BAB II

LANDASAN TEORI

2.1. Konsep Dasar

Konsep dasar adalah susunan dalam pembentukan pengetahuan ilmiah konsep itu bersifat abstrak dan berasal dari pemikiran.

2.1.1. Sistem

Menurut (Tohari, 2017) mengemukakan bahwa "Sistem adalah sekumpulan variabel yang saling berinteraksi, dan satu sama lain saling bergantung".

Berikut ini merupakan karakterisitik dari suatu sistem menurut (Tohari, 2017) adalah:

1. Komponen atau elemen (C*omponents*)
2. Batas Sistem (*Boundary*)
3. Lingkungan luar subsistem (*Environtment*)
4. Penghubung sistem (*Interface*)
5. Masukan (*Input*)
6. Luaran (*Output*)
7. Pengolah (*Process*)
8. Sasaran (*Objektive*)

2.1.2. Informasi

Menurut (Tohari, 2017) mengemukakan bahwa "Informasi merupakan data yang telah diklasifikasikan atau diolah atau diinterpretasikan untuk digunakan dalam pross pengambilan keputusan".

2.1.3. Sistem Informasi Akuntansi

Menurut (Tresnawati et al., 2017) "Sistem Informasi Akuntansi adalah olahan data keuangan yang di rancang menjadi laporan keuangan (informasi), yang di tujukan kepada pihak *internal* maupun *eksternal* perusahaan".

Tujuan dari sistem informasi menurut (Setiorini et al., 2018) yaitu menghasilkan *output* yang diperlukan bagi pihak pemakai informasi akuntansi yang dibedakan dalam dua kelompok yaitu pihak *ektern* dan pihak *intern*.

2.1.4. Pembelian

Menurut (Sholikhah et al., 2017) Mengemukakan bahwa "Aktivitas pembelian dapat disebut dengan prokuremen. Prokuremen merupakan suatu proses bisnis yang diawali dengan pemilihan sumber daya, aktivitas pembuatan order, dan perolehan barang dan jasa dari pemasok yang dilakukan oleh perusahaan".

(Sholikhah et al., 2017) juga menambahkan "Pembelian (*purchasing*) adalah akun yang digunakan untuk mencatat semua pembelian barang dagangan dalam satu periode".

1. Jurnal Pembelian Secara Tunai

Persediaan Barang Dagangan	XXX	
Kas		XXX

2. Jurnal Pembelian Secara Kredit

Persediaan Barang Dagangan	XXX	
Utang Usaha		XXX

3. Retur Pembelian Secara Tunai

Kas	XXX	
Persediaan Barang Dagangan		XXX

4. Retur Pembelian Secara Kredit

Utang Usaha XXX

Persediaan Barang Dagangan XXX

2.1.5. Program dan Bahasa Pemrograman *Java*

Menurut (Sukamto dan Shalahuddin, 2016) menjelaskan bahwa, "Pemrograman terstruktur adalah konsep-konsep atau pradigma atau sudut pandang pemrograman yang membagi-bagi program berdasarkan fungsi-fungsi atau prosedur-prosedur yang dibutuhkan program komputer".

Menurut (Jubilee Enterprise, 2017) mengemukakan bahwa "*Java* adalah Bahasa pemrograman yang powerful dan serbaguna untuk pengembangan perangkat lunak yang berjalan di perangkat seluler, komputer desktop, dan server".

2.1.6. *Netbeans* IDE 8.2

Menurut (Rusmayanti, 2015) mengemukakan bahwa "*Netbeans* merupakan salah satu IDE yang dikembangkan dengan bahasa pemrograman *java*. *Netbeans* mempunyai lingkup pemrograman java terintegrasi dalam suatu perangkat lunak yang didalamnya menyediakan pembanguanan GUI, *text editor*, *complier*, dan *interpreter*".

Sedangkan menurut (Nofriadi, 2018:4) menjelaskan bahwa, "Netbeans merupakan sebuah aplikasi *Integreted Decelopment Environmen* (IDE) yang berbasiskan *Java* dari Sun Microsystems yang berjalan diatas swing dan banyak digunakan sekarang sebagai editor untuk berbagai Bahasa pemrograman".

2.1.7. Basis Data (*Database*)

Menurut (Khambali & Prabowo, 2019) Mengemukakan bahwa "Basis data adalah kumpulan data berelasi yang disusun, diorganisasikan, dan disimpan secara sistematik dalam media simpan komputer mengacu kepada metode-metode tertentu

sedemikian rupa sehingga dapat diakses secara cepat dan mudah menggunakan program atau aplikasi komputer untuk memperoleh data dari basis data tersebut"

2.2. Peralatan Pendukung *(Tools System)*

Peralatan pendukung merupakan alat yang digunakan untuk menggambarkan bentuk logika model dari sistem yang akan dibuat. Adapun peralatan yang dipakai penulis sebagai berikut:

2.2.1. Konsep Dasar *Unitified Modelling Language(UML)*

Menurut (Lubis, 2016) Mengemukakan bahwa "*Unitified Modelling Language(UML)* Keluarga notasi grafis yang didukung oleh meta-model tunggal, yang membantu pendeskripsian dan desain sistem perangkat lunak, khususnya sistem yang dibangun menggunakan pemrograman berorientansi objek".

Menurut (Sukamto dan Shalahuddin, 2016) mengatakan bahwa "*Use Case* atau diagram *use case* merupakan pemodelan untuk kelakukan (*behavior*) sistem informasi yang akan dibuat. *Use Case* mendeskripsikan sebuah interaksi antara satu atau lebih aktor dengan sistem informasi yang akan dibuat. Secara kasar, *use case* digunakan untuk mengetahui fungsi apa saja yang ada didalam sebuah sistem informasi dan siapa saja yang berhak menggunakan fungsi-fungsi itu".

Syarat penamaan *use case* adalah nama didefinisikan sesimpel mungkin dan dapat dipahami menurut (Sukamto dan Shalahuddin, 2016), adalah:

1. Aktor merupakan orang, proses, atau sistem lain yang berinteraksi dengan sistem informasi yang akan dibuat diluar sistem informasi yang akan dibuat itu sendiri, jadi walaupun simbol dari aktor adalah gambar orang, tapi aktor belum tentu merupakan orang.

2. *Use case* merupakan fungsonalitas yang disediakan sistem sebagai unit-unit yang saling bertukar pesan antar unit atau aktor.

Menurut (Sukamto dan Shalahuddin, 2016) mengatakan bahwa "Diagram aktivitas atau *activity diagram* (aliran kerja) atau aktivitas dari sebuah sistem atau proses bisnis atau menu yang ada pada perangkat lunak".

Berikut menurut (Sukamto dan Shalahuddin, 2016) adalah:

1. Rancangan proses bisnis dimana setiap urutan aktivitas yang digambarkan merupakan proses bisnis sistem yang didefinisikan.
2. Urutan atau pengelompokan tampilan dari sistem/*user interface* dimana setiap aktivitas dianggap memiliki sebuah rancangan antarmuka tampilan.
3. Rancangan pengujian dimana setiap aktivitas dianggap memerlukan sebuah pengujian yang perlu didefinisikan kasus ujinya.
4. Rancangan menu yang ditampilkan pada perangkat lunak.

Menurut (Sukamto dan Shalahuddin, 2016) mengemukakan bahwa "Diagram *sequence* menggambarkan kelakukan objek pada *use case* dengan mendeskripsikan waktu hidup objek dan *message* yang dikirimkan dan diterima antar objek".

Banyaknya diagram sekuen yang 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. Oleh karena itu untuk menggambar diagram sekuen maka harus diketahui objek-objek yang terlibat dlam sebuah *use case* beserta metode-metode yang dimiliki kelas yang diinstansiasi menjadi objek itu.

Menurut (Tohari, 2017) menjelaskan bahwa, "Diagram *deployment* menunjukkan tata letak sebuah sistem secara fisik. Diagram ini akan menampakkan bagian-bagian *software* yang berjalan pada bagian-bagian *hardware* yang digunakan untuk mengimplementasikan sebuah sistem dan ketehubungan antara komponen-komponen *hardware* tersebut".

Untuk menggambar diagram *deployment*, ada beberapa hal yang harus diidentifikasikan terlebih dahulu menurut (Tohari, 2017) yaitu:

1. Menentukan *node*.
2. Hubungan antar *node.*

Penentuan *node*, diantaranya dapat dilakukan dengan beberapa pedoman menurut (Tohari, 2017) sebagai berikut:

1. *Node* merupakan elemen fisik yang sudah tersedia dalam sistem atau aplikasi. Amatilah dan jadikan elemen fisik sebagai *node* di diagram *deployment.*
2. *Node* mewakili sumber-sumber komputasi.
3. *Node* dilambangkan dengan kubus.

Sedangkan menurut (Tohari, 2017) menjelaskan bahwa, "*Interface* adalah satu *set operation* yang memberikan spesifikasi beberapa aspek dari perilaku dan operasi disuatu *class* ke *class* yang lain".

Misalnya, *keyboard* pada komputer sebenernya merupakan *interface* yang bisa dipakai ulang karena tombol-tombol *keyboard* sebenarnya berasal dari mesin ketik, hanya saja mungkin ada beberapa operasionalisasi tombol-tombol yang berbeda yang sudah ditransfer ke sistem yang lain. Misalnya tombol *control*, *page up*, *page down* dan lain-lain.

2.2.2. *Entity Relationship Diagram* (ERD)

Menurut (Junaidi, 2016) mengemukakan bahwa "*Entity Relationship Diagram* adalah suatu model jaringan yang menggunakan susunan data yang disimpan dalam secara abstrak".

Berdasarkan penjelasan diatas penulis menyimpulkan bahwa *Entity Relationship Diagram* adalah suatu model jaringan yang tersimpan secara sistem atau teknik menggambarkan suatu skema *database* dimana setiap komponen yang terlibat dari *Entity Relationship Diagram* memiliki atribut masing-masing yang mempresentasikan fakta dari dunia nyata yang sedang ditinjau.

2.2.3. *Logical Record Structure* (LRS)

Menurut (Puspitasari, 2016) mengemukakan bahwa :
"Sebuah model sistem yang digambarkan dengan sebuah diagram-ER akan mengikuti pola/ aturan pemodelan tertentu dalam kaitannya dengan konversi ke LRS, maka perubahan yang terjadi adalah mengikuti aturan - aturan berikut ini : Setiap entitas akan diubah kebentuk kotak, Sebuah atribut relasi disatukan dalam sebuah kotak bersama entitas jika hubungan yang terjadi pada diagram-ER 1:M (relasi bersatu dengan cardinality M) atau tingkat hubungan 1:1 (relasi bersatu dengan cardinality yang paling membutuhkan referensi), sebuah relasi dipisah dalam sebuah kotak tersendiri (menjadi entitas baru) jika tingkat hubungannya M:M (many to many) dan memiliki *foreign key* sebagai *primary key* yang diambil dari kedua entitas yang sebelumnya saling berhubungan".

2.2.4. Konsep Dasar *Black Box Testing*

Menurut (Sukamto dan Shalahuddin, 2016) "*Black-box testing* (pengujian kotak hitam), yaitu menguji perangkat lunak dari segi spesifikasi fungsional tanpa menguji desain dan kode program.