan objek beserta hubungan satu sama lain

seperti containment, pewarisan, asosiasi, agregasi dan hubungan dinamis.

Class memiliki tiga area pokok, yaitu:

a. Nama
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*.

Hubungan antar *Class*:

a. **Asosiasi**, yaitu hubungan statis antar class. Umumnya menggambarkan *class* yang memiliki atribut berupa *class* lain.
b. **Agregasi**, yaitu hubungan yang menyatakan bagian.
c. **Pewarisan**, yaitu hubungan hierarkis antar *class*. Class dapat diturunkan dari class lain dan mewarisi semua stribut dan metoda class asalnya dan menambahkan fungsionalitas baru, sehingga ia disebut anak dari *class* yang diwarisinya.

d. **Hubungan dinamis**, yaitu rangkaian pesan yang di-*passing* dari satu *class* kepada *class* lain.

4. *Object Diagrams*

Object diagrams memodelkan pandangan statik terhadap sistem untuk memodelkan struktur objek.

Menurut Martin Fowler (2005 : 125), Object Diagram merupakan "sebuah gambaran tentang objek-objek dalam sebuah sistem pada satu titik waktu. Karena lebih menonjolkan perintah-perintah dari pada class, object diagram lebih sering disebut sebagai sebuah diagram perintah".

Pemodelan struktur objek melibatkan *snapshot* dari objek-objek sistem pada suatu waktu. *Object diagrams* merepresentasikan satu frame statik papn cerita dinamis dari diagram interaksi.

Kegunaan object diagram adalah mendeskripsikan bagaimana kumpulan objek tertentu saling berhubungan. *Object diagrams* adalah diagram instan yang mendeskripsikan instan-instan kelas. Diagram instan berguna untuk dokumentasi skenario, serta kasus pengujian dan mendiskusikan contoh-contoh instanisasi *class diagrams*.

5. *Component Diagrams*

Component Diagrams menunjukan organisasi dan kebergantungan di antara sekumpulan komponen. Diagram ini memodelkan pandangan implementasi fisik dari sistem. *Component diagrams* berisi:

a. Komponen
b. Antarmuka
c. *Dependency*, generalisasi, asosiasi dan realisasi
d. Paket, untuk mengelompokkan elemen-elemen model menjadi potongan-potongan besar.

6. *Deployment Diagrams*

Deployment Diagrams digunakan untuk memodelkan aspek fisik dari sistem berorientasi objek, yaitu memodelkan konfigurasi node-node pengolahan waktu jalan dan komponen-komponen yang tinggal di node-node itu.

Deployment Diagram menunjukkan susunan fisik sebuah sistem, menunjukkan bagian perangkat lunak mana yang berjalan pada perangkat keras mana. Deployment diagram sangatlah sederhana.

Deploy Diagrams bisa juga berisi komponen-komponen, masing-masing komponen itu berada di suatu node. Juga dapat berisi paket atau subsistem yang digunakan mengelompokkan elemen-elemen di model menjadi potongan-potongan.

Penggunaan model *deployment* yaitu:

a. Memodelkan *embedded system*
b. Memodelkan sistem *client/server*
c. Memodelkan sistem tersebar penuh

7. *Sequence Diagrams*

Sequence diagram terdiri antar dimensi vertikal (waktu) dan dimensi horizontal (objek-objek yang terkait).

Sebuah *Sequence diagram*, secara khusus, menjabarkan behavior sebuah scenario tunggal. Diagram tersebut menunjukkan sebuah objek contoh dan pesan-pesan yang melewati objek-objek ini di dalam *use case*.

*Sequence diagram*s biasa digunakan untuk menggambarkan skenario atau rangkaian langkah-langkah yang dilakukan sebagai *respons* dari sebuah *event* untuk menghasilkan *output* tertentu.

8. *Collaboration Diagrams*

Collaboration diagrams mendefinisikan peran-peran yang dimainkan ketika satu tugas dilakukan. Peran-peran dimainkan oleh instan-instan yang berinteraksi.

Collaboration diagrams menyatakan komunikasi diantara objek-objek yang menunjukan pesan-pesan yang ada, urutan pesan dan hubungan antar objek-objek.

Collaboration diagrams digunakan untuk:

a. Pandangan dalam arti perilaku sistem, berfokus pada link-link di antara objek-objek
b. Ilustrasi dari suatu diagram *use-case*
c. Menyatakan objek-objek yang diperlukan untuk merealisasikan suatu layanan
d. Memeriksa jalur-jalur pengaksesan.

Collaboration diagrams menekankan pada organisasi objek yang berpartisipasi di interaksi. *Sequence diagrams* dan *collaboration diagrams* sama-sama menunjukkan interaksi (aspek dinamis). *Sequence diagrams* fokus pada waktu sedangkan *collaboration diagrams* fokus pada ruang. Sebagaimana *sequence diagrams*, *collaboration diagrams* juga dapat digunakan untuk mengilustrasikan eksekusi satu operasi, *use-case* atau skenario interaksi di sistem.

9. *Statechart Diagrams*

Statechart merupakan perluasan *State diagrams. Statechart* diperkenalkan oleh Harel, kemudian dipopularkan oleh Rumbaugh dan kawan-kawan di OMT. *Statechart* mempermudah pemodelan perilaku sistem secara tidak ambigu.

Statechart menyediakan notasi dan sekumpulan konvensi fasilitas dekomposisi hirarki mesin-mesin *finite state* dan

mekanisme berkomunikasi di antara mesin-mesin *finite state* konkuren.

Satu *state diagrams* untuk satu kelas objek dimana perilaku dinamisnya penting dan menunjukkan pola aktivitas. Masing-masing mesin *state* berjalan secara konkuren dan dapat mengubah *state* secara independen. Diagram-diagram *state* beragam kelas membentuk satu model dinamis tunggal lewat kejadian bersama (*shared events*). *Statechart* mendeskripsikan objek berupa state-state yang dimilikinya, kejadian-kejadian yang dapat berlangsung beserta transisi.

10. *Activity Diagrams*

Activity diagrams adalah diagram *flowchart* yang diperluas yang menunjukkan aliran kendali satu aktivitas ke aktivitas lain. Kita menggunakan diagram ini untuk memodelkan aspek dinamis sistem. Aktivitas adalah eksekusi nonatomik yang berlangsung di *state machine*. Diagram aktivitas mendeskripsikan aksi-aksi dan hasilnya. Diagram aktivitas berupa operasi-operasi dan aktivitas-aktivitas di *use-case*.

Activity diagram dapat digunakan untuk:

a. Pandangan dalam yang dilakukan di operasi.
b. Pandangan dalam bagaimana objek-objek bekerja.

c. Pandangan dalam di aksi-aksi dan pengaruhnya pada objek-objek.

d. Pandangan dalam dari suatu *use-case.*

e. Logik dari proses bisnis.

3.3. Pengujian Software

3.3.1. *Black Box Testing*

Konsep kotak hitam digunakan mempresentasikan sistem yang cara kerja didlamnya tidak tersedia untuk diinpeksi. Di dalam kotak hitam, item-item yang diuji dianggap "gelap" karena logikanya tidak diketahui, yang diketahui hanya apa yang masuk dan apa yang keluar dari kotak hitam. Bambang Hariyanto (2004:577)

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:

a. *Graph-based testing*

b. *Equivalaence partitioning*

c. *Comparison testing*

d. *Orthogonal array testing*

Pada pengujian *black box* , kita mencoba beragam masukan dan memeriksa keluaran yang dihasilkan. Kita dapat mempelajari apa yang

dilakukan kotak, tapi tidak mengetahui sama sekali mengenai cara konversi dilakukan.

Teknik pengujian *black box* juga dapat digunakan untuk pengujian berbasis skenario, dimana isi dalam sistem mungkin tidak tersedia untuk diinspeksi tapi masukan dan keluaran yang didefinisikan dengan *use-case* dalam informasi analisis yang lain.

3.3.2. *White Box Testing*

Pengujian *white box* mengasumsikan bahwa logic spesifik adalah penting dan harus diuji untuk menjamin sistem melakukan fungsi dengan benar. Penggunaan utama *white box* adalah pengujian berbasis kesalahan ketika kita siap menguji semua objek di aplikasi da semua *metode* eksternal atau public dari objek. Pada pengujian *white box*, kita mencari *bug* yang mempunyai peluang eksekusi uyang rendah, atau diimplementasikan secara sembrono. Bambang Hariyanto (2004:578)

Pengujian kotak putih disebut juag pegujain struktur. Penurunan kasus kasus pengujian berdasarkan struktur program. Pengetahuan mengenai program digunakan untuk identifikasi kasus-kasus pengujian. Sasaran dari pengujian ini adalah memeriksa semua pernyataan program (bukan semua kombinasi jalur, tapi semua pernyataan program).

Tenik pengujian *white box* antara lain:

1. *Basis path testing*

   Metode ini lmelakukan pencarian ukuran kompleksitas perancangan prosedur dan menggunakanya untuk mendefinisikan sekumpulan jalur eksekusi dasar. Kasus uji diturunkan dari himpunan basis yang dijamin mengeksekusi seluruh pernyataan d program setidaknya satu kali selama pengujian.

   Langkah-langkah:

   a. Gunakan rancangan atau kode program sebagai dasar, kemudian buat *flow graph*nya.
   b. Tentukan kompleksitas siklomatik dari flowgraph yang dibuat.
   c. Tentukan himpunan basis dari jalur-jalur yang independen secara linear
   d. Persiapkan kasusu pengujian yang akan memmaksa eksekusi masing-masing alur himpunan basis.

2. *Control structure testing*

   Terdapat beragam *control structure testing,* yaitu :

   a. *Condition testing*
   b. *Data flow testing*
   c. *Loop testing*