BAB II

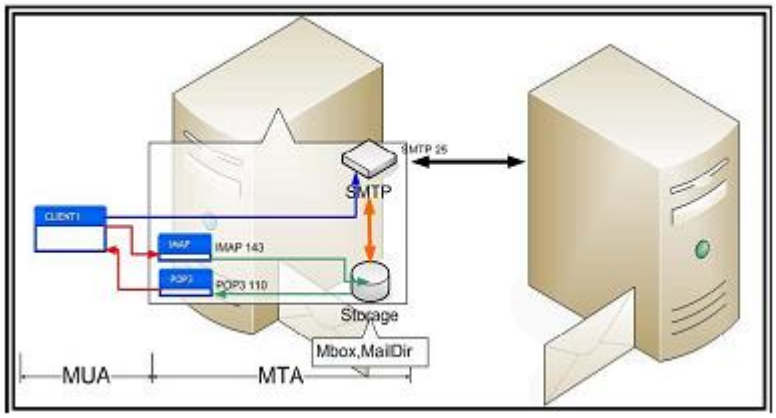
LANDASAN TEORI

Pada bab dua ini akan dibahas mengenai teori-teori yang digunakan penulis untuk membangun sistem yaitu mengenai pembangunan aplikasi penyaringan *spam e-mail* pada *mail server* dengan menggunakan metode *naïve bayes classifier*, metode rekayasa perangkat lunak dengan *waterfall*, perancangan sistem dengan *UML* dan teori lainya yang mendukung mengenai pembangunan aplikasi penyaringan *spam e-mail* pada *mail server* dengan menggunakan metode *naïve bayes classifier*.

2.1 *Mail* [11]

Electronic-Mail (*E-Mail*) merupakan sebuah metode untuk mengirimkan pesan dalam bentuk digital. Pesan ini biasanya dikirimkan melalui medium internet. Sebuah pesan elektronis terdiri dari isi, alamat pengirim, dan alamat-alamat yang dituju. Sistem *e-mail* yang beroperasi di atas jaringan berbasis pada model *store* and *forward*. Sistem ini mengaplikasikan sebuah sistem *server e-mail* yang menerima, meneruskan, mengirimkan, serta menyimpan pesan-pesan *user*, dimana *user* hanya perlu untuk mengkoneksikan *pc* mereka ke dalam jaringan.

E-mail dapat dianalogikan dengan kotak surat yang ada di kantor pos sedangkan *server e-mail* dapat diibaratkan sebagai kantor pos. Dengan analogi ini sebuah *mail server* dapat memiliki banyak *account e-mail* yang ada didalamnya.



Gambar 2.1 Cara kerja *e-mail*

Cara kerja *e-mail* yang dapat dilihat pada Gambar 2.1 menunjukkan bahwa *e-mail* yang dikirim belum tentu akan diteruskan ke komputer penerima (*end user*), tapi disimpan/dikumpulkan dahulu dalam sebuah komputer *server* (*host*) yang akan *online* secara terus menerus (*continue*) dengan media penyimpanan (*storage*) yang relatif lebih besar dibanding komputer biasa. Hal ini bisa diibaratkan dengan sebuah kantor pos, jika sesorang mempunyai alamat (*mailbox*), maka dia dapat memeriksa secara berkala jika dia mendapatkan surat. Komputer yang melayani penerimaan *e-mail* secara terus-menerus tersebut biasa disebut dengan *mailserver* atau *mailhost*.

2.2 *Spam Mail* [5]

Pendefinisian *spam e-mail* berbeda-beda, Undang-undang CAN-SPAM memberikan definisi utama *spam* dengan menjelaskan apa yang (dan apa yang tidak) diperbolehkan bila mengirim *e-mail* komersial pemasaran. Undang-undang tersebut disahkan pada tahun 2004 oleh *Federal Trade Commission*, yang diperbarui tahun 2008. Selain FTC terdapat badan-badan lain yang mengklasifikasikan *spam*, yaitu *Internet Service Provider* (ISP). *Internet Service Provider* juga memiliki bagian besar dalam menentukan apa yang dianggap *spam*.

ISP tidak mengandalkan CAN-SPAM sendirian untuk mendefinisikan *spam* karena di mata mereka *spam* didefinisikan oleh pengguna. Jika penerima *e-mail* mengelompokkan pesan *e-mail* sebagai *spam* dengan cara meletakkan didaftar pengirim yang diblokir mereka, menjatuhkannya di *folder spam* atau sekedar tidak konsisten membukanya, maka itu dianggap *spam* oleh ISP, terlepas dari apakah itu melekat pada masing-masing dan setiap CAN-SPAM aturan.

Berikut adalah tipe-tipe e-mail spam [9]:

a. Untuk Iklan: *Spam* dapat digunakan untuk mempromosikan suatu produk ataupun layanan, mulai dari produk *software*, perumahan *real estate* hingga produk kesehatan dan produk vitamin.
b. Untuk Mengirimkan *Malware*: *Spam* adalah salah satu cara utama untuk mendistribusikan *virus* dan *malware*. Dengan target yang bersifat individual, akan memperdaya korban untuk mempercayai bahwa

mereka menerima dokumen penting atau file tertentu, yang sebenarnya mengandung malware.

c. *Phishing*: Bersembunyi dibalik nama-nama besar perusahaan besar, lembaga keuangan, lembaga pemerintah, lembaga amal, para *phisher* mencoba memikat korban untuk mengunjungi *website* palsu, dimana melalui *website* tersebut mereka dapat mencuri data keuangan pribadi atau informasi dengan mengenai identitas korbannya.

d. *Scam*: Mengirimkan *e-mail* sebagai pangeran dari Nigeria, pegawai bank dari Swiss, seorang anak kecil yang sakit keras, dan beberapa tipe lainnya, para *scammer* berusaha memperoleh simpati.

e. Pesan yang tak berarti: Sebuah potongan pesan sampah seperti ini dapat memenuhi *inbox mail* kita. Bahkan beberapa pesan seperti ini dapat mengelabui teknologi *spam filter*, banyak pesan tak berarti ini dikirimkan tanpa tujuan yang jelas.

Perbedaan *Spam* dan Ham (bukan spam) berdasarkan struktur *e-mail* dapat diklasifikasikan sebagai berikut:

- *Header*

  E-mail header menunjukkan informasi perjalanan setiap e-mail. Secara umum, *e-mail header* terdiri dari pengirim, jaringan dan penerima *e-mail*.

- *Subject*

  Subject suatu *e-mail* biasanya merupakan suatu judul topik yang mewakili isi pada *e-mail*. *Subject e-mail* dapat dijumpai pada *header* setiap *e-mail*. Maka dapat dilihat pada gambar *header spam e-mail*, terdapat kata “VIAGRA”. Kata-kata tersebut sering dijumpai pada *subject spam e-mail*.

- *Body*

  Pada e-mail, body adalah isi dari suatu pesan e-mail, dan dengan adanya body e-mail, pengirim (sender) menyampaikan maksud yang akan disampaikan kepada penerima.

2.3 *Text Mining*[3]

Text mining, mengacu pada proses mengambil informasi berkualitas tinggi dari teks. Informasi berkualitas tinggi biasanya diperoleh melalui peramalan pola dan kecenderungan melalui sarana seperti pembelajaran pola statistik. *Text mining* biasanya melibatkan proses penataan teks input (biasanya *parsing*, bersama dengan penambahan beberapa fitur linguistik turunan dan penghilangan beberapa diantaranya, dan penyisipan *subsequent* ke dalam *database*), menentukan pola dalam data terstruktur, dan akhirnya mengevaluasi dan menginterpretasi *output*. "Berkualitas tinggi" di bidang *text mining* biasanya mengacu ke beberapa kombinasi relevansi, kebaruan, dan *interestingness*. Proses *text mining* yang khas meliputi kategorisasi teks, *text clustering*, ekstraksi konsep/entitas, produksi taksonomi granular, *sentiment analysis*, penyimpulan dokumen, dan pemodelan relasi entitas (yaitu, pembelajaran hubungan antara entitas bernama).

Klasifikasi/kategorisasi dokumen adalah masalah dalam ilmu informasi. Tugas kita adalah untuk menetapkan dokumen elektronik masuk dalam satu atau lebih kategori, berdasarkan isinya. Tugas klasifikasi dokumen dapat dibagi menjadi dua macam yaitu klasifikasi dokumen terawasi dimana beberapa mekanisme eksternal (seperti *feedback* manusia) memberikan informasi mengenai klasifikasi yang tepat untuk dokumen, dan klasifikasi dokumen tidak terawasi, dimana klasifikasi harus dilakukan sepenuhnya tanpa merujuk ke informasi eksternal. Ada juga klasifikasi dokumen semi-diawasi, dimana bagian dari dokumen diberi label oleh mekanisme eksternal.

Pendekatan *manual text mining* secara intensif dalam laboratorium pertama muncul pada pertengahan 1980-an, namun kemajuan teknologi telah memungkinkan ranah tersebut untuk berkembang selama dekade terakhir. *Text mining* adalah bidang inter disipliner yang mengacu pada pencarian informasi, pertambangan data, pembelajaran mesin, statistik, dan komputasi linguistik. Dikarenakan kebanyakan informasi (perkiraan umum mengatakan lebih dari 80%) saat ini disimpan sebagai teks, *text mining* diyakini memiliki potensi nilai komersial tinggi.

Saat ini *text mining* telah mendapat perhatian dalam berbagai bidang:

- Aplikasi Keamanan

  Banyak paket perangkat lunak *text mining* dipasarkan terhadap aplikasi keamanan, khususnya analisis *plain text* seperti berita internet. Hal ini juga mencakup studi enkripsi teks.

- Aplikasi Biomedis

  Berbagai aplikasi *text mining* dalam literatur biomedis telah disusun. Salah satu contohnya adalah PubGene yang mengkombinasikan *text mining* biomedis dengan visualisasi jaringan sebagai sebuah layanan internet. Contoh lain *text mining* adalah GoPubMed.org. Kesamaan semantik juga telah digunakan oleh sistem *text mining*, yaitu, GOAnnotator.

- Perangkat Lunak dan Aplikasi

  Departemen riset dan pengembangan perusahaan besar, termasuk IBM dan Microsoft, sedang meneliti teknik *text mining* dan mengembangkan program untuk lebih mengotomatisasi proses pertambangan dan analisis. Perangkat lunak *text mining* juga sedang diteliti oleh perusahaan yang berbeda yang bekerja di bidang pencarian dan pengindeksan secara umum sebagai cara untuk meningkatkan performansinya.

- Aplikasi Media *Online*

  Text mining sedang digunakan oleh perusahaan media besar, seperti perusahaan Tribune, untuk menghilangkan ambigu informasi dan untuk memberikan pembaca dengan pengalaman pencarian yang lebih baik, yang meningkatkan loyalitas pada *site* dan pendapatan. Selain itu, editor diuntungkan dengan mampu berbagi, mengasosiasi dan properti paket berita, secara signifikan meningkatkan peluang untuk menguangkan konten.

- Aplikasi Pemasaran

  Text mining juga mulai digunakan dalam pemasaran, lebih spesifik dalam analisis manajemen hubungan pelanggan. Coussement dan Van

den Poel (2008) menerapkannya untuk meningkatkan model analisis prediksi untuk *churn* pelanggan (pengurangan pelanggan).

- *Sentiment Analysis*

  Sentiment Analysis mungkin melibatkan analisis dari *review* film untuk memperkirakan berapa baik *review* untuk sebuah film. Analisis semacam ini mungkin memerlukan kumpulan data berlabel atau label dari efektifitas kata-kata. Sebuah sumber daya untuk efektivitas kata-kata telah dibuat untuk WordNet.

- Aplikasi Akademik

  Masalah *text mining* penting bagi penerbit yang memiliki *database* besar untuk mendapatkan informasi yang memerlukan pengindeksan untuk pencarian. Hal ini terutama berlaku dalam ilmu sains, dimana informasi yang sangat spesifik sering terkandung dalam teks tertulis. Oleh karena itu, inisiatif telah diambil seperti Nature's proposal untuk *Open Text Mining Interface* (OTMI) dan *Health's common Journal Publishing* untuk *Document Type Definition* (DTD) yang akan memberikan isyarat semantik pada mesin untuk menjawab pertanyaan spesifik yang terkandung dalam teks tanpa menghilangkan *barrier* penerbit untuk akses publik.

Text mining juga digunakan dalam beberapa *filter email spam* sebagai cara untuk menentukan karakteristik pesan yang mungkin berupa iklan atau materi yang tidak diinginkan lainnya.

2.4 *Naïve Bayes Classifier*[4]

2.4.1 Teorema *Naïve Bayes*

Naïve Bayes Classifier merupakan sebuah metoda klasifikasi yang berakar pada teorema *Bayes*. Ciri utama dari *Naïve Bayes Classifier* ini adalah asumsi yang sangat kuat (naïf) akan independensi dari masing-masing kondisi/kejadian. Sebelum menjelaskan *Naïve Bayes Classifier* ini, akan dijelaskan terlebih dahulu Teorema *Bayes* yang menjadi dasar dari metoda

tersebut. Pada teorema *Bayes*, bila terdapat dua kejadian yang terpisah (misalkan A dan B), maka teorema *Bayes* dirumuskan sebagai berikut:

$$P(A|B) = \frac{P(A)}{P(B)} P(B|A) \quad (1)$$

Teorema *Bayes* sering pula dikembangkan mengingat berlakunya hukum probabilitas total, menjadi seperti berikut:

$$P(A|B) = \frac{P(A)P(B|A)}{\sum_{i=1}^{n} P(Ai|B)}$$

dimana $A_1 \cup A_2 \cup \ldots \cup A_n = S$ (2)

Untuk menjelaskan teorema *Bayes*, perlu diketahui bahwa proses klasifikasi memerlukan sejumlah petunjuk untuk menentukan kelas apa yang cocok bagi sampel yang dianalisis tersebut. Karena itu, teorema *Bayes* di atas disesuaikan sebagai berikut:

$$P(C|F_1, \ldots, F_n) = \frac{P(C)P(F_1, \ldots, F_n|C)}{P(F_1, \ldots, F_n)} \quad (3)$$

Dimana variabel C merepresentasikan kelas, sementara variable F_1 ... F_n merepresentasikan karakteristik-karakteristik petunjuk yang dibutuhkan untuk melakukan klasifikasi. Maka rumus tersebut menjelaskan bahwa peluang masuknya sampel dengan karakteristik tertentu dalam kelas C (*posterior*) adalah peluang munculnya kelas C (sebelum masuknya sampel tersebut, seringkali disebut *prior*), dikali dengan peluang kemunculan karakteristik-karakteristik sampel pada kelas C (disebut juga *likelihood*), dibagi dengan peluang kemunculan karakteristik-karakteristik sampel secara global (disebut juga *evidence*). Karena itu, rumus (3) dapat pula ditulis secara sederhana sebagai berikut:

$$Posterior = \frac{prior \times likelihood}{evidence} \quad (4)$$

Nilai evidence selalu tetap untuk setiap kelas pada satu sampel. Nilai dari Posterior tersebut yang nantinya akan dibandingkan dengan nilai-nilai Posterior kelas lainnya untuk menentukan ke kelas apa suatu sampel akan diklasifikasikan. Penjabaran lebih lanjut rumus *Bayes* tersebut dilakukan dengan

menjabarkan $P(F_1...F_n|C)$ menggunakan aturan perkalian, menjadi sebagai berikut:

$$P(F_1, ..., F_n|C) = P(F_1|C)P(F_2, ..., Fn|C, F_1)$$
$$= P(F_1|C)P(F_2|C, F_1)P(F_3, ..., Fn|C, F_1, F_2)$$
$$\vdots$$
$$P(F_1, ..., F_n|C) = P(F_1|C)P(F_2|C, F_1)...P(F_n|C, F_1, F_2, ..., F_{n-1}) \quad (5)$$

Dapat dilihat bahwa hasil penjabaran tersebut menyebabkan semakin banyak dan semakin kompleksnya faktor-faktor syarat yang mempengaruhi nilai probabilitas, yang hampir mustahil untuk dianalisa satu-persatu. Akibatnya, perhitungan tersebut menjadi sulit untuk dilakukan. Di sinilah digunakan asumsi independensi yang sangat tinggi (naïf), bahwa masing-masing petunjuk (F_1, $F_2...F_n$) saling bebas (independen) satu sama lain. Dengan asumsi tersebut, maka berlaku suatu kesamaan sebagai berikut:

$$P(F_i|F_j) = \frac{P(F_i \cap F_j)}{P(F_j)} = \frac{P(F_i)P(F_j)}{P(F_j)} = P(F_i)$$

untuk $i \neq j$, sehingga

$$P(F_i|C, F_j) = P(F_i|C) \quad (6)$$

Dari persamaan di atas dapat disimpulkan bahwa asumsi independensi naïf tersebut membuat syarat peluang menjadi sederhana, sehingga perhitungan menjadi mungkin untuk dilakukan. Selanjutnya, penjabaran $P(F_1...F_n|C)$ dapat disederhanakan menjadi seperti berikut:

$$P(F_1...F_n|C) = P(F_1|C)P(F_2|C)...P(F_n|C)$$
$$P(F_1...F_n|C) = \prod_{i=1} P(F_i|C) \quad (7)$$

Dengan kesamaan di atas, persamaan teorema *Bayes* dapat dituliskan sebagai berikut:

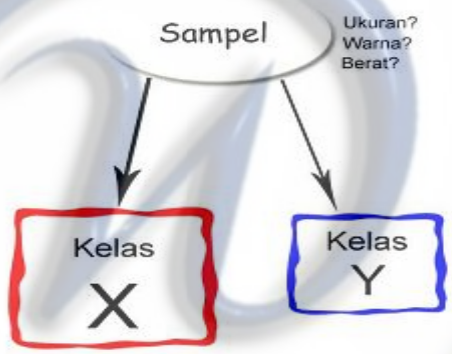
$$P(C|F_1...F_n) = \frac{1}{P(F_1, F_2, ..., F_n)} P(C) \prod_{i=1}^{n} P(F_i|C)$$
$$P(C|F_1...F_n) = \frac{P(C)}{Z} \prod_{i=1}^{n} P(F_i|C) \quad (8)$$

Persamaan di atas merupakan model dari teorema Naïve *Bayes* yang selanjutnya akan digunakan dalam proses klasifikasi dokumen. Adapun Z

merepresentasikan *evidence* yang nilainya konstan untuk semua kelas pada satu sampel.

2.4.2 Klasifikasi dengan *Naïve Bayes Classifier*

Klasifikasi adalah proses untuk menemukan model atau fungsi yang menjelaskan atau membedakan konsep atau kelas data, dengan tujuan untuk dapat memperkirakan kelas dari suatu obyek (Agus Mulyanto 2009). Oleh karena itu, kelas yang ada tentulah lebih dari satu. Penentuan kelas dari suatu dokumen dilakukan dengan cara membandingkan nilai probabilitas suatu sampel berada di kelas yang satu dengan nilai probabilitas suatu sampel berada di kelas yang lain.



Gambar 2.2 Ilustrasi contoh proses klasifikasi

Dengan persamaan teorema *Naïve Bayes* yang telah diturunkan di subbab A, kita mendapatkan nilai $P(C|F_1...F_n)$, yaitu nilai peluang suatu sampel dengan karakteristik F1...Fn berada dalam kelas C, atau dikenal dengan istilah *Posterior*. Umumnya kelas yang ada tidak hanya satu, melainkan lebih dari satu. Sebagai contoh, ahli statistik ingin mengklasifikasikan sampel kucing ke dalam jenis kelaminnya. Oleh karena itu, terdapat dua kelas yaitu jantan dan betina. Suatu sampel kucing akan diklasifikasikan ke dalam satu kelas saja, entah itu jantan atau betina, dengan melihat petunjuk-petunjuk yang ada (misalnya berat badan, panjang ekor, dan lain-lain).

Penentuan kelas yang cocok bagi suatu sampel dilakukan dengan cara membandingkan nilai *Posterior* untuk masing-masing kelas, dan mengambil

kelas dengan nilai *Posterior* yang tinggi. Secara matematis klasifikasi dirumuskan sebagai berikut:

$$C_{NB} = argmax_{c \in C}\, P(c) \prod_{i=1}^{n} P(f_i|c) \quad (9)$$

dengan c yaitu variabel kelas yang tergabung dalam suatu himpunan kelas C.

Dapat dilihat bahwa rumusan di atas tidak memuat nilai *Evidence* (Z). Hal ini disebabkan karena *evidence* memiliki nilai yang positif dan tetap untuk semua kelas sehingga tidak mempengaruhi perbandingan nilai *Posterior*. Karena itu, faktor Z ini dapat dihilangkan. Perlu menjadi perhatian pula bahwa metoda *Naïve Bayes Classifier* ini dapat digunakan bila sebelumnya telah tersedia data yang dijadikan acuan untuk melakukan klasifikasi.

Sebagai contoh, terdapat dua kelompok merek sepatu (X dan Y), dimana terdapat 3 petunjuk yang digunakan misalnya warna sepatu (merah, hitam), bahan sepatu (kulit, sintetis) dan model sepatu (Tali, Velkro). Sementara itu, terdapat pula 6 data seperti di bawah ini:

Warna	**Bahan**	**Model**	**Jenis**
Merah	Kulit	Tali	X
Hitam	Kulit	Tali	X
Merah	Sintetis	Velkro	Y
Hitam	Kulit	Velkro	Y
Hitam	Sintetis	Tali	Y
Hitam	Sintetis	Velkro	X

Tabel 2.1 Contoh data untuk klasifikasi metode *Naïve Bayes Classifier*

Bila terdapat sampel sepatu Hitam, Sintetis, Tali (tidak ada pada data di atas), klasifikasi dapat dilakukan dengan menggunakan *Naïve Bayes Classifier*. Pertama-tama harus dicari terlebih dahulu *Posterior* X dan Y untuk sampel tersebut.

P(X) = 3/6 = 0.5 P(Y) = 0.5

P(Hitam|X) = 2/3 = 0.66 P(Hitam|Y)=1/3=0.33

P(Sintetis|X)=1/3=0.33 P(Sintetis|Y)=2/3=0.66

P(Tali|X)=2/3=0.66 P(Tali|Y)=1/3=0.33

Posterior X = P(X) P(Hitam|X) P(Sintetis|X) P(Tali|X)

= 0.5 x 0.66 x 0.33 x 0.66 = 0.072

Posterior Y = P(Y) P(Hitam|Y) P(Sintetis|Y) P(Tali|Y)

= 0.5 x 0.33 x 0.66 x 0.33 = 0.034

Karena *Posterior* X > *Posterior* Y, maka sampel sepatu tersebut bermerek X.

2.5 Klasifikasi Dokumen

2.5.1 Metode *Naïve Bayes Classifier* untuk Klasifikasi Dokumen

Secara umum teknik klasifikasi dokumen sama seperti klasifikasi pada umumnya. Hal yang membedakan adalah karakteristik yang ditinjau. Pada klasifikasi secara umum (misalnya benda fisik), karakteristik yang dapat ditinjau merupakan karakteristik fisik yang beragam seperti ukuran, warna, bahan, dan lain-lain. Pada klasifikasi dokumen, karakteristik semacam itu tidak dapat ditemukan, karena umumnya dokumen hanya terdiri dari data-data literal (tulisan). Karena itu harus terdapat asumsi mengenai karakteristik yang ditinjau agar metoda *Naïve Bayes Classifier* dapat digunakan dalam klasifikasi dokumen.

Asumsi yang diambil dalam pengklasifikasian dokumen ini adalah dokumen dipandang sebagai kumpulan kata-kata yang saling bebas (independen) dan proses klasifikasi dokumen dilakukan dengan pengecekan kata-kata yang menyusun informasi di dalam dokumen tersebut. Penentuan kelas dari dokumen sampel dilakukan dengan cara menghitung besarnya peluang kata-kata pada dokumen suatu kelas yang muncul pada dokumen sampel yang dianalisis. Jadi, kata-kata dalam dokumen (w_i)berlaku seperti petunjuk-petunjuk yang telah dijelaskan sebelumnya (F_i), dan gabungan dari kata-kata tersebut menghasilkan suatu dokumen (D). Probabilitas sebuah kelas mengandung suatu dokumen merupakan produk dari probabilitas kata-kata dari dokumen tersebut yang terdapat pada kelas. Dengan kata lain:

$$P(D|C) = \prod_i P(w_i|C) \tag{10}$$

Persamaan teorema Naïve Bayes untuk klasifikasi dokumen menjadi sebagai berikut:

$$P(C|D) = \frac{P(C)}{P(D)} P(D|C) = \frac{P(C)}{P(D)} \prod_i P(w_i|C) \quad (11)$$

dimana P(C|D) menyatakan kemungkinan suatu dokumen diklasifikasikan pada kelas C.

2.5.2 Aplikasi

Aplikasi nyata yang sangat popular dari penggunaan metoda *Naïve Bayes Classifier* untuk klasifikasi dokumen dalam bidang teknologi informasi adalah penyaringan *spam* pada layanan surat elektronik (*spam filtering*). *Spam* adalah penyalahgunaan sistem pesan elektronik (termasuk media penyiaran dan system pengiriman digital) untuk mengirim berita iklan dan keperluan lainnya secara massal. Penyaringan surat elektronik *spam* (selanjutnya disebut "spam" saja) perlu dilakukan agar hanya informasi yang relevan saja yang tersampaikan dan menghindari diterimanya informasi yang tidak berguna, apalagi jika informasi tersebut merugikan penerimanya.

Salah satu langkah penting dalam penyaringan *spam* pada layanan surat elektronik adalah proses klasifikasi suar elektronik, yang akan memasukkan surat elektronik yang masuk (sampel) ke dalam salah satu dari dua kategori, yaitu kategori *spam* atau bukan *spam*. Ada banyak cara yang dapat digunakan untuk mendeteksi apakah suatu surat elektronik dikategorikan *spam* atau tidak, diantaranya:

1. Mendeteksi Alamat Pengirim

   Apabila alamat pengirim pernah dikategorikan sebagai *spam* sebelumnya (oleh pengguna layanan surat elektronik tertentu), maka surat elektronik yang dikirim melalui alamat tersebut otomatis terdeteksi sebagai *spam*. Cara ini memiliki banyak kelemahan, salah satunya adalah alamat surat elektronik yang begitu banyak di internet sehingga mendeteksi *spam* hanya dengan melihat alamat pengirim tidaklah efektif.

Terlebih lagi, sekarang ini sering terjadi pengiriman *spam* menggunakan alamat surat elektronik kerabat (oleh *automatic sender*) yang dengan mudah menipu perangkat lunak pendeteksi *spam*.

2. Mendeteksi Isi Surat Elektronik

Cara kedua ini menggunakan klasifikasi kata per kata untuk mengklasifikasikan surat elektronik sebagai spam atau bukan. Metoda klasifikasi yang paling popular digunakan saat ini adalah metoda *Naïve Bayes Classifier*. Cara ini didasarkan kepada statistik bahwa kebanyakan spam menggunakan pemilihan kata yang hampir sama untuk isi surat elektroniknya, misalnya menjurus kepada merek barang tertentu, atau kepada nama seseorang. Dengan menghitung kecocokan kata-kata dalam suatu surat elektronik dengan *library* kata-kata yang dikategorikan *spam* atau bukan *spam*, maka perangkat lunak dapat menentukan apakah suatu surat elektronik dimasukkan ke dalam kategori *spam* atau tidak. Kata-kata yang tersimpan di *library* dapat berasal dari surat-surat elektronik sebelumnya yang telah dinyatakan spam secara *manual* oleh pengguna layanan surat elektronik.

Ketika sebuah surat elektronik dikirimkan ke alamat pengirim tertentu, layanan surat elektronik akan mengimplementasikan perangkat lunak untuk mendeteksi apakah surat elektronik tersebut dikategorikan ke dalam *spam* atau tidak. Algoritma perangkat lunak tersebut didasarkan kepada persamaan (10) yang disesuaikan untuk kelas yang ada, yaitu "*spam*" dan "bukan *spam*". Nilai probabilitas surat elektronik tertentu diklasifikasikan pada kelas "*spam*" dirumuskan sebagai berikut. Misalkan variable S merepresentasikan kelas "*spam*":

$$P(S|D) = \frac{P(S)}{P(D)} \prod_i P(w_i|S) \tag{12}$$

Sementara itu, untuk kelas "bukan *spam*" yang merupakan negasi dari kelas "*spam*":

$$P(\bar{S}|D) = \frac{P(\bar{S})}{P(D)} \prod_i P(w_i|\bar{S}) \tag{13}$$

Penentuan kelas bagi surat elektronik yang dianalisis tersebut dilakukan dengan cara mencari nilai probabilistik yang paling maksimum antara kedua nilai yang didapat. Bila didapat bahwa P(S|D) > P(~S|D) maka surat elektronik tersebut masuk ke dalam kelas "*spam*". Sebaliknya, bila P(S|D) < P(~S|D) maka surat elektronik tersebut masuk ke dalam kelas "bukan *spam*".

Perlu diperhatikan bahwa jumlah kata-kata yang terdapat pada setiap surat elektronik berbeda-beda. Hal ini mengindikasikan bahwa faktor w_i bersifat kuantitatif (dapat dihitung), bukan kualitatif seperti contoh pada subbab 2B, dan karenanya memiliki kurva distribusi beserta data statistik seperti rata-rata dan simpangan baku kemunculan suatu kata dalam sebuah surat elektronik.

Katakanlah bahwa kata "Hubungi" diduga sebagai kata yang menjurus kepada *spam*, dan besar kemunculan suatu kata "Hubungi" dalam sebuah surat elektronik memiliki distribusi peluang berbentuk distribusi normal, maka dalam menganalisis suatu surat elektronik yang masuk, nilai probabilitas untuk P("Hubungi"|S) dihitung dengan metode distribusi normal baku.

$$Z = \frac{X - \mu}{\sigma} \tag{14}$$

Dengan X adalah jumlah kata "Hubungi" yang ada di dalam surat elektronik, μ adalah rata-rata kemunculan kata "Hubungi" dalam sebuah surat elektronik (didapat dari data-data sebelumnya), dan σ adalah simpangan baku. Bila nilai Z yang didapat mendekati 0, artinya nilai X mendekati nilai rata-rata, yang juga mengindikasikan bahwa ada tingkat kecocokan yang tinggi antara surat elektronik yang dianalisis dengan surat-surat elektronik lainnya, yang telah diklasifikasikan sebagai spam. Maka, nilai probabilitas P("Hubungi"|S) tinggi. Sebaliknya, bila nilai Z jauh dari 0, maka nilai probabilitas P("Hubungi"|S) rendah.

2.6 Metode N-Gram[6]

2.6.1 Pengertian

N-gram adalah potongan N-karakter yang diambilkan dari suatu string. Untuk mendapatkan N-gram yang utuh ditempuh dengan menambahkan blank pada awal dan akhir string. Misalnya suatu string “TEXT” setelah ditambah aal dan akhir dengan “_” sebagai pengganti blank akan didapat N-gram sebagai berikut :

Unigram : T,E,X,T

Bigram : _T, TE, EX,XT, dan T

Trigram : _TE,TEX,EXT, XT_ dan T_ _

Quadgram : _TEX, TEXT, EXT_, EX_ _, X_ _ _

Dapat disimpulkan bahwa untuk string berukuran n akan dimiliki n unigram dan n+1 bigram, n+1 trigram, n+1 quadgram dan seterusnya. Penggunaan N-gram untuk matching kata memiliki keuntungan sehingga dapat diterapkan pada recovery pada input karakter ASCII yang terkena noise, interpretasi kode pos, information retrieval dan berbagai aplikasi dalam pemrosesan bahasa alami.

Keuntungan N-gram dalam matching string adalah berdasarkan karakteristik N-gram sebagai bagian dari suatu string, sehingga kesalahan pada sebagian string hanya akan berakibat perbedaan pada sebagian N-gram. Sebagai contoh jika N-gram dari dua string dibandingkan, kemudian kita menghitung cacah N-gram yang sama dari dua string tersebut maka akan didapatkan nilai similaritas atau kemiripan dua string tersebut yang bersifat resistan terhadap kesalahan tekstual.

Kemiripan antara kata JOKO dengan JOKI (ada perbedaan 1 huruf), dapat diukur derajat kesamaan dengan cara menghitung berapa buah N-gram yang diambil dari dua kata tersebut yang bernilai sama, yaitu :

JOKO: _J, JO, OK,KO,O_ , JOKI : _J, JO, OK,KI, I_ kesamaan :3

Sementara antara kata JOKO dengan JONI (ada perbedaan 2 huruf), nilai kesamaan adalah :

JOKO: _J, JO, OK,KO,O_ , JONI : _J, JO, ON,NI,I_ kesamaan : 2

Sehingga dapat disimpulkan bahwa kemiripan atau kesamaan antara JOKO-JOKI dari pada antara JOKO-JONI.

2.6.2 Pembentukan Model N-gram Dokumen

Model *n-gram* dokumen dibentuk berdasarkan frekuensi *n-gram* yang muncul di dalam dokumen. Dokumen akan dibaca kata per kata, dan untuk setiap kata akan dibuat *n-gram* dari kata tersebut. Untuk setiap *n-gram* yang dibangkitkan, akan dicatat dalam sebuah *table* dengan *n-gram* sebagai kunci dan jumlah sebagai isi. Apabila *n-gram* tersebut sudah pernah muncul di dalam dokumen maka frekuensi untuk *n-gram* itu akan ditambah satu, jika belum maka *n-gram* tersebut akan ditambahkan ke dalam *table* dengan jumlah kemunculan satu.

Sebagai contoh untuk pembentukan model *n-gram* dokumen yang menggunakan *bi-gram* pada sebuah dokumen yang hanya berisi satu kalimat, ”*pengenalan bahasa suku bangsa indonesia berbasis teks dengan menggunakan metode ngram*”. Akan menghasilkan *bi-gram* (*2-gram*) sebagai berikut:

No	Kata	*Bi-gram*
1	pengenalan	_p; pe; en; ng; ge; en; na; al; la; an; n_
2	bahasa	_b; ba; ah; ha; as; sa; a_
3	suku	_s; su; uk; ku; u_
4	bangsa	_b; ba; an; ng; gs; sa; a_
5	indonesia	_i; in; nd; do; on; ne; es; si; ia; a_
6	berbasis	_b; be; er; rb; ba; as; si; is; s_
7	teks	_t; te; ek; ks; s_
8	dengan	_d; de; en; ng; ga; an; n_
9	menggunakan	_m; me; en; ng; gg; gu; un; na; ak; ka; an; n_
10	metode	_m; me; et; to; od; de; e_
11	ngram	_n; ng; gr; ra; am; m_

Tabel 2.2 Contoh Bi-gram Kalimat kesatu

Sebagai perbandingan berikut ini adalah proses yang sama dilakukan pada dokumen kedua yang berisi satu kalimat: ”*natural language identification by using ngram method”.* Akan menghasilkan *bi-gram* (*2-gram*):

No	Kata	*Bi-gram*
1	pengenalan	_p; pe; en; ng; ge; en; na; al; la; an; n_
2	bahasa	_b; ba; ah; ha; as; sa; a_
3	suku	_s; su; uk; ku; u_
4	bangsa	_b; ba; an; ng; gs; sa; a_
5	indonesia	_i; in; nd; do; on; ne; es; si; ia; a_
6	berbasis	_b; be; er; rb; ba; as; si; is; s_
7	teks	_t; te; ek; ks; s_
8	dengan	_d; de; en; ng; ga; an; n_
9	menggunakan	_m; me; en; ng; gg; gu; un; na; ak; ka; an; n_
10	metode	_m; me; et; to; od; de; e_
11	ngram	_n; ng; gr; ra; am; m_

Tabel 2.3 Contoh *Bi-gram* Kalimat Kedua

Setelah proses tersebut dilakukan pada seluruh kata dalam dokumen, maka *ngram* akan diurutkan berdasarkan frekuensi kemunculannya dalam dokumen. Daftar *n-gram* yang terurut berdasarkan kemunculannya inilah yang disebut sebagai model *n-gram* dokumen. Selanjutnya model *n-gram* dokumen akan digunakan untuk menentukan bahasa yang digunakan oleh dokumen tersebut. *Bi-gram* tersebut akan diurutkan berdasarkan frekuensi kemunculannya sehingga menjadi daftar seperti berikut:

No	Kata	*Bi-gram*
1	pengenalan	_p; pe; en; ng; ge; en; na; al; la; an; n_
2	bahasa	_b; ba; ah; ha; as; sa; a_
3	suku	_s; su; uk; ku; u_
4	bangsa	_b; ba; an; ng; gs; sa; a_
5	indonesia	_i; in; nd; do; on; ne; es; si; ia; a_
6	berbasis	_b; be; er; rb; ba; as; si; is; s_
7	teks	_t; te; ek; ks; s_
8	dengan	_d; de; en; ng; ga; an; n_
9	menggunakan	_m; me; en; ng; gg; gu; un; na; ak; ka; an; n_
10	metode	_m; me; et; to; od; de; e_
11	ngram	_n; ng; gr; ra; am; m_

Tabel 2.4 Model *N-gram* Dokumen

2.6.3 Deteksi Bahasa Dengan N-Gram

Penggunaan n-gram untuk deteksi bahasa didasarkan pada anggapan bahwa pola sebaran n-gram dari suatu bahasa bersifat unik karena ini terkait dengan frekuensi penggunaan huruf, atau pasangan huruf baik itu vokal atau konsonan dari suatu bahasa yang umumnya berbeda dengan bahasa yang lain. Untuk unigram misalnya, yang jika dihitunmg frekuensinya adalah frekuensi keumunculan huruf dalam teks bahasa tertentu yang akan uniq untuk bahasa yang berbeda. Untuk teks bahasa Indonesia vokal a akan merupakan vokal yang frekuensi munculnya paling tinggi, sementara untuk bahasa inggris vokal e merupakan vokal yang frekuensinya paling tinggi. Demikian juga jika digunkan abi-gram dan tri-gram, keunikan pola n-gram dari suatu bahasa akan nampak lebih menonjol.

2.6.4 Teknik Pengklasifikasian Bahasa

Teknik pengklasifikasian disini merupakan suatu teknik untuk mendapatkan keluaran dari sistem pengenalan bahasa. Pada dasarnya teknik ini sangat sederhana, setelah model *n-gram* dokumen dari masing-masing dokumen yang telah diketahui bahasanya (disebut juga sampel bahasa) didapatkan dari dokumen pelatihan bahasa dan model *n-gram* dokumen untuk pengujian didapatkan dari dokumen inputan pengujian (dokumen uji atau data uji), maka dapat dilakukan pengklasifikasian dengan membandingkan kedua model *n-gram* dokumen tersebut.

2.7 Metode Rekayasa Perangkat Lunak dengan *Waterfall*

Waterfall mengusulkan sebuah pendekatan kepada perkembangan perangkat lunak yang sistematik dan sekuensial yang mulai pada tingkat dan kemajuan sistem pada seluruh analisis, desain, kode, pengujian, dan pemeliharaan. Dimodelkan setelah siklus rekayasa *konvensional*, model *waterfall* melingkupi aktifitas-aktifitas berikut :

1. Rekayasa dan pemodelan sistem/ informasi. Karena perangkat lunak selalu merupakan bagian dari sebuah sistem yang lebih besar, kerja dimulai dengan membangun syarat dari semua elemen dari semua sistem dan

mengalokasikan beberapa subset dari kebutuhan perangkat lunak tersebut. Pandangan sistem ini penting ketika perangkat lunak harus berhubungan dengan elemen-elemen yang lain seperti perangkat lunak, manusia, dan *database*. Rekayasa dan analisis sistem menyangkut pengumpulan kebutuhan pada tingkat sistem dengan sejumlah kecil analisis serta desain tingkat puncak. Rekayasa informasi mencakup juga pengumpulan kebutuhan pada tingkat bisnis strategis dan tingkat area bisnis.

2. Analisis kebutuhan perangkat lunak. Proses pengumpulan kebutuhan diintensifkan dan difokuskan, khususnya pada perangkat lunak. Untuk memahami sifat program yang yang dibangun, perekayasa perangkat lunak (analis) harus memahami domain informasi, tingkah laku, unjuk kerja, dan antar muka yang diperlukan. Kebutuhan baik untuk sistem maupun perangkat lunak didokumentasikan dan dilihat lagi dengan pelanggan.
3. Desain. Desain perangkat lunak sebenarnya adalah proses multi langkah yang berfokus pada empat atribut sebuah program yang berbeda; struktur data arsitektur perangkat lunak, *refresentasi interface* dan detail (algoritma) prosedural.
4. Generasi kode. Desain harus diterjemahkan ke dalam bentuk mesin yang bisa dibaca. Langkah pembuatan kode melakukan tugas ini. Jika desain dilakukan dengan cara yang lengkap, pembuatan kode dapat diselesaikan secara mekanis.
5. Pengujian. Sekali kode dibuat, pengujian program dimulai. Program dimulai. Proses pengujian berfokus pada logika internal perangkat lunak, memastikan bahwa semua pernyataan sudah diuji, dan pada ekternal fungsional – yaitu mengarahkan pengujian untuk menemukan kesalahan-kesalahan dan memastikan bahwa input yang dibatasi akan memberikan hasil yang aktual yang sesuai dengan hasil yang dibutuhkan.
6. Pemeliharaan. Perangkat lunak akan mengalami perubahan setelah disampaikan kepada pelanggan (perkecualian yang mungkin adalah perangkat lunak yang dilekatkan). Perubahan akan terjadi karena kesalahan-kesalahan ditentukan, karena perangkat lunak harus disesuaikan untuk mengakomodasi perubahan-perubahan dilingkungan eksternalnya

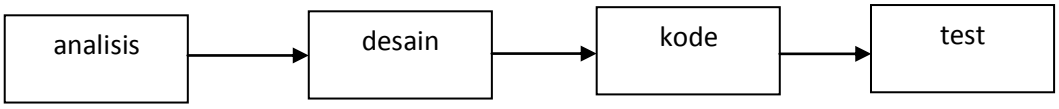
(contohnya perubahan yang dibutuhkan sebagai akibat dari perangkat *pheriperal* atau sistem operasi yang baru), atau karena pelanggan membutuhkan perkembangan fungsional atau unjuk kerja. Pemeliharaan perangkat lunak mengaplikasikan lagi setiap *fase* program sebelumnya dan tidak membuat yang baru lagi

Model *sekuensial linier* adalah paradigma rekayasa perangkat luas yang paling luas dipakai dan paling tua. Tetapi kritik dari paradigma tersebut telah menyebabkan dukungan aktif untuk mempertanyakan kehandalannya[HAN95]. Masalah-masalah yang kadang-kadang terjadi ketika model ini diaplikasikan adalah :

1. Jarang sekali proyek nyata mengikuti aliran sekensial yang dianjurkan oleh model. Meskipun model linier bisa mengakomodasi iterasi, model itu melakukannya dengan cara tidak langsung. Sebagai hasilnya, perubahan-perubahan dapat menyebabkan keraguan pada saat tim proyek berjalan.
2. Kadang-kadang sulit bagi pelanggan untuk menyatakan semua kebutuhannya secara eksplisit. Model *linier sekuensial* memerlukan hal ini dan mengalami kesulitan untuk mengakomodasi ketidakpastian natural yang ada pada bagian awal proyek.
3. Pelanggan harus bersikap sabar. Sebuah versi kerja dari program-program itu tidak akan diperoleh sampai akhir proyek dilalui. Sebuah kesalahan besar, jika tidak terdeteksi samapi program yang bekerja tersebut dikaji ulang, bisa menjadi petaka.
4. Pengembang sering melakukan penundaan yang tidak perlu. Di dalam anlisis yang menarik tentang proyek aktual, Bradac [BRA94] mendapatkan bahwa sifat alami dari siklus kehidupan klasik membawa *kepada blocking state* dimana banyak anggota tim proyek harus menunggu tim yang lain untuk melengkapi tugas yang saling memiliki ketergantungan. Kenyataannya, waktu yang dipakai untuk menunggu bisa mengurangi waktu untuk usaha produktif.

Masing-masing dari masalah tersebut bersifat riil. Tetapi paradigma siklus kehidupan klasik memiliki tempat yang terbatas namun penting di dalam rekayasa perangkat lunak. Paradigma itu memberikan template dimana metode analisis,

desain, pengkodean, pengujian, dan pemeliharaan bisa dilakukan. Siklus kehidupan klasik tetap menjadi model bagi rekayasa perangkat lunak yang paling luas dipakai. Sekalipun memiliki kelemahan, secara signifikan dia lebih baik daripada pendekatan yang sifatnya sembarang kepada pengembang perangkat lunak.



Gambar 2.3 Model *Waterfall*

(Sumber gambar : Pressman, Roger. *Rekayasa Perangkat Lunak*, Buku Satu. Andi. Yogyakarta, 2002.)

2.8 *Unified Modeling Language* (*UML*)[9]

Unified Modelling Language (*UML*) adalah sebuah "bahasa" yg telah menjadi standar dalam industri untuk visualisasi, merancang dan mendokumentasikan sistem perangkat lunak. *UML* menawarkan sebuah standar untuk merancang model sebuah sistem. Dengan menggunakan *UML* kita dapat membuat model untuk semua jenis *aplikasi* perangkat lunak, dimana aplikasi tersebut dapat berjalan pada perangkat keras, sistem operasi dan jaringan apapun, serta ditulis dalam bahasa pemrograman apapun.

UML mendefinisikan notasi dan *syntax*/semantik. Notasi *UML* merupakan sekumpulan bentuk khusus untuk menggambarkan berbagai diagram perangkat lunak. Setiap bentuk memiliki makna tertentu, dan *UML syntax* 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*).

Dimulai pada bulan Oktober 1994 Booch, Rumbaugh dan Jacobson, yang merupakan tiga tokoh yang boleh dikata metodologinya banyak digunakan mempelopori usaha untuk penyatuan metodologi pendesainan berorientasi objek. Pada tahun 1995 direlease draft pertama dari UML (versi 0.8).Sejak tahun 1996

pengembangan tersebut dikoordinasikan oleh **Object Management Group** (OMG – http://www.omg.org).

Tahun 1997 UML versi 1.1 muncul, dan saat ini versi terbaru adalah versi 1.5 yang dirilis bulan Maret 2003.Booch, Rumbaugh dan Jacobson menyusun tiga buku serial tentang UML pada tahun 1999. Sejak saat itulah UML telah menjelma menjadi standar bahasa pemodelan untuk *aplikasi* berorientasi objek.

2.8.1 Konsepsi Dasar *UML*

Dari berbagai penjelasan rumit yang terdapat di dokumen dan buku-buku UML. Sebenarnya konsepsi dasar *UML* bisa kita rangkumkan dalam gambar dibawah :

Major Area	***Views***	***Diagrams***	***Main Concepts***
Structural	*Use Case View*	*Use Case Diagram*	*use case, actor, association, extend, Include, use case generalization*
Dynamic	*Interaction View*	*Sequence Diagram*	*interaction, object, message, activation*
Model Management	*Management View*	*Class Diagram*	*package, subsystem, model*

Tabel 2.5 Konsepsi *UML*

(Sumber: www.ilmukomputer.com, yanti_uml.pdf)

Seperti juga tercantum pada gambar diatas *UML* mendefinisikan diagram-diagram sebagai berikut:

- *use case diagram*
- *sequence diagram*
- *class diagram*

2.8.2 *Use Case Diagram*

Use case diagram menggambarkan fungsionalitas yang diharapkan dari sebuah sistem. Yang ditekankan adalah “apa” yang diperbuat sistem, dan bukan

"bagaimana".Sebuah *use case* merepresentasikan sebuah interaksi antara aktor dengan sistem. *Use case* merupakan sebuah pekerjaan tertentu, misalnya login ke sistem, meng-*create* sebuah daftar belanja, dan sebagainya. Seorang/sebuah aktor adalah sebuah entitas manusia atau mesin yang berinteraksi dengan sistem untuk melakukan pekerjaan-pekerjaan tertentu.

Use case diagram dapat sangat membantu bila kita sedang menyusun *requirement* sebuah sistem, mengkomunikasikan rancangan dengan klien, dan merancang kasus uji untuk semua bagianyang ada pada sistem. 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 *usecase* yang meng-*include* dieksekusi secara normal. Sebuah *use case* dapat di-*include* oleh lebih dari satu *use case* lain, sehingga duplikasi fungsionalitas dapat dihindari dengan cara menarik keluar fungsionalitas yang sama. Sebuah *use case* juga dapat meng-*extend use case* lain dengan *behaviour*-nya sendiri. Sementara hubungan generalisasi antar *use case* menunjukkan bahwa *use case* yang satu merupakan spesialisasi dari yang lain.

2.8.3 *Class Diagram*

Class adalah sebuah spesifikasi yang jika diinstansiasi akan menghasilkan sebuah objek dan merupakan inti dari pengembangan dan desain berorientasi objek. *Class* menggambarkan keadaan (atribut/ properti) suatu sistem, sekaligus menawarkan layanan untuk memanipulasi keadaan tersebut (metoda/fungsi).

Class diagram menggambarkan struktur dan deskripsi *class, package* dan objek beserta hubungan satu sama lain seperti *containment*, pewarisan, asosiasi, dan lain-lain.

Class memiliki tiga area pokok :

1. Nama
2. Atribut
   - Nilai atau elemen-elemen data yang dimiliki oleh objek dalam kelas objek
   - Merupakan ciri dari sebuah objek

- Dipunyai secara individual oleh sebuah objek

3. Metoda

Atribut dan metoda dapat memiliki salah satu sifat berikut :

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

Hubungan Antar *Class*

1. Asosiasi, yaitu hubungan statis antar *class*. Umumnya menggambarkan *class* yang memiliki atribut berupa *class* lain, atau *class* yang harus mengetahui eksistensi *class* lain. Panah *navigability* menunjukkan arah *query* antar *class*.
2. Agregasi, yaitu hubungan yang menyatakan bagian ("terdiri atas..").
3. Pewarisan, yaitu hubungan hirarkis antar *class*. *Class* dapat diturunkan dari *class* lain dan mewarisi semua atribut dan metoda *class* asalnya dan menambahkan fungsionalitas baru, sehingga ia disebut anak dari *class* yang diwarisinya. Kebalikan dari pewarisan adalah generalisasi.
4. Hubungan dinamis, yaitu rangkaian pesan (*message*) yang di-*passing* dari satu *class* kepada *class* lain. Hubungan dinamis dapat digambarkan dengan menggunakan *sequence diagram* yang akan dijelaskan kemudian.

2.8.4 *Sequence Diagram*

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 atas 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 terjadi 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.

2.8.5 Pemodelan Penggunaan Sistem

Pemodelan pada penggunaan sistem terdapat beberapa tahap, diantaranya : Pemodelan *Use-Case*

- Identifikasi actor
- Identifikasi *use-case*
- Pembuatan diagram *use-case*
- Pembuatan diagram sekuen atau diagram kolaborasi untuk memperjelas masing-masing *use-case*
- Pembuatan *diagram* aktivitas

2.8.6 Pemodelan Perilaku Kelas Objek

Perilaku kelas pada objek terdapat beberapa poin, yaitu :

1. Evaluasi semua *use-case* agar dapat memahami sepenuhnya sekuen interaksi di dalam sistem.
2. Identifikasi kejadian-kejadian yang menuntun sekuen interaksi dan pahami bagaimana kejadian-kejadian ini berhubungan dengan objek-objek tertentu.
3. Pembuatan *diagram sekuen* untuk masing-masing *use-case*.
4. Pembuatan *diagram kolaborasi* untuk masing-masing kelas.
5. Pembuatan *diagram aktivitas* untuk memperjelas masing-masing kelas atau operasi.
6. Pembuatan *diagram statechart* untuk sistem.
7. Lakukan *review* model perilaku objek yang diperoleh untuk verifikasi akurasi dan konsistensi.

2.9 Perancangan Berorientasi Objek

Pada setiap disiplin rekayasa, perancangan merupakan pendekatan berdisiplin untuk menemukan solusi masalah. Perancangan merupakan penghubung antara spesifikasi kebutuhan dan implementasi. Perancangan menekankan pada solusi logik mengenai cara sistem memenuhi kebutuhan.

Terdapat banyak metode perancangan berorientasi objek. Perbedaan-perbedaan pada metode perancangan berorientasi objek bukan pada langkah-langkah esensi, hanya rincian-rincian, sehingga dapat disimpulkan langkah-langkah perancangan berorientasi objek adalah sebagai berikut :

1. Perancangan sistem meliputi arsitektur sistem dan pendeskripsian subsistem-subsistem dan alokasinya di pemroses dan proses.
2. Pemilihan strategi perancangan untuk implementasi manajemen data, dukungan antarmuka dan manajemen proses/memori, penanganan kesalahan.
3. Perancangan mekanisme kendali yang cocok untuk sistem.
4. Perancangan rinci kelas objek dalam hal struktur data dan algoritmanya.
5. Perancangan pertukaran pesan menggunakan kolaborasi antar objek dan hubungan objek.
6. Penciptaan model pertukaran pesan.
7. Melakukan *review* atas model rancangan dan melakukan iterasi bila perlu untuk perbaikan model rancangan yang sebelumnya.

2.10 Implementasi

2.10.1 *PHP*

PHP adalah kependekan dari *Hypertext Preprocessor* atau Profesional Homepage, yaitu sebuah bahasa scripting yang dieksekusi di sisi *server* (*Server-side Scripting Language*). Fungsinya adalah membuat sebuah web yang interaktif dan dinamis. *PHP* dibuat pertama kali pada tahun 1995 oleh Rasmus Lerdorf, seorang *software engineer* anggota tim pengembangan *web server Apache*. Pada tahun 1996 Rasmos menulis kode *script Perl* untuk diterapkan pada hal yang lebih komplek dengan bahasa pemrograman C.

Pada bulan Oktober 2000 *PHP* merilis versi 4.0 dengan mengintegrasikan Zend Engine, dan merupakan versi pengembangan dari *PHP* 3 *Script Engine*. Keunggulannya dari sifatnya yang *server*-side tersebut antara lain :

1. *PHP* mudah dibuat dan cepat dijalankan.
2. *PHP* dapat berjalan dalam web *server* yang berbeda dan sistem opersai yang berbeda pula.
3. *PHP* diterbitkan secara gratis.
4. PHP dapat berjalan pada web *server* Microsoft Personal Web *Server*, Apache, IIS dan sebagainya.
5. Tidak diperlukan kompatibilitas browser atau harus menggunakan browser tertentu, karena *server*-lah yang akan mengerjakan skrip PHP. Hasil yang dikirimkan kembali ke browser apapun.
6. Dapat memanfaatkan sumber-sumber aplikasi yang dimiliki oleh *server*, misalnya koneksi ke *database*.
7. PHP termasuk bahasa yang *embedded* (bisa diletakan dalam ag HTML).

Dengan output berupa tag-tag HTML biasa, maka output tersebut pasti dapat dieksekusi oleh browser apapun. Adapun kelebihan dari PHP yaitu dapat "melakukan" semua aplikasi program CGI, seperti mengambil nilai form, menghasilkan halaman web yang dinamis, mengirim dan menerima cookie. PHP juga dapat berkomunikasi dengan layanan-layanan yang menggunakan protokol IMAP, SNMP, NNTP, POP3, HTTP, dan lain-lain.

Namun tampaknya kelebihan PHP yang paling signifikan adalah kemampuannya untuk melakukan koneksi dengan berbagai macam *database*. Saat ini, *database* yang didukung PHP adalah : Adabas D, InterBase, PostgreSQL, dBase, FrontBase, Solid, Empress, mSQL, Sybase, FilePro(*read-only*), Direct MS-SQL, Velocis, IBM DB2, MySQL, Unix dbm, Informix, Semua *database* yang mempunyai provider ODBC, Ingres, Oracle (OCI7 and OCI8).

2.10.2 MySQL

MySQL adalah sebuah *database server*, dapat juga berperan sebagai *client* sehingga sering disebut *database client*/ *server* yang *open source* dengan kemampuan dapat berjalan baik di OS (*operating System*) manapun, dengan

platform Windows maupun Linux. Selain itu *database* ini memiliki beberapa kelebihan dibanding *database* lain, di antaranya adalah :

1. MySQL adalah sebuah software *database* yang open source, artinya program ini bersifat *free* atau bebas digunakan oleh siapa saja tanpa harus membeli dan membayar lisensi kepada pembuatnya.
2. MySQL merupakan sebuah *database client*. Selain menjadi *server* yang melayani permintaan, MySQL juga dapat melayani *query* yang mengakses *database* pada *server*. Jadi MySQL dapat juga berperan sebagai *client*.
3. MySQL mampu menerima *query* yang bertumpuk dalam satu permintaan atau yang disebut *multi-Threading*.
4. MySQL merupakan sebuah *database* yang mampu menyimpan data berkapasitas sangat besar hingga berukuran Gigabyte sekalipun.
5. MySQL memiliki kecepatan dalam pembuatan tabel maupun peng-*update*-an tabel.

2.10.3 XAMPP

Xampp merupakan tool yang menyediakan paket perangkat lunak ke dalam satu buah paket. Dengan menginstall XAMPP maka tidak perlu lagi melakukan instalasi dan konfigurasi *web server Apache*, PHP dan MySQL secara manual. XAMPP akan menginstalasi dan mengkonfigurasikannya secara otomatis untuk anda atau auto konfigurasi.

Software XAMPP versi ini terdiri atas:

a. APACHE

Apache sudah berkembang sejak versi pertamanya. Sampai saat ditulisnya artikel ini versi terakhirnya yang ada yaitu *Apache ver 2.0.54*. *Apache* bersifat open source, artinya setiap orang boleh menggunakannya, mengambil dan bahkan mengubah kode programnya.

Tugas utama *apache* adalah menghasilkan halaman web yang benar kepada peminta, berdasarkan kode PHP yang dituliskan oleh pembuat halaman web. Jika diperlukan juga berdasarkan kode PHP yang dituliskan, maka dapat saja suatu *database* diakses terlebih dahulu (misalnya dalam MySQL) untuk mendukung halaman web yang dihasilkan.

b. PhpMyAdmin

Pengelolaan *database* dengan *MySql* harus dilakukan dengan mengetikkan baris-baris perintah yang sesuai (*command line*) untuk setiap maksud tertentu. Jika ingin membuat *database*, ketikkan baris perintah yang sesuai untuk membuat database. Jika ingin menghapus tabel, ketikkan baris perintah yang sesuai untuk menghapus tabel. Hal tersebut tentu cukup menyulitkan karena kita harus hafal dan mengetikkan perintahnya satu persatu.

Banyak sekali perangkat lunak yang dapat dimanfaatkan untuk mengelola data base dalam *MySQL*, salah satunya adalah *phpMyAdmin*. Dengan *phpMyAdmin* kita dapat membuat tabel, mengisi data dan lain-lain dengan mudah tanpa harus hafal perintahnya. Untuk mengaktifkan phpMyAdmin langkah-langkahnya adalah : yang pertama setelah XAMPP kita terinstall, kita harus mengaktifkan *web server Apache* dan MySQL dari control panel XAMPP. Yang kedua, jalankan *browser* (IE, Mozilla Firefox atau Opera) lalu mengetikkan alamat web berikut : http://localhost/phpmyadmin/ pada address bar lalu tekan Enter. Langkah ketiga apabila telah nampak interface (tampilan antar muka) phpMyAdmin, kita bisa memulainya dengan mengetikkan nama database, nama tabel dan seterusnya.