# BAB II
# LANDASAN TEORI

## 2.1. Rekayasa Perangkat Lunak

Rekayasa perangkat lunak atau *software engineering* merupakan proses pemecahan problem yang dihadapi *customer* dengan pengembangan dan evolusi sistem perangkat lunak yang berkualitas tinggi dengan mempertimbangkan kekangan biaya dan waktu. Rekayasa perangkat lunak merupakan disiplin ilmu teknik yang menaruh perhatian pada aspek produk perangkat lunak dan harus mengadopsi suatu pendekatan yang sistematik dan terorganisir serta menggunakan *tools* dan teknik yang bergantung pada problem yang akan dipecahkan, kekangan yang dihadapi dan sumber-sumber (*resources*) yang dimiliki.

Tujuan dari rekayasa perangkat lunak ialah:

1) Menghasilkan perangkat lunak yang bebas kesalahan (*fault-free software*).
2) Perangkat lunak yang selesai tepat waktu, sesuai anggaran biaya.
3) Perangkat lunak yang memenuhi kebutuhan pemakai.

### 2.1.1. Analisis Perangkat Lunak

Analisis perangkat lunak merupakan proses menganalisis permasalahan yang muncul dan menentukan spesifikasi kebutuhan atas sistem perangkat lunak yang akan dibuat. Untuk menentukan batasan masalah dan

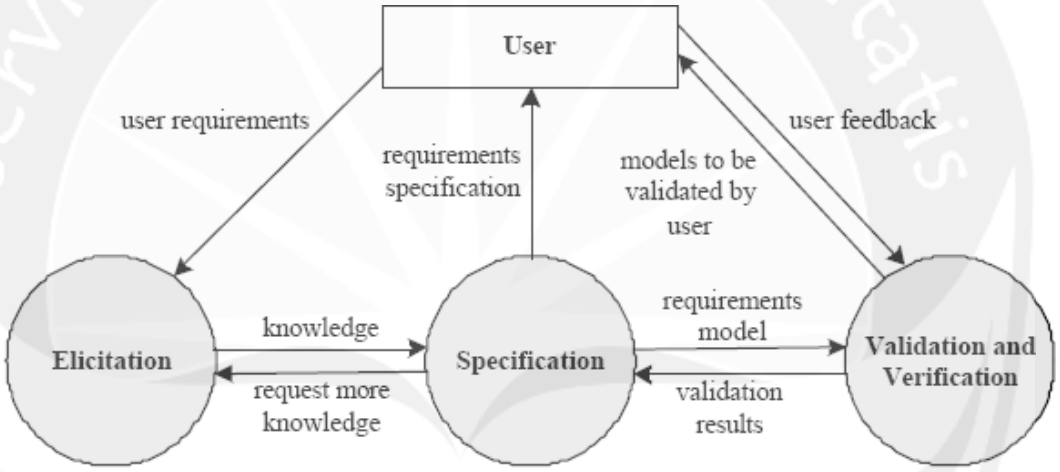
spesifikasi kebutuhan perangkat lunak diperlukan suatu proses analisis kebutuhan (*requirements engineering*).

*Requirements engineering* adalah 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) (Zave, 1997).

Definisi lain dari *requirements engineering* adalah fase terdepan dari proses rekayasa perangkat lunak (*software engineering)*, dimana *software requirements* (spesifikasi kebutuhan) dari *user* (pengguna) dan *customer* (pelanggan) dikumpulkan, dipahami dan ditetapkan. Para pakar rekayasa perangkat lunak (*software engineering*) sepakat bahwa *requirements engineering* adalah suatu pekerjaan yang sangat penting, terutama berdasarkan fakta bahwa kebanyakan kegagalan pengembangan *software* disebabkan karena adaya ketidakkonsistenan (*inconsistent*), ketidaklengkapan (*incomplete*), maupun ketidakbenaran (*incorrect*) dari *requirements specification* (spesifikasi kebutuhan) (Romi, 2003).

Hasil dari fase *requirements engineering* terdokumentasi dalam *requirements specification*. *Requirements specification* berisi kesepakatan bersama tentang permasalahan yang ingin dipecahkan antara pengembang dan customer, dan merupakan titik *start* menuju proses berikutnya yaitu desain perangkat lunak

(*software design)*. Sistemisasi proses negosiasi pengembang dan *customer* dalam *requirements engineering* dibagi dalam 3 proses besar yaitu: *elicitation, specification, validation and verification*. Formula ini kemudian juga dikenal dengan nama *The Three Dimensions of Requirements Engineering* (Gambar 2.1). Proses *requirements engineering* ini dilakukan secara terus menerus (iterasi) dengan mengakomodasi adanya *feedback* dari *customer* (*user*).




Gambar 2.1 *Requirements Engineering Process (Romi, 2003)*

#### 2.1.1.1. *Requirements Elicitation*

Adalah proses mengumpulkan dan memahami requirements dari user. Kadang masalah yang muncul berakar dari gap masalah *knowledge domain* (perbedaan disiplin ilmu yang dimiliki). *Customer* adalah *expert* pada domain yang *software*-nya ingin dikembangkan (*domain specialist*), dilain pihak sang pengembang (*requirements analyst*) adakalanya sama sekali buta

terhadap *knowledge domain* tersebut, meskipun tentu memahami dengan benar bagaimana sebuah *software* harus dikembangkan. Gap *knowledge domain* tersebut yang diharapkan bisa diatasi dengan adanya interaksi terus menerus dan berulang (iterasi) antara pengembang dan *customer*. Proses interaksi tersebut kemudian dimodelkan menjadi beberapa teknik dan metodologi diantaranya adalah *interviewing*, *brainstorming*, *prototyping*, *use case*, dan sebagainya.

#### 2.1.1.2. *Requirements Specification*

Setelah masalah berhasil dipahami, pengembang mendeskripsikannya dalam bentuk dokumen spesifikasi dokumen. Spesifikasi ini berisi tentang fitur dan fungsi yang diinginkan oleh *customer*, dan sama sekali tidak membahas bagaimana metode pengembangannya. IEEE mengeluarkan standar untuk dokumen spesifikasi kebutuhan yang terkenal dengan nama *IEEE Recommended Practice for Software Requirements Specifications* (IEEE, 1998). Dokumen spesifikasi kebutuhan bisa berisi *functional requirements*, *performance requirements*, *external interface requirements*, *design constraints*, maupun *quality requirements*.

#### 2.1.1.3. *Requirements Validation and Verification*

Setelah spesifikasi kebutuhan perangkat lunak berhasil dibuat, perlu dilakukan dua usaha:

1) *Validation* (validasi), yaitu proses evaluasi sistem atau komponen sebelum atau sesudah proses development untuk memastikan apakah sistem atau

komponen sesuai dengan kebutuhan perangkat lunak (IEEE-1012).

2) *Verification* (verifikasi), yaitu proses evaluasi sistem atau komponen untuk memastikan bahwa produk dari suatu fase tertentu telah sesuai dengan kondisi awal fase tersebut (IEEE-1012).

Proses validasi dan verifikasi ini melibatkan *customer* sebagai pihak yang menilai dan memberi *feedback* berhubungan dengan *requirements*.

### 2.1.2. Desain Perangkat Lunak

Analisis dan Desain perangkat lunak merupakan bagian awal dari proses panjang rekayasa perangkat lunak. Analisis dan desain perangkat lunak merupakan proses yang digunakan untuk membuat (*aotumated*) sistem perangkat lunak.

Desain perangkat lunak ialah proses perancangan sistem berdasarkan hasil analisis yang telah dilakukan. Perancangan dilakukan untuk mendapatkan deskripsi arsitektural perangkat lunak, deskripsi data dan deskripsi prosedural. Hasil perancangan berupa dokumen Deskripsi Perancangan Perangkat Lunak (DPPL).

Perancangan antarmuka perangkat lunak yaitu merancang media komunikasi yang efektif antara manusia dan komputer. Ada 3 hal yang perlu diperhatikan dalam merancang antarmuka perangkat lunak, antara lain yaitu sebagai berikut:

1. Tempatkan pengguna dalam kontrol

   Menempatkan pengguna dalam kontrol yaitu menyediakan interaksi yang fleksibel, interaksi yang dapat diinterupsi dan dibatalkan (*undo*) oleh pengguna,

interaksi yang dapat di *customised* dan interaksi secara langsung dengan obyek yang terlihat pada layar.

2. Minimalkan ingatan pengguna
   Meminimalisasi ingatan pengguna yaitu mengurangi kebutuhan pengguna untuk mengingat (misalnya: *shortcut*), memberikan standar (*default*) dalam penamaan kontrol dan mendefinisikan *shortcut* secara intuitif serta menggunakan *layout* berdasarkan metafora pada dunia nyata.
3. Antarmuka yang konsisten
   Antarmuka yang konsisten akan memungkinkan pengguna untuk menempatkan tugas (*task*) yang sedang dilakukan dalam konteks yang *meaningful* (pengguna mengetahui *task* yang sedang dikerjakan). Antarmuka yang konsisten berarti mempertahankan konsistensi bentuk antarmuka perangkat lunak, *layout*, posisi kontrol dan lainnya untuk semua *form* aplikasi perangkat lunak, dimana perubahan model interaktif hanya boleh dilakukan dengan alasan tertentu.

## 2.2. Basis Data

Dalam membangun suatu basis data diperlukan suatu pemahaman tentang basis data itu sendiri mulai dari apa itu basis data, komponen sistem basis data dan teknik-teknik normalisasi yang ada untuk mendapatkan suatu basis data yang terstruktur.

### 2.2.1. Pengertian Basis Data

Basis data adalah suatu kumpulan data terhubung yang disimpan secara bersama-sama pada suatu media,

tanpa adanya pengulangan (*redundancy*) yang tidak perlu sehingga mudah untuk digunakan atau ditampilkan kembali; dapat digunakan oleh satu atau lebih program aplikasi secara optimal; data tersimpan tanpa mengalami ketergantungan pada program yang akna menggunakannya; data disimpan sedemikian rupa sehingga penambahan, pengambilan dan modifikasi data dapat dilakukan dengan mudah dan terkontrol. Prinsip utama dari basis data ialah pengaturan data yang baik dan tujuan utamanya adalah kemudahan dan kecepatan dalam pengambilan kembali data tersebut.

Terdapat beberapa aturan yang harus dipatuhi pada file basis data agar dapat memenuhi kriteria sebagai suatu basis data. Beberapa aturan itu berhubungan dengan:

a. Kerangkapan data, yaitu muncul data-data yang sama secara berulang pada file basis data yang semestinya tidak diperlukan.
b. Inkonsistensi data, yaitu munculnya data yang tidak konsisten pada medan yang sama untuk beberapa file dengan kunci yang sama.
c. Data terisolasi, disebabkan oleh pemakaian beberapa file basis data. Program aplikasi yang digunakan tidak dapat mengakses file tertentu dalam sistem basis data tersebut, kecuali bila program aplikasi diubah atau ditambah sehingga seolah-olah ada file yang terpisah atau terisolasi terhadap file yang lain.
d. Keamanan data, berhubungan dengan masalah keamanan data dalam sistem basis data. Pada prinsipnya file

basis data hanya boleh digunakan oleh pemakai tertentu yang mempunyai wewenang untuk mengakses.

e. Integrasi data, berhubungan dengan unjuk kerja sistem agar dapat melakukan kendali atau kontrol pada semua bagian sistem sehingga sistem selalu beroperasi dalam pengendalian penuh.

### 2.2.2. Komponen Sistem Basis Data

#### 2.2.2.1. Perangkat Keras

Perangkat keras yang biasanya terdapat dalam sebuah sistem basis data adalah:

1. Komputer (satu untuk sistem yang *stand-alone* atau lebih dari satu untuk sistem jaringan).
2. Memori sekunder yang *online* (*harddisk*).
3. Memori sekunder yang *offline* (*tape* atau *removable disk*) untuk keperluan *backup* data.
4. Media/perangkat komunikasi (komunikasi untuk sistem jaringan).

#### 2.2.2.2. Perangkat Lunak

Aplikasi (perangkat lunak) ini bersifat optional. Artinya ada atau tidaknya tergantung pada kebutuhan. DBMS yang digunakan lebih berperan dalam pengorganisasian data dalam basis data, sementara bagi pemakai basis data (khususnya yang menjadi *end-user/native-user*) dapat dibuatkan atau disediakan program khusus/lain untuk melakukan pengisian, pengubahan dan pengambilan data. Program ini ada yang sudah disediakan bersama dengan DBMS-nya, ada juga yang harus dibuat sendiri dengan menggunakan

aplikasi lain yang khusus untuk itu (*development tools*).

#### 2.2.2.3. Pemakai

Ada beberapa jenis atau tipe pemakai terhadap suatu sistem basis data yang akan dibedakan cara berinteraksinya terhadap sistem:

1. Programmer Aplikasi
   Pemakai yang berinteraksi dengan basis data melalui *Data Manipulation Language* (DML), yang disertakan (*embedded*) dalam program yang ditulis dalam bahasa pemrograman.
2. User Mahir (*Casual User*)
   Pemakai yang berinteraksi dengan sistem tanpa menulis modul program. Mereka menyatakan *query* (untuk akses data) dengan bahasa *query* yang telah disediakan oleh suatu DBMS.
3. User Umum (*End User/Native User*)
   Pemakai yang berinteraksi dengan sistem basis data yang melalui pemanggilan satu program aplikasi permanen (*executable program*) yang telah ditulis atau disediakan sebelumnya.
4. User Khusus (*Specialized User*)
   Pemakai yang menulis aplikasi basis data non konvensional, tetapi untuk keperluan-keperluan khusus, seperti untuk aplikasi *Artificial Intelligence (AI)*, Sistem Pakar, Pengolahan Citra, dan lain-lain, yang bisa saja mengakses basis data dengan atau tanpa DBMS yang bersangkutan.

#### 2.2.2.4. Prosedur

Prosedur adalah suatu urut-urutan operasi klerikal (tulis menulis), biasanya melibatkan beberapa orang. Di dalam satu atau lebih departemen, yang diterapkan untuk menjamin penanganan yang seragam dari transaksi-transaksi bisnis yang terjadi.

Suatu prosedur adalah urut-urutan yang tepat dari tahapan-tahapan instruksi yang menerangkan apa (*what*) yang harus dikerjakan, siapa (*who*) yang mengerjakan, kapan (*when*) dikerjakan dan bagaimana (*how*) mengerjakan.

## 2.3. Logika Fuzzy

Logika fuzzy dikatakan sebagai logika baru yang lama, sebab ilmu tentang logika fuzzy modern dan metode baru ditemukan beberapa tahun yang lalu, padahal sebenarnya konsep tentang logika fuzzy itu sendiri sudah ada sejak lama.

Logika fuzzy adalah suatu cara yang tepat untuk memetakan suatu ruang input kedalam suatu ruang output. Sebagai contoh:

1. Manajer pergudangan mengatakan pada manajer produksi seberapa banyak persediaan barang pada akhir minggu ini, kemudian manajer produksi akan menetapkan jumlah barang yang harus diproduksi esok hari.
2. Pelayan restoran memberikan pelayanan terhadap tamu, kemudian tamu akan memberikan tip yang sesuai atas baik tidaknya pelayanan yang diberikan.

3. Anda mengatakan pada saya seberapa sejuk ruangan yang anda inginkan, saya akan mengatur putaran kipas yang ada pada ruangan ini.

Sejak diperkenalkan oleh Lotfi A Zadeh pada tahun 1965, himpunan fuzzy dan logika fuzzy semakin banyak diminati oleh para peneliti baik untuk diaplikasikan pada bidang ilmu tertentu maupun dilakukan pengembangan terhadap konsep yang telah diberikan. Teori himpunan fuzzy merupakan kerangka matematis yang digunakan untuk merepresentasikan ketidakpastian, ketidakjelasan, ketidaktepatan, kekurangan informasi, dan kebenaran parsial (Tettamanzi, 2001).

Ketidakpastian dapat digunakan untuk mendeskripsikan sesuatu yang berhubungan dengan ketidakpastian yang diberikan dalam bentuk informasi linguistik atau intuisi. Namun demikian dalam bentuk semantik, ketidakjelasan (*vague*) dan fuzzy secara umum tidak dapat dikatakan bersinonim.

Ada beberapa alasan mengapa orang mengguankan logika fuzzy (Cox, 1995), antara lain:

1. Konsep logika fuzzy mudah dimengerti. Konsep matematis yang didasari penalaran fuzzy sangat sederhana dan mudah dimengerti.
2. Logika fuzzy sangat fleksibel.
3. Logika fuzzy memiliki toleransi terhadap data-data yang tidak tepat.
4. Logika fuzzy mampu memodelkan fungsi-fungsi nonlinear yang sangat kompleks.
5. Logika fuzzy dapat membangun dan mengaplikasikan pengalaman-pengalaman para pakar secara langsung tanpa harus melalui proses pelatihan.

6. Logika fuzzy dapat bekerjasama dengan teknik-teknik kendali secara konvensional.
7. Logika fuzzy didasarkan pada bahas alami.

### 2.3.1. Himpunan fuzzy

Pada himpunan tegas (*crisp*), nilai keanggotaan suatu item x dalam suatu himpunan A, yang sering ditulis dengan $\mu_A[x]$, memiliki 2 kemungkinan, yaitu:

1. Satu (1), yang berarti bahwa suatu item menjadi anggota dalam suatu himpunan, atau
2. Nol (0), yang berarti bahwa suatu item tidak menjadi anggota dalam suatu himpunan.

Kalau pada himpunan tegas (*crisp*), nilai keanggotaan hanya ada 2 kemungkinan, yaitu 0 dan 1, pada himpunan fuzzy nilai keanggotaan terletak pada rentang 0 sampai 1. Apabila x memiliki nilai keanggotaan fuzzy $\mu_A[x]$=0 berarti x tidak menjadi anggota himpunan A, demikian pula apabila x memiliki nilai keanggotaan fuzzy $\mu_A[x]$=1 berarti x menjadi anggota penuh pada himpunan A.

Terkadang kemiripan antara keanggotaan fuzzy dengan probabilitas menimbulkan kerancuan, keduanya memiliki nilai pada interval [0, 1], namun interpretasi nilainya sangat berbeda antara kedua kasus tersebut. Pada teori himpunan fuzzy, komponen utama yang sangat berpengaruh adalah derajat keanggotaan. Fungsi keanggotaan merepresentasikan derajat kedekatan suatu obyek terhadap atribut tertentu, sedangkan pada teori probabilitas lebih pada penggunaan frekuensi relatif (Ross, 2005).

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 grup yang mewakili suatu keadaan atau kondisi tertentu dengan menggunakan bahasa alami.
2. Numerik, yaitu suatu nilai (angka) yang menunjukkan ukuran dari suatu variabel.

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

1. Variabel fuzzy

   Variabel fuzzy merupakan variabel yang hendak dibahas dalam suatu sistem fuzzy. Contoh: umur dan temperatur.

2. Himpunan fuzzy

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

3. Semesta pembicaraan

   Semesta pembicaraan adalah keseluruhan nilai yang diperbolehkan untuk 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.

4. Domain

Domain himpunan fuzzy adalah keseluruhan nilai yang diijinkan dalam semsesta 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.

### 2.3.2. Fungsi keanggotaan

Fungsi keanggotaan (*membership function*) adalah suatu kurva yang menunjukkan pemetaan titik-titik input data kedalam nilai keanggotaannya (sering juga disebut dengan derajat keanggotaaan) yang memiliki interval antara 0 sampai 1. Salah satu cara yang dapat digunakan untuk mendapatkan nilai keanggotaan adalah dengan melalui pendekatan fungsi.

### 2.3.3. Operasi Dasar Zadeh untuk Himpunan Fuzzy

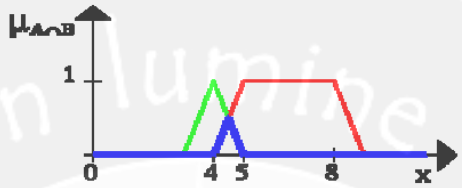
Seperti halnya himpunan konvensional, ada beberapa operasi yang didefinisikan secara khusus untuk mengkombinasi dan memodifikasi himpunan fuzzy. Nilai keanggotaan sebagai hasil dari operasi 2 himpunan sering dikenal dengan nama *fire strength* atau $\alpha$-predikat. Ada 3 operator dasar yang diciptakan oleh Zadeh, yaitu:

1. Operator AND

Operator ini berhubungan dengan operasi interseksi pada himpunan. $\alpha$-predikat sebagai hasil operasi dengan operator AND diperoleh dengan mengambil nilai

keanggotaan terkecil antar elemen pada himpunan-himpunan yang bersangkutan.

$$\mu_{A\cap B} = \min(\mu_A[x], \mu_B) \quad (2.1)$$

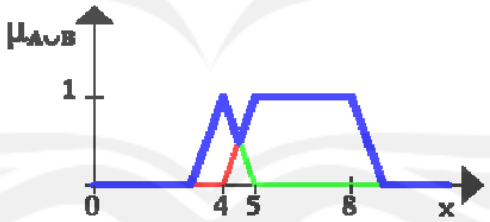



Gambar 2.2 *Fuzzy AND (Hellmann, 2001).*

2. Operator OR

Operator ini berhubungan dengan operasi union pada himpunan. α-predikat sebagai hasil operasi dengan operator OR diperoleh dengan mengambil nilai keanggotaan terbesar antar elemen pada himpunan-himpunan yang bersangkutan.

$$\mu_{A\cup B} = \max(\mu_A[x], \mu_B) \quad (2.2)$$

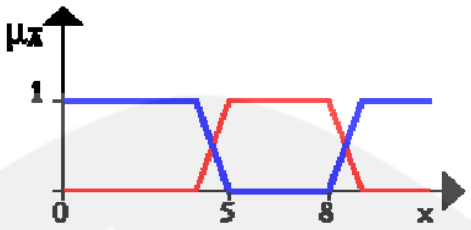



Gambar 2.3 *Fuzzy OR (Hellmann, 2001).*

3. Operator NOT

Operator ini berhubungan dengan operasi komplemen pada himpunan. α-predikat sebagai hasil operasi dengan operator NOT diperoleh dengan mengurangkan nilai keanggotaan elemen pada himpunan yang bersangkutan dari 1.

$$\mu_{A'} = 1 - \mu_A[x] \quad (2.3)$$




Gambar 2.4 *Fuzzy NOT (Hellmann, 2001).*

## 2.4. Data Mining

Data Mining adalah serangkaian proses untuk menggali nilai tambah dari suatu kumpulan data berupa pengetahuan yang selama ini tidak diketahui secara manual. Patut diingat bahwa kata mining sendiri berarti usaha untuk mendapatkan sedikit barang berharga dari sejumlah besar material dasar. Karena itu Data Mining sebenarnya memiliki akar yang panjang dari bidang ilmu seperti kecerdasan buatan (*artificial intelligence*), *machine learning*, statistik dan *database*.

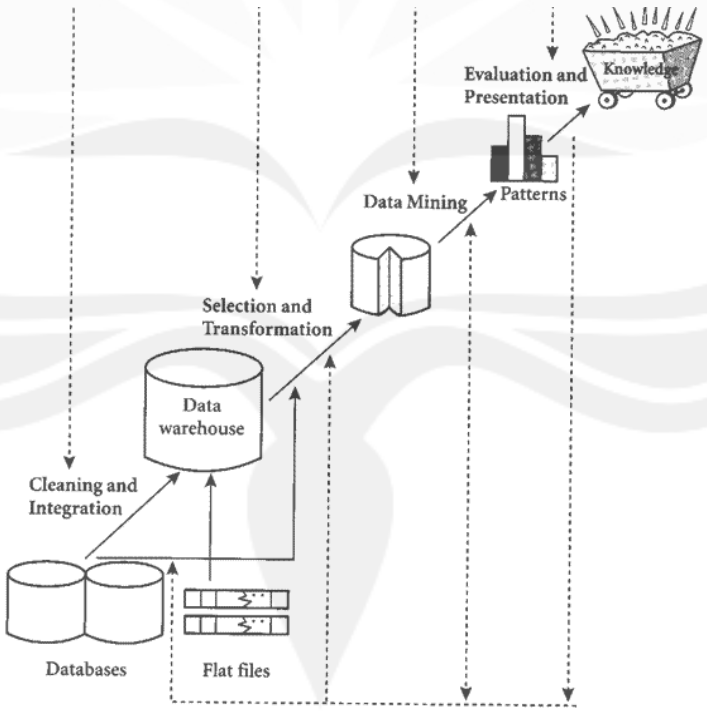
Data mining adalah proses ekstraksi (*non-trivial*, implisit, belum diketahui sebelumnya dan mungkin sangat berguna) untuk mencari informasi atau pola (*patterns*) dari data yang besar di database.

Beberapa teknik yang sering disebut-sebut dalam literatur Data Mining antara lain: *clustering, classification, association rule mining, neural network, genetic algorithm* dan lain-lain.

### 2.4.1. Tahap-Tahap Data Mining

Karena Data Mining adalah suatu rangkaian proses, Data Mining dapat dibagi menjadi beberapa tahap yang diilustrasikan di Gambar 2.5:

1. Pembersihan data (untuk membuang data yang tidak konsisten dan *noise*)
2. Integrasi data (penggabungan data dari beberapa sumber)
3. Transformasi data (data diubah menjadi bentuk yang sesuai untuk di-mining)
4. Aplikasi teknik Data Mining
5. Evaluasi pola yang ditemukan (untuk menemukan yang menarik/bernilai)
6. Presentasi pengetahuan (dengan teknik visualisasi)




Gambar 2.5 *Tahap-tahap Data Mining (Pramudiono, 2003).*

### 2.4.2. **Teknik-Teknik Data Mining**

Dengan definisi Data Mining yang luas, ada banyak jenis teknik analisa yang dapat digolongkan dalam Data Mining, yaitu:

***1.Classification***

*Classification* adalah proses untuk menemukan model atau fungsi yang menjelaskan atau membedakan konsep atau kelas data, dengan tujuan untuk dapat memperkirakan kelas dari suatu objek yang labelnya tidak diketahui. Model itu sendiri bisa berupa aturan "jika-maka", berupa *decision tree*, formula matematis atau *neural network*.

Metode-metode *classification* adalah *Decision tree, Bayesian, neural network, genetic algorithm, fuzzy, case-based reasoning, dan k-nearest neighbor*.

Proses *classification* biasanya dibagi menjadi dua fase: *learning* dan *test*. Pada fase *learning*, sebagian data yang telah diketahui kelas datanya diumpankan untuk membentuk model perkiraan. Kemudian pada fase *test* model yang sudah terbentuk diuji dengan sebagian data lainnya untuk mengetahui akurasi dari model tersebut. Bila akurasinya mencukupi model ini dapat dipakai untuk prediksi kelas data yang belum diketahui.

***2.Association Rule Discovery***

*Association rule mining* adalah teknik mining untuk menemukan aturan assosiatif antara suatu kombinasi item. Contoh dari aturan assosiatif dari analisa pembelian di suatu pasar swalayan adalah bisa diketahui berapa besar kemungkinan seorang

pelanggan membeli roti bersamaan dengan susu. Dengan pengetahuan tersebut. pemilik pasar swalayan dapat mengatur penempatan barangnya atau merancang kampanye pemasaran dengan memakai kupon diskon untuk kombinasi barang tertentu. Penting tidaknya suatu aturan assosiatif dapat diketahui dengan dua parameter, *support* yaitu persentase kombinasi item tersebut. dalam database dan *confidence* yaitu kuatnya hubungan antar item dalam aturan assosiatif.

Algoritma yang paling populer dikenal sebagai Apriori dengan paradigma *generate* and *test*, yaitu pembuatan kandidat kombinasi item yang mungkin berdasar aturan tertentu lalu diuji apakah kombinasi item tersebut memenuhi syarat *support* minimum. Kombinasi item yang memenuhi syarat tersebut. disebut *frequent itemset*, yang nantinya dipakai untuk membuat aturan-aturan yang memenuhi syarat *confidence* minimum. Algoritma baru yang lebih efisien bernama FP-Tree.

***3.Clustering***

Berbeda dengan *association* rule *mining* dan *classification* dimana kelas data telah ditentukan sebelumnya, *clustering* melakukan pengelompokan data tanpa berdasarkan kelas data tertentu. Bahkan *clustering* dapat dipakai untuk memberikan label pada kelas data yang belum diketahui itu. Karena itu *clustering* sering digolongkan sebagai metode *unsupervised learning*.

Prinsip dari *clustering* adalah memaksimalkan kesamaan antar anggota satu kelas dan meminimumkan kesamaan antar kelas/klaster. *Clustering* dapat

dilakukan pada data yang memiliki beberapa atribut yang dipetakan sebagai ruang multidimensi.

Banyak algoritma *clustering* memerlukan fungsi jarak untuk mengukur kemiripan antar data, diperlukan juga metode untuk normalisasi bermacam atribut yang dimiliki data.

Beberapa kategori algoritma *clustering* yang banyak dikenal adalah metode partisi dimana pemakai harus menentukan jumlah k partisi yang diinginkan lalu setiap data dites untuk dimasukkan pada salah satu partisi, metode lain yang telah lama dikenal adalah metode hierarki yang terbagi dua lagi: *bottom-up* yang menggabungkan klaster kecil menjadi klaster lebih besar dan *top-down* yang memecah klaster besar menjadi klaster yang lebih kecil. Kelemahan metode ini adalah bila bila salah satu penggabungan/pemecahan dilakukan pada tempat yang salah, tidak dapat didapatkan klaster yang optimal.

Adapun beberapa metode *clustering* diantarnya *K-Means Clustering, Fuzzy C-Means Clustering (FCM), Mountain Clustering Method, Fuzzy Substractive Clustering, Partition Simplification Fuzzy C-Means (psFCM)* dan metode cluster lainnya.

***4.Regression***

Memprediksi sebuah nilai variabel yang diberikan secara terus menerus (*continuous*) berdasarkan nilai dari variabel lainnya, seperti ketergantungan antara model linear dan model non-linear.

***5.Deviation Detection***

Mendeteksi perubahan/penyimpangan signifikan dari perilaku normal, misalnya deteksi kesalahan *credit card*.

## 2.5. Fuzzy Clustering

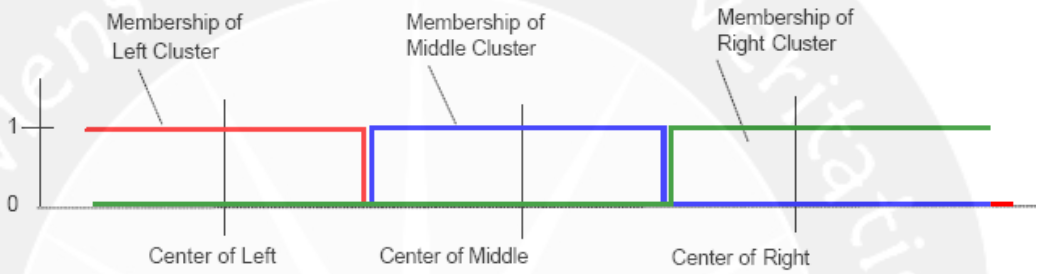
Pada logika fuzzy, ada beberapa metode yang dapat digunakan untuk melakukan pengelompokan sejumlah data yang dikenal dengan nama *fuzzy clustering*. Pada kebanyakan situasi, *fuzzy clustering*, lebih alami jika dibandingkan dengan pengklasteran secara klasik. Suatu algoritma *clustering* dekatakan sebagai algoritma *fuzzy clustering* jika dan hanya jika alogaritma tersebut menggunakan parameter strategi adaptasi secara *soft competitive (non-crisp)* (Baraldi, 1998). Sebagian besar algoritma fuzzy clustering didasarkan atas optimasi fungsi obyektif atau modifikasi dari fungsi obyektif tersebut.

Pemilihan algoritma *clustering* yang tepat, sangatlah penting demi kesuksesan proses *clustering*. Secara umum, algoritma pengklasteran dicirikan berdasarkan ukuran kedekatan dan kriteria pengklasteran (Vazirgiannis, 2003). Ukuran kedekatan menunjukan seberapa dekat fitur antara 2 data; sedangkan kriteria pengklasteran biasanya diekspresikan dengan menggunakan fungsi biaya atau tipe aturan yang lain.

1. **Partisi Klasik (*Hard Partition*)**

   Konsep partisi menjadi bagian yang sangat penting bagi proses pengclusteran. Tujuan proses pengklasteran pada partisi klasik adalah membagi

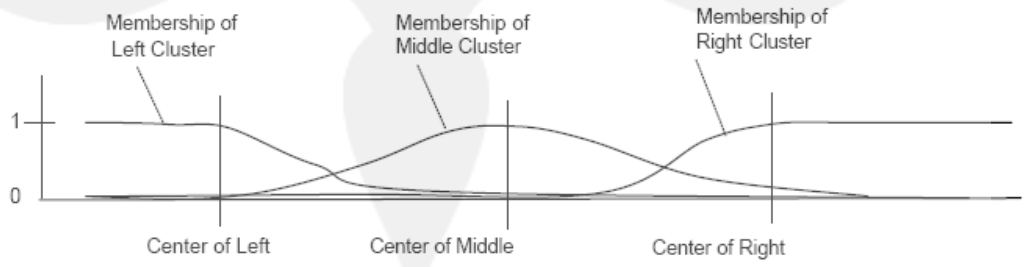
himpunan data X kedalam c kelompok (grup atau kelas), dengan asumsi bahwa c diketahui (Babuska, 2005). Dengan menggunakan teori himpunan klasik, partisi klasik X dapat didefinisikan sebagai suatu keluarga dari himpunan bagian–himpunan bagian $(A_i | 1 \le i \le c\} \subset$ P(X), P(X) adalah *power set* dari X (Bezdek, 1981).




Gambar 2.6 *Partisi Klasik/Hard (Cheok, 2005).*

**2. Partisi Fuzzy (*Fuzzy Partition*)**

Jika pada partisi klasik, suatu data secara eksklusif menjadi anggota hanya pada satu klaster saja, tidak demikian halnya dengan partisi fuzzy. Pada partisi fuzzy, nilai keanggotaan suatu data pada suatu klaster, $\mu_{ik}$ , terletak pada interval [0, 1] (Ruspini, 1970).




Gambar 2.7 *Partisi Fuzzy (Cheok, 2005).*

3. **Partisi Posibilistik (*Possibilistic Partition*)**

Tidak seperti halnya kedua partisi di atas, pada partisi posibilistik jumlah nilai keanggotaan suatu data pada semua klaster tidak harus 1, namun untuk menjamin suatu data menjadi anggota dari (paling tidak) satu klaster, maka diharuskan ada nilai keanggotaan yang bernilai lebih dari 0 (Krishnapuram, 1993).

### 2.5.1. Ukuran Fuzzy

Ukuran fuzzy menunjukkan derajat kekaburan dari himpunan fuzzy. Secara umum ukuran kekaburan dapat ditulis sebagai suatu fungsi (Yan, 1994):

$$f : p(X) \rightarrow R$$

Dengan P(X) adalah himpunan semua subset dari X; f(A) adalah suatu fungsi yang memetakan subset A ke karakteristik derajat kekaburannya.

Dalam mengukur nilai kekaburan, fungsi f harus mengikuti hal-hal sebagai berikut (Yan, 1994):

1. f(A) = 0 jika dan hanya jika A adalah himpunan crisp.
2. Jika A < B, maka f(A) <= f(B). Disini, A < B berarti B lebih kabur dibanding A (atau A lebih tajam dari B). relasi ketajaman A < B didefinisikan dengan:

   $\mu_A(x) \leq \mu_B(x)$, jika $\mu_B(x) \leq 0{,}5$ dan

   $\mu_A(x) \geq \mu_B(x)$ jika $\mu_B(x) \geq 0{,}5$.

f(A) akan mencapai maksimum jika dan hanya jika A benar-benar kabur secara maksimum. Tergantung pada

interprestasi derajat kekaburan, nilai fuzzy maksimal biasanya terjadi pada saat $\mu_A[x]=0{,}5$ untuk setip x.

### 2.5.2. Indeks Kekaburan

Indeks kekaburan adalah jarak antara suatu himpunan fuzzy A dengan himpunan *crisp* C yang terdekat. Himpunan *crisp* C terdekat dari himpunan fuzzy dinotasikan sebagai $\mu_C[x]=0$, jika $\mu_A[x]\le 0{,}5$, dan $\mu_C[x]=1$ jika $\mu_A[x]\ge 0{,}5$ (Yan, 1994).

Ada beberapa fungsi jarak yang dapat digunakan dalam mencari indeks kekaburan, yaitu (Pedrycz, 2005):

1. Hamming distance: $d(x,y)=\sqrt{\sum_{i=1}^{n}|x_i-y_i|}$ (2.4)

2. Euclidean distance: $d(x,y)=\sqrt{\sum_{i=1}^{n}(x_i-y_i)^2}$ (2.5)

3. Tehebyschev distance: $d(x,y)=\max_i|x_i-y_i|; i=1,2,\ldots,n$ (2.6)

4. Minkowski distance: $d(x,y)=\sqrt[p]{\sum_{i=1}^{n}(x_i-y_i)^p; p>0}$ (2.7)

5. Canberra distance: $d(x,y)=\sum_{i=1}^{n}\frac{|x_i-y_i|}{x_i+y_i};$ (2.8)

   dengan $x_i$ dan $y_i$ bernilai positif.

6. Angular separation : $d(x,y)=\frac{\sum_{i=1}^{n}x_iy_i}{\left[\sum_{i=1}^{n}x_i^2\sum_{i=1}^{n}y_i^2\right]^{1/2}}$ (2.9)

### 2.5.3. Fuzzy Entropy

Fuzzy entropy didefinisikan dengan fungsi (Yan, 1994):

$$f(A) = -\sum \{\mu_A[x]\log \mu_A[x] + [1-\partial\mu_A[x]]\log[1-\mu_A[x]]\} \quad (2.10)$$

### 2.5.4. Ukuran Kesamaan

Ukuran kesamaan digunakan untuk menunjukkan derajat perbedaan antara 2 himpunan fuzzy. Perbedaan antara premis suatu aturan dengan input fuzzy-nya kemudian dapat digunakan untuk menentukan nilai $\alpha$ pada suatu aturan.

## 2.6. Fuzzy Subtractive Clustering

*Subtractive Clustering* didasarkan atas ukuran densitas (potensi) titik-titik data dalam suatu ruang (variabel). Konsep dasar dari *Subtractive Clustering* adalah menentukan daerah-daerah dalam suatu variabel yang memiliki densitas tinggi terhadap titik-titik disekitarnya. Titik dengan jumlah tetangga terbanyak akan dipilih sebagai pusat klaster. Titik yang sudah terpilih sebagai pusat klaster ini kemudian akan dikurangi densitasnya. Kemudian algoritma akan memilih titik lain yang memiliki tetangga terbanyak untuk dijadikan pusat klaster yang lain. Hal ini akan dilakukan berulang-ulang hingga semua titik-titik data diuji.

Apabila terdapat N buah data: $X_1$, $X_2$, ..., $X_N$ dan dengan menganggap bahwa data-data tersebut sudah dalam keadaan normal, maka densitas titik $X_k$ dapat dihitung sebagai:

$$D_k = \sum_{j=1}^{N} \exp\left( - \frac{\| X_k - X_j \|}{(r/2)^2} \right) \quad (2.11)$$

Dengan $\| X_k - X_j \|$ adalah jarak antara $X_k$ dengan $X_j$, dan r adalah konstanta positif yang kemudian dikenal dengan nama jari-jari (*influence range*). Jari-jari berupa vektor yang akan menentukan seberapa besar pengaruh pusat klaster pada tiap-tiap variabel. Dengan demikian, suatu titik data akan memiliki densitas yang besar jika dia memiliki banyak tetangga dekat.

Setelah menghitung densitas tiap-tiap titik, maka titik dengan densitas tertinggi akan dipilih sebagai pusat klaster. Misalkan $X_{c1}$ adalah titik yang terpilih sebagai pusat klaster, sedangkan $D_{c1}$ adalah ukuran densitasnya. Selanjutnya densitas dari titik-titik disekitarnya akan dikurangi menjadi:

$$D_k = D_k - D_{c1} * \exp\left( - \frac{\| X_k - X_{c1} \|}{(r_b/2)^2} \right) \quad (2.12)$$

Dengan $r_b$ adalah konstansta positif. Hal ini berarti bahwa titik-titik yang berada dekat dengan pusat klaster $u_{c1}$ akan mengalami pengurangan densitas besar-besaran. Hal ini akan berakibat titik-titik tersebut akan sangat sulit untuk menjadi pusat klaster berikutnya. Nilai $r_b$ menunjukkan suatu lingkungan yang mengakibatkan titik-titik berkurang ukuran densitasnya. Biasanya $r_b$ bernilai lebih besar dibanding dengan r, $r_b = q*r_a$ (biasanya *squash factor* (q) = 1,5).

Setelah densitas tiap-tiap titik diperbaiki, maka selanjutnya akan dicari pusar klaster yang kedua yaitu $X_{c2}$. Sesudah $X_{c2}$ didapat, ukuran densitas setiap titik akan diperbaiki kembali, demikian seterusnya.

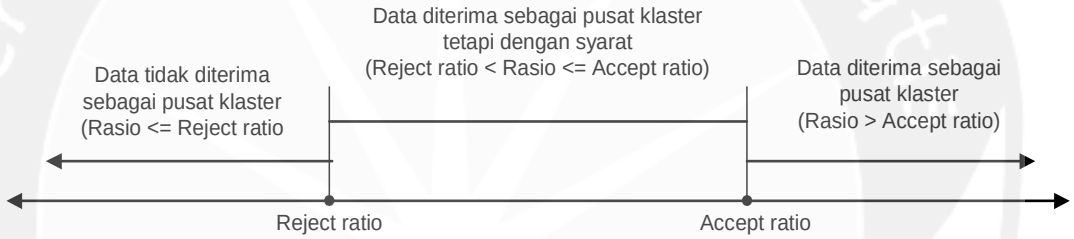
Pada implementasinya, bisa digunakan 2 pecahan sebagai faktor pembanding, yaitu *Accept ratio* dan *Reject ratio*. Baik *Accept ratio* maupun *Reject ratio* keduanya merupakan suatu bilangan pecahan yang bernilai 0 sampai 1. *Accept ratio* merupakan batas bawah dimana suatu titik data yang menjadi kandidat (calon) pusat klaster diperbolehkan untuk menjadi pusat klaster. Sedangkan *Reject ratio* merupakan batas atas dimana suatu titik data yang menjadi kandidat (calon) pusat klaster tidak diperbolehkan untuk menjadi pusat klaster. Pada suatu iterasi, apabila telah ditemukan suatu titik data dengan potensi tertinggi (misal $X_k$ dengan potensi $D_k$), kemudian akan dilanjutkan dengan mencari Rasio potensi titik data tersebut dengan potensi tertinggi suatu titik data pada awal iterasi (misal: $X_h$ dengan potensi $D_h$).

Hasil bagi antara $D_k$ dengan $D_h$ ini kemudian disebut dengan Rasio (Rasio = $D_k/D_h$). Ada 3 kondisi yang bisa terjadi dalam suatu iterasi (Gambar 2.8):

1. Apabila Rasio > Accept ratio, maka titik data tersebut diterima sebagai pusat klaster baru.
2. Apabila Reject ratio < Rasio ≤ Accept ratio, maka titik data tersebut baru akan diterima sebagai pusat klaster baru hanya jika titik data tersebut terletak pada jarak yang cukup jauh dengan pusat klaster yang lainnya (hasil penjumlahan antara Rasio dan jarak terdekat titik data tersebut dengan pusat klaster lainnya yang telah ada ≥ 1). Apabila hasil penjumlahan antara Rasio dan jarak terdekat titik data tersebut dengan pusat klaster

lainnya yang telah ada < 1, maka selain titik data tersebut tidak akan diterima sebagai pusat klaster yang baru, dia sudah tidak dipertimbangkan lagi untuk menjadi pusat klaster baru (potensinya diset sama dengan nol).

3. Apabila Rasio ≤ Reject ratio, maka sudah tidak ada lagi titik data yang akan dipertimbangkan untuk menjadi kandidat pusat klaster, iterasi dihentikan.




*Gambar 2.8 Rasio, Accept ratio dan Reject ratio (Sri K dan Hari P, 2004)*

Perbedaan lain dengan metode-metode terawasi (misalnya: *K-Mean*, *Fuzzy C-Mean* (FCM)) adalah, jika pada metode-metode terawasi pusat klaster bisa jadi bukan merupakan salah satu dari data yang diklaster, tidak demikian halnya dengan *Subtractive Clustering*. Pada metode *Substractive Clustering*, suatu pusat klaster pasti merupakan salah satu data yang ikut diklaster, yaitu data dimana derajat keanggotaannya pada klaster tersebut sama dengan 1. Pada FCM misalnya, penjumlahan semua derajat keanggotaan selalu bernilai sama dengan 1. Sedangkan pada *Subtractive*

*Clustering*, penjumlahan semua derajat keanggotaanya belum tentu (bahkan jarang) bernilai sama dengan 1.

### 2.6.1. Algoritma Fuzzy Subtractive Clustering

Algoritma *fuzzy subtractive clustering* adalah sebagai berikut (Sri K dan Hari P, 2004):

1. Input data yang akan diklaster: $X_{ij}$, dengan i = 1, 2,...,n; dan j = 1, 2,...,m.

2. Tetapkan nilai:
   a. r (jari-jari atribut data)
   b. q (*squash factor*);
   c. Accept ratio;
   d. Reject ratio;
   e. Xmin (minimum data diperbolehkan);
   f. Xmax (maksimim data diperbolehkan);

3. Normalisasi.

$$X_{ij} = \frac{X_{ij} - X\min_j}{X\max_j - X\min_j}, \qquad (2.13)$$

   i = 1, 2,...,n; j = 1,2,...,m

4. Tentukan potensi awal tiap-tiap titik data.
   a. i = 1
   b. Kerjakan hingga i = n,
      b.1. $T_j = X_{ij}$; j = 1, 2,…,m
      b.2. Hitung:

$$Dist_{kj} = \left( \frac{T_j - X_{kj}}{r} \right)$$

   j=1, 2,..,m; k=1, 2,..,n

b.3. Potensi awal:

$$D_i = \sum_{k=1}^{n} e^{-4\left(\sum_{j=1}^{m} Dist_{kj}^2\right)}$$

b.4. i = i+1

5. Cari titik dengan potensi tertinggi.
   a. M = max[$D_i$ |i = 1, 2,…,n];
   b. h = i , sedemikian hingga $D_i$ = M;

6. Tentukan pusat klaster dan kurangi potensinya terhadap titik-titik di sekitarnya.
   a. Center = []
   b. $V_j = X_{hj}$; j = 1, 2,…,m;
   c. C = 0 (jumlah klaster);
   d. Kondisi = 1;
   e. Z = M;
   f. kerjakan selama (Kondisi ≠ 0) AND (Z ≠ 0):
      f.1. Rasio = Z/M
      f.2. Jika Rasio >= Accept ratio
           Maka Kondisi = 1; (ada calon pusat baru).
      f.3. Jika Rasio > Reject ratio
           (calon baru akan diterima sebagai pusat jika keberadaanya akan memberikan keseimbangan terhadap data-data yang terletak cukup jauh dengan pusat klaster yang telah ada), maka kerjakan:
           f.3.1. Md = -1;
           f.3.2. Kerjakan untuk i=1 sampai i=C:

$$G_{ij} = \frac{V_j - Center_{ij}}{r} \quad j = 1, 2, \ldots, m$$

$$Sd_i = \sum_{j=1}^{m} \left(G_{ij}\right)^2$$

Jika (Md < 0) ATAU (Sd < Md)
Maka Md = Sd

f.3.3. Smd = √Md;

f.3.4. Jika (Rasio + Smd) ≥ 1
Maka Kondisi = 1; (data diterima sebagai pusat klaster)

f.3.5. Jika tidak, (Rasio + Smd) < 1
Maka Kondisi = 2; (data tidak akan dipertimbangkan kembali sebagai pusat klaster)

f.4. Jika Rasio < Reject Ratio
Maka Kondisi = 0;

f.5. Jika Kondisi = 1 (calon pusat baru diterima sebagai pusat baru), kerjakan:

f.5.1. C = C+1;

f.5.2. $Center_c$ = V;

f.5.3. Kurangi potensi dari titik-titik didekat pusat klaster.

f.5.3.1. $S_{ij} = \frac{V_j - X_{ij}}{r_j * q}$ ;

j=1,2,..,m; i=1,2,..,n

f.5.3.2. $Dc_i = M * e^{-4\left[\sum_{j=1}^{m}(S_{ij})^2\right]}$;

i=1, 2,..,n

f.5.3.3. D = D – Dc;

f.5.3.4. Jika $D_i \leq 0$, maka $D_i=0$; i=1,2,…,n

f.5.3.5. Z = max[$D_i$ |i = 1, 2,…,n];

f.5.3.6. Pilih h = i, sedemikian hingga $D_i$=Z;

f.6. Jika kondisi = 2 (calon pusat baru tidak diterima sebagai pusat baru), maka:

f.6.1. $D_h$ = 0;

f.6.2. Z = max[$D_i$ |i = 1, 2,…,n];

f.6.3. Pilih h=i, sedemikian hingga $D_i$= Z;

7. Kembalikan pusat klaster dari bentuk ternormalisasi ke bentuk semula.

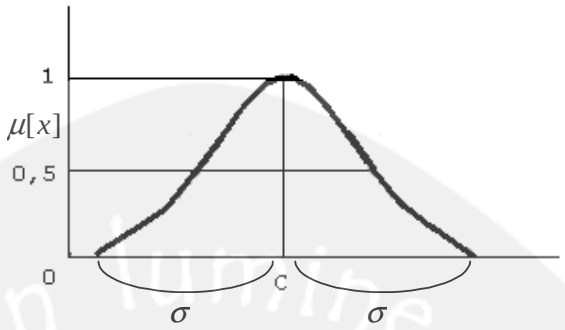
$Center_{ij} = Center_{ij} *(Xmax_j – Xmin_j) + Xmin_j$; (2.14)

8. Hitung nilai sigma klaster.

$$\sigma_j = r_j * (X\max_j - X\min_j) \Big/ \sqrt{8} \qquad (2.15)$$

9. Tentukan derajat keanggotaan titik-titik data.

Hasil algoritma *Subtractive Clustering* ini berupa matriks pusat klaster (C) dan *sigma* (σ) akan digunakan untuk menentukan nilai parameter fungsi keanggotaan *Gauss*.




*Gambar 2.9 Kurva Gauss (Sri K dan Hari P, 2004)*

Dengan kurva Gauss, maka derajat keanggotaan suatu titik data $X_i$ pada klaster ke-k, adalah:

$$\mu_{ki} = e^{-\sum_{j=1}^{m} \frac{\left(X_{ij} - C_{kj}\right)^2}{2\sigma_j^2}} \qquad (2.16)$$