BAB II

LANDASAN TEORI

2.1. Konsep Dasar Sistem

A. Sistem

Dalam konsep dasar sistem terdapat beberapa pendekatan sistem yaitu pendekatan yang menekankan pada prosedur dan pendekatan pada elemen-elemen atau komponennya.

Menurut Hutahaean dalam (Sahara, 2018:15) “Sistem Informasi adalah suatu sistem didalam suatu organisasi yang mempertemukan kebutuhan pengelolaan transaksi harian, mendukung operasi, bersifat manajerial, dan kegiatan strategi dari suatu organisasi dan menyediakan pihak luar tertentu dengan laporan-laporan yang dibutuhkan.”

Sistem adalah kumpulan dari elemen-elemen yang berinteraksi untuk mencapai satu tujuan tertentu. Suatu sistem dapat didefinisikan sebagai suatu kesatuan yang terdiri dari dua atau lebih komponen atau subsistem yang berinteraksi untuk mencapai suatu tujuan. Sistem informasi adalah suatu sistem di dalam suatu organisasi yang mempertemukan kebutuhan pengolahan transaksi harian yang mendukung fungsi operasi organisasi yang bersifat manajerial dengan kegiatan strategi dari suatu organisasi untuk dapat menyediakan kepada pihak luar tertentu dengan laporan-laporan yang diperlukan (Lubis, 2016).

Menurut (Purnama, 2016) Sistem informasi manajeman digambarkan sebagai sebuah bangunan piramida dimana lapisan dasarnya terdiri dari informasi, penjelasan transaksi, penjelasan status, dan sebagainya. Lapisan berikutnya terdiri dari sumber-

sumber informasi dalam mendukung operasi manajemen sehari-hari. Lapisan keriga terdiri dari sumber daya sistem informasi untuk membantu perencanaan taktis dan pengambilan keputusan untuk pengendalian manajemen. Lapisan puncak terdiri dari sumber daya informasi utnuk mendukung perencanaan dan perumusan kebijakan oleh tingkat manajemen.

1. Komponen (*components*). Bagian-bagian atau elemen-elemen, yang dapat berupa benda atau manusia, berbentuk nyata atau abstrak, dan disebut subsistem.
2. Penghubung antarbagian (*interface*). Sesuatu yang bertugas menjembatani satu bagian dengan bagian lain, dan memungkinkan terjadinya interaksi/komunikasi antarbagian.
3. Batas (*boundary*). Sesuatu yang membedakan antara sistem dengan sistem atau sistem-sistem lain.
4. Lingkungan (*environment*). Segala sesuatu yang berada diluar sistem dan dapat besifat menguntungkan atau merugikan sistem yang bersangkutan.
5. Masukan (*input*). Sesuatu yang merupakan bahan untuk diolah atau diproses oleh sistem.
6. Mekanisme pengolahan (*processing*). Perangkat dan prosedur untuk mengubah masukan menjadi keluaran dan menampilkannya.
7. Keluaran (*output*). Berbagai macam bentuk hasil atau produk yang dikeluarkan dari pengolahan.
8. Tujuan (*goal/objective*). Sesuatu keadaan yang ingin dicapai oleh sistem,baik dalam jangkan pendek maupun jangka panjang.
9. Sensor dan kendali (*sensor & control*). Sesuatu yang bertugas memantau dan menginformasikan perubahan-perubahan di dalam lingkungan dan dalam diri sistem kepada sistem.

10. Umpan-balik (*feedback*). Informasi tentang perubahan-perubahan lingkungan dan perubahan-perubahan (penyimpangan) dalam diri sistem.

B. Definisi Penggajian

Menurut Rivai (2009 : 360) mendefinisikan gaji adalah balas jasa yang berbentuk uang yang diterima oleh karyawan sebagai konsekuensi kedudukannya sebagai karyawan yang memberikan kontribusi dan pikiran untuk mencapai tujuan perusahaan.

Menurut undang-undang Tenaga Kerja No. 13 Tahun 2000, Ban I, Pasal 1, Ayat 30 Upah adalah hak pekerja/buruh yang diterima dan dinyatakan dalam bentuk uang sebagai imbalan dari pengusaha/pemberi kerja kepada pekerja/buruh yang ditetapkan dan dibayarkan menurut suatu perjanjian kerja, kesepakatan, atau peraturan perundang-undangan termasuk tunjangan bagi pekerja/buruh dan keluarganya atas suatu pekerjaan dan/atau jasa yang telah atau akan dilakukan.

Tujuan utama pemberian gaji adalah mewujudkan pembayaran yang sama untuk pembayaran yang adil. Menurut Dale Yoder dalam (2009) , tujuan pemberian gaji adalah :

1. Mendapatkan karyawan-karyawan yang cakap

   Gaji yang cukup tinggi dapat menarik pelamar karena perusahaan-perusahaan akan bersaing dalam pasar tenaga kerja untuk mendapatkan karyawan yang cakap. Oleh sebab itu perusahaan harus mengikuti permintaan dan penawaran gaji yang dapat bersaing.

2. Mempertahankan karyawan yang sudah ada di dalam perusahaan.

   Bila tingkat gaji didalam perusahaan tidak bersaing dengan perusahaan lain maka akan meningkatkan potensi karyawan meninggalkan perusahaan. Untuk

mencegah hal tersebut terjadi maka pemberian gaji harus dapat bersaing dengan tingkatan gaji perusahaan lainnya.

3. Menjamin keadilan

   Administrasi gaji akan selalu berusaha untuk mendapatkan keadilan intern dan ekstern. Keadilan ekstern mengharuskan pembayaran gaji sesuai nilai relatif sesuai dengan jabatan. Artinya tingkatan jabatan yang sama harus mendapatkan jumlah pembayaran gaji yang sama. Keadilan ekstern membutuhkan pembayaran karyawan pada satu tingkat yang sama dengan pembayaran yang diterima oleh karyawan yang sama di perusahaan lainnya.

4. Menghargai Perilaku yang di inginkan

   Dengan pembayaran gaji ini harus dapat memperkuat perilaku karyawan yang di inginkan seperti prestasi kerja yang baik, pengalaman, kesetiaan, tanggung jawab baru dan perilaku lainnya yang dapat di hargai dengan rencana administrasi gaji yang efektif.

5. Mengawasi biaya.

   Dengan adanya administrasi gaji yang rasional maka dapat membantu organisasi untuk mendapatkan atau mempertahankan tenaga kerja dengan biaya yang layak.

   Tanpa adanya struktur gaji yang sistematik maka perusahaan berkemungkinan membayar gaji kepada karyawannya secara berlebih atau kurang dari yang seharusnya.

6. Menaati Undang-undang

   Tujuan pemberian gaji yang layak kepada karyawan merupakan salah satu cara perusahaan untuk mematuhi perundang-undangan yang berlaku.

   Pembayaran gaji yang baik harus juga mempertimbangan pembatasan nilai gaji karyawan dan juga menjamin semuat peraturan pemerintah di taati.

Berikut beberapa fungsi dari penggajian antara lain :

1. Pengalokasian Sumber Daya Manusia secara efisien

   Fungsi gaji ini dapat menggambarkan bahwa pemberian gaji yang cukup dapat mendorong karywan untuk dapat berprestasi lebih baik dan akan lebih mendorong karyawan untuk berkerja lebih produktif.

2. Penggunaan Sumber daya Manusia secara efisien dan efektif

   Dengan memberikan gaji yang tinggi kepada karyawan bisa menggambarkan bahwa perusahaan atau organisasi akan memanfaatkan karyawan dengan cara yang efisien dan efektif. Hal ini dapat memberikan dampak bagi perusahaan karena dapat mendapatkan manfaat dan keuntungan yang maksimal.

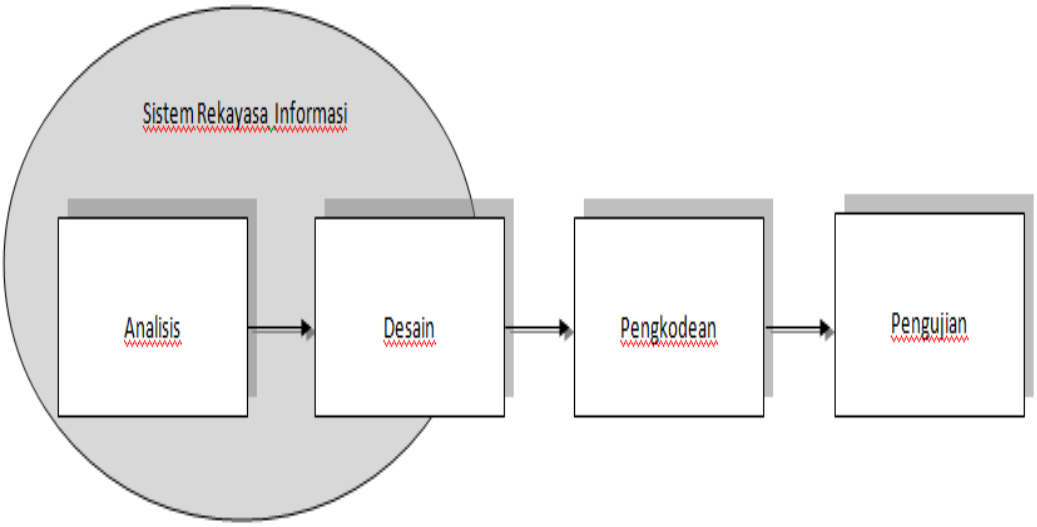
3. Mendorong stabilitas dan pertumbuhan ekonomi

   Dampak dari penggunaan sumber daya manusia di dalam organisasi secara efektif dan efisien maka diharapkan pemberian gaji dapat secara langsung membantu stabilitas perusahaan dan pertumbuhan ekonomi.

C. Model Pengembangan Perangkat Lunak *(Waterfall)*

Menurut Rosa dan Shalahuddin dalam (Bakhri, 2015) Model *waterfall* adalah model SDLC *(Software Development Life Cycle)* yang paling sederhana. Model ini hanya cocok untuk pengembangan perangkat lunak dengan spesifikasi yang tidak berubah-ubah.

Model SDLC air terjun *(waterfall)* sering juga disebut model sekuensial linier *(sequential linear)* atau alur hidup klasik *(classic life cycle).* Model air terjun menyediakan pendekatan alur hidup perangkat lunak secara sekuensial atau terurut dimulai dari analisis, desain, pengkodean, pengujian dan tahap pendukung *(support).*



Sumber : (Rosa, A.S., 2018)

Gambar II.1.
Ilustrasi model *waterfall*

1. Analisa kebutuhan perangkat lunak

   Analisis sistem dilakukan untuk mengidentifikasikan dan mengevaluasi permasalahan-permasalahan, hambatan-hambatan yang terjadi dan kebutuhan-kebutuhan yang diharapkan sehingga dapat diusulkan perbaikan-perbaikan.

2. Desain

   Desain perangkat lunak adalah proses multi langkah yang focus pada desain pembuatan program perangkat lunak termasuk struktur data, arsitektur perangkat lunak, representasi antarmuka, dan prosedur pengodean. Tahap ini mentranslasi kebutuhan perangkat lunak dari tahap analisis kebutuhan ke representasi desain agar dapat diimplementasikan menjadi program pada tahap selanjutnya

3. Pengkodean

   Desain harus ditranslasikan ke dalam program perangkat lunak. Hasil dari tahap ini adalah program komputer sesuai dengan desain yang telah dibuat pada tahap desain.

4. Pengujian

   Pengujian fokus pada perangkat lunak secara dari segi lojik dan fungsional dan memastikan bahwa semua bagian sudah diuji. Hal ini dilakukan untuk meminimalisir kesalahan (error) dan memastikan keluaran yang dihasilkan sesuai yang diinginkan

Menurut (Rosa, A.S., 2018), Dari kenyataan yang terjadi sangat jarang model air terjun dapat dilakukan sesuai alurnya karena sebab berikut :

1. Perubahan spesifikasi perangkat lunak terjadi ditengah alur pengembangan.
2. Sangat sulit bagi pelanggan untuk mendefinisikan semua spesifikasi di awal jalur pengembangan. Pelanggan sering kali butuh contoh *(prototype)* untuk menjabarkan spesifikasi kebutuhan lebih lanjut.
3. Pelanggan tidak mungkin bersabar mengakomodasikan perubahan yang diperlukan di akhir alur pengembangan.

Dengan berbagai kelemahan yang dimiliki model air terjun model air terjun ini juga memiliki kelebihan, diantaranya :

1. Model ini telah menjadi dasar dari model-model lain dalam melakukaperbaikan model pengembangan perangkat lunak.
2. Model air terjun ini sangat cocok digunakan kebutuhan pelanggan sudah sangat dipahami dan kemungkinan terjadinya perubahan kebutuhan selama pengembangan perangkat lunak kecil.
3. Hal positif dari model air terjun adalah struktur tahap pengembangan sistem jelas, dokumentasi dihasilkan di setiap tahap pengembangan, dan sebuah tahap dijalankan setelah tahapsebelumnya selesai dijalankan (tidak ada tumpang tindih pelaksanaan tahap).

2.2. Teori Pendukung

A. ERD *(Entity Relationship Diagram)*

Menurut Munawar dalam (Bakhri, 2015:71) "*Entity Relationship Diagram* merupakan suatu model untuk menjelaskan hubungan antar data dalam basis data berdasarkan suatu persepsi bahwa *real world* terdiri dari *object-object* dasar yang mempunyai hubungan atau relasi antar *object-object* tersebut. Relasi antar *object* dilukiskan dengan menggunakan symbol-symbol grafis tertentu".

Komponen yang terdapat dalam *Entity Relationship Diagram* adalah sebagai berikut:

1. *Entity* (entitas)

   Entitas adalah suatu data yang dapat disimpan dan berguna bagi badan atau perusahaan, dengan kata lain suatu obyek yang dapat dibedakan dengan objek lainnya. Entitas digambarkan dengan kotak persegi panjang.

   Terdapat juga Entitas Lemah *(Weak Entity),* yaitu suatu entitas sangat bergantung dengan entitas biasa, dengan kata lain, entitas lemah tidak akan ada apabila tidak ada entitas biasa. Entitas lemah digambarkan dengan kotak persegi panjang dengan garis ganda.

2. Atribut

   Atribut menunjukan karakteristik dari tiap-tiap entitas. Atribut digambarkan dengan bentuk oval.

3. Relasi

   Relasi menunjukkan hubungan yang terjadi diantara entitas. Relasi digambarkan dengan bentuk belah ketupat atau diamond.

4. *Line Connector*

   Line Connector digambarkan dengan bentuk garis tunggal.

5. Atribut Utama

   Atribut utama digambarkan dengan bentuk oval, dengan keterangan diberi garis bawah *absolute.*

6. Atribut Pilihan

   Atribut Pilihan digambarkan dengan bentuk oval dengan keterangan diberi garis bawah putus-putus.

7. Kardinalitas

   Kardinalitas merupakan tingkat hubungan yang terjadi diantara entitas di dalam sebuah sistem. Terdapat tiga tingkat hubungan yang terjadi, yaitu:

   a. Hubungan Satu pada Satu (*One to One* atau 1:1)

      Tingkat hubungan dinyatakan satu pada satu jika satu kejadian pada entitas pertama hanya mempunyai satu hubungan dengan suatu kejadian pada entitas kedua. Demikian juga sebaliknya satu kejadian pada entitas kedua hanya bisa mempunyai satu hubungan dengan satu kejadian pada entitas yang pertama.

   b. Hubungan Satu pada Banyak (*One to Many* atau 1:M)

      Tingkat hubungan satu pada banyak (1:M) adalah sama dengan banyak pada satu (M:1), tergantung dari arah mana hubungan tersebut dilihat. Untuk satu kejadian pada entitas yang pertama dapat mempunyai banyak hubungan dengan kejadian pada entitas yang kedua. Sebaliknya satu kejadian pada entitas yang kedua, hanya bisa mempuyai satu hubungan dengan satu kejadian pada entitas yang pertama.

   c. Hubungan Banyak pada Banyak (*Many to Many* atau M:N)

      Tingkat hubungan banyak pada banyak (M:N) terjadi jika tiap kejadian pada sebuah entitas akan mempunyai banyak hubungan dengan kejadian pada

entitas lainnya, baik dilihat dari sisi entitas yang pertama maupun dilihat dari sisi entitas yang kedua.

B. LRS *(Logical Relationship Structure)*

Menurut Pratama dalam (Rosidin & Lubis, 2017) "LRS merupakan transformasi dari penggambaran ERD dalam bentuk yang lebih jelas dan mudah untuk dipahami". Penggambaran LRS hampir mirip dengan penggambaran normalisasi file, hanya saja tidak digambarkan simbol *asterix* (*) sebagai simbol *primary key* (kunci utama) dan *foreign key* (kunci tamu).

LRS merupakan hasil pemodelan *Entity Relationship* (ER) beserta atributnya sehingga bisa terlihat hubugan-hubungan antar entitas.

Dalam pembuatan LRS terdapat tiga hal yang dapat mempengaruhi, yaitu:

1. Jika tingkat hubungan (*cardinality*) satu pada satu *(one to one)*, maka digabungkan dengan entitas yang lebih kuat *(strong entity)*, atau digabungkan dengan entitas yang memiliki atribut yang lebih sedikit.
2. Jika tingkat hubungan (*cardinality*) satu pada banyak (*one to many*), maka hubungan relasi atau digabungkan denga entitas yang tingkat hubungannya banyak.
3. Jika tingkat hubungan *(cardinality*) banyak pada banyak *(many to many*), maka hubungan relasi tidak akan digabungkan dengan entitas manapun, melainkan menjadi sebuah LRS.

C. UML (*Unified Modeling Language*)

Pada perkembangan teknik pemrograman berorientasi objek, munculaah sebuah standarisasi bahasa pemodelan untuk pembangunan perangkat lunak yang

dibangaun dengan menggunakan teknik pemrograman berorientasi objek, yaitu *Unified Modelling Language(UML). UML* muncul karena adanya kebutuhan pemodelan visual untuk menspesifikasikan, menggambarkan, membangun, dan dokumentasi dari sistem perangkat lunak. UML Merupakan bahasa visual untuk pemodelan dan komunikasi mengenai sebuah sistem dengan menggunakan diagram dan teks-teks pendukung. (Sukamto & Shalahuddin, 2013)

Seperti bahasa-bahasa lainnya, *UML* mendefinisikan notasi dan *syntax/semantik.*Notasi *UML* merupakan sekumpulan bentuk khusus untuk menggambarkan berbagai diagram piranti lunak.Setiap bentuk memiliki makna tertentu, dan *UMLsyntax* mendefinisikan bagaimana bentuk-bentuk tersebut dapat dikombinasikan. Notasi *UML* terutama diturunkan dari 3 notasi yang telah ada sebelumnya : Grady Booch *OOD* (*Object Oriented Design*), Jim Rumbaugh *OMT* (*Object Modeling Technique*), dan Ivar Jacobson *OOSE* (*Object Oriented Software Engineering*).

Abstraksi konsep dasar *UML* terdiri dari *structural classification, dynamic behavior* dan *model management.*Abstraksi konsep dasar UML terdiri dari *structural, class, ification, dynamic, behavior* dan *model manajemen.* UML mendefinisikan diagram-diagram sebagai berikut:

1. *Use Case* Diagram

Use case adalah pemodelan untuk kelakuan sistem informasi yang akan dibuat. Kesimpulannya *use case* digunakan untuk mengetahui fungsi apa saja yang ada di dalam sebuah sistem informasi dan siapa saja yang berhak menggunakan fungsi itu, (Sukamto & Shalahuddin, 2013).

Stereotype adalah sebuah model khusus yang terbatas untuk kondisi tertentu. Untuk menunjukkan *stereotype* digunakan symbol “<<” diawalnya dan ditutup “>>”

diakhirnya. <<*extend*>> digunakan untuk menunjukkan bahwa satu use case merupakan tambahan fungsional dari use case yang lain jika kondisi atau syarat tertentu yang dipenuhi. Sedangkan <<*include*>> digunakan untuk menggambarkan bahwa suatu *use case* seluruhnya merupakan fungsionalitas dari *use case* lainnya.Biasanya <<*include*>> digunakan menghindari pengcopian suatu use case karena sering dipakai. Sebuah skema *level use case*, *use case* utama ada pada pada *level* „ *sea level'*. *Sea level use case* mewakili interaksi *diskret* diantara *actor* utama dan *system*. *Use case* ini akan mendeliver beberapa nilai ke *actor* utama dan biasanya membutuhkan beberapa menit sampai setengah jam bagi *actor* untuk melakukan. *Use case* yang ada karena di *include*oleh *use case sea level* disebut *fish level*.

2. Activity Diagram

"*Activity Diagram* menggambarkan *workflow* (aliran kerja) atau aktivitas dari sebuah sistem atau proses bisnis atau menu yang ada pada perangkat lunak. (Sukamto & Shalahuddin, 2013).

Aktivitas menggambarkan proses yang berjalan, sementara *Use Case* menggambarkan bagaimana actor menggunakan sistem untuk melakukan aktivitas. Sama seperti state, standart UML menggunakan segi empat dengan sudut membulat untuk menggambarkan aktivitas.

Untuk mengilustrasikan proses paralel (*fork and join*) digunakan titik sinkronisasi yang dapat berupa titik, garis horizontal atau vertikal. *Activity Diagram* dapat dibagi menjadi beberapa *object swimlane* untuk menggambarkan objek mana yang bertanggung jawab untuk aktivitas tertentu.

3. Class Model / Class Diagram

Class diagram menggambarkan struktur sistem dari segi pendefinisian kelas-kelas yang akan dibuat untuk membangun sistem. Kelas memiliki apa yang disebut atribute dan metode atau operasi.

a. Atribut merupakan variabel-variabel yang dimiliki oleh suatu kelas.
b. Operasi atau metode adalah funsi-fungsiyang dimiliki oleh suatu kelas.

Diagram kelas dibuat agar pembuat program atau programmer membuat kelas-kelas sesuai rancangan didalam diagram kelas agar antara dokumentasi perancangan dan perangkat lunak sinkron. (Rosa, A.S., 2018)

4. Sequence Diagram

Diagram sekuen menggambarkan kelakuan objek pada use case dengan mendeskripsikan waktu hidup objek dan message yang dikirimkan atau yang diterima antar objek. Oleh karena itu untuk menggambar diagram sekuen maka harus diketahui objek-objek yang terlibat dalam sebuah usecase beserta metode-metode yang dimiliki kelas yang diintansiasi menjadi objek itu. Membuat diagran sequen juga dibutuhkan untuk melihat skenario yang ada pada usecase.

Banyaknya diagram sekuen harus digambar adalah minimal sebanyak pendefinisian use case yang memiliki proses sendiri atau yang penting semua use case yang telah didefinisikan interaksi jalannya pesan sudah dicakup pada diagram sekuen sehingga semakin banyak use case yang didefinisikan maka diagram sekuen yang harus dibuat juga semakin banyak. (Rosa, A.S., 2018).