BAB II

TINJAUAN PUSTAKA

2.1 Pengertian Sistem Informasi

Menurut A.A. Ngurah Adhi Jaya (2016) Sistem : kumpulan dari unsur/elemen-elemen yang saling berkaitan/berinteraksi dan saling mempengaruhi dalam melakukan kegiatan bersama untuk mencapai suatu tujuan tertentu. Syarat-Syarat Sistem :

A. Sistem harus dibentuk untuk menyelesaikan tujuan.

B. Elemen sistem harus mempunyai rencana yang ditetapkan.

C. Adanya hubungan diantara elemen sistem.

D. Unsur dasar dari proses (arus informasi, energi dan material) lebih penting daripada elemen sistem.

E. Tujuan organisasi lebih penting dari pada tujuan elemen.

Sistem Informasi adalah sekelompok orang, prosedur, input, output dan pengolahannya secara bersama-sama menghasilkan informasi yang akurat, tepat waktu dan relevan bagi penggunanya. Kemudian menambahkan bahwa, Sistem informasi berbasis komputer mempunyai 6 bagian: hardware, software, data/informasi, proseder, komunikasi dan orang. SI ditentukan dalam perusahaan bergantung pada sifat dan struktur bisnisnya. Ini berarti SI bersifat modifikatif terhadap kebutuhan organisasi. Komponen prosedur dalam SI berkaitan dengan prosedur manual dan prosedur berbasis komputer serta standar untuk mengolah data menjadi informasi yang berguna. Suatu prosedur adalah urutan langkah yang

dilakukan untuk menyelesaikan satu atau lebih aktifitas pengolahan informasi. Pengolahan informasi ini dapat dikerjakan dengan pengguna, atau kombinasi pengguna dan staff TI.

2.2 Pengertian Aplikasi Web

Aplikasi Web adalah sebuah program yang bila dieksekusi akan menghasilkan sebuah aplikasi yang dapat bekerja sesuai dengan yang diinginkan. Aplikasi web dibangun dengan menggunakan bahasa HTML(Hypertext Markup Language). Pada masa kini aplikasi web dikembangkan untuk memperluas kemampuan HTML dengan PHP dan ASP pada skrip objek. Aplikasi web dapat dibagi menjadi dua bagian yaitu aplikasi web dinamis dan aplikasi web statis.

Aplikasi web merupakan sebuah aplikasi yang mengunakan teknologi browser untuk menjalankan aplikasi dan diakses melalui jaringan komputer (Remick, 2011). Arsitektur aplikasi web meliputi klien, web server, middleware dan basis data. Klien berinteraksi dengan web server. Secara internal, web server berkomunikasi dengan middleware dan middleware yang berkomunikasi dengan basis data. Contoh middleware adalah PHP dan ASP. Pada mekanisme aplikasi web dinamis, terjadi tambahan proses yaitu server menerjemahkan kode PHP menjadi kode HTML. Kode PHP yang diterjemahkan oleh mesin PHP yang akan diterima oleh klien.(Abdul Kadir, 2009). Aplikasi adalah sebuah „mahakarya Symphony Orchestra‟ dari pelaku Teknologi Informasi, yang merupakan hasil kerjasama antara sumber daya manusia, tools dan pengguna dalam sebuah manajemen yang terintegrasi dimana didalamnya ada perencanaan, ujicoba, pelaksanaan dan pemeliharaan, dengan tujuan akhir untuk mendukung aktifitas manusia agar lebih efisien dan efektif (Muhammad Safri Lubis, 2011).

Jadi aplikasi adalah sebuah program hasil karya yang siap pakai. Program yang terbuat dengan beberapa tahapan yang melaksanakan suatu fungsi yang telah diperintahkan.

2.3 PHP

PHP (*Hypertext Preprocessor*) dikembangkan pertama kali tahun 1995 oleh Ramsus Leodorf yang merupakan salah satu anggota group Appache. PHP pertama kali didesain dengan alat tracking pengunjung website ledorf. Kemudian, fungsinya diperlebar dan dihubungkan dengan *Appache*. PHP dikembangkan sepenuhnya untuk bahasa script *server-side* programming. PHP bersifat open source dan dapat digabungkan dengan berbagai server yang berbeda-beda.

Abdul Kadir mengatakan bahwa menurut dokumen resmi PHP, PHP singkatan dari *Hypertext Preprocessor* yang merupakan bahasa berbentuk script yang ditempatkan di server dan di proses di server. Hasilnya akan dikirim ke *client* tempat pemakai menggunakan browser. PHP adalah bahasa *server-side* programming yang powerfull untuk membuat halaman web yang dinamis dan interaktif. Sintak PHP mirip dengan bahasa Perl dan C. PHP biasanya sering digunakan bersama *web server Appache* diberagam sistem operasi (Sunyoto, 2007).

2.4 *Hypertext Markup Language* (HTML)

HTML akronim dari *Hypertext Markup Language*. HTML adalah file text murni yang dapat dibuat dengan editor text sembarang. Dokumen ini dikenal sebagai *web* page. File-file HTML ini berisi intruksi-intruksi yang kemudian diterjemahkan oleh browser yang ada di komputer client (*user*) sehingga isi formatnya dapat ditampilkan secara visual dikomputer pengguna. (kadir, Abdul 2009).

2.5 Java Script

Java Script adalah bahasa yang berbentuk kumpulan skrip berjalan pada suatu dokumen HTML. Bahasa ini adalah pemrograman untuk memberikan kemampuan tambahan terhadap bahasa HTML dengan mengizinkan mengeksekusi perintah-perintah di sisi *user* artinya browser bukan di sisi server *web.* Java Script adalah bahasa yang "*case sensitive*" artinya membedakan penamaan variabel dan fungsi yang menggunakan huruf besar dan huruf kecil.

2.6 *MYSQL*

Menurut Betha Sidik, dalam buku yang "Pemrograman *web* Dengan PHP (2012 : 333)" menyebutkan bahwa : "MySQL merupakan *software database* yang termasuk paling popular di lingkungan Linux, kepopuleran ini karena ditunjang karena performansi *query* dari *database* nya yang saat itu bisa dikatakan paling cepat dan jarang bermasalah". MySQL merupakan aplikasi *database* server. SQL kepanjangan dari *Structured Query Language*. SQL merupakan bahasa terstruktur yang digunakan untuk mengolah *database.* MySQL dapat digunakan untuk membuat dan mengelola database. Fungsi dari MySQL adalah untuk menambahkan, mengubah, dan menghapus data di dalam database. MySQL berawal dari proyek yang dimulai oleh kedua orang developer, yakni Michael Widenius dan David Axmark di tahun 1994. Proyek ini didasari karena ingin membuat suatu sistem *database* yang murah, meskipun ketika itu ada *database* yang power full yakni *oracle*, namun database ini bersifat komersil yang harganya mahal, dan begitu menguasai pasar.

2.7 *Unified Modeling language* (UML)

Unified Modeling language (UML) adalah sebuah standart bahasa pemodelan yang memungkinkan untuk menspesifikasi, memvisualisasi, membangun, dan mendokumentasikan sebuah sistem perangkat lunak. Tujuan dari pemodelan ini adalah memodelkan sistem perangkat lunak dari segi pembangunan, produktifitas, kualitas, pengurangan biaya, dan juga waktu.

Diagram UML diklarifikasikan menjadi dua kategori struktural dan *behavioral*. Diagram struktural memodelkan aspek statis dari sistem dan juga *fitur-fitur structural,* sedangkan diagram *behavioral* menggambarkan perilaku dinamis sebuah sistem. Diagram struktural yang akan dibahas dalam penulisan ini adalah *Class Diagram,* dan diagram *behavioral* yang akan dibahas adalah *Use Case Diagram* dan *Sequence* Diagram.

UML tepat digunakan untuk memodelkan sistem dari mulai memodelkan informasi sistem untuk perusahaan hingga aplikasi *web*, bahkan untuk sistem yang rumit sekalipun. UML menggunakan *class* dan *operation* dalam konsep dasarnya, maka lebih cocok untuk penulisan piranti lunak dalam bahasa-bahasa berorientasi objek.(Salahudin M., & Sukamto, R.A. 2011).

UML juga dapat diartikan sebuah bahasa grafik standart yang dugunakan untuk memodelkan perangkat lunak berbasis objek. UML pertama kali dikembangkan pada pertengahan tahun 1990an dengan kerjasama antara James Rumbaugh, Grady Booch, dan Ivar Jacobson, yang masing-masing telah mengembangkan notasi mereka sendiri di awal tahun 1990an. (Lethbride dan Leganiere, 2002, p151).

UML terdiri dari diagram-diagram, dimana setiap diagramnya didalam UML memperlihatkan sistem dari berbagai sudut pandang yang berbeda, berikut diagram-diagram tersebut :

2.7.1 *Class Diagram*

Class Diagram merupakan bangunan utama dalam pemodelan berorientasi objek. Diagram ini menggambarkan sebuah pandangan dari satu aspek tertentu dari model atau keseluruhan, menggambarkan struktur elemen beserta hubungan mereka. *Class Diagram* terutama digunakan untuk membangun sebuah arsitektur sistem dengan menangkap dan mendefinisikan *class-class* dan *interface* dan hubungan antara mereka. Sebuah *class Diagram* menggambarkan hubungan antara kelas dari pada hubungan objek *Class* itu sendiri bisa disebut sebagai bagian yang digunakan untuk membangun dan mendefinisikan objek-objek. Sehingga setiap hal yang dibangun dari sebuah *class* disebut sebagai objek atau *instance*. Deskripsi dari sebuah *class* mencangkup dua bagian, yaitu informasi yang akan dimiliki oleh objek dan *behaviour* yang akan didukung objek tersebut.

Class memiliki tiga area pokok, yaitu :

1. Nama
2. Atribut
3. Metoda

Atribut dan metoda dapat dimiliki salah satu sifat berikut :

1. *Public*, dapat dipanggil di luar *class* yang digunakan.
2. *Private*, tidak dapat dipanggil dari luar *class* yang bersangkutan.
3. *Protected*, hanya dapat dipanggil oleh *class* yang bersangkutan dan anak-anak yang mewarisinya.

Dibawah ini adalah hubungan-hubungan antar *class* yang bisa terjadi :

1. Asosiasi, yaitu hubungan statis antar *class*. Umumnya menggambarkan *class* yang memiliki atribut berupa *class* lain, atau *class* yang harus mengetahui esksistensi *class* lain. Panah *navigability* menunjukan arah *query* antar *class.*
2. Agregasi, yaitu hubungan yang menyatakan bagian ("terdiri atas..").
3. Pewarisan, yaitu hubungan hirarki antar *class*. *Class* dapat diturunkan dari *class* lain dan mewarisi semua atribut dan metoda *class* asalnya dan menambahkan fungsionalitas baru, sehingga disebut anak dari *class* yang di warisinya. Kebalikan dari pewarisan adalah generelasi. (Salahudin, M., & Sukamto, R.A. 2011).

Kelas (Schmuller, 1999, p8) adalah sebuah kategori atau pengelompokan dari halhal yang mempunyai atribut dan fungsi yang sama. *Class* diagram (Rumbaugh, 1999, p190) adalah sebuah gralik presentasi dari gambaran statis yang menunjukkan sekumpulan model elemen yang terdeklarasi (statis), seperti kelas, tipe dan isinya serta hubungannya. Sebuah *class* diagram terdiri dari sejumlah class yang dihubungkan dengan garis yang menunjukkan hubungan antar kelas yang disebut dengan Associations (Rambaugh, 1999, p47).

Jenis-jenis Associations (Rambaugh, 1999, pp49-53) yaitu :

1. *Aggregation*

   Associations yang menggambarkan hubungan antar kelas di mana *class* yang satu mempakan bagian dari *class* yang lainnya.

2. *Composition*

Associations yang menggambarkan hubungan erat antar kelas di mana kelas composite mempunyai segala tanggung jawab untuk mengatm‘ kelas lainnya dan kedua kelas mempunyai *lifetime* yang sama.

3. *Bidirectionality*

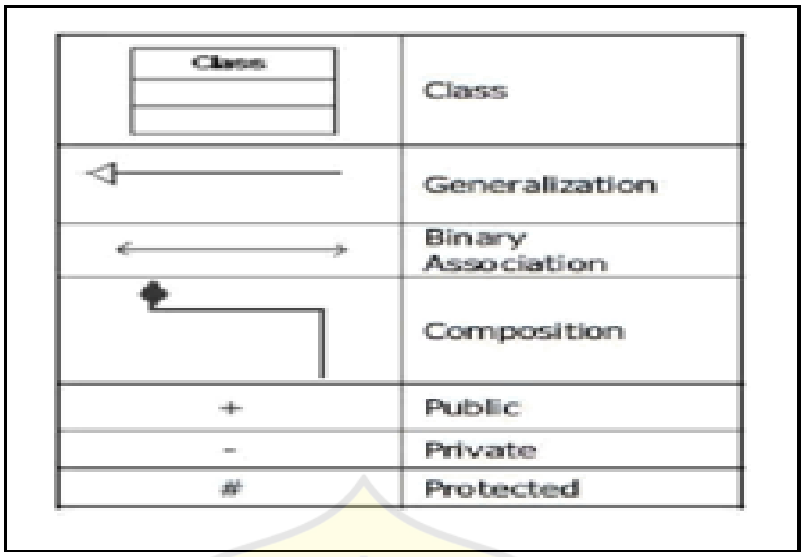
Associations yang menghubungkan antara dua kelas atau lebih yang berbeda *object* tapi tidak berganttmg satu sama lainnya, sehingga apabila salah satu kelas dihilangkan, kelas yang lain dapat tetap digunakan.

4. *Generalization*

Associations yang menghubungkan dua kelas atau lebih untuk membedakan antara kelas yang umum dengan kelas yang khusus.

5. *Inheritance*

Associations yang menghubungkan dua kelas atau lebih yang dapat menurunkan *properties* seperti *attributes, operations* antara kelas induk dengan kelas anak.



Gambar 2.1 Komponen-Komponen *Class Diagram*

2.7.2 *Use Case Diagram*

Use Case menggambarkan interaksi antara sistem dengan pelaku yang ada. Diagram ini mendeskripsikan siapa saja yang menggunakan sistem dan bagaimana cara mereka berinteraksi dengan sistem. *Usecase* digunakan untuk menggambarkan bagaimana system terlihat pada pengguna (Mathiassen et.al, 2000, p120).

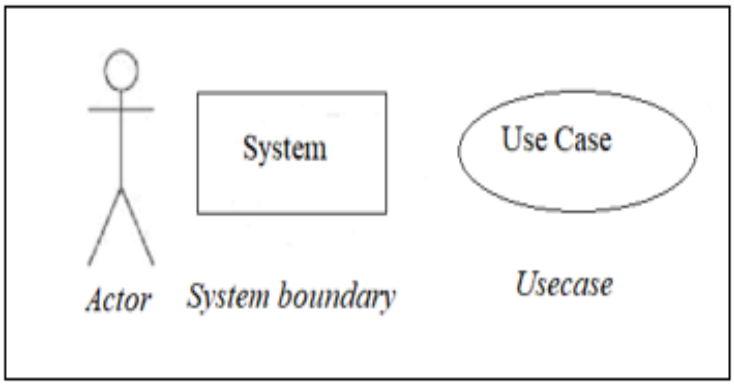
Use case diagram menggambarkan sebuah skenario yang menampilkan interaksi antara pengguna dan sistem. Seorang pengguna dapat berupa seseorang atau sistem lain. Sebuah *use case* mengacu pada tindakan bahwa sistem dapat dilakukan dengan berinteraksi dengan sebuah aktor. Oleh karena itu, *use case* diagram menunjukkan serangkaian tindakan serta aktor yang dapat melakukan tindakan tersebut.

Yang ditekankan dalam *use case* diagram adalah "apa" yang diperbuat sistem, dan bukan "bagaimana". Secara umum, *use case* diagram terdiri dari tiga bagian antara lain :

1. Aktor, adalah sebuah peran dari *user*, subsystem, atau piranti yang berinteraksi dengan sistem. Aktor digambarkan dengan figur' ‘stick’.
2. *Use case* menyediakan aktor rangkaian aksi yang bisa dilakukan. *Use case* direpresentasikan dengan bentuk oval yang digambarkan secara horizontal.
3. *Association* adalah hubungan dua atau lebih c*lassifier*, yaitu hubungan koneksi yang terdapat diantara instansi-instansi yang ada dan digambarkan dengan menggunakan garis.

Sebuah *use case* dapat meng-*include* fungsionalitas *use case* lain sebagai bagian dari proses dalam dirinya. Secara umum diasumsikan bahwa *use case* yang di *include* akan dipanggil setiap kali *use case* yang meng-*include* dieksekusi secara normal. Sebuah *use case* dapat di-i*nclude* oleh lebih dari satu *use case* lain, sehingga duplikasi fungsionalitas dapat dihindari dengan cara menarik keluar fungsionalitas yang *common.*

Sebuah *use case* juga dapat meng-*extend use case* lain dengan *behaviour*-nya sendiri. Sementara hubungan generalisasi antara *use case* menunjukan bahwa *use case* yang satu merupakan spesialisasi dari yang lain. (Salahudin, M., & Sukamto, R.A. 2011).



Gambar 2.2 Komponen *Use Case* Diagram

Jenis jenis *Use Case* Relationship (Rambaugh, 1999, p65) antara lain :

1. *Associaton*

   Garis yang menghubungkan antara actor dengan *use case*.

2. *Extend*

   Menghubungkan antara dua atau lebih *use case* yang merupakan tambahan dari *base use case* yang biasanya untuk mengatasi kasus pengecualian.

3. *Generalization*

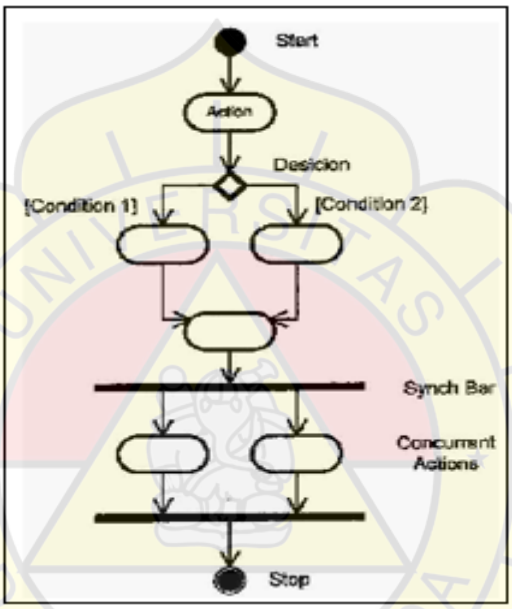
   Hubungan antara *use case* umum denga *use case* yang lebih khusus.

4. *Inlcude*

   Mengubungkan antara 2 atau lebih *use case* untuk menunjukan *use case* tersebut merupakan bagian dari *base use case.*

2.7.3 *Activity Diagram*

Activity Diagram Diagram *activity*, secara grafis digunakan untuk mengambarkan rangakaian aliran altivitasbaik proses bisnis atau *use-case*. Diagram ini juga dapat digunakan untuk memodelkan *action* yang akan dilakukan saat sebuah operasi dieksekusi, dan memodelkan hasil dari *action* tersebut.



Gambar 2.3 Komponen *Activity Diagram*

2.7.4 *Sequence Diagram*

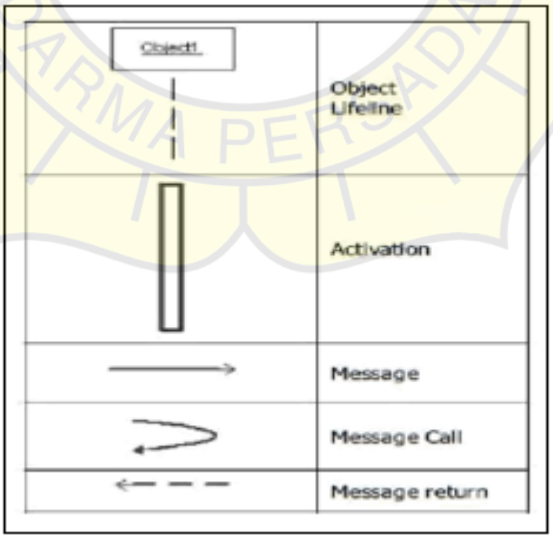
Sebuah *sequence* diagram (Lethbridge dan Laganiere, 2002, p270), menunjukkan urutan pertukaran pesan yang dilakukan oleh sekumpulan objek atau aktor yang mengerjakan pekerjaan.

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

digunakan untuk menggambarkan skenario atau rangkaian langkah-langkah yang dilakukan sebagai respons dari sebuah *event* untuk menghasilkan *output* tertentu. Diawali dari apa yang men-*trigger* aktivitas tersebut, proses dan perubahan apa saja yang teljadi secara internal dan *output* apa yang dihasilkan.

Masing-masing objek, termasuk aktor, memiliki *lifeline* vertikal. *Message* digambarkan sebagai garis berpanah dari satu objek ke objek lainnya. Pada fase desain berikutnya, *message* akan dipetakan menjadi operasi/metoda dari *class*. *Activation bar* menunjukkan lamanya eksekusi sebuah proses, biasanya diawali dengan diterimanya sebuah *message.*

Message Caller adalah sebutan untuk participant (suatu objek) yang mengirim *message* dan *Message Receiver* untuk participant (objek lainnya) yang menerima *message. Time* menunjukkan urutan dimana semua interaksi berlangsung sesuai dehgan waktu. *Time* pada *sequencé diagram* ditunjukkan dengan garis titik-titik vertikal. (Salahudin, M., & Sukamto, RA. 2011).



Gambar 2.4 Komponen *Sequence Diagram*

2.8 Pengertian Restoran

Restoran adalah istilah umum untuk menyebut usaha gastronomi yang menyajikan hidangan kepada masyarat dan menyediakan tempat untuk menikmati hidangan itu serta menetapkan tarif tertentu untuk makanan dan pelayanannya, umumnya restoran menyajikan makanan ditempat, tetapi ada juga restoran yang menyediakan layanan *take-away dinning* dan delivery service untuk melayani konsumennya. Restoran biasanya memiliki spesialisasi dalam jenis makanan yang dihidangkannya.

Restoran adalah tempat usaha yang komersial yang ruang lingkup kegiatannya menyediakan pelayanan makanan dan minuman untuk umum di tempat usahanya (Suarthana, 2006).

2.9 Pengertian Kepuasan Pelanggan

Perasaan senang atau kecewa yang muncul setelah membandingkan kinerja (hasil) produk yang dipikirkan terhadap kinerja (atau hasil) yang diharapkan. (Kotler, 2014)

Perusahaan perlu mengukur kepuasan pelanggan guna melihat umpan balik maupun masukan yang dapat diambil oleh perusahaan untuk keperluan pengembangan dan implementasi strategi peningkatan kepuasan pelanggan (Kotler,1999) mengemukakan bahwa terdapat empat metode untuk mengukur kepuasan pelanggan, yaitu:

1. Sistem keluhan dan saran. Setiap perusahaan yang berpusat pada pelanggan (customer centered) perlu memberikan kesempatan bagi pelanggannya untuk menyampaikan saran, pendapat, dan keluhan mereka. Banyak restoran dan hotel yang memberikan formulir bagi tamu untuk

mengetahui kesukaan dan keluhan mereka. Alur informasi ini memberikan banyak gagasan baik dan perusahaan dapat bergerak cepat untuk menyelesaikan masalah.

2. Survei kepuasan pelanggan. Perusahaan tidak dapat beranggapan bahwa sistem keluhan dan saran dapat menggambarkan secara lengkap kepuasan dan kekecewaan pelanggan. Perusahaan yang responsif mengukur kepuasan pelanggan dengan mengadakan survey berkala. Mereka mengirimkan daftar pertanyaan atau menelpon suatu kelompok acak dari pembeli mereka untuk mengetahui perasaan mereka terhadap berbagai aspek kinerja perusahaan. Perusahaan juga menanyakan pendapat pembeli mengenai kinerja perusahaan pesaing.

3. Ghost Shopping (pembeli bayangan). Metoda ini dilaksanakan dengan cara mempekerjakan beberapa orang (ghost shopper) untuk berperan sebagai pelanggan atau pembeli potensial produk perusahaan pesaing untuk melaporkan titik-titik kuat maupun titik-titik lemah yang mereka alami waktu membeli produk perusahaan maupun produk pesaing. Ghost shopper juga dapat mengamati cara penanganan setiap keluhan.

4. Lost customer analysis (analisis pelanggan yang beralih). Perusahaan sebaiknya menghubungi para pelanggan yang telah berhenti membeliatau yang telah berpindah pemasok agar dapat memahami mengapa hal ini terjadi dan supaya dapat mengambil kebijakasanaan perbaikan atau penyempurnaan selanjutnya

2.10 Logika *Fuzzy*

2.10.1 Pengertian Logika *Fuzzy*

Dalam logika konvensional, nilai kebenaran mempunyai kondisi yang pasti yaitu benar atau salah *(true or false*), dengan tidak ada kondisi antara. Prinsip ini telah mendominasi pemikiran logika di dunia sampai sekarang. Tentu saja, pemikiran mengenai logika konvensional dengan nilai kebenaran yang pasti yaitu benar atau salah dalam kehidupan yang nyata sangatlah tidak mungkin. logika *fuzzy* menawarkan suatu logika yang dapat merepresentasikan keadaan dunia nyata (Kusumadewi & Purnomo, 2010).

Teori himpunan logika *fuzzy* di kembangkan oleh Professor Lofti A. Zadeh pada tahun 1965. Ia ber-pendapat bahwa logika benar dan salah dari logika booleanlkonvensional tidak dapat mengatasi masalah gradasi yang berada pada dunia nyata.Untuk mengatasi masalah gradasi yang tidak terhingga tersebut, Zadeh mengembangkan sebuah himpunan *fuzzy*. Tidak seperti logika boolean, logika *fuzzy* mempunyai nilai yang kontinu. *Fuzzy* dinyatakan dalam derajat dari suatu keanggotaan dan derajat dari kebenaran. Oleh sebab itu sesuatu dapat dikatakan sebagian benar dan sebagian salah pada waktu yang sama.

Berdasarkan hal tersebut diatas Logika *fuzzy* dapat digunakan untuk memodelkan suatu permasalahan yang matematis, dimana konsep matematis yang mendasari penalaran *fuzzy* sangat sederhana dan mudah dimengerti.

Logika *fuzzy* merupakan generalisasi dari logika klasik (Crisp Set) yang hanya memiliki dua nilai keanggotaan yaitu 0 dan 1. Dalam logika *fuzzy* nilai kebenaran suatu pernyataan berkisar dari sepenuhnya benar sampai dengan sepenuhnya salah.

2.10.2 Himpunan Fuzzy

Himpunan *fuzzy* merupakan suatu group yang mewakili suatu kondisi atau keadaa tertentu dalam suatu variabel *fuzzy*. Pada himpunan tegas (crisp), nilai keanggotaan suatu item x dalam suatu himpunan A, yang sering ditulis dengan flA[x], memiliki dua kemungkinan, yaitu : Satu (I), yang berarti bahwa suatu item menjadi anngota dalam suatu himpunan atau Nol (0), yang berarti bahwa suatu item tidak menjadi anggota dalam suatu himpunan.

Pada himpunan *fuzzy* nilai keanggotaan terletak pada rentang 0 sampai 1. Apabila x memiliki nilai keanggotaan *fuzzy* flA[x] = 0 berarti x tidak menjadi anggota himpunan A, demikian pula apabila x memiliki nilai keanggotaan *fuzzy* flA[x] = 1 berarti x menjadi anggota penuh pada himpunanA.

Kemiripan antara keanggotaan *fuzzy* dengan probabilitas terkadang menimbulkan kerancuan, karena memiliki nilai pada interval [0,1], namun interpretasi nilainya sangat berbeda. Keanggotaan *fuzzy* memberikan suatu ukuran terhadap pendapat atau keputusan, sedangkan probabilitas mengindikasikan proporsi terhadap keseringan suatu hasil bernilai benar dalam jangka panjang.

Himpunan *fuzzy* memiliki 2 atribut, yaitu :

1. Linguistik, yaitu penamaan suatu group yang mewakili suatu keadaan atau kondisi tertentu dengan menggunakan bahasa alami, seperti : Muda, Parobaya, Tua.
2. Numeris, yaitu suatu nilai (angka) yang menunjukkan ukuran dari suatu variabel seperti : 25,40,60.

Ada beberapa hal yang perlu diketahui dalam memahami suatu sistem *fuzzy*, yaitu :

1. Variabel *fuzzy*

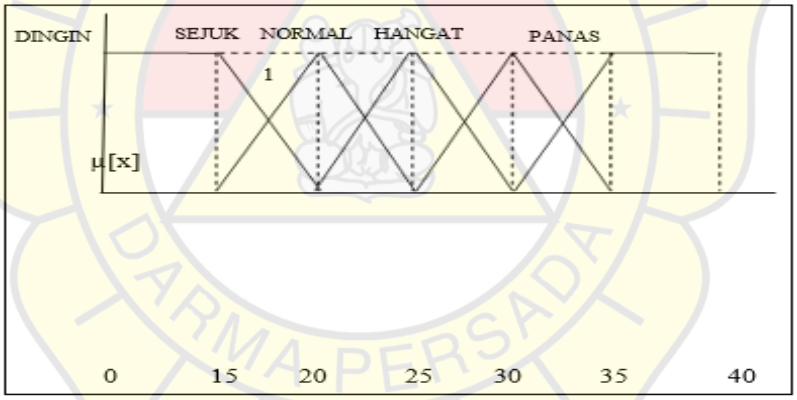
Variabel *fuzzy* merupakan variabel yang hendak dibahas dalam suatu system *fuzzy*. Contoh : umur, temperatur, permintaan, dsb.

2. Himpunan *fuzzy*

Himpunan *fuzzy* merupakan suatu grup yang mewakili suatu kondisi atau keadaan tertentu dalam suatu variabel *fuzzy*.

Contoh :

1. Variabel umur, terbagi menjadi 3 himpunan *fuzzy*, yaitu : MUDA, PAROBAYA, TUA.
2. Variabel temperatur, terbagi menjadi 5 himpunan *fuzzy*, yaitu : DINGIN, SEJUK, NORMAL, HANGAT, dan PANAS.



Gambar 2.5 Himpunan *Fuzzy* pada variabel temperatur

3. Semesta Pembicaraan

Semesta pembicaraan adalah keseluruhan nilai yang diperbolehkan umtuk dioperasikan dalam suatu variabel *fuzzy*. Semesta pembicaraan merupakan himpunan bilangan real yang senantiasa naik (bertambah) secara monoton dari kiri ke kanan. Nilai semesta pembicaraan dapat berupa bilangan positif maupun negatif. Adakalanya nilai semesta pembicaraan ini tidak dibatasi

batas atasnya.

Contoh :

1. Semesta pembicaraan untuk variabel umur : [0 +∞]
2. Semesta pembicaraan untuk variabel temperatur : [0 40]
4. Domain

Domain himpunan *fuzzy* adalah keseluruhan nilai yang diijinkan dalam semesta pembicaraan dan boleh dioperasikan dalam suatu himpunan *fuzzy*. Seperti halnya semesta pembicaraan, domain merupakan himpunan bilangan real yang senantiasa naik (bertambah) secara monoton dari kiri ke kanan. Nilai domain dapat berupa bilangan positif maupun negatif.

Contoh domain himpunan *fuzzy* :

1. MUDA : [0 45]
2. PAROBAYA : [33 45]
3. PAROBAYA : [33 45]

2.10.3 Metode Tsukamoto

Dalam membangun sebuah sistem *fuzzy* dikenal beberapa metode penalaran, antara lain: metode *Tsukamoto*, metode *Mamdani* dan metode *Sugeno.*

Pada metode Tsukamoto, setiap konsekuen pada aturan yang berbentuk IF-THEN harus direpresentasikan dengan suatu himpunan *fuzzy* dengan fungsi keanggotaan yang monoton. Sebagai hasilnya, output hasil inferensi dari tiap-tiap aturan diberikan dengan tegas (*crisp*) berdasarkan α-predikat (*fire strength*). Hasil akhirnya diperoleh dengan menggunakan rata-rata terbobot.

Misal ada 2 variabel input, var-1(x) dan var-2(y) serta 1 variabel output var-3(z), dimana var-1 terbagi atas 2 himpunan yaitu A1 dan A2 dan var-2 terbagi atas himpunan B1 dan B2. Sedangkan var-3 juga terbagi atas 2 himpunan yaitu C1 dan C2. (Kusumadewi, 2003).

Ada dua aturan yang digunakan yaitu:

[R1] IF (x is A1) and (y is B2) THEN (z is C1)

[R2] IF (x is A2) and (y is B1) THEN (z is C2)

[R3] IF Permintaan NAIK And Persediaan BANYAK

THENProduksi Barang BERTAMBAH;

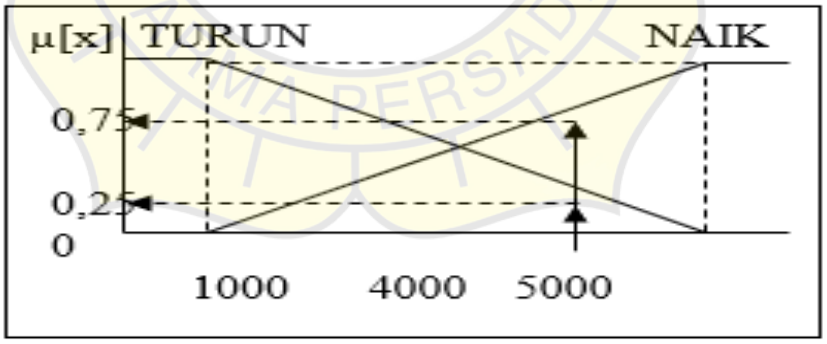
[R4] IF Permintaan NAIK And Persediaan SEDIKIT

THENProduksi Barang BERTAMBAH;

Solusi :

Ada 3 variabel yang akan dimodelkan, yaitu :

1. Permintaan, terdiri atas 2 himpunan *fuzzy* : NAIK dan TURUN.



Gambar 2.6 Fungsi Keanggotaan Variabel Permintaan

$$\mu PmtTURUN[x] = \begin{cases} 1, & x \leq 1000 \\ \frac{5000 - x}{4000}, & 1000 \leq x \leq 5000 \\ 0, & x \geq 5000 \end{cases}$$

$$\mu PmtNAIK[x]=\begin{cases}0, & x \leq 1000\\ \dfrac{x-1000}{4000}, & 1000 \leq x \leq 5000\\ 1, & x \geq 5000\end{cases}$$

Mencari nilai keanggotaan :

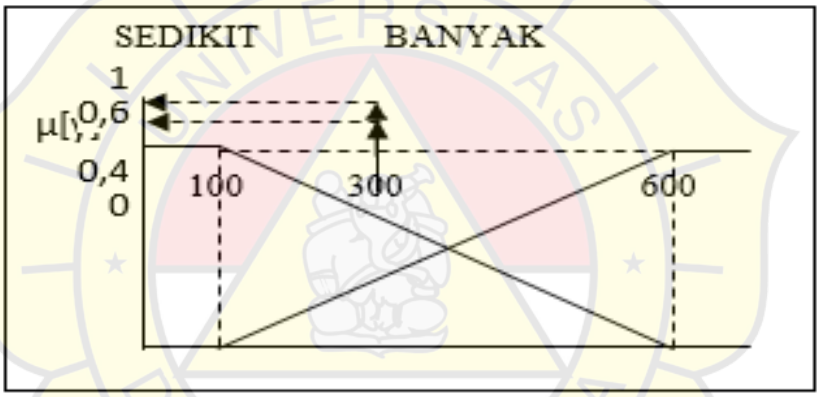
μPmtTURUN(4000) = (5000-4000)/4000

= 0,25

μPmtNAIK(4000) = (4000-1000)/4000

= 0,75

2. Persediaan, terdiri atas 2 himpunan *fuzzy* :SEDIKIT dan BANYAK



Gambar 2.7 Fungsi Keanggotaan Variabel Persediaan

$$\mu PsdSEDIKIT[y]=\begin{cases}1, & y \leq 100\\ \dfrac{600-y}{500}, & 100 \leq y \leq 600\\ 0, & y \geq 600\end{cases}$$

$$\mu PsdBANYAK[y]=\begin{cases}1, & y \leq 100\\ \dfrac{y-100}{500}, & 100 \leq y \leq 600\\ 0, & y \geq 600\end{cases}$$

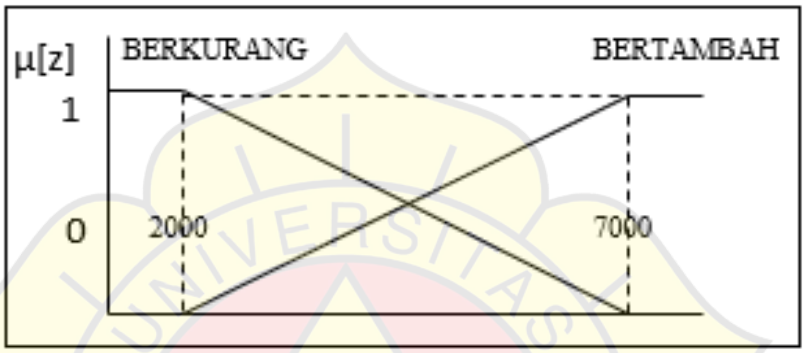
Mencari nilai keanggotaan :

μPsdSEDIKIT(300) = (600-300)/500

= 0,6

μPsdBANYAK(300) = (300-100)/500= 0,4

3. Produksi, terdiri atas 2 himpunan *fuzzy* : BERKURANG dan BERTAMBAH.



Gambar 2.8 Fungsi Keanggotaan Variabel Produksi Barang

$$\mu PrBrgBERKURANG[z]=\begin{cases}1, & z \le 2000\\ \dfrac{7000-z}{5000}, & 2000 \le z \le 7000\\ 0, & z \ge 7000\end{cases}$$

$$\mu PrBrgBERTAMBAH[z]=\begin{cases}0, & z \le 2000\\ \dfrac{z-2000}{5000}, & 2000 \le z \le 7000\\ 1, & z \ge 7000\end{cases}$$

Cari nilai z untuk setiap aturan dengan menggunakan fungsi MIN pada aplikasi fungsi implikasinya :

[R1] IF Permintaan TURUN And Persediaan BANYAK THEN Produksi Barang BERKURANG;

α-predikat$_1$ = μPmnTURUN $\cap$ PsdBANYAK

= min(μPmnTURUN(4000), μPsdBANYAK(300))

= min(0,25;0,4)

= 0,25

Lihat himpunan Produksi barang BERKURANG,

(7000-z)/5000 = 0,25 ⟶ z_1 = 5750

[R2] IF Permintaan TURUN And Persediaan SEDIKIT THENProduksi Barang BERKURANG;

α-predikat$_2$ = μPmtTURUN $\cap$ PsdSEDIKIT

= min(μPmtTURUN(4000), μPsdSEDIKIT(300))

= min(0,25;0,6)

= 0,25

Lihat himpunan Produksi barang BERKURANG,

(7000-z)/5000 = 0,25 ⟶ z_2 = 5750

[R3] IF Permintaan NAIK And Persediaan BANYAK THENProduksi Barang BERTAMBAH;

α-predikat$_3$ = μPmtNAIK $\cap$ PsdBANYAK

= min(μPmtNAIK(4000), μPsdBANYAK(300))

= min(0,75;0,4)

= 0,4

Lihat himpunan Produksi barang BERTAMBAH,

(z-2000)/5000 = 0,4 ⟶ z_3 = 4000

[R4] IF Permintaan NAIK And Persediaan SEDIKIT

THENProduksi Barang BERTAMBAH;

α-predikat$_4$ = μPmtNAIK $\frown$ PsdSEDIKIT

= min(μPmTNAIK(4000), μPsdSEDIKIT(300))

= min(0,75;0,6)

= 0,6

Lihat himpunan Produksi barang BERTAMBAH,

(z-2000)/5000 = 0,6 ⟶ z_4 = 4000

Nilai z dapat dicari dengan cara sebagai berikut :

$$z = \frac{\alpha pred_1 * z_1 + \alpha pred_2 * z_2 + \alpha pred_3 * z_3 + \alpha pred_4 * z_4}{0{,}25 + 0{,}25 + 0{,}4 + 0{,}6}$$

$$z = \frac{0{,}25 * 5750 + 0{,}25 * 5750 + 0{,}4 * 4000 + 0{,}6 * 5000}{0{,}25 + 0{,}25 + 0{,}4 + 0{,}6} = \frac{7475}{1{,}5} = 4983$$

Jadi jumlah makanan kaleng jenis ABC yang harus diproduksi sebanyak **4983** kemasan.

2.11 Kerangka Pemikiran

Suatu diagram yang menjelaskan secara garis besar alur logika berjalannya suatu penelitian.