BAB III

OBJEK DAN METODE PENELITIAN

3.1. OBJEK PENELITIAN

Objek penelitian merupakan sasaran utama dalam kegiatan penelitian untuk dilakukan tahap-tahap atau langkah-langkah dalam metode pendekatan. Sesuai dengan judul penelitian ini, maka yang menjadi objek penelitian adalah anak tingkat SMP di SMPN 14 Bandung.

3.1.1. Sejarah

Menurut Drs. Dwi Markoniandi sebagai Rintisan Sekolah Bertaraf Internasional Mandiri, SMPN 14 Bandung merintis program Bilingual. Program Bilingual yang dimiliki bahkan yang pertama ada di kota Bandung. Program ini dimulai sejak tahun ajaran 2006-2007, SMP Negeri 14 Bandung sudah melaksanakan program RSBI, yaitu program pembelajaran dengan menggunakan bahasa Inggris sebagai bahasa pengantar.

Sumber daya manusia di SMPN 14 Bandung benar-benar ditangani secara serius sebagai persiapan menuju Sekolah Bertaraf Internasional. Salah satunya dengan mengadakan pelatihan ESQ dan pelatihan sholat khusuk.”Maintenance, guru-guru dan karyawan saya ikutkan kegiatan sholat khusuk oleh Abu Sangkan. Saya menyentuhnya pada ruhani. Kalau ruhani sudah disentuh dan tahu siapa mereka Insya Allah kulaitas mereka juga akan naik,” katanya. Selain kegiatan di

atas ada juga program tahajud dan doa bersama bagi yang beragama Islam dan renungan bagi non muslim.

Bagi para guru ada kegiatan bimbingan rohani. Bimbingan ini sengaja diisi dengan materi keagamaan Islam karena mayoritas Islam maka bicara lewat Al Quran. Meskipun bimbingan rohani ini berlabel Islam namun bagi mereka yang beragama lain juga diperbolehkan ikut. Menurut Dwi Markoniandi mereka boleh ikut hanya sebagai pendengar sebagai bentuk silaturahmi.Bimbingan rohani ini masih sebagai bagian dari berbagai program lain seperti tadarus, dan baca Asmaul khusna, doa bagi siswa di tiap pagi.

Sebenarnya pada tahun 2008 SMPN 14 ditunjuk sebagai sekolah RSBI di kota Bandung, namun karena kurang nya lahan membuat status SMPN 14 yang RSBI kembali diturunkan menjadi SSN.

3.1.2. Visi dan Misi

Visi

• Membina akhlak

• Meraih prestasi

• Berwawasan Nusantara yang dilandasi nilai-nilai budaya luhur sesuai dengan ajaran agama.

Misi

• Menanamkan keyakinan / akidah melalui pengamalam ajaran agama

• Mengoptimalkan proses pembelajaran dan bimbingan

• Mengembangkan pengetahuan di bidang Iptek (Ilmu Pengetahuan dan Tehnologi) , bahasa, olah raga, dan seni budaya sesuai dengan bakat minat dan potensi siswa

• Menjalin kerja sama yang harmonis antara warga sekolah dan lingkungan

3.1.3. Struktur Organisasi



Gambar 3.1 Struktur Organisasi

3.1.4. Deskripsi Kerja

1. Kepala sekolah

Kepala Sekolah sebagai edukator, manager, administrator, dan supervisor.

2. Wakil Kepala Sekolah

Wakil Kepala Sekolah membantu Kepala Sekolah dalam kegiatan – kegiatan. sebagai berikut:

3. Kepala TU (Tata Usaha)

Kepala Tata Usaha Sekolah mempunyai tugas melaksanakan ketata usahaan sekolah dan bertanggung jawab terhadap kepala sekolah dalam kegiatan-kegiatan sebagai berikut:

1) Membuat perangkat program pengajaran
2) Melaksanakan kegiatan pembelajaran

3) Melaksankan kegiatan penilaian proses belajar, ulangan harian, ulangan umum, dan ujian akhir.

4) Melaksanakan analis hasil ulangan harian

5) Menyusun dan melaksanakan program perbaikan dan pengadaan

6) Mengisi daftar nilai siswa

7) Melaksankan kegiatan membimbing (penghimbauan pengetahuan) kepada guru lain dalam kegiatan proses belajar mengajar

8) Membuat alat pelajaran/alat peraga

9) Menumbuhkembangkan sikap mengharagai karya seni

10) Mengikuti kegiatan pengembangan dan pemasyarakatan kurikulum

11) Melaksanakan pengembangan program pengajaran yang menjadi tanggung jawabnya

12) Membuat catatan tentang kemajuan hasil belajar siswa

13) Mengisi dan meneliti daftar siswa sebelum memulai pelajaran

14) Melaksanakan tugas tertentu di sekolah

15) Mengatur kebersihan ruang keras dan ruang praktikum

16) Mengumpulkan dan menghitung a ngka kredit untuk kenaikan pangkatnya.

4. Wali Kelas

Wali Kelas membantu Kepala Sekolah dalam kegiatan sebagai berikut:

1) Pengelolaan Kelas

2) Penyelenggaraan administrasi kelas

3) Penyusunan atau pembuatan statistik bulanan siswa

4) Pengisian daftar kumpulan nilai siswa (legger)

5) Pembuatan catatan khusus tentang siswa

6) Pencatatan mutasi siswa

7) Pengisian buku laporan penilaian hasil belajar siswa

8) Pembagian buku laporan penilaian hasil belajar

5. Guru BP

Bimbingan Konseling (BP) membantu kepala sekolah dalam kegiatan - kegiatan sebagai berikut:

1) Penyusunan program dan pelaksanaan bimbingan konseling

2) Koordinasi dengan wali kelas dalam rangka mengatasi masalah - masalah yang dihadapi tentang kesulitan belajar

3) Memberikan layanan bimbingan kepada siswa agar lebih berprestasi dalam kegiatan belajar

4) Memberikan saran dan pertimbangan kepada siswa dalam memperoleh gambaran tentang kelanjutan pendidikan dan lapangan pekerjaan yang sesuai

5) Mengadakan penilaian pelaksanaan bimbingan konseling

6) Melaksanakan kegiatan-kegiatan analisi evaluasi belajar

7) Menyusun statistik hasil penilaian bimbingan konseling

8) Menyusun dan melaksanakan program tindak lanjut bimbingan konseling

9) Menyusun laporan bimbingan dan konseling

3.2. Metode Penelitian

Metode merupakan cara bagaimana secara teknis membangun perangkat lunak yang harus berada pada komitmen dasar menuju kualitas. Untuk membangun aplikasi e-learning Matematika Dasar untuk anak tingkat SD, menggunakan metode *system development life cycle* (SDLC). Sesuai dengan namanya metode ini dimulai dengan suatu tahapan sampai tahapan terakhir dan kembali lagi ketahapan awal membentuk suatu siklus hidup atau daur hidup sebuah sistem. Model yang penulis gunakan adalah model waterfall.

3.2.1. Jenis dan Metode Pengumpulan Data

3.2.1.1. Metode Pengumpulan Data

Pengumpulan data bertujuan untuk melihat kondisi saat ini dari sistem pembelajaran Matematika Dasar di SDN CIJAGRA Bandung, kendala yang dihadapi dalam menerapkan sistem pembelajaran, keinginan dari para Pelajar dan guru mengenai sistem pembelajaran dimasa yang akan datang, untuk itu penulis menggunakan 4 macam cara yaitu melalui wawancara dan pengamatan langsung.

3.2.1.2. Sumber Data Primer

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

a. Wawancara

   pengumpulan data yang dilakukan dengan cara Tanya jawab dengan unsure-unsur individu yang terlibat guna mendapatkan data yang akurat

b. Observasi

Dilakukan dalam pengamatan secara langsung terhadap system pembelajaran bahasa Inggris di SD Jelegong 3

3.2.1.3. Sumber Data Sekunder

Data sekunder merupakan data yang di peroleh secara tidak langsung yaitu berupa referensi dari buku, makalah atau dari berupa sumber internet yang berhubungan dengan tema penelitian yang sedang dilakukan.

3.2.2. Metode Pendekatan

Metode pendekatan dan pengembangan sistem adalah beberapa metode yang digunakan pada penelitian ini, diantaranya metode pengembangan sistem dan metode pendekatan sistem dan alat bantu analisis dan perancangan, Dibawah ini akan diuraikan tentang pemahaman dari metode tersebut

3.2.2.1. Metode Pendekatan Sistem

Metode pendekatan sistem yang akan digunakan adalah analisis dan perancangan terstruktur, karena setelah melakukan wawacara maka kebutuhan atau spesifikasi sistem telah dapat ditentukan. Berdasarkan informasi tersebut maka dapat dilakukan proses desain sistem dan perangkat lunak. Proses desain sistem membagi kebutuhan-kebutuhan menjadi sistem perangkat lunak atau perangkat keras. Proses tersebut menghasilkan sebuah arsitektur sistem secara keseluruhan. Desain Perangkat lunak termasuk menghasilkan fungsi sistem perangkat lunak.

Metode pendekatan sistem merupakan serangkaian langkah-langkah pemecahan masalah yang memastikan bahwa masalah dipahami, solusi alternative dipertimbangkan dan solusi yang dipilih bekerja. Proses pemecahan masalah secara sistematis bermula dari John Dewey, seorang professor filosofi di Columbia University pada awal abad ini. Dalam bukunya 1910 diidentifikasi ada 3 seri penilaian dalam memecahkan suatu kontroversi yang memadai :

1. Mengenali kontroversi.
2. Menimbang klaim alternatif.
3. Membentuk penilaian.

Metode pendekatan sistem yang penulis gunakan adalah metode pendekatan sistem yang berorientasi objek (*Object-Oriented)*

3.2.2.2. Metode Pengembangan Sistem

Pada tahap ini, sistem diintegrasikan dan diuji sebagai sebuah kesatuan sistem yang utuh. Salah satu kelemahan metode SDLC adalah hanya memberikan tahapan-tahapan (apa yang harus dilakukan) dalam mengembangkan sistem tetapi tidak memberikan cara (bagaimana mengembangkannya) dan alat apa yang harus digunakan untuk mengembangkannya. Untuk itu dalam penggunaan cara dan alat pengembangan system, penulis menggunakan metode pengembangan sistem terstruktur. Metode pengembangan sistem terstruktur terdapat dua pendekatan yaitu pendekatan top down dan pendekatan bottom up. Pendekatan top down adalah mengidentifikasi kebutuhan informasi untuk pemakai sampai kepada identifikasi data yang dibutuhkan untuk informasi tersebut. Untuk pendekatan bottom up adalah pendekatan dimulai dari identifikasi ketersediaan informasi

sampai kepada informasi yang dibutuhkan pemakai sistem. Dalam penelitian ini penulis menggunakan pendekatan top down karena sesuai dengan tujuan penelitian yaitu membantu Untuk membantu anak tingkat SMP belajar Matematika Dasar, berawal dengan mengetahui rumus-rumus dalam Matematika Dasar.

Dalam perancangan sebuah sistem hal yang sangat penting dan harus diperhatikan agar menghasilkan suatu sistem yang baik dan dapat melengkapi informasi yang dibutuhkan merupakan kajian teknis formal. Untuk itu digunakan metode – metode perancangan sistem. Pengembangan sistem yang penulis gunakan dalam pelaporan tugas akhir ini adalah dengan menggunakan pendekatan model SDLC (*System Development Life Cycle*).

Dikutip dari http://id.wikipedia.org/wiki/SDLC. *Siklus hidup pengembangan sistem (Systems Development Life Cycle) adalah proses pembuatan dan pengubahan sistem serta model dan metodologi yang digunakan untuk mengembangkan sistem-sistem tersebut. Konsep ini umumnya merujuk pada sistem komputer atau informasi.*

SDLC berfungsi untuk menggambarkan tahapan-tahapan utama dan langkah-langkah dari setiap tahapan yang secara garis besar terbagi dalam tiga kegiatan utama, yaitu :

a. *Analysis*

Digunakann untuk membuat keputusan, apabila sistem saat ini mempunyai masalah atau sudah tidak berfungsi secara baik, dan hasil analisisnya digunakan sebagai dasar untuk memperbaiki system.

b. *Design* (perancangan)

Tahapan perancangan / desain memiliki tujuan untuk mendesain sistem baru yang dapat menyelesaikan masalah-masalah yang dihadapi perusahaan yang diperoleh dari pemlihan alternatif sistem yang terbaik. Kegiatan yang dilakukan dalam tahapan perancangan ini meliputi perancangan output, input, dan file.

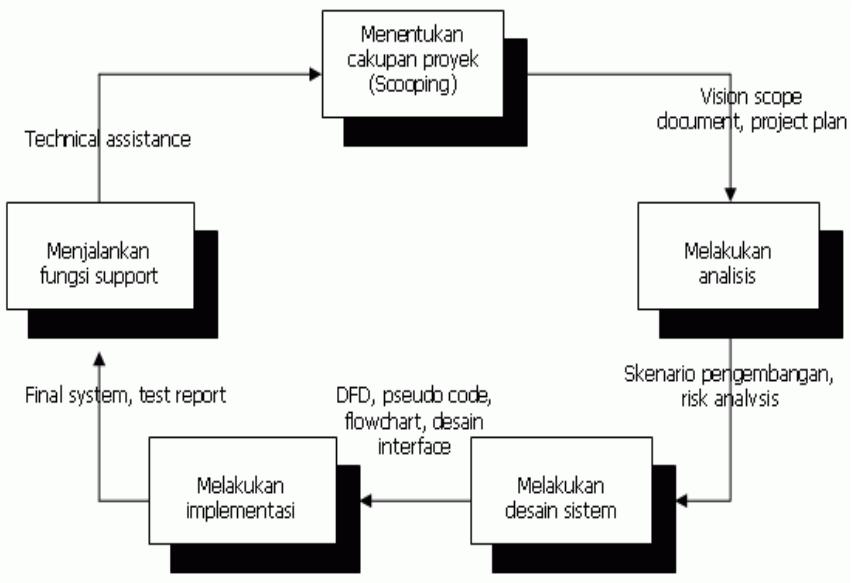
c. Implementation

Memiliki beberapa tujuan yaitu untuk melakukan kegiatan spesifikasi rancangan logikal ke dalam kegiatan yang sebenarnya dari sistem informasi yang akan dibangunnya atau dikembangkan ny, lalu mengimplementasikan sistem yang baru tersebut ke dalam salah satu bahasa pemrograman yang paling sesuai. Pada tahap ini juga harus menjamin bahwa sistem yang baru dapat berjalan secara optimal. Kegiatan yang dilakukan dalam tahap implementasi ini adalah pembuatan program dan test data, pelatihan dan pergantian sistem.

Dalam penerapan tahapan pengembangan sistem informasi penulis menggunakan sistem pengembangan *prototype*. *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 dengan tidak mengesampingkan segi-segi teknis dan pelanggan akan mengetahui proses-proses dalm menyelasaikan system 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 actual direkayasa dengan kualitas dan implementasi yang sudah ditentukan.



Gambar 3.2 Proses *Prototype*

(Sumber: http://www.ristinet.com/2010/04/16)

a) Pengumpulan kebutuhan

Pelanggan dan pengembang bersama-sama mendefinisikan format seluruh perangkatlunak, mengidentifikasikan semua kebutuhan, dan garis besar sistem yang akandibuat.

b) Membangun *prototyping*

Membangun prototyping dengan membuat perancangan sementara yang berfokuspada penyajian kepada pelanggan (misalnya dengan membuat input dan formatoutput)

c) Evaluasi *protoptyping*

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

d) Mengkodekan Sistem

Dalam tahap ini *prototyping* yang sudah disepakati diterjemahkan ke dalam bahasa pemrograman yang sesuai

e) Menguji Sistem

Setelah sistem sudah menjadi suatu perangkat lunak yang siap pakai, harus ditesdahulu sebelum digunakan. Pengujian ini dilakukan dengan *White Box*, *Black Box*, *Basis Path*, pengujian arsitektur dan lain-lain

f) Evaluasi Sistem

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

g) Menggunakan system

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

Adapun alasan mengapa penulis menggunakan metode ini karena metode ini mempunyai bebrapa keunggulan diantaranya :

Keunggulan *prototyping* adalah:

1. Adanya komunikasi yang baik antara pengembang dan pelanggan
2. Pengembang dapat bekerja lebih baik dalam menentukan kebutuhan pelanggan
3. Pelanggan berperan aktif dalam pengembangan system
4. Lebih menghemat waktu dalam pengembangan system
5. Penerapan menjadi lebih mudah karena pemakai mengetahui apa yang diharapkannya.

Kelemahan *prototyping* adalah :

1. Pelanggan kadang tidak melihat atau menyadari bahwa perangkat lunak yang ada belum mencantumkan kualitas perangkat lunak secara keseluruhan dan juga belum memikirkan kemampuan pemeliharaan untuk jangka waktu lama.
2. Pengembang biasanya ingin cepat menyelesaikan proyek. Sehingga menggunakan algoritma dan bahasa pemrograman yang sederhana untuk membuat prototyping lebih cepat selesai tanpa memikirkan lebih lanjut bahwa program tersebut hanya merupakan cetak biru sistem.

3. Hubungan pelanggan dengan komputer yang disediakan mungkin tidak mencerminkan teknik perancangan yang baik

Prototyping bekerja dengan baik pada penerapan-penerapan yang berciri sebagai berikut:

1. Resiko tinggi Yaitu untuk masalah-masalah yang tidak terstruktur dengan baik, ada perubahan yang besar dari waktu ke waktu, dan adanya persyaratan data yang tidak menentu.
2. Interaksi pemakai penting. Sistem harus menyediakan dialog on-line antara pelanggan dan komputer.
3. Perlunya penyelesaian yang cepat
4. Perilaku pemakai yang sulit ditebak
5. Sitem yang inovatif. Sistem tersebut membutuhkan cara penyelesaian masalah dan penggunaan perangkat keras yang mutakhir
6. Perkiraan tahap penggunaan sistem yang pendek

3.2.2.3. Pengertian UML

UML adalah bahasa untuk mengspesifikasi, memvisualisasi, membangun dan mendokumentasikan artefacts (bagian dari informasi yang digunakan atau dihasilkan oleh proses pembuatan perangkat lunak, seperti pada pemodelan bisnis dan sistem non perangkat lunak lainnya. (*Modul Unaki,2003, hal: 2*)

Unified Modelling Language (UML) adalah keluarga notasi grafis yang didukung oleh meta-model tunggal, yang membantu pendeskripsian dan desain sistem perangkat lunak, khususnya sistem yang dibangun menggunakan pemograman berorientasi objek (Fowler, 2005:1). Selain itu UML juga dapat

diartikan sebagai sebuah bahasa yang telah menjadi standar dalam industri untuk visualisasi, merancang dan mendokumentasikan sistem perangkat lunak (Dharwiyanti dan Wahono, 2003:2). *Unified Modelling Language* (UML) menawarkan sebuah standar untuk merancang model sebuah sistem. Tujuan UML adalah :

1. Memberikan model yang siap pakai, bahasa pemodelan visual yang ekspresif untuk mengembangkan dan saling menukar model dengan mudah dan dimengerti secara umum.
2. Memberikan bahasa pemodelan yang bebas dari berbagai bahasa pemrograman dan proses rekayasa.
3. Menyatukan praktik-praktik terbaik yang terdapat dalam pemodelan.

UML menyediakan beberapa *notasi dan artifact standar* yang bisa digunakan sebagai alat komunikasi bagi para pelaku dalam proses analisis dan desain. Artifact didalam UML didefinisikan sebagai informasi dalam bentuk yang digunakan atau dihasilkan dalam proses pengembangan perangkat. Contohnya adalah source code yang dihasilkan oleh proses pemrograman.

3.2.2.4. Notasi-notasi dalam Unified Modelling Languaage (UML)

Notasi Dasar

Merupakan bentuk pemodelan yang digunakan untuk mempresentasikan elemen-elemen dalam UML, terdiri dari :

1.Notasi Struktual, Terdiri dari

Notasi Aktor

Merupakan segala sesuatu yang berinteraksi dengan system aplikasi komputer. Aktor dapat berupa orang, perangkat, atau objek lain dalam sistem yang sama.

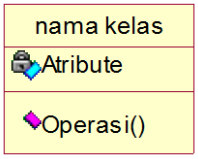


aktor

Gambar 3.2 : Simbol Aktor (*Modul Unaki,2003, hal: 3)*

2. Notasi Kelas

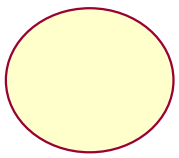
Merupakan pembentuk utama dari sistem berorientasi objek karena kelas menunjukan kumpulan objek yang mempunyai attribute dari operasi yang sama. Kelas digunakan untuk mengabstraksikan elemen-elemen dari sistem yang sedang dibuat.



Gambar 3.3 : Simbol Notasi Kelas(*Modul Unaki,2003, hal: 4)*

3. Notasi Antar Muka *(interface)*

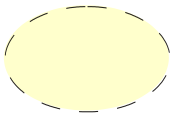
Merupakan kumpulan operasi tanpa implementasi dari suatu kelas. Implementasi operasi dalam antar muka dijabarkan oleh operasi dalam kelas, sehingga keberadaan anatar muka selalu disertai oleh kelas yang mengimplementasikan operasinya.



Gambar 3.4 : Simbol Notasi Antar Muka (*Modul Unaki,2003, hal :4*)

4. *Notasi Kolaborasi*

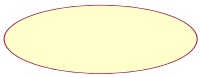
   Mempresentasikan sekelompok peranan dan elemen lainya yang bekerja bersama untuk menyediakan perilaku bersama dari semua elemen yang ada didalamnya



Gambar 3.5 : Simbol Notasi Kolaborasi (*Modul Unaki, 2003, hal: 4)*

5. *Notasi Use Case*

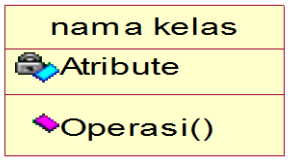
   Gambaran fungsionalitas dari suatu sistem, sehingga pengguna sistem paham dan mengerti mengenai kegunaan sistem yang akan dibangun



Gambar 3.6 : Simbol Notasi Use Case (*Modul Unaki, 2003, hal: 5)*

6. *Notasi kelas aktif*

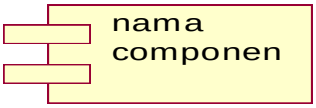
   Mirip dengan sebuah kelas, hanya saja objek - objeknya merepresentasikan elemen - elemen yang memiliki konkuren dengan elemen-elemen yang lain.



Gambar 3.7 : Simbol Notasi Kelas Aktif (*Modul Unaki, 2003, Hal :5)*

7. *Notasi Komponen*

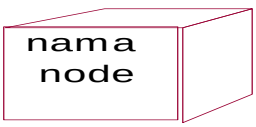
   Merupakan bentuk fisik yang dapat dipindahkan dari sebuah sistem yang membentuk dan menyediakan realisasi dari sekumpulan antar muka.



Gambar 3.8 : Simbol Notasi Komponen (*Modul Unaki,2003, hal: 5)*

8. *Notasi Node*

   Merupakan elemen fisik yang harus ada pada saat aplikasi dijalankan dan mempresentasikan sebuah sumber daya komputasi, yang memiliki memori, dan kemampuan pengolahan data.



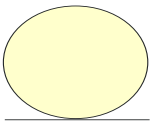
Gambar 3.9 : Simbol Notasi Node (*Modul Unaki, 2003, hal: 6)*

9. *Objek Desain*

Sruktur sebuah sistem berbasis objek dapat didentifikasikan dan diklasifikasikan dalam tiga tipe objek yaitu :

1) Objek entity

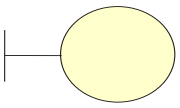
Merupakan sebuah objek yang berisi informasi yang berhubungan dengan kehidupan nyata yang bersifat menetap dan disimpan dalam suatu database yang memungkinkan pemanggilan dan manipulasi di wktu yang akan datang.



Gambar 3.10 : Simbol Objek Entitiy (*Modul Unaki, 2003, hal 6)*

2) Objek Antar Muka

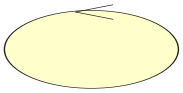
Merupakan sebuah objek yang menyediakan peralatan dimana pengguna dapat berkomunikasi dengan system



Gambar 3.11 : Simbol Objek Antar Muka (*Modul Unaki, 2003, hal: 6)*

3) Objek Control

Merupakan sebuah objek yang berisi logika aplikasi yang bukan merupakan tanggung jawab objek entity. Contoh logika tersebut adalah aturan dan perhitungan bisnis yang melibatkan banyak objek.



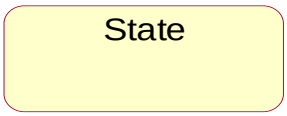
Gambar 3.12 : Simbol Objek Kontrol (*Modul Unaki, 2003, hal:7)*

Notasi Perilaku

Notasi Perilaku terdiri dari :

1. Notasi State

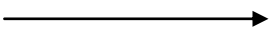
   Adalah suatu kondisi selama kehidupan sebuah objek ketika objek memenuhi beberapa kondisi, melakukan beberapa aksi, atau menunggu beberapa aksi, atau menuggu sebuah event.



Gambar 3.13 :Simbol Notasi State (*Modul Unaki, 2003, hal :7)*

2. Notasi Interaksi.

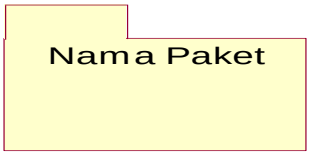
   Mempresentasikan pertukaran pesan antar objek dalam sebuah objek dalam sebuah konteks tertentu guna melakukan sebuah tujuan tertentu..



Gambar 3.14 : Simbol Notasi Interaksi (*Modul Unaki, 2003, hal: 8)*

Notasi Pengelompokan

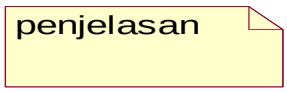
Jika sistem hanya mempunyai sedikit kelas, dapat diatur dengan mudah. Sebagian besar sistem dibuat dari banyak kelas sehingga diperlukan mekanisme untuk mengelompokan, untuk memudahkan dalam penggunaan kembali.



Gambar 3.15 : Simbol Notasi Penggelompokan (*Modul Unaki, 2003,hal:8)*

Notasi Penjelasan

Penjelasan digunakan untuk memberikan penjelasan dan catatan dalam pemodelan yang dilakukan

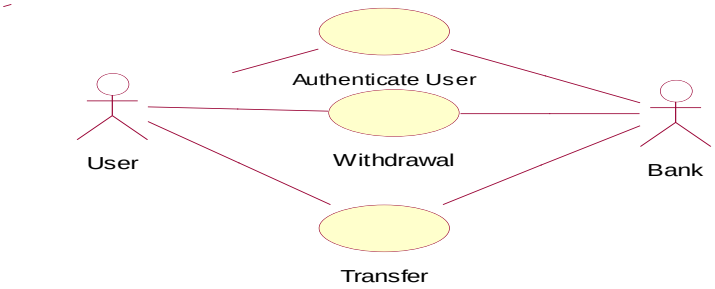


Gambar 3.16 :Simbol Notasi Penjelasan (*Modul Unaki, 2003, hal: 8)*

3.2.2.5. Tipe – tipe Diagram UML

1. *Use Case Diagram*

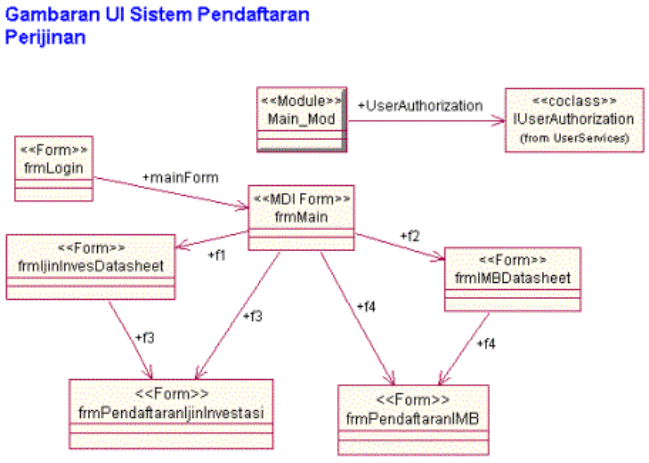
   Menggambarkan sejumlah eksternal aktor dan hubungannya ke use case yang diberikan oleh sistem



Gambar 3.17 : Use Case Diagram (*Modul Unaki, 2003, hal :32)*

2. *Class Diagram*

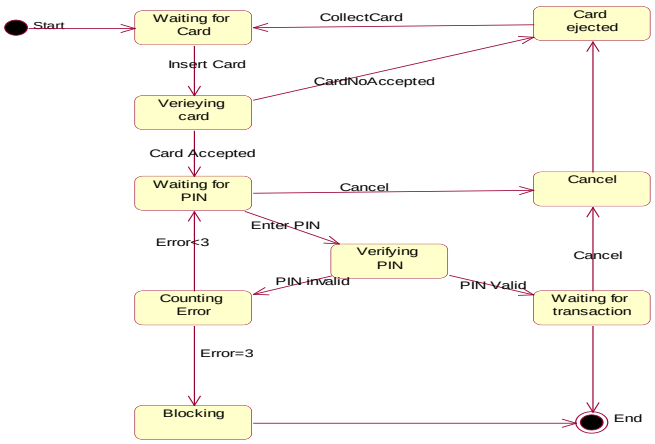
Menggambarkan struktur dan deskripsi class, package dan objek beserta hubungan satu sama lain seperti containment, pewarisan, asosiasi dan lain-lain.



Gambar 3.18 : Class Diagram (*Modul Unaki, 2003, hal: 53*)

3. *Statechart Diagram*

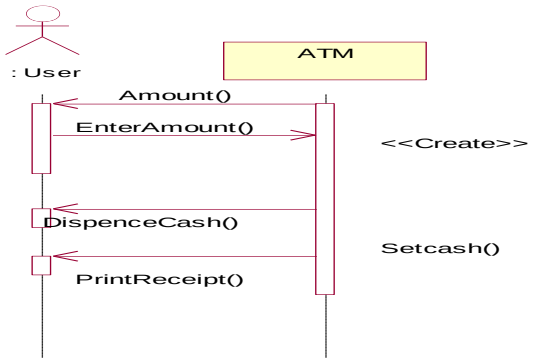
Menggambarkan semua state yang dimiliki oleh suatu objek dari suatu class dan keadaan yang menyebabkan state berubah.



Gambar 3.19 : Statechart Diagram (*Modul Unaki, 2003,hal: 60)*

4. *Sequance Diagram*

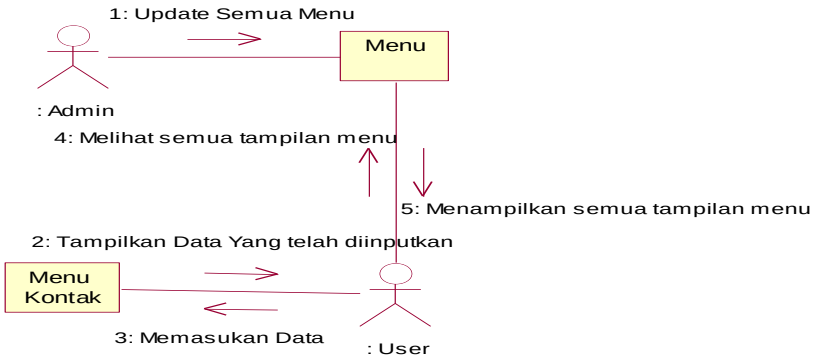
Menggambarkan kolaborasi dinamis antara sejumlah objek dan untuk menunjukkan rangkaian pesan yang dikirim antara objek juga interaksi antara objek, sesuatu yang terjadi pada titik tertentu dalam eksekusi sistem



Gambar 3.20 : Sequance Diagram (*Modul Unaki, 2003, hal: 39)*

5. *Collaboration Diagram*

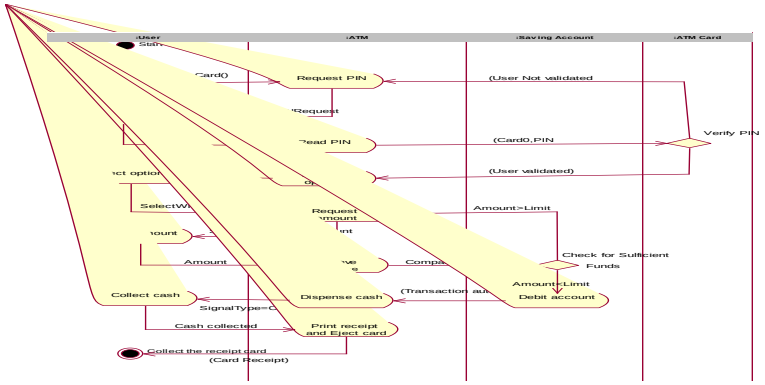
Menggambarkan kolaborasi dinamis seperti sequence diagram. Dalam menunjukan pertukaran pesan, collaboration diagram menggambarkan objek dan hubungannya



Gambar 3.21 :Collaboration Diagram (*Modul Unaki, 2003, hal: 41)*

6. *Activity Diagram*

Menggambarkan rangkaian aliran dari aktivitas, digunakan untuk mendeskipsikan aktifitas yang dibentuk dalam suatu operasi sehingga dapat juga digunakan untuk aktifitas lainya seperti Use Case atau interaksi.



Gambar 3.22 : Activity Diagram (*Modul Unaki, 2003, hal: 62)*

3.3. Evaluasi Software

3.3.1. Black Box

Pengujian Software adalah suatu cara atau teknik untuk menguji perangkat lunak yang digunakan secara lengkap dan mempunyai kemungkinan tinggi untuk menemukan kesalahan.

Pengujian *black-box* sering disebut juga dengan pengujian tingkah laku (*behavoiur testing*) yang lebih terfokus kepada kebutuhan fungsional dari perangkat lunak, pengujian *black-box* memungkinkan pembuat perangkat lunak untuk menentukan kondisi yang terjadi untuk suatu masukan yang akan menjalankan semua kebutuhan fungsional (Pressman 2002:551).

Jadi dapat penulis simpulkan bahwa Pengujian *Black Box* adalah pengujian yang dilakukan untuk antarmuka Perangkat Lunak. Pengujian ini dilakukan untuk memperlihatkan demo bahwa fungsi-fungsi Perangkat Lunak bekerja dengan baik dalam arti semua masukan diterima dengan benar dan keluaran yang dihasilkan benar-benar tepat dan pengintegrasian dari eksternal data berjalan dengan baik.

Pengujian *black-box* dilakukan untuk menentukan beberapa macam kesalahan, yaitu :

1. Fungsi-fungsi yang tidak benar atau hilang
2. Kesalahan interface
3. Kesalahan dalam struktur data atau database eksternal
4. Kesalahan kinerja
5. Inisialisasi dan kesalahan terminasi

Pengujian ini dilakukan pada tahap akhir dalam membuat perangkat lunak tidak seperti pengujian *white-box* yang dilakukan di awal pembuatan, hal tersebut dikarenakan pengujian *black-box* dengan sengaja menghiraukan struktur kendali dimana perhatian lebih di utamakan pada domain informasi.

Black-Box Testing terfokus pada spesifikasi fungsional dari perangkat lunak Tester dapat mendefinisikan kumpulan kondisi input dan melakukan pengetesan pada spesifikasi fungsional program. Black Box Testing bukanlah solusi alternatif dari White-Box Testing tapi lebih merupakan pelengkap untuk menguji hal-hal yang tidak dicakup oleh White-Box Testing.

Black-Box Testing cenderung untuk menemukan hal-hal berikut:

- Fungsi yang tidak benar atau tidak ada
- Kesalahan antarmuka (interface errors)
- Kesalahan pada struktur data dan akses basis data
- Kesalahan performansi (performance errors)
- Kesalahan inisialisasi dan terminasi.

Pengujian didesain untuk menjawab pertanyaan-pertanyaan berikut:

- Bagaimana fungsi-fungsi diuji agar dapat dinyatakan valid?
- Input seperti apa yang dapat menjadi bahan kasus uji yang baik?
- Apakah sistem sensitif pada input-input tertentu?
- Bagaimana sekumpulan data dapat diisolasi?
- Berapa banyak rata-rata data dan jumlah data yang dapat ditangani sistem?
- Efek apa yang dapat membuat kombinasi data ditangani spesifik pada operasi sistem?

Equivalence Partitioning

Membagi input menjadi kelas-kelas data yang dapat digunakan untuk menggenerasi kasus uji. Bertujuan untuk menemukan kelas-kelas kesalahan. Berdasarkan pada kesamaan kelas-kelas kondisi input. Sebuah kelas yang ekivalen merepresentasikan kumpulan status/kondisi yang valid atau tidak valid Sebuah kondisi input dapat berupa nilai numerik yang spesifik, rentang nilai, kumpulan nilai yang berkaitan, atau kondisi boolean. Kelas ekivalen dapat didefinisikan dengan kondisi berikut:

- Jika kondisi input mensyaratkan rentang nilai atau nilai yang spesifik, maka sebuah kelas ekivalen yang valid dan dua buah kelas ekivalen yang tidak valid akan terbentuk
- Jika sebuah kondisi input mensyaratkan sebuah boolean atau anggota dari sebuah himpunan, maka sebuah kelas ekivalen yang valid dan sebuah kelas ekivalen yang tidak valid akan terbentuk.
- Kasus uji untuk setiap domain input item data harus dikembangkan dan dieksekusi.

Aturan Kelas Ekivalen

- Jika sebuah kondisi mengacu pada nilai masukan yang memiliki rentang maka nilai yang ada dalam rentang masuk menjadi sebuah kelas ekivalen valid, dan nilai masukan diluar kedua sisi ambang batas menjadi dua buah kelas ekivalen yang tidak valid.

- Jika sebuah kondisi mengacu pada satu atau sejumlah nilai masukan dengan nilai tertentu maka akan menjadi sebuah kelas ekivalen yang valid yang berisi satu atau sejumlah nilai yang valid dan dua buah kelas ekivalen yang tidak valid yaitu kelas yang berisi masukan kosong dan masukan diluar nilai yang dispesifikasikan
- Jika sebuah kondisi mengacu pada himpunan nilai masukan tertentu maka akan menjadi sejumlah anggota himpunan kelas ekivalen yang valid dan sebuah kelas ekivalen yang tidak valid dimana jika masukan memiliki nilai diluar anggota himpunan
- Jika sebuah kondisi mengekspresikan sebuah kalimat yang berisi "harus" misal, awal masukan harus berupa karakter maka akan menjadi sebuah kelas ekivalen yang valid yaitu semua masukan yang diawal karakter dan sebuah kelas ekivalen yang tidak valid yaitu masukan yang tidak diawali karakter
- Jika ada alasan yang menyatakan bahwa elemen di dalam sebuah kelas ekivalen diproses dengan cara yang berbeda maka kelas ekivalen itu harus dibagi-bagi lagi menjadi kelas-kelas ekivalen yang lebih kecil berdasarkan pendekatan lojik dari sub-program pada level algoritma.
- Jika sub-program memiliki beberapa masukan yang independen, maka kelas ekivalen atau kasus uji harus dibuat berdasarkan permasukan dan kombinasi kemungkinan per masukan dimana permasukan dibuat dua buah kelas ekivalen yang valid dan sebuah kelas ekivalen yang tidak valid
- Tulis kasus uji untuk setiap kelas ekivalen yang tidak valid

Contoh:

Sebuah spesifikasi menyatakan untuk memasukkan masukan berupa tipe dan nomor mobil

- Untuk menguji sebuah sub-program yang memiliki beberapa masukan yang saling bergantung maka kelas ekivalen harus dibuat berdasarkan kombinasi masukan

Boundary Value Analysis / Limit Testing

Banyak kesalahan terjadi pada kesalahan masukan. BVA mengijinkan untuk menyeleksi kasus uji yang menguji batasan nilai input. BVA merupakan komplemen dari equivalence partitioning. Lebih pada memilih elemen-elemen di dalam kelas ekivalen pada bagian sisi batas dari kelas. Contoh:

- Untuk rentang yang dibatasi a dan b maka uji (a-1), a, (a+1), (b-1), b, (b+1).
- Jika kondisi input mensyaratkan sejumlah n nilai maka uji dengan sejumlah (n-1), n dan (n+1) nilai.
- Aplikasikan dua aturan sebelumnya pada kondisi output (buat table pengujian hasil outputnya untuk nilai maksimal dan minimal).
- Jika struktur data internal dari program memiliki cakupan (misal: ukuran buffer, batas array) gunakan data input yang menguji batas cakupan.

Comparison Testing

Pada beberapa aplikasi reliability dari sebuah perangkat lunak sangat penting. Redundansi perangkat keras dan perangkat lunak mungkin digunakan untuk

meminimalisir kesalahan (error). Untuk redundansi perangkat lunak, gunakan tim yang terpisah untuk mengembangkan setiap versi perangkat lunak yang independen. Uji setiap versi dengan data yang sama untuk memastikan semua versi menghasilkan keluaran yang sama. Jalankan semua versi dengan paralel dan perbandingan keluaran

secara real-time. Walau hanya dijalankan sebuah versi pada akhirnya, untuk beberapa

aplikasi yang penting dapat mengembangkan versi independen dan menggunakan comparison testing atau back-to-back testing. Ketika output dari versi berbeda, maka setiap versi diinvestigasi jika ada kemungkinan defect. Metode ini tidak untuk menemukan kesalahan dari spesifikasi.

Sample and Robustness Testing

- Sample Testing melibatkan beberapa nilai yang terpilih dari sebuah kelas ekivalen, mengintegrasikan nilai pada kasus uji, nilai-nilai yang terpilih mungkin dipilih dengan urutan tertentu atau interval tertentu
- Robustness Testing data input dipilih diluar spesifikasi yang telah didefinisikan Tujuan dari pengujian ini adalah membuktikan bahwa tidak ada kesalahan jika masukan tidak valid

Behavior Testingn dan Performance Testing

- Behavior Testing

Hasil uji tidak dapat dievaluasi jika hanya melakukan pengujian sekali, tapi dapat dievaluasi jika pengujian dilakukan beberapa kali, misalnya pada pengujian struktur data stack

- Performance Testing

Mengevaluasi kemampuan program untuk beroperasi dengan benar dipandang dari sisi acuan kebutuhan misalnya: aliran data, ukuran pemakaian memori, kecepatan eksekusi, dll, untuk mencari tahu beban kerja atau kondisi konfigurasi program, spesifikasi mengenai performansi didefinisikan pada saat tahap spesifikasi atau desain, dapat digunakan untuk menguji batasan lingkungan program

Requirement Testing

Spesifikasi kebutuhan yang terasosiasi dengan perangkat lunak (input/output/fungsi/performansi) diidentifikasi pada tahap spesifikasi kebutuhan dan desain. Requirement testing melibatkan pembuatan kasus uji untuk setiap spesifikasi kebutuhan yang terkait dengan program. Untuk memfasilitasinya, setiap spesifikasi kebutuhan bias ditelusuri dengan kasus uji dengan menggunakan traceability matrix

Endurance Testing

Endurance Testing melibatkan kasus uji yang diulangulang dengan jumlah tertentu dengan tujuan untuk mengevaluasi program apakah sesuai dengan spesifikasi kebutuhan. Contoh:

- Untuk menguji keakuratan operasi matematika (floating point, rounding off, dll)
- Untuk menguji manajemen sumber daya sistem (resources) (pembebasan sumber daya yang tidak benar, dll)

- input/outputs (jika menggunakan framework untuk memvalidasi bagian input dan output)

Spesifikasi kebutuhan pengujian didefinisikan pada tahap spesifikasi kebutuhan atau desain.

Cause-effect Relationship Testing

Teknik ini merupakan suplemen dari equivalence testing dengan menyediakan cara untuk memilih kombinasi data input melibatkan kondisi input (cause) dan kondisi output (effect) untuk mencegah pendefinisian kasus uji yang terlalu banyak. Langkah :

- Bagi-bagi spesifikasi kebutuhan menjadi bagian yang memiliki kemungkinan kerja
- Definisikan cause dan effect berdasarkan spesifikasi kebutuhan
- Analisa spesifikasi kebutuhan untuk membuat hubungan logika
- Tandai graf untuk jalur yang tidak mungkin berhubungan dengan kombinasi cause/effect sesuai dengan batasan spesifikasi kebutuhan
- Ubah graf menjadi tabel keputusan
- kolom --> test case
- baris --> cause/effect
- Ubah kolom dari tabel keputusan menjadi kasus uji