**BAB II**

**LANDASAN TEORI**

1. **Aplikasi**

Menurut Martin Fowler (2002). Dalam bukunya yang berjudul Patterns of Enterprise Application Architecture, menjelaskan bahwa Aplikasi, dalam konteks teknologi informasi, merujuk pada perangkat lunak yang dirancang untuk memenuhi kebutuhan tertentu dari pengguna. Aplikasi ini dapat bervariasi dalam kompleksitas dan fungsi, dari program sederhana seperti pengolah kata hingga sistem yang lebih kompleks seperti perangkat lunak manajemen perusahaan.

**2.2 Sistem Informasi**

Menurut Jogiyanto HM (2005). Dalam bukunya yang berjudul Analisis & Desain, menjelaskan bahwa “Sistem adalah suatu jaringan kerja dari prosedur – prosedur yang saling berhubungan, berkumpul bersama – sama untuk melakukan suatu kegiatan atau menyelesaikan suatu sasaran yang tertentu.”

Menurut Jogiyanto HM (2002) Informasi dapat didefenisikan data yang diolah menjadi bentuk yang lebih berguna dan lebih berarti bagi yang menerimanya.

Untuk menghasilkan informasi yang berkualitas maka dibuatlah sistem informasi. Sistem informasi didefenisikan oleh Robert A. Laitch dan K.Roscoe Bavis dalam buku karangan Jogiyanto HM (2005) “Sistem informasi adalah suatu sistem di dalam suatu organisasi yang mempertemukan kebutuhan pengolahan transaksi harian, mendukung operasi, bersifat manajerial dan kegiatan strategi dari suatu organisasi dan menyediakan pihak luar tertentu dengan laporan-laporan yang diperlukan.” Semua sistem informasi memiliki tiga kegiatan utama, yaitu:

1. Menerima data sebagai masukan (input);
2. Melakukan pemrosesan dengan mengerjakan perhitungan, penggolongan,unsur data, dan pemutakhiran (updating); dan
3. Memperoleh informasi sebagai keluaran (output).

**2.3 Pemodelan Berorientasi Objek (UML)**

Menurut Nugroho (2005) Pemodelan (modeling) adalah proses merancang piranti lunak sebelum melakukan pengkodean (coding). Membuat moel dari sebuah sistem yang kompleks sangat penting agar dapat memahami sistem secara menyeluruh. Semakin kompleks sebuah sistem, semakin penting pula penggunaan tehnik pemodelan yang baik. Dengan menggunakan model, diharapkan pengembangan piranti lunak dapat memenuhi semua kebutuhan pengguna dengan lengkap dan tepat.

Berdasarkan penjelasan di atas penulis menggunakan perancangan sistem dengan pemodelan berorientasi objek menggunakan Unified Modelling Language (UML).

Nugroho (2005) berpendapat bahwa UML merupakan bahasa untuk visualisasi, spesifikasi, kontruksi, serta dokumentasi. Sejalan dengan itu, dengan menggunakan UML dapat dibuat model untuk semua jenis aplikasi piranti lunak dimana aplikasi tersebut dapat berjalan pada piranti keras, sistem operasi dan jaringan apapun serta ditulis dalam bahasa pemograman apapun, karena UML juga menggunakan *class* dan operation dalam konsep dasarnya, maka lebih cocok untuk penulisan piranti lunak dalam bahasa berorientasi objek. Walaupun demikian, UML tetap dapat digunakan untuk modeling aplikasi prosedural dalam VB atau C.

Setiap sistem yang kompleks seharusnya bisa dipandang dari sudut yang berbeda-beda sehingga bisa didapatkan pemahaman secara menyeluruh, UML menyediakan sembilan jenis diagram yaitu diagram kelas, diagram objek, Use Case Diagram, Sequence Diagram, Collaboration Diagram, Statechart Diagram, Activity Diagram, component diagram, dan Deployment Diagram (Nugroho,2005).

1) Use Case Diagram

Use Case diagram bersifat statis, diagram ini memperlihatkan himpunan use case dan aktor-aktor (suatu jenis khusus dari kelas). Diagram ini sangat penting untuk mengorganisasi dan memodelkan perilaku dari suatu system yang dibutuhkan serta diharapkan pengguna (Nugroho, 2005). Simbol-simbol yang digunakan dalam Use Case Diagram dijelaskan

|  |  |  |
| --- | --- | --- |
| No | Simbol | Keterangan |
| 1 | A. Use Case Diagram Merupakan orang, proses atau sistem lain yang  berinteraksi dengan sistem yang akan dibuat. Jadi walaupun si | Aktor: Mewakili peran orang, system yang lain, atau alat ketika berkomukasi dengan usecase |
| 2 |  | Use Case: Abstraksi dan interaksi antara system dan actor |
| 3 |  | Association: Abstraksi dari penghubung antara actor dengan use case |
| 4 |  | Generalisasi: Menunjuka spesialisasi actor untuk dapat berpartisipasi dengan use case |
| 5 | <<include>> | Menunjukan bahwa suatu use case seluruhnya merupakan fungsionalitas dari use case lainnya |
| 6 | <<extend>> | Menunjukan bahwa suatu use case merupakan tambahan fungsional dari use case lainnya jika suatu kondisi terpenuhi |

Tabel 2.1 *Use case* Diagram

2) *Class* Diagram

*Class* adalah sebuah spesifikasi yang akan menghasilkan objek dan merupakan inti dari pengembangan dan desain berorientasi objek. *Class* menggambarkan keadaan (atribut/properti) suatu sistem. *Class* diagram menggambar struktur dan deskripsi kelas, package beserta hubungan satu sama lain Tohari, (2014).

Simbol-simbol yang digunakan dalam *class* diagram dijelaskan pada tabel berikut:

|  |  |  |  |
| --- | --- | --- | --- |
| No | Gambar | Nama | Keterangan |
| 1 |  | *Generalization* | Hubungan dimana objek anak (decendent) berbagi perilaku dan struktur data dari objek yang ada di atasnya objek induk (ancestor). |
| 2 |  | *Nary Association* | Upaya untuk menghindari asosiasi dengan lebih dari 2 objek |
| 3 |  | *Class* | Himpunan dari objek – objek yang berbagi atribut serta operasi yang sama |
| 4 |  | *Collaboration* | Deskripsi dari urutan aksi-aksi yang ditampilkan system yang menghasilkan suatu hasil yang terukur bagi suatu aktor |
| 5 |  | *Realization* | Operasi yang benar-benar dilakukan oleh suatu objek |
| 6 |  | *Despendency* | Hubungan dimana perubahan yang terjaadi pada suatu elemen mandiri (independent) akan mempengaruhi elemen yang bergantung padanya elemen yang tidak mandiri |
| 7 |  | *Association* | Menghubungkan antara objek satu dengan objek yang lainnya |

Tabel 2.2 *Class* Diagram

3). Activity Diagram

Hamim Tohari (2014) berpendapat bahwa, Activity diagram worklfow proses bisnis dan urutan aktivitas dalam sebuah proses. Diagram ini sangat mirip dengan flowchart karena memodelkan workflow dari satu aktivitas ke aktivitas lainnya atau dari aktivitas ke status. Membuat activity diagram pada awal pemodelan proses cukup menguntungkan untuk membantu memahami keseluruhan proses. Activity diagram juga bermanfaat untuk menggambarkan parallel behaviour atau menggambarkan interaksi antara beberapa use case.

|  |  |  |  |
| --- | --- | --- | --- |
| No | Simbol | Nama | Keterangan |
| 1 |  | Status awal | Sebuah diagram aktivitas memiliki sebuah status awal |
| 2 |  | Aktivitas | Aktivitas yang dilakukan system, aktivitas biasanya diawali dengan kata kerja |
| 3 |  | Percabangan/  Decision | Percabangan dimana ada pilihan aktivitas yang lebih dari satu |
| 4 |  | Pergabungan/  Join | Pergabungan dimana yang lebih dari satu aktivitas lalu digabungkan menjadi satu |
| 5 |  | Status akhir | Status aklhir yang dilakukan sebuah system, sebuah diagram aktivitas memiliki sebuah status akhir |
| 6 |  | Swimlane | Swimlane memisahkan organisasi bisnis yang bertanggung jawab terhadap aktivitas yang terjadi |

Tabel 2.3 *Activity* Diagram

2) *Sequence* Diagram

Sequence diagram adalah alat visual dalam pemodelan sistem yang menunjukkan interaksi antar objek dalam urutan tertentu, menggambarkan bagaimana proses berjalan dari satu objek ke objek lain. Diagram ini biasanya digunakan dalam pengembangan perangkat lunak untuk memperjelas alur komunikasi dalam sistem. (Martin Fowler, 2018)

|  |  |  |  |
| --- | --- | --- | --- |
| No | Simbol | Nama | Keterangan |
| 1 |  | Entity class | Gambaran system sebagai landasan dalam Menyusun basis data |
| 2 |  | Boundary Class | Menangani komunikasi antar lingkungan sistem |
| 3 |  | Control Class | Bertanggung jawab terhadap kelas-kelas terhadap objek yang berisi logika |
| 4 | Creating a Recursive Message in a Sequence Diagram | Recursive | Pesan untuk dirinya |
| 5 |  | Activation | Mewakili proses durasi aktivasi sebuah operasi |
| 6 |  | Lifeline | Komponen yang digambarkan garis putus terhubung dengan objek |

Tabel 2.4 *Sequence* Diagram

Simbol-simbol yang digunakan dalam *class* diagram dijelaskan pada tabel berikut:

|  |  |  |  |
| --- | --- | --- | --- |
| No | Gambar | Nama | Keterangan |
| 1 |  | *Generalization* | Hubungan dimana objek anak (decendent) berbagi perilaku dan struktur data dari objek yang ada di atasnya objek induk (ancestor). |
| 2 |  | *Nary Association* | Upaya untuk menghindari asosiasi dengan lebih dari 2 objek |
| 3 |  | *Class* | Himpunan dari objek – objek yang berbagi atribut serta operasi yang sama |
| 4 |  | *Collaboration* | Deskripsi dari urutan aksi-aksi yang ditampilkan system yang menghasilkan suatu hasil yang terukur bagi suatu aktor |
| 5 |  | *Realization* | Operasi yang benar-benar dilakukan oleh suatu objek |
| 6 |  | *Despendency* | Hubungan dimana perubahan yang terjaadi pada suatu elemen mandiri (independent) akan mempengaruhi elemen yang bergantung padanya elemen yang tidak mandiri |
| 7 |  | *Association* | Menghubungkan antara objek satu dengan objek yang lainnya |

Menjelaskan Aplikasi: https://martinfowler.com/books/eaa.html