**BAB I**

**PENDAHULUAN**

* 1. **Latar Belakang**

Sejalan dengan perkembangan teknologi informasi, manusia semakin hari semakin dimudahkan dalam mencari atau memperoleh sebuah informasi yang diperlukannya. Dengan hadirnya teknologi yang begitu canggih membuat manusia dapat memperoleh informasi dari mana saja dan kapan pun mereka mau tanpa bersusah payah untuk mendapatkannya. Selain itu juga dengan adanya kemajuan teknologi yang semakin hari semakin canggih membuat pekerjaan manusia menjadi lebih mudah untuk diselesaikan.

Perkembangan suatu lembaga pendidikan merupakan hal yang wajib diperhatikan, karena jika setiap tahunnya jumlah peminat (mahasiswa) yang mendaftar masuk ke lembaga pendidikan tersebut terus meningkat. Maka sudah pasti pihak dari lembaga itu harus mempersiapkan sistem yang lebih canggih lagi untuk membantu mempermudah proses kegiatan akademiknya. Meningkatnya jumlah mahasiswa yang mendaftar pada lembaga pendidikan membuat proses pendaftaran memerlukan tambahan waktu proses.

*World Wide Web Consortium (W3C)*, yang merupakan organisasi yang mengembangkan standart-standart dalam dunia *web*, mendefinisikan *web service* sebagai perangkat lunak sistem yang dirancang untuk mendukung interaksi antar mesin dengan mesin melalui jaringan. Lucky (2008 : 2)

Prosedur yang dimiliki oleh IPI-Leppindo dalam melayani pendaftaran calon mahasiswa baru yaitu bisa secara *offline* dan *online* dan kedua aplikasi tersebut berbeda-beda bahasa pemrogramannya. Namun, kedua aplikasi tersebut tidak terhubung satu sama lain sehingga aplikasi *web* dan aplikasi *desktop* tidak dapat saling berkomunikasi untuk bertukar data maupun informasi.

Mengidentifikasikan masalah pada IPI-LEPPINDO PALEMBANG maka digunakanlah metode *Requirements Engineering*. *Requirements engineering* merupakan cabang dari *software engineering* yang mengurusi masalah yang berhubungan dengan: tujuan (dunia nyata), fungsi, dan batasan-batasan pada sistem *software*. Termasuk hubungan faktor-faktor tersebut dalam menetapkan spesifikasi yang tepat dari suatu *software*, proses evolusinya baik berhubungan dengan masalah waktu maupun dengan *software* lain (dalam satu famili). Untuk menguhubungkan aplikasi itu maka menggunakan *tools* bernama *nusoap* yang berfungsi untuk menghubungkan aplikasi yang berbeda bahasa pemrogramannya.

Sistem yang dikembangkan yaitu pada sistem akademik IPI-LEPPINDO PALEMBANG seperti pendaftaran mahasiswa secara *online* yang menggunakan *web application* dan pendaftaran secara *offline* dengan menggunakan *desktop application*, yang berbeda bahasa pemrogramannya.

Berdasarkan latar belakang diatas maka topik penelitian ini yaitu **“Implementasi *Web Service* dalam Sikronisasi Aplikasi *Desktop* dengan Aplikasi *Web* pada Ipi-Leppindo Palembang”.**

* 1. **Rumusan Masalah**

Berdasarkan uraian pada latar belakang, permasalahan yang teridentifikasi adalah membuat aplikasi pendaftaran pada *web* dan aplikasi pendaftaran pada *desktop* bisa terhubung dan saling bertukar data atau informasi sehingga permasalahan tersebut dirumuskan sebagai berikut: “Bagaimana menerapkan teknologi *web service* pada sistem akademik IPI-LEPPINDO PALEMBANG dengan metode *requirements engineering*”.

* 1. **Batasan Masalah**

Agar pembahasan lebih terarah dan tidak terlalu menyimpang maka permasalahan yang akan dibahas yaitu dibatasi dengan pemanfaatan *web service* pada sistem pendaftaran PMB secara *online* dengan pendaftaran PMB secara *offline.*

* 1. **Tujuan dan Manfaat Penelitian**
     1. **Tujuan Penelitian**

Tujuan penelitian pada Ipi-Leppindo Palembang ini adalah untuk meng-imlementasikan *web service* dalam sikronisasi aplikasi *desktop* dengan aplikasi *web* pada ipi-leppindo palembang”.

* + 1. **Manfaat Penelitian**

Manfaat dari penelitian ini sebagai berikut :

1. Bagi lembaga pendidikan indonesia (Leppindo), dapat mempermudah dalam hal mengakses dan pengolahan data.
2. Bagi calon mahasiswa, dapat mempermudah proses pendaftaran pada Ipi-Leppindo Palembang dimana saja asalkan ada koneksi internet.
3. Bagi penulis, dapat menerapkan ilmu yang telah dipelajari selama ada dibangku kuliah.
   1. **Metode Penelitian**
      1. **Waktu Penelitian**

Waktu penelitian ini dilakukan pada bulan November 2012 sampai dengan bulan Februari 2013.

* + 1. **Tempat Penelitian**

Lokasi penelitian yang dilakukan yaitu di IPI-LEPPINDO PALEMBANG yang beralamat di Jl. Jend Sudirman No.122 (Depan Kodam), Palembang Sumatera Selatan.

* + 1. **Metode Pengumpulan Data**

Metode pengumpulan data pada penelitian ini yaitu :

1. **Wawancara,** Pada metode ini maka peneliti mengumpulkan data dengan cara bertanya langsung kepada pihak yang terlibat dalam penelitian yang sedang dilakukan saat ini dengan harapan pihak yang bersangkutan dapat memberikan berbagai macam informasi yang diperlukan.
2. **Kepustakaan,** Mengumpulkan data dengan cara mencari dan mempelajari buku-buku dan referensi lainnya yang berhubungan dalam penulisan laporan proposal ini. Buku-buku yang digunakan oleh penulis sebagai referensi dalam penelitian ini dapat dilihat pada bagian daftar pustaka.
3. **Observasi,** Observasi adalah suatu metode yang dilakukan dengan cara mengamati langsung keadaan dan kegiatan yang ada pada IPI-LEPPINDO PALEMBANG sebagai objek guna mendapatkan keterangan yang akurat.
   * 1. **Metode Pengembangan Sistem**

Metode yang digunakan pada penelitian ini yaitu menggunakan metode *Requirements Engineering.* Adapun tahapan-tahapan yang dilakukan pada penelian yaitu sebagai berikut:

1. *Requirements elicitation*
2. *Requirements modeling and analysis*
3. *Requirements documentation*
4. *Requirements validation*
5. *Requirements management and evolution*
   1. **Sistematika Penulisan**

Pada Sistematika penulisan skripsi ini yang dilakukan adalah sistematika penulisan yang terdiri atas 5( lima) bab dan pada setiap bab akan membahas hal-hal sebagai berikut :

**BAB I PENDAHULUAN**

Uraian yang terdapat pada bab ini yaitu meliputi latar belakang, perumusan masalah, tujuan penelitian, ruang lingkup dan batasan masalah, manfaat penelitian, metodologi penelitian, dan sistematika penulisan.

**BAB II LANDASAN TEORI**

Pada bab ini berisi ringkasan mengenai teori-teori yang didapat pada penelitian yang sudah ada sebelumnya dan berhubungan dengan judul dan masalah yang diteliti pada skripsi ini.

**BAB III ANALISIS DAN PERANCANGAN SISTEM**

Bagian bab berikut ini yaitu memuat analisis sistem yang akan dibuat nantinya besertaa rancangan desain sistemnya berdasarkan metode pengembangan sistem.

**BAB IV HASIL DAN PEMBAHASAN**

Bab ini menghasilkan sebuah implementasi sistem yang telah dilakukan.

**BAB V KESIMPULAN DAN SARAN**

Bab ini berisi kesimpulan dan saran dari penelitian yang telah dilakukan.

**BAB II**

**LANDASAN TEORI**

* 1. **Metode *Requirements Engineering***

Menurut Zave (1997) memberikan salah satu definisi yang paling jelas dari RE: *Requirements engineering is the branch of software engineering concerned with the real-world goals for, functions of, and constraints on software systems. It is also concerned with the relationship of these factors to precise specifications of software behavior, and to their evolution over time and across software families.*

Terdapat lima aktivitas utama di dalam proses *requirements engineering,* yaitu :

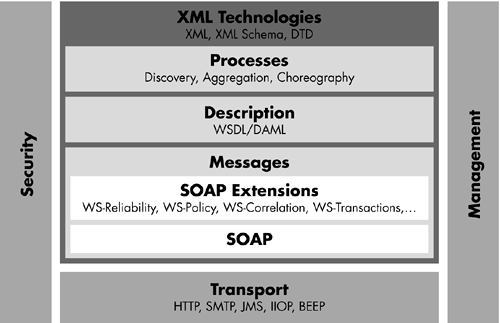
1. *Requirements elicitation*, Pada tahap ini dikumpulkan berbagai *requirement* dari para *stakeholder*. Seorang pelanggan mempunyai masalah yang dapat ditangani oleh solusi berbasis komputer*.*
2. *Requirements analysis and negotiation,* Sebuah model adalah perwakilan dari benda lain yang mempunyai rincian yang cukup untuk membantu penyelesaian tugas-tugas tertentu*.* Data *modeling* bertujuan untuk mendapatkan pengertian dari pemrosesan serta pengaturan informasi.
3. *Requirements documentation, dokumen* ini sebaiknya*:*
4. Hanya menetapkan perilaku sistem sebagaimana terlihat dari luar
5. Menetapkan batasan-batasan *(constraints)* yang diberikan kepada implementasinya.
6. Mudah diubah.
7. Berguna sebagai alat referensi untuk pemeliharaan sistem.
8. Memuat gambaran akan siklus kehidupan sistem di masa yang akan datang.
9. *Requirements validation,* Dalam tahap ini, dokumen dari tahap sebelumnya diperiksa agar memenuhi kriteriakriteria sebagai berikut :
10. Lengkap.
11. Konsisten.
12. Tunduk pada keputusan-keputusan yang diambil pada tahap *requirements analysis.*
13. *Requirements management and evolution* , Sebuahsoftware yang sukses pasti akan berevolusi mengikuti perubahan lingkungannya. Sebaliknya*, software* yang sudah tidak diperbaharui berarti telah ditinggalkan oleh para penggunanya. Dalam perjalanan evolusi sebuah *software, requirement* akan *software* tersebut akan bertambah*,* berubah, atau terkadang berkurang. Agar perubahan ini terkendali, perlu adanya aktivitas *requirements management (RM).*
    1. ***Web Service***

*World Wide Web Consortium (W3C),* yang merupakan organisasi yang mengembangkan standart-standart dalam dunia *web*, mendefinisikan *web service* sebagai perangkat lunak sistem yang dirancang untuk mendukung interaksi antara mesin dengan mesin melalui jaringan. Lucky (2008 : 2)

*Web service* merupakan sebuah entitas komputasi yang dapat diakses melalui jaringan internet maupun intranet dengan standart *protokol* yang sudah ditentukan dalam *platform* dan antarmuka bahasa pemprograman yang independen. Tujuan pengembangannya adalah untuk “menjembatani komunikasi antar program”, sehingga aplikasi yang satu dan aplikasi lain yang terdapat suatu jaringan yang sama atap pada jaringan berbeda dapat saling berkomunikasi asalkan menggunakan standar *protokol* yang ditetapkan oleh *web service*. Hal ini bisa terjadi, karena standar *protokol* itu tidaklah terikat pada suatu *platform* atau bahasa pemrograman. *Protokol* itu sendiri dibangun oleh *Extensible Markup Language* (XML) yang memang kenyataanya telah didukung oleh banyak *platform*, bahasa pemograman, dan oleh developer di seluruh dunia.

Beberapa alasan mengapa digunakannya *web service* yaitu:

1. *Web service* bisa digunakan mentransformasika satu atau beberapa bisnis *logic* atau *class* dan objek yang terpisah pada suatu ruang lingkup yang telah menjadi satu, sehingga tingkat keamanannya dapat diatasi dengan baik.
2. *Web Service* juga mempunyai kemudahan pada proses *deployment*-nya, karena *web service* tidak memerlukan registrasi (mendaftar) khusus ke dalam suatu sistem operasi yang bersangkutan. *Web service* hanya cukup di *upload* ke *web server* dan siap di akses oleh pihak-pihak yang telah diberikan otorisasi / izin.
3. *Web Service* tidak memerlukan konfigurasi khusus di sisi *firewalll* karena *web service* berjalan pada *port* 80 yang merupakan *protokol* standar HTTP.
   * 1. **Arsitektur *Web Service***

Arsitektur *web service* dibangun oleh beberapa layer dan teknologi yang saling berhubungan. Banyak cara untuk memvisualisasikan layanan (*service*), sama banyaknya dengan cara untuk membangun dan menggunakan *web service.*Gambar berikut ini merupakan salah satu cara untuk menggambarkan arsitekrut dari *web service*.

*Web service* disusun oleh beberapa komponen standar, yaitu sebagai berikut.

1. *Extnsible Markup Language* (XML)

XML merupakan dasar yang penting atas terbentuknya *Web Service*. *Web Service* dapat berkomunikasi dengan aplikasi-aplikasi yang memanggilnya dengan menggunakan XML. Karena XML berbentuk teks sehingga mudah untuk ditransportasikan menggunakan *protokol* HTTP. Selain itu XML juga bersifat *platform* independen sehingga informasi di dalamnya bisa dibaca oleh aplikasi apapun pada *platform* apapun selama aplikasi tersebut menerjemahkan tag-tag XML.

1. SOAP atau *Simple Object Access Protokol,* merupakan suatu format standart dokumen berbentuk XML yang digunakan untuk melakukan proses *request* dan *response* antara *web service* dengan aplikasi yang memanggilnya.
2. WSDL atau *Web Service Definition Language*, sebelum mengakses *Web Service,* tentu perlu mengetahui method-method apa saja yang disediakan oleh *Web Service* tersebut. Untuk mengetahuinya maka diperlukan sebuah dokumen yang bernama WSDL. WSDL merupakan sebuah dokumen dalam format XML yang isinya menjelaskan informasi detail sebuah *web service.*
3. *Universal Description, Discovery, and Integration* (UDDI) merupakan salah satu komponen pendukung *web service* yaitu UDDI. UDDI merupakan *directory service* yang digunakan untuk meregistrasikan dan mencari *Web Service.*
   * 1. **Keamanan *Web Service***

*Web service* bersifat *open standart* (standar pada *protokol* *universal* dan diketahui oleh semua pihak) dan belum memiliki sistem penanganan keamanan [Pullier, Eric, Taylor, Hugh. Understanding Enterprise SOA.Manning 2006]. Untuk itu pada sistem keamanannya menjadi perhatian serius bagi pengembang aplikasi jika mengimplementasikan *web service.*

Beberapa alasan mengapa digunakannya *web service* yaitu:

1. *Web service* bisa digunakan mentransformasika satu atau beberapa bisnis *logic* atau *class* dan objek yang terpisah pada suatu ruang lingkup yang telah menjadi satu, sehingga tingkat keamanannya dapat diatasi dengan baik.
2. *Web Service* juga mempunyai kemudahan pada proses *deployment*-nya, karena *web service* tidak memerlukan registrasi (mendaftar) khusus ke dalam suatu sistem operasi yang bersangkutan. *Web service* hanya cukup di *upload* ke *web server* dan siap di akses oleh pihak-pihak yang telah diberikan otorisasi / izin.
3. *Web Service* tidak memerlukan konfigurasi khusus di sisi *firewalll* karena *web service* berjalan pada *port* 80 yang merupakan *protokol* standar HTTP.
   1. **Aplikasi**

Aplikasi berasal dari *Application* yang menurut kamus komputer eksekutif (1993:9), *Application* adalan masalah yang memakai teknik pemrosesan data aplikasi biasanya mengacu pada komputasi yang diinginkan, atau pemrosesan data. Pengertian Aplikasi menurut Yan Tirtobisono (1999:12) adalah istilah yag digunakan untuk pengguna komputer bagi pemecahan masalah. Biasanya istilah sebuah aplikasi dipadukan dengan *software* contohnya *Visual Basic 6.0,* yang akan memberikan arti atau makna yaitu suatu program yang dibuat untuk menyelesaikan masalah tertentu.

Menurut Widodo (2001:19) Aplikasi adalah suatu sub kelas perangkat lunak kompute yang memanfaatkan kemampuan komputer langsung untuk melakukan suatu tugas yang diinginkan pengguna.

Sedangkan menurut Jogiyanto (2000:12), aplikasi adalah penggunaan dalam suatu komputer, instruksi (instruction) atau pernyataan *(statment*) yang disusun sedemikian rupa sehingga komputer dapat memproses input menjadi *output.*

* + 1. **Aplikasi *Desktop***

Menurut Konixbam (2009) *Desktop Based Application* adalah suatu aplikasi yang dapat berjalan sendiri atau independen tanpa menggunakan *browser* atau koneksi *Internet* di suatu komputer otonom dengan *operating system* atau *platform* tertentu. Aplikasi *Desktop* difokuskan kepada aplikasi yang lebih independen. Hal tersebut bertujuan untuk mempermudahkan para pengguna aplikasi *desktop* dalam hal memodifikasi pengaturan aplikasi sehingga efektifitas, efesinsi waktu, dana, dan tenaga dapat lebih ditekankan semaksimal mungkin.

Secara garis besar pada pemrograman terutama pada aplikasi yang berbasis *desktop* dapat dibedakan menjadi dua macam yaitu pemrograman konvensional dan pemrograman visual.

1. Pemrograman konvensional merupakan metode mendesain suatu aplikasi, pemrograman dituntut untuk bisa menerapkan baris demi baris kode program agar bisa menghasilkan sebuah bentuk tampilan aplikasi yang dibuat dan akan memakan waktu lama.
2. Pemrograman visual merupakan metode pembuatan program dimana seorang programmer membuat koneksi antar objek-objek dengan cara menggambar, menunjuk, dan mengkilik pada diagram dan ikon dengan berinteraksi denga diagram jalur.

Beberapa keunggulan yang dimiliki oleh aplikasi *desktop* yaitu:

1. Dapat berjalan dengan independen, tidak perlu menggunakan sebuah *web browser.*
2. Tidak memerlukan koneksi internet
3. Prosesnya lebih cepat dibanding aplikasi *web*

Kekurangnnya yaitu :

1. Harus menginstal aplikasinya terlebih dahulu jika ingin menjalankannya
2. Bermasalah pada lisenci karena membutuhkan banyak lisensi pada setiap komputer yang bereda-beda
3. Biasayan memerlukan *hardware* yang mempunyai kualitas yang baik atau cukup tinggi.
   * 1. **Aplikasi *Web***

Menurut Roger S. Pressman (2005), *web-application* adalah satuan aplikasi yang cukup luas, pada bentuk paling sederhana, *web-application* dapat berupa serangkaian *hypertext files* yang terhubung yang memberikan informasi berupa *text* dengan sedikit gambar/grafik.

Aplikasi web sendiri terbagi menjadi dua bagian yaitu :

1. Web statis, dibentuk dengan menggunakan HTML saja. Kekurangan yang dimiliki aplikasi *web statis* yaitu terletak pada pemeliharaan program secara terus menerus yang harus mengikuti setiap perubahan yang akan terjadi. Kelemahan yang dimiliki oleh *web statis* ini dapat diatasi dengan model aplikasi berupa *web dinamis.*
2. Web dinamis, dengan memperluas kemampuan yang dimiliki HTML yaitu dengan cara menggunakan *software* tambahan, perubahan suatu informasi pada halaman-halaman web dapat diatasi dengan melakukan perubahan data, bukan melakukan perubahan program.
   * + 1. **Atribut Aplikasidan Sistem *Web***

Menurut Pressman (2005: 502) *Web application* berbeda dari kategori lainnya dalam perangkat lunak komputer. Atribut yang mengikuti sangat banyak ditemui di dalam *web application* adalah *Network intensiveness, Concurrency, Unpredicable load, Performance, Availability, Data driven, Content sensitive, Continuous evolution, Immediacy, Security,* dan  *Aesthetics.* Sedangkan yang termasuk ke dalam kategori aplikasi yang sering ditemui di dalam kerja *web engineering* adalah *Informational, Download, CustomizeableInteraction, User input, Transaction-oriented, Service-oriented, Portal, Database access, Data warehouse.*

* + - 1. **Komponen Pemrograman *Web***

Dalam pemrograman *web* sangat diperlukan beberapa komponen penting yang harus terdapat dalam sebuah komputer yaitu sebagai berikut :

1. *Web Browser*, merupakan perangkat lunak yang berfungsi untuk menjalankan aplikasi *web.*
2. *Web Server*, Sebuah perangkat yang harus dimiliki untuk membuat *web* dinamis, misalnya *Apache Web Server.*
3. *Database Server*, merupakan sebuah tempat penyimpanan data dalam *web*.
4. *Web Editor*, merupakan perangkat lunak yang berfungsi untuk membuat, mengedit atau menghapus perintah-perintah script yang digunakan, salah satu contohnya *Macromedia Dreamweaver.*
5. *Image Editor,* Perangkat lunak yang digunakan untuk mengedit atau mengatur gambar dan animasi yang akan digunakan dalam aplikasi *web,* seperti *Adobe Photoshop.*
   1. ***Unified Modeling Language* (UML)**

Menurut munawar ( 2005 : 17 ), *Unified Modeling Language* (UML) adalah salah satu alat bantu yang sangat handal dalam perkembangan sistem beroritasi objek. Hal ini disebabkan karena UML menyediakan bahasa pemodelan *visual* yang memungkinkan bagi pengembang *system* untuk membuat cetak biru atau visi mereka dalam bentuk yang baku, mudah dimengerti serta dilengkapi dengan mekanisme yang efektif untuk berbagi (*sharing*) dan mengkomunikasikan rancangan mereka dengan yang lain.

Menurut kristanto (2004 : 113), UML *(Unified Modeling Language)* adalah sebuah bahasa pemrograman yang telah menjadi standart untuk merancang dan mendokumentasikan sistem perangkat lunak. UML menawarkan sebuah standart yang dibuat untuk merancang model sebuah sistem yang diinginkan dan sudah digunakan oleh orang secara luas.

* + 1. ***Use Case Diagram***

Menurut Munawar (2005 : 63 ) *Use Case* adalah deskripsi fungsi dari sebuah system dari perspektif pengguna. *Use case* bekerja dengan cara mendeskripsikan tipikal interaksi antara *user* (pengguna) sebuah *system* dengan sistemnya sendiri melalui sebuah cerita bagaimana sebuah *system* dipakai. Urutan langkah-langkah yang menerangkan antara pengguna dan *system* disebut skenario. Setiap skenario mendeskripsikan urutan kejadian. *Use case* diagram menampilkan *actor, use case,* dan hubungan antara meraka:

1. Aktor mana yang menggunakan *use case* mana.
2. *Use case* mana yang memasukkan *use case* lain.

**Tabel 2.1**. Simbol *use case Diagram.*

|  |  |  |
| --- | --- | --- |
| No | Simbol | Keterangan |
| 1 | *Aktor* | *Actor* merupakan pemain atau pengguna sistem yang memperhatikan himpunan atau sesuatu yang berinteraksi dengan sistem yang akan di kembangkan. |
| 2 | *Use Case* | Menspesifikasikan fungsi dari suatu sistem. |
| 3 | ------------------------>  *Include* | Menspesifikasikan bahwa *use case* sumber secara *eksplisit.* |
| 4 | ------------------------  *Extend* | Menspesifikasikan bahwa *use case* target memperluas perilaku dari *use case* sumber pada suatu titik yang diberikan. |
| 5 | *Association* | Apa yang menghubungkan antara objek satu dengan yang lainnya. |

Sumber : Munawar (2005)

* + 1. ***Activity Diagram***

Menurut Munawar (2005 : 109) *Activity diagram* adalah teknik untuk mendeskripsikan logika *procedural*, proses bisnis dan aliran kerja dalam banyak kasus. *Activity Diagram* mempunyai peran seperti halnya *flowchart*, akan tetapi perbedaannya dengan *flowchart* adalah *activity diagram* bias mendukung perilaku paralel sedang *flowchart* tidak bisa.

Menurut Whitten et al. (2004:419), *activity diagram* secara grafis digunakan untuk menggambarkan rangkaian aliran aktivitas use case. *Activity Diagram* dapat digunakan untuk memodelkan *action* yang akan dilakukan saat sebuah operasi dieksekusi dan memodelkan hasil dari *action* tersebut.

Sedangkan menurut Nugroho (2005:61), *activity diagram* adalah cara untuk memodelkan *event-event* yang terjadi dalam suatu *use case*. *Activity diagram* digunakan untuk memodelkan aspek dinamis dari sistem. *Activity diagram* secara essensial mirip dengan diagram alir (*flowchart)*. Activity diagram menekankan aliran kontrol dari suatu aktivitas ke aktivitas lainnya.

**Tabel 2.2** Simbol *Activity Diagram*

|  |  |  |
| --- | --- | --- |
| No. | Nama dan Simbol | Keterangan |
| 1 | Start | Mendeskripsikan suatu tindakan sebelum aktivitas dimasukan. |
| 2 | Arus Kegiatan | Mendeskripsikan ke mana aliran kegiatan. |
| 3 | Proses / Kegiatan | Mendeskripsikan tentang suatu tindakan aktivitas, proses kegiatan. |
| 4 | *Decisions* | Mendeskripsikan tentang suatu tindakan untuk menghasilkan keputusan. |
| 5 | *Fork* | Menunjukkan kegiatan yang dilakukan secara parelel. |
| 6 | *Final Activity* | Mendeskripsikan suatu tindakan sesudah aktivitas. |

Sumber : Munawar (2005)

* + 1. ***Class Diagram***

Menurut Whitten et al. (2004:418), *class* diagram menggambarkan struktur objek sistem*. Class* diagrammenggambarkan kelas objek yang menyusun sistem dan juga hubungan antara kelas objek tersebut. Sedangkan menurut Nugroho (2005:110), *class* diagram adalah diagram yang digunakan untuk menampilkan beberapa kelas serta paket-paket yang ada dalam sistem/perangkat lunak yang sedang dikembangkan. *Class* diagram memberikan gambaran/diagram statis tentang sistem/perangkat lunak dan relasi-relasi yang ada di dalamnya.

*Class diagram* menggambarkan struktur dan deskripsi *class, package* dan objek beserta hubungan satu sama lain seperti pewarisan, *containmen*t, asosiasi, dan lain-lain. *Class* diagram juga berfungsi untuk menggambarkan atau menjelaskan hubungan antar *class-class* pada sebuah sistem yang sedang dibuat sehingga dapat mengetahui bagaimana caranya mereka saling berkolaborasi untuk mencapai sebuah tujuan tertentu.

*Class* diagram memiliki 3 (tiga) area pokok (utama) yaitu nama, atribut, dan operasi. Namaberfungsi untuk member identitas pada sebuah *class.* Atribut berfungsi untuk member karakteristik pada data yang dimiliki suatu objek di dalam *class*. Operasi berfungsi untuk menjelaskan atau memberikan sebuah fungsi ke sebuah objek.

* + 1. ***Statechart Diagram***

*Statechart diagram* menggambarkan transisi dan perubahan keadaan (dari satu *state* ke *stat*e lainnya) suatu objek pada system sebagai akibat dari stimuli yang diterima. Pada umumnya *statechart* diagram menggambarkan *class* tertentu (sutu class dapat memiliki lebih dari satu *statechart diagram*). Menurut Whitten et al. (2004:419), *statechart* diagram digunakan untuk memodelkan behavior objek khusus yang dinamis. *Statechart* diagram mengilustrasikan siklus hidup objek, yaitu berbagai keadaan yang dapat diasumsikan oleh objek dan *event-event* yang menyebabkan objek beralih dari satu *state* ke state lain.

Sedangkan menurut Nugroho (2005:19), *statechart diagram* memperlihatkan *state-state* pada sistem, memuat *state,* transisi, event, serta aktivitas. *Statechart* diagram penting untuk memperlihatkan sifat dinamis dari antarmuka (*interface*), kelas, kolaborasi, dan penting pada pemodelan sistem-sistem yang reaktif. Perilaku dinamis suatu objek adalah perilaku berbeda yang dimiliki suatu objek sesuai dengan perubahan nilai atribut yang dimilikinya.

Sedangkan perilaku dinamis interface tergantung dengan relasi-relasi yang dimiliki suatu kelas. Relasi yang memiliki *multiplisitas* 0 (nol), mengindikasikan bahwa relasi yang bersangkutan bersifat pilihan (optional). Dengan demikian objek atau *interface* memiliki perilaku berbeda saat relasi tidak ada (multiplisitas nol) dan saat ada relasi (multiplisitas lebih dari nol).

* + 1. ***Sequence Diagram***

Menurut Whitten et al. (2004:419), *sequence diagram* secara grafis menggambarkan bagaimana objek berinteraksi satu sama lain melalui pesan pada eksekusi sebuah *use case* atau operasi. *Sequence diagram* mengilustrasikan bagaimana pesan terkirim dan diterima di antara objek dan dalam sekuensi/rangkaian.

Sedangkan menurut Nugroho (2005:92), *sequence diagram* memperlihatkan *event-event* yang berurutan sepanjang berjalannya waktu. Masing-masing sequence diagram menggambarkan aliran-aliran pada suatu use case dengan melihat objek-objek dan pesan-pesan. Setiap objek memiliki garis hidup (*lifeline*), yang digambarkan sebagai garis vertikal di bawah nama suatu objek. *Lifeline* dimulai saat suatu objek terbentuk (diinstansiasi) dan berakhir saat objek yang bersangkutan dihancurkan. Pesan-pesan digambarkan di antara *lifeline* yang dimiliki dua objek untuk memperlihatkan bagaimana objek-objek itu saling berkomunikasi.

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

* + 1. ***Collaboration Diagram***

*Collaboration diagram* menurut Munawar (2005:101) adalah perluasan dari objek diagram. Objek diagram menunjukan objek-objek yang hubungannya satu dengan yang lain. *Collaboration diagram* menunjukkan *message-message* objek yan dikirim satu sama lain.

*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 numbe*r, di mana *massage* dari level tertinggi memiliki 1, *messages* dari level yang sama memiliki prefiks yang sama.

Dalam suatu *collaboration diagram* dimana untuk merepresentasikan urutan-urutan proses yang ada maka dapat dilakukan dengan cara pemberian nomor pada label setiap pesan(*message)*, dimana nomor tersebut berhubungan dengan urutan waktu proses.

* + 1. ***Component Diagram***

*Commponent diagram* menurut Munawar (2005:119) , mempersentsikan dunia rill item yaitu *component software. component software* adalah bagian fisik dari sebuah sistem karena menetap di komputer. *component software* mengandung *component, interface* dan *relationship.*

*Component diagram* menjelaskan atau menggambarkan struktur fisik kode dari komponen kodede. Suatu komponen bisa berupa komponen source code, komponen *executable* atau komponen binar. Komponen mempunyai informasi mengenai *logical class*. Pada ketergantungan diantara komponen dapat memudahkan untuk menganalisa bagaimana komponen yang lain dapat terpengaruh dengan perubahan sebuah *component.*

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

* + 1. ***Deployment Diagram***

*Deployment/Physical diagram* menurut Munawar (2005:125), menunjukan tata letak sebuah sistem secara fisik, manampakkan bagian-bagian *software* yang berjalan pada bagian-bagian *hardware.*

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

Dengan model-model yang baik kita dapat menggunakan seluruh cara dari sebuah *node* pada arsitektur fisik ke komponen *ke class* implementasi ke objek yang sedang berinteraksi dan selajutnya ke dalam *use case.*

* 1. **PHP**

PHP singkatan dari PHP *Hypertexy Preprocessor* yang digunakan sebagai bahasa *script* *server-side* dalam pengembangan *Web* yang disisipkan pada dokumen HTML. Penggunaan PHP memungkinkan *web* dapat dibuat dinamis sehingga *maintanance* situs *web* tersebut menjadi lebih mudah dan efisien. Kasiman Peranginangin (2006 : 2)

Menurut Swastika (2006) PHP merupakan bahasa berbentuk skrip yang ditempatkan dalam *server* dan diproses di *server.* Hasilnya akan dikirimkan ke *client*, tempat pemakai menggunakan *browser*. PHP dikenal sebagai sebuah bahasa *scripting*, yang menyatu dengan tag-tag *HTML*, dieksekusi di *server*, dan digunakan untuk membuat halaman *web* yang dinamis seperti halnya *Active Server Pages (ASP)* atau *Java Server Pages (JSP).* PHP merupakan sebuah *software open source*.

Abdul Kadir (2002:512), “menyatakan PHP merupakan skrip yang dijalankan di *server*, dimana kode yang menyusun program tidak perlu diedarkan ke pemakai sehingga kerahasiaan kode dapat dilindungi”.

* 1. ***Visual Basis* 6.0**

Kata *visual* menunjukan cara yang digunakan untuk membuat *graphical user interface* (GUI). Dengan cara ini maka tidak perlu lagi menuliskan instruksi pemrograman dalam kode-kode baris, tetapi secara mudah dapat melakukan *drag* dan *drop* objek-objek yang akan digunakan. Kata *Basic* merupakan bagian bahasa *basic* (*Beginners All Purpose Symbolic Instruction Code*), yaitu sebuah bahasa pemrograman yang dalam sejarahnya sudah bayak digunakan oleh programer untuk menyusun aplikasi. Andi (2003 : 1)

Menurut Widodo Budiharto, *Visual Basic* adalah bahasa pemrograman *event-driven* yang berasal dari *Basic. Event driven* artinya program menunggu sampai adanya respon dari pemakai berupa kejadian tertentu, misalnya tombol diklik, atau menu dipilih. Ketika *Event driven* terdeteksi, *event* yang berhubungan akan melakukan aksi sesuai dengan kode yang diberikan.

*Visual Basic* menurut Krisna D. Oktovhina (2003:1) adalah suatu bahasa pemrograman komputer, bahasa pemrograman komputer adalah perintah-perintah yang dimengerti oleh komputer untuk melakukan tugas-tugas tertentu. Sedangkan menurut Wahana (2001:2) menyatakan bahwa *Microsoft Visual Basic 6.0* adalah suatu bahasa pemrograman yang bersifat *objek oriented*.

* 1. **MySQL**

MySQL (*My Structure Query Language*) merupakan sebuah program pembuat *database* yang bersifat *open source*, artinya semua orang dapat menggunakannya dan dapat dijalankan pada semua *patform* baik *window* maupun *linux*. MySQL juga merupakan sebuah perangkat lunak sistem manajemen basis data SQL yang bersifat jaringan sehingga dapat digunakan untuk aplikasi *multi user* (banyak pengguna). Andri Kristanto(2010: 219)

Menurut Abdul Kadir (2008) *MySQL* merupakan *software* yang tergolong sebagai *DBMS (Database Management System)* yang bersifat *open source*. *Open source* menyatakan bahwa *software* ini dilengkapi dengan *source code* (kode yang dipakai untuk membuat *MySQL*), selain tentu saja bentuk sistem operasi, dan bisa diperoleh dengan cara men-*download* (mengunduh) di internet secara gratis.

Menurut Allen dan Hornberger (2002, p220) MySQL merupakan bahasa pemrograman *open source* yang paling populer dan banyak digunakan dilingkungan Linux. Kepopuleran ini karena ditunjang oleh performasi *query* dari *database*-nya yang jarang bermasalah.

* 1. **HTML**

HTML (*Hypertext Mark Up Language*) bukan bahasa pemrograman tetapi lebih merujuk ke bahasa *markup* yang digunakan untuk menentukan format atau *style* dari teks yang ditandai. Fendy Novento (204: 2)

Dokumen-dokumen HTML disimpan di dalam web server, yang mendistribusikan file-file web browser yang terhubung ke web tersebut. File-file HTML kemudian ditampilkan kepada pengguna seperti apa adanya, dan juga diinterpretasi oleh web browser, yag menciptakan halaman web tersebut menurut instruks-instruksi yang merupakan bagian dari kode HTML (Amor, 2002, p429).

HTML mudah dipelajari dan diimplementasikan, sehingga dapat digunakan oleh perusahaan manapun untuk mempresentasikan dirinya secara multimedia dan sangat interaktif. HTML juga dapat mengurangi biaya secara signifikan. Hal ini juga dapat mengurangi penggunaan kertas yang digunakan untuk mencetak tulisan (printing) (Amor, 2002, P434).