BAB II

LANDASAN TEORI

II.1 Bahasa Inggris

Bahasa inggris merupakan bahasa tertua yang ada di dunia dan berasal dari dataran britania yaitu sekitar abad ke 8, selain bahasa inggris merupakan bahasa tertua yang ada di dunia bahasa inggris mempunyai perkembangan kosa kata yang sangat pesat. Menurut tim riset gabungan peneliti Havard University dan Google mencatat penambahan kosa kata bahasa tertua di dunia itu mencapai 8.500 kata pertahun. Kini jumlah total telah mencapai 1.022.000 kata. itu merupakan jumlah kosakata yang jauh lebih banyak dibandingkan dengan negara kita tercinta ini. jika dibandingkan dengan bahasa Indonesia[3].

Menurut pusat bahasa kemendikbud yang baru baru ini telah selesai dan menerbitkan KBBI (Kamus Besar Bahasa Indonesia) edisi ke empat dengan total jumlah kosa kata yang terdapat di dalamnya cuma berkisar 91.000 kosa kata, kamus edisi ke empat ini merupakan pengembangan dari edisi sebelumnya yang hanya mencakup 82.000 kosa kata. memang perkembangan jumlah kosa kata ini lah yang menjadi patokan layak atau tiidaknya sebuah bahasa dianggap sebagai bahasa internasional

II.1.1 *Tenses*

Tenses merupakan suatu bentuk kata kerja dalam bahasa Inggris untuk menunjukkan waktu (sekarang, masa depan, atau masa lalu) terjadinya suatu perbuatan atau peristiwa. *Tenses* terdiri dari empat macam waktu yaitu *Present Tense* (Masa Kini), *Past Tense* (Masa Lalu), *Future Tense* (Masa yang Akan Datang), dan *Past Future Tense* (Masa Kini Untuk Dikerjakan Dimasa depan). Dari empat macam tenses yang telah dipaparkan sebelumnya, terbagi kembali menjadi empat jenis pekerjaan yang akan dilakukan yaitu *simple* (Sederhana),

Continuous (yang sedang berlangsung), *Perfect* (telah terjadi), dan *Perfect Continuous* (telah dan sedang terjadi) [4].

II.2 Pembelajaran Berbasis Komputer

Pembelajaran bebrbasis komputer merupakan pembelajaran yang melibatkan peran serta komputer. Pembelajaran merupakan proses alamiah mengkonstruksi sesuatu pada alam fikiran manusia, sehingga manusia mendapatkan pengetahuan yang berbeda dengan sebelumnya. Secara alamaiah pula seseorang akan mendapatkan sebuah pengalaman baru. Manusia pada umumnya telah melakukan kegiatan belajar secara terus menerus dengan melibatkan peran serta perangkat apapun.

Terminologi aplikasi komputer dalam pembelajaran terus berkembang, seirama dengan perkembangan teknologi komputer dan teknologi informasi dalam meningkatkan mutu pembelajaran. *E-learning* disebut juga dengan pembelajaran berbantuan komputer. Secara umum, *e-learning* terdapat dua katagori yaitu (1) belajar melalui komputer mandiri *(standalone*) dan (2) belajar melalui komputer dalam jaringan. Pengembangan lingkungan belajar berbasis Web dapat diciptakan dan diakseskan melalui jaringan internet atau intranet. Pengembangan lingkungan belajar dengan berbantuan Web, identik dengan kegiatan dosen memberikan informasi untuk kegiatan belajar mahasiswa. Pembelajaran berbasis Web mampu digunakan belajar untuk membentuk kompetensi mahasiswa melalui variasi tugas, menggunakan berbagai perangkat lunak, belajar berkolaborasi untuk suatu proyek dan tukar informasi[5].

II.3 Pengujian Sistem

II.3.1 Pengujian *White Box*

Pengujian white box adalah pengujian yang didasarkan pada pengecekan terhadap detail perancangan, menggunakan struktur kontrol dari desain program secara procedural untuk membagi pengujian ke dalam beberapa kasus pengujian. Secara sekilas dapat diambil kesimpulan white box testing merupakan petunjuk untuk mendapatkan program yang benar secara 100%.

Penggunaan metode pengujian white box dilakukan untuk :

1. Memberikan jaminan bahwa semua jalur independen suatu modul digunakan minimal satu kali
2. Menggunakan semua keputusan logis untuk semua kondisi true atau false
3. Mengeksekusi semua perulangan pada batasan nilai dan operasional pada setiap kondisi.
4. Menggunakan struktur data internal untuk menjamin validitas jalur keputusan.

Persyaratan dalam menjalankan strategi White Box Testing:

1. Mendefinisikan semua alur logika
2. Membangun kasus untuk digunakan dalam pengujian
3. Mengevaluasi semua hasil pengujian
4. Melakukan pengujian secara menyeluruh

II.3.2 Pengujian *Black Box*

Black-box testing adalah metode pengujian perangkat lunak yang tes fungsionalitas dari aplikasi yang bertentangan dengan struktur internal atau kerja (lihat pengujian white-box). Pengetahuan khusus dari kode aplikasi / struktur internal dan pengetahuan pemrograman pada umumnya tidak diperlukan. Uji kasus dibangun di sekitar spesifikasi dan persyaratan yakni aplikasi apa yang seharusnya dilakukan. Menggunakan deskripsi eksternal perangkat lunak, termasuk spesifikasi, persyaratan, dan desain untuk menurunkan uji kasus.

Pengujian *black box* dapat menjadi fungsional atau non-fungsional, meskipun biasanya fungsional. Pengujian *black box* memilih input yang valid dan tidak valid dan menentukan output yang benar. Metode pengujian *black box* memfokuskan pada keperluan fungsional dari software. Karena itu ujicoba *black box* memungkinkan pengembang perangkat lunak untuk membuat himpunan kondisi input yang akan melatih seluruh syarat-syarat fungsional suatu program. Ujicoba blackbox bukan merupakan alternatif dari ujicoba *white box*, tetapi merupakan pendekatan yang melengkapi untuk menemukan kesalahan lainnya, selain menggunakan metode *white box*.

Pengujian *black box* berusaha untuk menemukan kesalahan dalam beberapa kategori, diantaranya :

1. Fungsi-fungsi yang salah atau hilang

2. Kesalahan interface

3. Kesalahan dalam struktur data atau akses database eksternal

4. Kesalahan performa

5. Kesalahan inisialisasi dan terminasi

II.3.3 Skala Likert

Skala likert adalah suatu skala psikometrik yang digunakan dalam kuesioner dan merupakan salah satu teknik yang dapat digunakan dalam evaluasi suatu program atau kebijakan perencanaan. Rensis Likert telah mengembangkan sebuah skala untuk mengukur sikap masyarakat di tahun 1932 yang sekarang terkenal dengan nama skala Likert. Skala Likert ini merupakan skala yang dapat dipergunakan untuk mengukur sikap, pendapat, dan persepsi seseorang atau sekelompok orang mengenai suatu gejala atau fenomena. Pada evaluasi, skala likert digunakan untuk

a. Menilai keberhasilan suatu kebijakan atau program
b. Menilai manfaat pelaksanaan suatu kebijakan atau program

c. Mengetahui kepuasan stakeholder terhadap pelaksanaan suatu kebijakan atau program.

Skala Likert terdapat dua bentuk pernyataan yaitu pernyataan positif yang berfungsi untuk mengukur sikap positif, dan pernyataan negative yang berfungsi untuk mengukur sikap negative objek. Skor pernyataan positif dimulai dari

1. Untuk sangat tidak setuju (STS)
2. Untuk tidak setuju (TS)
3. Untuk ragu-ragu (R)
4. Untuk setuju (S)
5. Untuk sangat setuju (SS)

Skor pernyataan negative dimulai dari

1. Untuk sangat setuju (SS)
2. Untuk setuju (S)
3. Untuk ragu-ragu (R)
4. Untuk tidak setuju (TS)
5. Untuk sangat tidak setuju (STS).

Beberapa menghilangkan option “Ragu-ragu” dalam instrument untuk memudahkan dalam melihat angket yang responden isikan. Skala Likert digunakan untuk mengukur kesetujuan dan ketidaksetujuan seseorang terhadap sesuatu rencana program, pelaksanaan program ataupun tingkat keberhasilan suatu program.

II.4 Algoritma Genetika

Algoritma genetika adalah algoritma komputasi yang diinspirasi teori evolusi yang kemudian diadopsi menjadi algoritma komputasi untuk mencari solusi suatu permasalahan dengan cara yang lebih “alamiah”. Salah satu aplikasi algoritma genetika adalah pada permasalahan optimasi kombinasi, yaitu mendapatkan suatu nilai solusi optimal terhadap suatu permasalahan yang mempunyai banyak kemungkinan solusi. Dalam tulisan ini akan dibahas teori

dasar algoritma genetika beserta contoh aplikasinya dalam menyelesaikan suatu permasalahan optimasi kombinasi sederhana.

Algoritma genetika yang dikembangkan oleh Goldberg adalah algoritma komputasi yang diinspirasi teori evolusi Darwin yang menyatakan bahwa kelangsungan hidup suatu makhluk dipengaruhi aturan "yang kuat adalah yang menang". Darwin juga menyatakan bahwa kelangsungan hidup suatu makhluk dapat dipertahankan melalui proses reproduksi, crossover, dan mutasi. Konsep dalam teori evolusi Darwin tersebut kemudian diadopsi menjadi algoritma komputasi untuk mencari solusi suatu permasalahan dengan cara yang lebih "alamiah".

Sebuah solusi yang dibangkitkan dalam algoritma genetika disebut sebagai chromosome, sedangkan kumpulan chromosome-chromosome tersebut disebut sebagai populasi. Sebuah chromosome dibentuk dari komponen-komponen penyusun yang disebut sebagai gen dan nilainya dapat berupa bilangan numerik, biner, simbol ataupun karakter tergantung dari permasalahan yang ingin diselesaikan.

Chromosome-chromosome tersebut akan berevolusi secara berkelanjutan yang disebut dengan generasi. Dalam tiap generasi chromosome-chromosome tersebut dievaluasi tingkat keberhasilan nilai solusinya terhadap masalah yang ingin diselesaikan (fungsi_objektif) menggunakan ukuran yang disebut dengan fitness. Untuk memilih chromosome yang tetap dipertahankan untuk generasi selanjutnya dilakukan proses yang disebut dengan seleksi. Proses seleksi chromosome menggunakan konsep aturan evolusi Darwin yang telah disebutkan sebelumnya yaitu chromosome yang mempunyai nilai fitness tinggi akan memiliki peluang lebih besar untuk terpilih lagi pada generasi selanjutnya.

Chromosome-chromosome baru yang disebut dengan offspring, dibentuk dengan cara melakukan perkawinan antar chromosome-chromosome dalam satu generasi yang disebut sebagai proses crossover. Jumlah chromosome dalam populasi yang mengalami crossover ditetukan oleh paramater yang disebut dengan

crossover_rate. Mekanisme perubahan susunan unsur penyusun mahkluk hidup akibat adanya faktor alam yang disebut dengan mutasi direpresentasikan sebagai proses berubahnya satu atau lebih nilai gen dalam chromosome dengan suatu nilai acak. Jumlah gen dalam populasi yang mengalami mutasi ditentukan oleh parameter yang dinamakan mutation_rate. Setelah beberapa generasi akan dihasilkan chromosome-chromosome yang nilai gen-gennya konvergen ke suatu nilai tertentu yang merupakan solusi terbaik yang dihasilkan oleh algoritma genetika terhadap permasalahan yang ingin diselesaikan[6].

II.5 Object Oriented Programing

OOP (*Object Oriented Programming*) atau yang dikenal dengan Pemrograman Berorientasi Objek merupakan paradigma pemrograman yang berorientasikan kepada objek. Semua data dan fungsi di dalam paradigma ini dibungkus ke dalam kelas-kelas atau objek-objek. Model data berorientasi objek dikatakan dapat memberi fleksibilitas yang lebih, kemudahan mengubah program, dan digunakan luas dalam teknik piranti lunak skala besar. Lebih jauh lagi, pendukung OOP mengklaim bahwa OOP lebih mudah dipelajari bagi pemula dibanding dengan pendekatan sebelumnya, dan pendekatan OOP lebih mudah dikembangkan dan dirawat.

Dengan menggunakan OOP maka dalam melakukan pemecahan suatu masalah kita tidak melihat bagaimana cara menyelesaikan suatu masalah tersebut (terstruktur) tetapi objek-objek apa yang dapat melakukan pemecahan masalah tersebut. Sebagai contoh anggap kita memiliki sebuah departemen yang memiliki manager, sekretaris, petugas administrasi data dan lainnya. Misal manager tersebut ingin memperoleh data dari bag administrasi maka manager tersebut tidak harus mengambilnya langsung tetapi dapat menyuruh petugas bagian administrasi untuk mengambilnya. Pada kasus tersebut seorang manager tidak harus mengetahui bagaimana cara mengambil data tersebut tetapi manager bisa mendapatkan data tersebut melalui objek petugas adminiistrasi. Jadi untuk menyelesaikan suatu masalah dengan kolaborasi antar objek-objek yang ada

karena setiap objek memiliki deskripsi tugasnya sendiri. Pemrograman orientasi-objek menekankan konsep-konsep yaitu[7]:

1. *Class*

Kumpulan atas definisi data dan fungsi-fungsi dalam suatu unit untuk suatu tujuan tertentu. Sebagai contoh 'class of dog' adalah suatu unit yang terdiri atas definisi-definisi data dan fungsi-fungsi yang menunjuk pada berbagai macam perilaku/turunan dari anjing. Sebuah class adalah dasar dari modularitas dan struktur dalam pemrograman berorientasi objek. Sebuah class secara tipikal sebaiknya dapat dikenali oleh seorang non-programmer sekalipun terkait dengan domain permasalahan yang ada, dan kode yang terdapat dalam sebuah class sebaiknya (relatif) bersifat mandiri dan independen (sebagaimana kode tersebut digunakan jika tidak menggunakan OOP). Dengan modularitas, struktur dari sebuah program akan terkait dengan aspek-aspek dalam masalah yang akan diselesaikan melalui program tersebut. Cara seperti ini akan menyederhanakan pemetaan dari masalah ke sebuah program ataupun sebaliknya.

2. *Object*

Membungkus data dan fungsi bersama menjadi suatu unit dalam sebuah program komputer. Objek merupakan dasar dari modularitas dan struktur dalam sebuah program komputer berorientasi objek.

3. *Abstract*

Kemampuan sebuah program untuk melewati aspek informasi yang diproses olehnya, yaitu kemampuan untuk memfokus pada inti. Setiap objek dalam sistem melayani sebagai model dari "pelaku" abstrak yang dapat 23 melakukan kerja, laporan dan perubahan keadaannya, dan berkomunikasi dengan objek lainnya dalam sistem, tanpa mengungkapkan bagaimana kelebihan ini diterapkan. Proses, fungsi atau metode dapat juga dibuat abstrak, dan beberapa teknik digunakan untuk mengembangkan sebuah pengabstrakan.

4. *Encapsulation*

Memastikan pengguna sebuah objek tidak dapat mengganti keadaan dalam dari sebuah objek dengan cara yang tidak layak; hanya metode dalam objek tersebut yang diberi ijin untuk mengakses keadaannya. Setiap objek mengakses

interface yang menyebutkan bagaimana objek lainnya dapat berinteraksi dengannya. Objek lainnya tidak akan mengetahui dan tergantung kepada representasi dalam objek tersebut.

5. *Polymorphism*

Melalui pengiriman pesan. Tidak bergantung kepada pemanggilan subrutin, bahasa orientasi objek dapat mengirim pesan. Metode tertentu yang berhubungan dengan sebuah pengiriman pesan tergantung kepada objek tertentu di mana pesa tersebut dikirim.

6. *Inheritance*

Mengatur polimorfisme dan enkapsulasi dengan mengijinkan objek didefinisikan dan diciptakan dengan jenis khusus dari objek yang sudah ada objek-objek ini dapat membagi (dan memperluas) perilaku mereka tanpa harus mengimplementasi ulang perilaku tersebut (bahasa berbasis-objek tidak selalu memiliki inheritas).