BAB II
LANDASAN TEORI

II.1. Perancangan Aplikasi

II.1.1. Perancangan

Perancangan adalah tahap setelah analisis dari siklus pengembangan sistem yang dapat berupa penggambaran, perencanaan dan pembuatan sketsa atau pengaturan dari beberapa elemen yang terpisah ke dalam satu kesatuan yang utuh dan berungsi, termasuk menyangkut mengkonfigurasi dari komponen-komponen perangkat lunak dan perangkat keras dari suatu sistem (Jogiyanto 2005 : 196).

Perancangan adalah satu kegiatan yang memiliki tujuan untuk mendesain sistem baru yang dapat menyelesaikan masalah-masalah yang dihadapi perusahaan yang diperoleh dari pemilihan alternatif sistem yang terbaik (Ladjamudin 2005:39). Berdasarkan definisi tersebut, penulis menyimpulkan bahwa perancangan merupakan suatu solusi untuk menyelesaikan permasalahan yang selanjutnya dikembangkan dengan suatu sistem.

II.1.2. Aplikasi

Definisi aplikasi menurut Griffin dkk (2006 : 86) “aplikasi merupakan paket *software* yang ditulis oleh orang lain”. Definisi lain dari aplikasi menurut Kadir (2005:222) “perangkat lunak aplikasi (*aplication software*) adalah program yang biasa dipakai oleh pemakai untuk melakukan tugas - tugas yang spesifik ; misalnya untuk membuat dokumen, memanipulasi foto, atau membuat laporan keuangan”. Berdasarkan definisi di atas, penulis menyimpulkan bahwa aplikasi

merupakan *software* yang dibuat oleh orang lain atau *programmer* yang memiliki fungsi tertentu untuk melakukan tugas-tugas tertentu.

II.2. Algoritma

Algoritma adalah merupakan jantung ilmu komputer atau informatika. Banyak cabang ilmu komputer yang diacu dalam terminologi algoritma. Namun jangan beranggapan algoritma selalu identik dengan ilmu komputer saja. Cara membuat kue atau masakan yang dinyatakan dalam resep masakan, itu juga merupakan algoritma. Ibu-ibu yang mencoba resep masakan tersebut akan membaca satu persatu langkah satu persatu pembuatannya, lalu mengerjakan proses (melakukan aksi) sesuai yang ia baca. Secara umum, pihak yang mengerjakan proses disebut pemroses (*processor*). Pemrosesan dapat berupa manusia, komputer, robot, alat mekanik, alat elektronik dll. Melaksanakan algoritma berarti mengerjakan langkah-langkah yang tertulis dalam algortima tersebut.

Dalam matematika dan komputasi, algoritma merupakan kumpulan perintah untuk menyelesaikan suatu masalah. Perintah-perintah ini dapat diterjemahkan secara bertahap dari awal hingga akhir. Masalah tersebut dapat berupa apa saja, dengan catatan untuk setiap masalah, ada kriteria kondisi awal yang harus dipenuhi sebelum menjalankan algoritma. Algoritma akan dapat selalu berakhir untuk semua kondisi awal yang memenuhi kriteria, dalam hal ini berbeda dengan heuristik. Algoritma sering mempunyai langkah pengulangan (iterasi) atau memerlukan keputusan (logika Boolean dan perbandingan) sampai tugasnya

selesai. Desain dan analisis algoritma adalah suatu cabang khusus dalam ilmu komputer yang mempelajari karakteristik dan performa dari suatu algoritma dalam menyelesaikan masalah, terlepas dari implementasi algoritma tersebut. Dalam cabang disiplin ini algoritma dipelajari secara abstrak, terlepas dari sistem komputer atau bahasa pemrograman yang digunakan. (Jurnal Saintikom Vol 4 No 1: 2008)

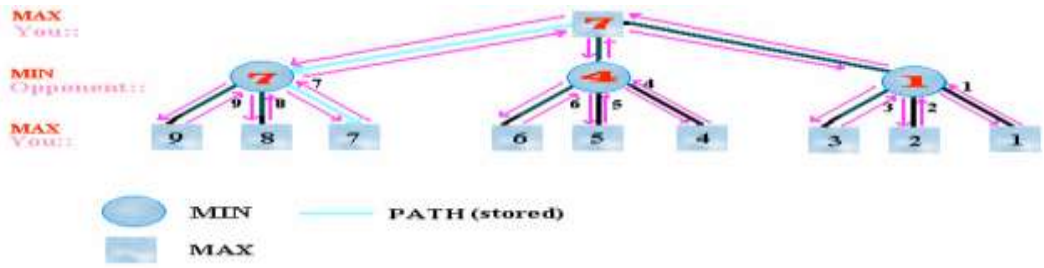
II.2.1. Algoritma Minimax

Tujuan sebenarnya algoritma Minimax digunakan pada permainan capsah ini yaitu untuk memperkirakan kartu apakah yang dapat dikeluarkan pada 1 langkah ke depan. [Kusumadewi, 2003] dan [Russel, 2003]. Komponen - komponen yang membentuk Minimax yaitu langkah pemain utama (max), langkah lawan (min), dan fungsi evaluasi. Langkah lawan dalam permainan Capsah Banting ini dilakukan oleh virtual hand, yaitu tumpukan kartu simulasi yang digunakan oleh komputer dan berisi semua kartu dikurangi kartu yang telah dibuang, kemudian dikurangi oleh kartu pemain tersebut. Sedangkan untuk dapat menghitung nilai evaluasi diperlukan beberapa rumus, yaitu rumus untuk penilaian kartu, penilaian kombinasi kartu, dan total nilai evaluasi.

Dalam ilmu komputer, sebuah algoritma pencarian dijelaskan secara luas adalah sebuah algoritma yang menerima masukan berupa sebuah masalah dan menghasilkan sebuah solusi untuk masalah tersebut, yang biasanya didapat dari evaluasi beberapa kemungkinan solusi. Sebagian besar algoritma yang dipelajari oleh ilmuwan komputer adalah algoritma pencarian. Himpunan semua kemungkinan solusi dari sebuah masalah disebut ruang pencarian. Algortima

pencarian *brute-force* atau pencarian naif/*uninformed* menggunakan metode yang sederhana dan sangat intuitif pada ruang pencarian, sedangkan algoritma pencarian informed menggunakan heuristik untuk menerapkan pengetahuan tentang struktur dari ruang pencarian untuk berusaha mengurangi banyaknya waktu yang dipakai dalam pencarian. Sebuah algoritma pencarian uninformed adalah algoritma yang tidak mempertimbangkan sifat alami dari permasalahan. Oleh karena itu algoritma tersebut dapat diimplementasikan secara umum, sehingga dengan implementasi yang sama dapat digunakan pada lingkup permasalahan yang luas, hal ini berkat abstraksi. Kekurangannya adalah sebagian besar ruang pencarian adalah sangat besar, dan sebuah pencarian uninformed (khususnya untuk pohon) membutuhkan banyak waktu walaupun hanya untuk contoh yang kecil. Sehingga untuk mempercepat proses, kadang-kadang hanya pencarian informed yang dapat melakukannya. (Jurnal Informatika Vol 4 No 1 : 2008 ; 4)

Dalam repersentasi pohon dalam algoritma Minimax, terdapat dua jenis node, yaitu node *min* dan node *max*. *Max* node akan memilih langkah dengan nilai tertinggi dan *min* node akan memilih langkah dengan nilai terendah. Berikut merupakan gambar pohon untuk algoritma Minimax.



Gambar II.1. Pohon Pencarian Algoritma Minimax
(Sumber : Makalah IF2251 ; 2008)

II.3. Algoritma Pencarian

Dalam ilmu komputer, sebuah algoritma pencarian dijelaskan secara luas adalah sebuah algoritma yang menerima masukan berupa sebuah masalah dan menghasilkan sebuah solusi untuk masalah tersebut, yang biasanya didapat dari evaluasi beberapa kemungkinan solusi. Sebagian besar algoritma yang dipelajari oleh ilmuwan komputer adalah algoritma pencarian. Himpunan semua kemungkinan solusi dari sebuah masalah disebut ruang pencarian. Algortima pencarian *brute-force* atau pencarian naif/uninformed menggunakan metode yang sederhana dan sangat intuitif pada ruang pencarian, sedangkan algoritma pencarian informed menggunakan heuristik untuk menerapkan pengetahuan tentang struktur dari ruang pencarian untuk berusaha mengurangi banyaknya waktu yang dipakai dalam pencarian.

Sebuah algoritma pencarian uninformed adalah algoritma yang tidak mempertimbangkan sifat alami dari permasalahan. Oleh karena itu algoritma tersebut dapat diimplementasikan secara umum, sehingga dengan implementasi yang sama dapat digunakan pada lingkup permasalahan yang luas, hal ini berkat abstraksi. Kekurangannya adalah sebagian besar ruang pencarian adalah sangat besar, dan sebuah pencarian uninformed (khususnya untuk pohon) membutuhkan banyak waktu walaupun hanya untuk contoh yang kecil. Sehingga untuk mempercepat proses, kadang-kadang hanya pencarian informed yang dapat melakukannya. (Jurnal Informatika Vol 4 No 1 : 2008 ; 4)

II.3..1. Pencarian List

Algoritma pencarian list mungkin adalah algoritma pencarian paling dasar. Tujuannya adalah mencari sebuah elemen dari sebuah himpunan dengan suatu kunci (kemungkinan memuat informasi yang terkait dengan kunci). Oleh karena hal ini adalah masalah yang lazim dalam ilmu komputer, kompleksitas komputasi algoritma-algoritma tersebuh telah dipelajri dengan baik. Algoritma paling sederhana adalah pencarian linear, yang secara sederhana melihat setiap elemen dari list secara berurutan. Waktu pengerjaan algoritma ini adalah O(n), dimana n adalah banyaknya elemen dalam list, dan dapat digunakan langsung pada list yang belum diproses. Algoritma pencarian list yang lebih canggih adalah pencarian biner; waktu pengerjaannya adalah O(log n). Waktu pengerjaannya jauh lebih baik daripada pencarian linear untuk list yang memiliki data banyak, tetapi sebelum dilakukan pencarian list terlebih dahulu harus terurut (lihat algoritma pengurutan) dan juga harus dapat diakses secara acak (pengaksesan acak). Pencarian interpolasi adalah lebih baik dari pencarian biner untuk list terurut yang sangat besar dan terdistribusi merata. Algoritma Grover adalah sebuah algoritma kuantum yang menawarkan percepatan kuadrat dibandingkan pencarian linear klasik untuk list tak terurut.

Tabel *hash* juga digunakan untuk pencarian *list*, hanya memerlukan waktu yang konstan untuk mencari pada kasus rata-rata, tetapi memiliki overhead ruang yang lebih dan pada kasus terburuk waktu pengerjaannya adalah O (n). Pencarian lain yang berdasarkan struktur data khusus, menggunakan pohon pencarian biner yang *self-balancing* (*self-balancing binary search tree*) dan membutuhkan waktu

pencarian O (log n); hal ini dapat dipandang sebagai pengembangan dari ide utama pencarian biner untuk memungkinkan penyisipan dan penghapusan yang cepat. Lihat array asosiatif untuk diskusi lanjut dari struktur data pencarian *list*. Sebagian besar algoritma pencarian, seperti pencarian linear, pencarian biner dan pohon pencarian biner yang *self-balancing*, dapat dikembangkan dengan sedikit tambahan costuntuk menemukan semua nilai yang kurang dari atau lebih dari sebuah kunci, operasi ini disebut pencarian jangkauan (*range search*). Pengecualin ada pada tabel *hash*, yang tidak dapat melakukan pencarian tersebut secara efisien.

II.3.2. Pencarian Pohon

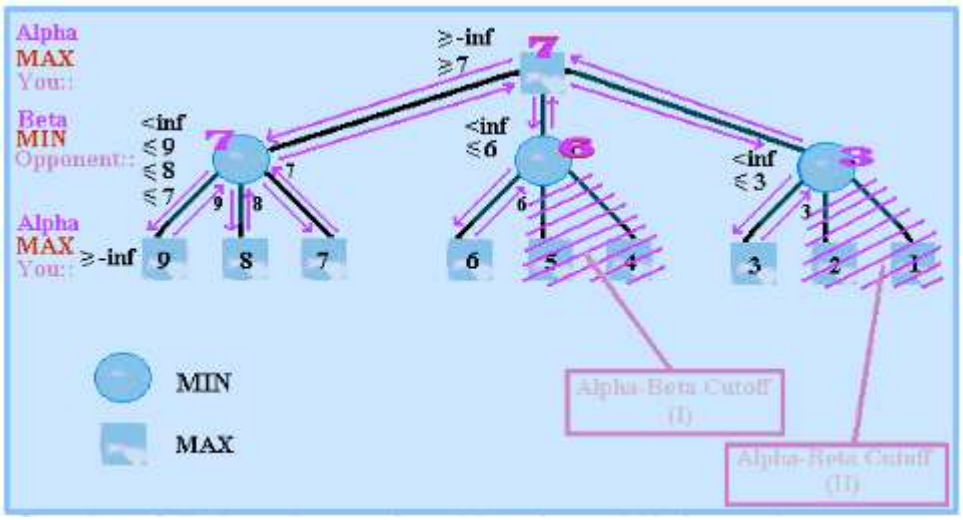
Algoritma pencarian pohon adalah jantung dari teknik-teknik pencarian. Algoritma tersebut mencari node dari pohon, terlepas apakah pohon tersebut eksplisit atau implisit (dibangkitkan saat pengerjaan). Prinsip dasarnya adalah sebuah node diambil dari sebuah struktur data, suksesornya diperiksa dan ditambahkan pada struktur data. Dengan memanipulasi struktur data, pohon dieksplorasi dalam urutan yang berbeda-beda, dieksplore dari satu tingkat ke tingkat berikutnya (pencarian *Breadth-first*) atau mengunjungi node pucuk terlebih dahulu kemudian lacak balik/backtracking (pencarian Depth-first). Contoh lain dari pencarian pohon antara lain pencarian *iterative-deepening*, pencarian berbatas kedalaman, pencarian dwiarah dan pencarian *uniform-cost*.

II.3.3. Depth First Search

Pada Depth First Search (DFS), proses akan dilakukan pada semua anaknya sebelum dilakukan pencarian ke node-node (titik) yang selevel. Pencarian dimulai dari node akar ke level yang lebih tinggi. Proses ini diulangi terus hingga ditemukannya solusi. Stack atau tumpukan adalah struktur data yang setiap proses baik penambahan maupun penghapusan hanya bisa dilakukan dari posisi teratas tumpukan. Cara kerja stack adalah LIFO *(Last In First Out)*, dimana data yang terakhir masuk akan keluar pertama.

II.4. Alpha-Beta Pruning

Dalam algoritma Minimax, pencarian dilakukan pada seluruh bagian pohon, sementara sebagian pohon tidak seharusnya diperiksa. *Alpha-beta pruning* merupakan modifikasi dari algoritma Minimax, yang akan mengurangi jumlah node yang dievaluasi oleh pohon pencarian. Pencarian untuk node berikutnya akan dipikirkan terlebih dahulu. Algoritma ini akan berhenti mengevaluasi langkah ketika terdapat paling tidak satu kemungkinan yang ditemukan dan membuktikan bahwa langkah tersebut lebih buruk jika dibandingkan dengan langkah yang diperiksa sebelumnya. Sehingga, langkah berikutnya tidak perlu dievaluasi lebih jauh. Dengan algoritma ini hasil optimasi dari suatu algoritma tidak akan berubah. Berikut merupakan pohon dengan algoritma *alpha-beta pruning*.



Gambar II.2. Pohon Pencarian Algoritma Minimax dengan Alpha-Beta Pruning

II.5. Jenis-Jenis *Game*

Game atau permainan terdiri dari beberapa bentuk yang di kategorikan berdasarkan fungsinya. Berikut ini beberap jenis game antara lain

1. *Edutainment Game*

Game jenis ini biasanya dibuat lebih spesifikasi untuk tujuan tertentu, misalnya untuk balita untuk sekedar mengenal warna dan objek. Ada juga yang ditujukan ke anak sekolah, sebagai contoh game tentang pelajaran biologi dimana di dalam *game* tersebut menyediakan konten misalnya tentang fungsi organ tubuh manusia.

2. *First Person Shooter*

Jenis *game* ini menampilkan sudut pandang orang pertama, biasanya yang nampak hanya tangan dan senjata *player* saja. Contoh game ini adalah CS (*Counter Strike*), *Saurbatten*, dll.

3. *Real Time Strategy*

Game ini lebih menekankan pada kehebatan strategi pemainnya, dan biasanya pemain tidak hanya memainkan satu karakter melainkan lebih dari satu karakter.

4. *Fighting*

Game ini menuntut pemainnya untuk lincah, cepat tanggap, respon yang baik. Sedikit berbeda dari *game fighting* lainnya yang hanya melawan AI atau komputer saja, melainkan *game* ini akan teruji jika pemain sudah bisa mengalahkan pemain lainnya atau dengan kata lain game ini merupakan *multi player*.

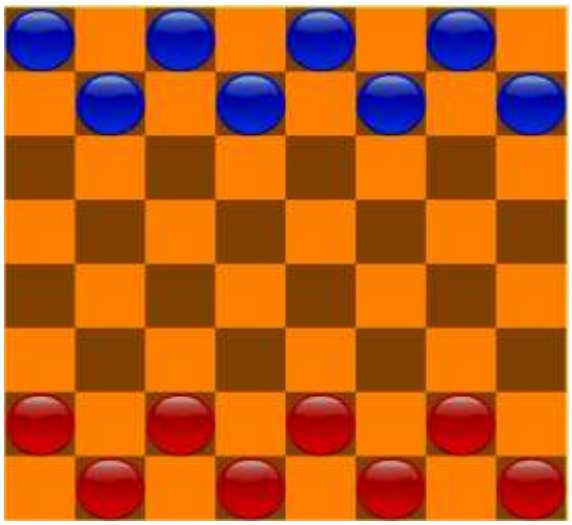
5. *Adventure*

Berbeda dengan *game* lain yang menuntut pemainnya untuk lincah, refleks, respon. Dalam *game* petualangan pemain dituntut kemampuan berfikir untuk menganalisa tempat secara visual, memecahkan teka-teki maupun menyimpulkan rangkaian peristiwa dan percakapan karakter, menggunkan benda-benda yang tepat dan diletakan ditempat yang tepat. (Jurnal Ilmiah Komputer dan Informatika : 2016 ; 2)

II.6. Permainan Checkers

Permainan checkers (dalam bahasa Inggris Amerika) atau disebut *draughts* (dalam bahasa Inggris British) merupakan permainan yang menggunakan strategi abstrak dimainkan oleh dua pemain dengan menggunakan langkah diagonal token dan menangkap dengan melompati token musuh. Permainan ini telah dimainkan

di Eropa sejak abad ke 16, dikembangkan dari permainan *alquerque*. Bentuk yang paling populer dari pemainan ini adalah international *draughts*, yang dimainkan pada papan 10x10. Bentuk yang juga populer adalah *English draughts*, yang disebut American checkers, dimainkan pada papan 8x8.



Gambar II.3. *International Checkers*
(Sumber : Wikipedia ; Akses 27 Nopember 2015)

III.6.1. Peraturan Checkers

Dimainkan oleh dua orang, dengan pemain berada pada sisi yang berlawanan dari papan. Salah satu pemain memiliki kepingan berwarna merah dan pemain lain berwarna biru. Pemain dengan kepingan berwarna merah melakukan langkah pertama, kecuali telah ditentukan sebelumnya. Kepingan akan bergerak diagonal dan kepingan lawan ditangkap dengan meloncatinya. Kepingan yang ditangkap akan dihilangkan dari papan. Gerak kepingan pada papan hanya dapat dilakukan pada kotak yang tidak ditempati. Permukaan yang dapat menjadi papan permainan hanya kotak dengan warna gelap. Pemain yang kalah adalah pemain

yang tidak memiliki kepingan yang tersisa atau tidak dapat melakukan langkah lagi.

Kepingan tanpa mahkota disebut orang, akan bergerak satu langkah maju diagonal dan menangkap kepingan dengan melakukan dua langkah pada arah yang sama, melompati kepingan lawan pada kotak tengah. Sejumlah kepingan lawan dapat ditangkap dengan satu loncatan, tidak harus pada arah yang sama tapi bisa zig zag. Pada English draughts kepingan hanya dapat ditangkap maju, tetapi pada *international draughts* kepingan dapat ditangkap mundur. Ketika mencapai baris terjauh, kepingan berubah menjadi raja, ditandai dengan memberikan mahkota. Kepingan raja ini memiliki kekuatan tambahan untuk berjalan dan menangkap mundur (pada jenis yang tidak dapat melakukaknnya). Pada international draughts, raja dapat begerak sejauh yang ia inginkan secara diagonal.

II.7. Multimedia

Multimedia adalah penggunaan komputer untuk menyajikan dan menggabungkan teks, suara, gambar, animasi dan video dengan alat bantu *(tool)* dan koneksi *(link)* sehingga pengguna dapat bernavigasi, berinteraksi, berkarya dan berkomunikasi. Multimedia sering digunakan dalam dunia hiburan. Selain dari dunia hiburan, Multimedia juga diadopsi oleh dunia Game.

Menurut Vaughan (2004), Multimedia merupakan kombinasi teks, seni, suara, gambar, animasi, dan video yang disampaikan dengan komputer atau dimanipulasi secara digital dan dapat disampaikan dan/atau dikontrol secara interaktif. (Jurnal Pemikiran Alternatif Kependidikan : 2009 ; 2)

II.8. *Microsoft Visual Studio* 2010

Visual Basic 2010 adalah inkarnasi dari bahasa Visual Basic yang sangat popoler dan telah dilengkapi dengan fitur serta fungsi yang setara dengan bahasa tingkat tinggi lainnya seperti C++. Anda dapat menggunakan Visual Basic 2010 untuk membuat aplikasi Wondows, mobile, Web, dan Office yang kompleks dengan menggunakan kode yang Anda tulis, atau kode yang telah ditulis oleh orang lain dan kemudian dimasukkan kedalam program Anda. (Christopher Lee : 1).

II.9. UML

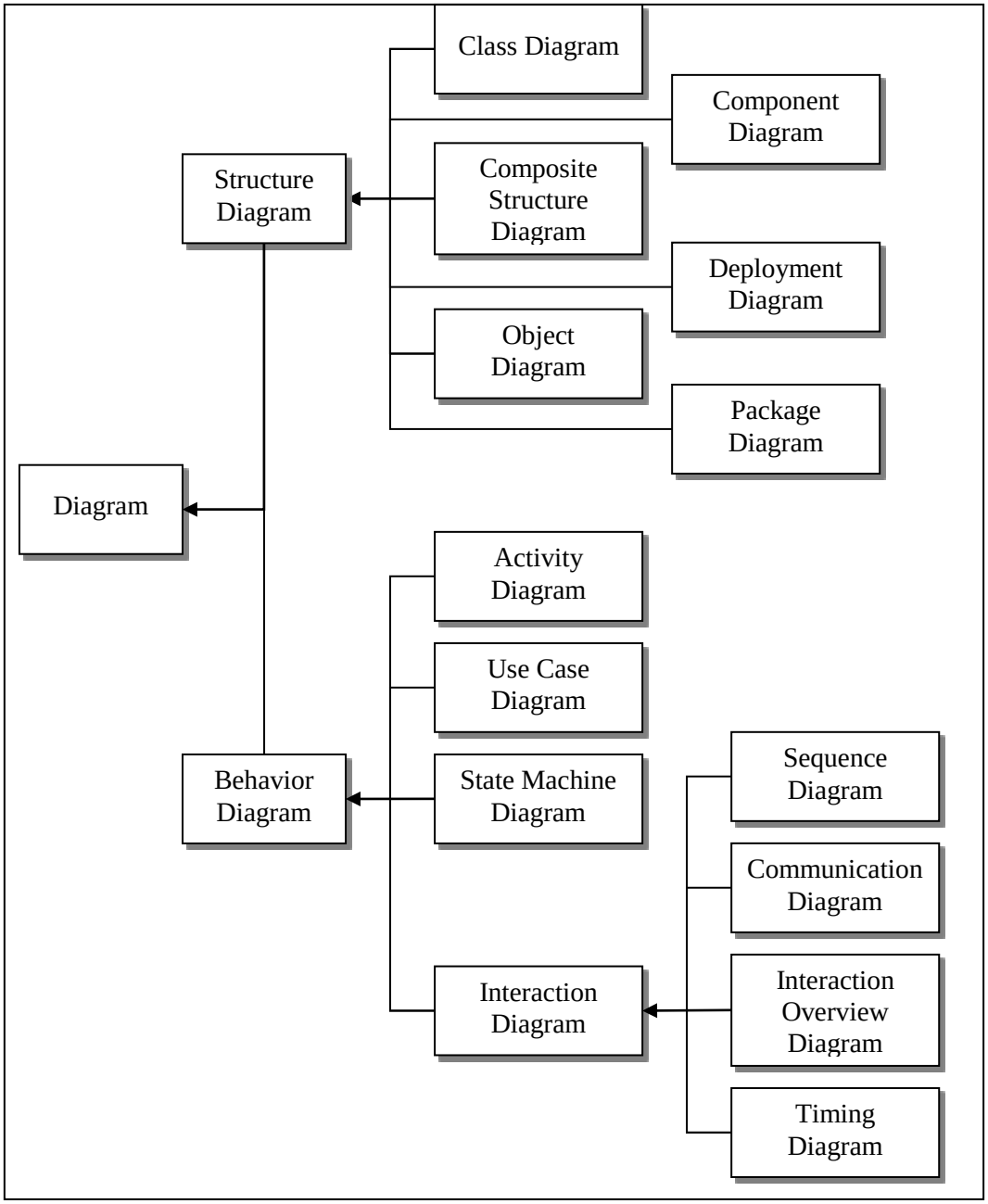
Pada perkembangan teknik pemrograman berorientasi objek, munculah sebuah standarisasi bahasa pemodelan untuk pembangunan perangkan lunak yang dibangun dengan menggunakan teknik pemrograman berorientasi objek, yaitu *Unified Modeling Languange* (UML). UML muncul karena adanya kebutuhan pemodelan visual untuk menspesifikasikan, menggambarkan, membangun, dan dokumentasi dari sistem perangkat lunak. UML merupakan bahasa *visual* untuk pemodelan dan komunikasi mengenai sebuah sistem dengan menggunakan diagram dan teks-teks pendukung. UML hanya berfungsi untuk melakukan pemodelan. Jadi penggunaan UML tidak terbatas pada metodologi tertentu, meskipun pada kenyataan UML paling banyak digunakan pada metodologi berorientasi objek.

Seperti yang kita ketahui bahwa banyak hal di dunia sistem informasi yang tidak dapat dibakukan, semua tergantung kebutuhan, lingkungan dan konteksnya.

Begitu juga dengan perkembangan penggunaan UML bergantung pada *level* abstraksi penggunaannya. Jadi belum tentu pandangan yang berbeda dalam penggunaan UML adalah suatu yang salah, tapi perlu ditelaah dimanakah UML digunakan dan hal apa yang ingin divisualkan. Secara *analogi* jika dengan bahasa yang kita gunakan sehari-hari, belum tentu penyampaian bahasa dengan puisi adalah hal yang salah. Sistem informasi bukanlah ilmu yang pasti, maka jika ada banyak perbedaan dan interprestasi di dalam bidang sistem informasi merupakan hal yang sangat wajar. (Munawar ; 2005 : 17-18)

II.9.1. Diagram-Diagram UML

Pada UML 2.3 terdiri dari 13 macam diagram yang dikelompokkan dalam 3 kategori. Pembagian kategori dan macam-macam diagram tersebut dapat dilihat pada gambar di bawah :



Gambar II.4. Klasifikasi Jenis Diagram UML
(Sumber : Munawar ; 2005 : 24)

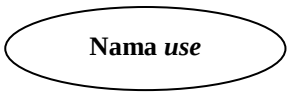
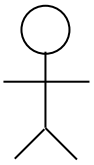
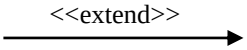
Berikut ini jenis-jenis diagram yang terdapat pada UML antara lain :

1. *Use Case* Diagram

Use case atau diagram *use case* merupakan pemodelan untuk kelakuan (*behavior*) sistem informasi yang akan dibuat. *Use case* mendeskripsikan

sebuah interaksi antara satu atau lebih aktor dengan sistem informasi yang akan dibuat. Secara kasar, *use case* digunakan untuk mengetahui fungsi apa saja yang ada di dalam sebuah sistem informasi dan siapa saja yang berhak menggunakan fungsi - fungsi itu.

Berikut adalah simbol-simbol yang ada pada diagram *use case* :

Simbol	Deskripsi
Use case 	Fungsionalitas yang disediakan sistem sebagai unit-unit yang saling bertukar pesan antar unit atau aktor; biasanya dinyatakan dengan menggunakan kata kerja di awal di awal frase nama *use case*
Aktor / actor 	Orang, proses, atau sistem lain yang berinteraksi dengan sistem informasi yang dibuat di luar sistem informasi yang akan dibuat sendiri, jadi walaupun simbol dari aktor adalah gambar orang, tapi aktor belum tentu merupakan orang; biasa dinyatakan menggunakan kata benda di awal frase nama aktor
Asosiasi / association	Komunikasi antara aktor dan *use case* yang berpartisipasi pada *use case* atau *use case* memiliki interaksi dengan aktor
Ekstensi / *extend* 	Relasi *use case* tambahan ke sebuah *use case* dimana *use case* yang ditambahkan dapat berdiri sendiri walau tanpa *use case* tambahan .

Gambar II.5. *Use Case Model*
(Sumber : Munawar ; 2005 : 63 - 64)

Use case nantinya akan menjadi kelas proses pada diagram kelas sehingga perlu dipertimbangkan penamaan yang dilakukan apakah sudah layak menjadi kelas atau belum sesuai dengan aturan pendefinisian kelas yang baik.

2. *Activity Diagram*

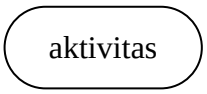
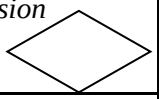
Diagram aktivitas atau *activity diagram* menggambarkan *workflow* (aliran kerja) atau aktivitas dari sebuah sistem atau sebuah proses bisnis

atau menu yang ada pada perangkai lunak. Yang perlu diperhatikan disini adalah bahwa diagram aktivitas menggambarkan aktivitas sistem bukan apa yang dilakukan aktor, jadi aktivitas yang dapat dilakukan oleh sistem.

Diagram aktivitas juga banyak digunakan untuk mendefinisikan hal - hal berikut :

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

Berikut adalah simbol-simbol yang ada pada diagram aktivitasnya :

Simbol	Deskripsi
Status awal 	Status awal aktivitas sistem, sebuah diagram aktivitas memiliki sebuah status awal
Aktivitas 	Aktivitas yang dilakukan sistem, aktivitas biasanya diawali dengan kata kerja
Percabangan / *decision* 	Asosiasi percabangan dimana jika ada pilihan aktivitas lebih dari satu
Penggabungan / *join* 	Asosiasi penggabungan dimana lebih dari satu aktivitas digabungkan menjadi satu
Status akhir 	Status akhir yang dilakukan sistem, sebuah diagram aktivitas memiliki sebuah status akhir

Gambar II.5. Simbol Activity Diagram
(Sumber : Munawar ; 2005 : 109)