BAB II
LANDASAN TEORI

II.1 Sampah

Sampah adalah suatu bahan yang terbuang atau dibuang dari sumber hasil aktivitas manusia maupun proses alam yang belum memiliki nilai ekonomis. Bentuk sampah bisa berada dalam setiap fase materi, yaitu padat, cair, dan gas [4].

Berangkat dari pandangan tersebut sehingga sampah dapat dirumuskan sebagai bahan sisa dari kehidupan sehari-hari masyarakat.

Sampah padat pada umumnya dapat di bagi menjadi dua bagian :

1. Sampah Organik

Sampah organik (biasa disebut sampah basah) dan sampah anorganik (sampah kering). Sampah Organik terdiri dari bahan-bahan penyusun tumbuhan dan hewan yang diambil dari alam atau dihasilkan dari kegiatan pertanian, perikanan atau yang lain. Sampah ini dengan mudah diuraikan dalam proses alami. Sampah rumah tangga sebagian besar merupakan bahan organik, misalnya sampah dari dapur, sisa tepung, sayuran dan sebagainya.

2. Sampah Anorganik

Sampah Anorganik berasal dari sumber daya alam tak terbarui seperti mineral dan minyak bumi, atau dari proses industri. Beberapa dari bahan ini tidak terdapat di alam seperti plastik dan aluminium. Sebagian zat anorganik secara keseluruhan tidak dapat diuraikan oleh alam, sedang sebagian lainnya hanya dapat diuraikan dalam waktu yang sangat lama. Sampah jenis ini pada tingkat rumah tangga, misalnya berupa botol, botol, tas plastik.

Kertas, koran, dan karton merupakan pengecualian. Berdasarkan asalnya, kertas, koran, dan karton termasuk sampah organik. Tetapi karena kertas, koran, dan karton dapat didaur ulang seperti sampah anorganik lain (misalnya gelas, kaleng, dan plastik), maka dimasukkan ke dalam kelompok sampah anorganik.

II.2 Pembelajaran Dengan Media Komputer

Pembelajaran dengan media komputer merupakan sarana yang baik digunakan dalam proses belajar mengajar karena dapat menjadikan pembelajaran menjadi lebih efektif dan efisien. Perkembangan teknologi khususnya di bidang teknologi informasi yang semakin pesat serta semakin ketatnya persaingan mengharuskan para pendidik harus lebih inovatif dalam pengoptimalan proses pembelajaran.

II.2.1 Jenis Pembelajaran Dengan Media Komputer

Ada lima tipe pembelajaran dengan media komputer, yaitu [5]:

1. *Drill & Practice*, menyajikan materi pelajaran untuk dipelajari secara berulang.
2. *Tutorial*, menyajikan materi yang telah diajarkan atau materi baru yan akan dipelajari.
3. *Simulation*, memberi kesempatan untuk menguji kemampuan pada aplikasi nyata dengan menciptakan situasi yang mengikutsertakan siswa untuk bertindak pada situasi tersebut.
4. *Problem Solving*, menyajikan masalah-masalah untuk diselesaikan siswa berdasarkan kemampuan yang mereka peroleh.
5. *Educational Game*, merupakan paket *software* yang menciptakan kemampuan pada lingkungan permainan yang diberikan sebagai alat bantu untuk memotivasi atau membantu siswa untuk melalui prosedur permainan secara teliti untuk mengembangkan kemampuannya. Berdasarkan kriteria-kriteria ini, maka aplikasi yang dirancang dapat ditentukan sebagai aplikasi *educational game*.

II.3 Game

Game merupakan kata dalam bahasa inggris yang berarti permainan. Permainan adalah sesuatu yang dapat dimainkan dengan aturan tertentu sehingga ada yang menang dan ada yang kalah, biasanya dalam konteks tidak ada serius atau dengan tujuan *refreshing.* Suatu cara belajar yang digunakan dalam

menganalisa interaksi antara sejumlah pemain maupun perorangan yang menunjukan strategi-strategi yang rasional.

Teori permainan pertama kali ditemukan oleh sekelompok ahli Matematika pada tahun 1944. Teori itu dikemukakan oleh John Von Neumann dan Oskar Morgenstern yang berisi:

"Permainan terdiri atas sekumpulan peraturan yang membangun situasi bersaing dari dua sampai beberapa orang atau kelompok dengan memilih strategi yang dibangun untuk memaksimalkan kemenangan sendiri atau pun untuk meminimalkan kemenangan lawan. Peraturan-peraturan menentukan kemungkinan tindakan untuk setiap pemain, sejumlah keterangan diterima setiap pemain sebagai kemajuan bermain, dan sejumlah kemenangan atau kekalahan dalam berbagai situasi" [6].

II.3.1 Pengertian Game

Pengertian *game* menurut para ahli :

1. Menurut Agustinus Nilwan dalam bukunya Pemograman Animasi dan *Game Profesional* terbitan Media Komputindo, *game* merupakan permainan komputer yang dibuat dengan teknik dan metode animasi. Jika ingin mendalami penggunaan animasi haruslah memahami pembuatan *game.* Atau jika ingin membuat *game,* maka haruslah memahami teknik dan metode animasi, sebab keduanya saling berkaitan [7].
2. Menurut Clark C. Abt, *Game* adalah kegiatan yang melibatkan keputusan pemain, berupaya mencapai tujuan dengan dibatasi oleh konteks tertentu (misalnya, dibatasi oleh peraturan).
3. Menurut Chris Crawford, seorang *computer game designer* mengemukakan bahwa *game*, pada intinya adalah sebuah interaktif. Aktivitas yang berpusat pada sebuah pencapaian, ada pelaku aktif (*player*), ada pelaku pasif (*NPC*).
4. Menurut David Parlett, *game* adalah sesuatu yang memiliki "akhir dan cara mencapainya" : artinya ada tujuan, hasil dan serangkaian peraturan untuk mencapai keduanya.

5. Menurut Roger Caillois, seorang sesiolog Perancis dalam bukunya yang berjudul *Les Jeux Et Les Hommes* menyatakan *game* adalah aktivitas yang mencakup karakteristik berikut: *fun* (bebas bermain adalah pilihan bukan kewajiban), *separate* (terpisah), *uncertain*, *non-productive*, *governed by rules* (ada aturan), *fictitious* (pura-pura).
6. Menurut Greg Costikyan, *game* adalah "sebentuk karya seni dimana peserta, yang disebut pemain, membuat keputusan untuk mengelola sumber daya yang dimilikinya melalui benda di dalam *game* demi mencapai tujuan".

II.3.2 Klasifikasi *Game*

Game biasanya digunakan untuk mengkatagorikan sebuah *game* berdasarkan dari interaksi *gameplay* daripada perbedaan tampilan atau narasi. Sebuah jenis *game* dapat didefinisikan dengan kumpulan dari sebuah tipe permainan dari *game* tersebut. *Game* dapat diklasifikasikan menjadi dua, yaitu:

II.3.2.1 Berdasarkan Jenis Platform yang Digunakan

Tiga kategori utama platform pada *game* adalah *personal computer (PC), console,* dan *mobile* [8] :

1. *Personal Computer* (PC)

   PC game adalah *game* yang dibuat untuk computer baik berbasis Windows, Mac, ataupun Linux. *PC* menyediakan kekuatan grafis dan pemrosesan yang kuat yang memungkinkan pengembang untuk membuat *game* yang mutakhir. Tetapi kelemahan *game PC* adalah mahal karena pengguna harus menggunakan hardware yang *up-to-date* untuk memainkan *game PC* dengan baik. Selain itu, kelemahan *game PC* bagi pengembang adalah banyaknya varian dari konfigurasi *PC* membuat sulit bagi pengembang untuk memastikan *game* tersebut berjalan dengan benar pada semua *setup PC*.

2. *Console*

   Console adalah *hardware* yang dibuat oleh pihak ketiga seperti Sony, Microsoft, dan Nintendo. *Console* terhubung ke televisi dan tujuan utamanya adalah untuk bermain *game*. *Game Console* sangat menarik bagi *game* pengembang karena mereka hanya perlu memikirkan satu konfigurasi *hardware* ketika membuat *software* untuk konsol. Sangat kontras dengan *PC* yang memiliki opsi konfigurasi yang tak terbatas.

3. *Mobile*

   Mobile platform terdiri atas sesuatu yang *portable* dan bisa digenggam, termasuk ponsel, PDA, iPods, dan *handheld game* seperti Nintendo DSi atau Sony PSP. *Game mobile* memiliki kontrol yang sederhana (terutama jika dibandingkan dengan *PC*).

II.3.2.2 Berdasarkan Gendre Permainan

Genre *game* bisa dibagi menjadi 9, yaitu:

1. *Action*

   Action game adalah *game* dimana kebanyakan dari tantangan yang disajikan merupakan dari tes *physical skill* dan koordinasi pemain. Salah satu sub-genre *action game* adalah *shooters game*, baik yang 2D maupun 3D seperti *First Person, Shooters* (FPS).

2. *Strategy*

   Strategy game menantang pemain untuk mencapai kemenangan dengan perencanaan, khususnya melalui perencanaan serangkaian tindakan yang dilakukan melawan satu lawan atau lebih. Kemenangan diraih dengan perencanaan matang dan pengambilan keputusan yang optimal.

3. *Role Playing Game* (RPG)

   RPG adalah *game* dimana pemain mengontrol satu atau lebih karakter yang biasanya di desain oleh pemain itu sendiri, dan memandu mereka melewati berbagai rintangan yang diatur oleh komputer. Perkembangan karakter dalam hal kekuatan dan kemampuannya adalah kunci dari *game* jenis ini.

4. *Sports*

   Sports game mensimulasikan berbagai aspek dari olahraga atletik nyata maupun imajiner, apakah itu memainkan pertandingan, me-*manage* tim dan karir, atau keduanya. Salah satu contoh *game* jenis ini adalah *Pro Evolution Soccer 2012* (PES 2012), dimana pemain bisa memainkan pertandingan, menjadi manajer tim, maupun menjadi pemain dan mengembangkan karirnya sendiri.

5. *Vechicle Simulation*

   Vechicle simulation membuat *feeling* mengendarai kendaraan, baik kendaraan nyata maupun kendaraan imajiner. Performa dan karakteristik kasar mesin harus menyerupai kenyataan, kecuali jika yang didesain adalah kendaraan imajiner.

6. *Construction and Management Simulation*

   CMS game adalah *game* tentang proses. Tujuan pemain bukan untuk mengalahkan musuh, tetapi membangun sesuatu dengan konteks proses yang sedang berjalan. Semakin pemain mengerti dan mengontrol proses, semakin sukses sesuatu yang ia bangun. *Game* seperti ini biasanya menyediakan dua jenis permainan, yaitu mode bebas dimana pemain bebas membangun sesuatu, dan mode misi dimana terdapat skenario hal apa yang harus dibangun oleh pemain.

7. *Adventure*

   Adventure game adalah cerita interaktif tentang karakter protagonist yang dimainkan oleh pemain. Penyampaian cerita dan eksplorasi adalah elemen inti dari *game* ini. Penyelesaian teka-teki dan tantangan konseptual adalah bagian besar dari permainan.

8. *Artificial Life and Puzzle Game*

   Artificial Life game adalah *game* yang membuat tiruan dari kehidupan sebenarnya. Biasanya ada dua jenis *game* ini, tiruan kehidupan manusia, contohnya The SIMS, dan tiruan kehidupan binatang, contohnya Tamagochi.

9. *Online Game*

   Istilah *online game* disini mengacu kepada *multiplayer game* dimana mesin dari para pemain terhubung dengan jaringan.

II.3.3 Game Edukasi

Game edukasi adalah *game* digital yang dirancang untuk pengayaan pendidikan (mendukung pengajaran dan pembelajaran), menggunakan teknologi multimedia interatif [9].

II.3.3.1 Kriteria Game Edukasi Ideal

1. Rasa ingin tahu, fantasi dan kontrol pengguna
   a. Motivasi intrinsik, menyenangkan untuk digunakan. Pengguna memiliki kontrol terhadap permainan.
   b. Integrasi antara materi edukasi dan aspek fantasi permainan
   c. Pengguna memegang kendali permainan dan dapat mengatur kecepatan bermain.
   d. Mendorong keingintahuan, pengguna dapat melakukan eksplorasi bebas. Permainan mengandung rahasia tersembunyi.
   e. Pengguna dapat menyimpan kemajuan permainan.
   f. Menyediakan lebih dari jalur. Ada beberapa cara untuk memenangkan permainan
   g. Pengguna dapat melakukan trial-error untuk menyelesaikan permainan.
   h. Keberhasilan ditentukan oleh pengetahuan, bukan kebetulan.
   i. Task disajikan secara incremental dan bertahap.
   j. Simulasi realistik dunia.
   k. Materi edukasi disesuaikan dengan materi dunia nyata.
   l. Personalisasi: pengguna dapat mengubah karakter dan objek-objek.
   m. Menyediakan banyak pilihan dan tema.
   n. Hindari pengulangan, hindari *drill-practice.*
   o. Sediakan elemen tidak terduga dan kejutan.

p. Point tidak hilang saat menjawab pertanyaan.

2. Tantangan
   a. Kinerja pemain terukur dan didefinisikan dengan baik.
   b. Tantangan diberikan secara terus menerus tapi disesuaikan dengan tingkat pemahaman pemain.
   c. Kompleksitas semakin berkembang sejalan dengan kemampuan pengguna. Tersedia banyak *level*.
   d. Memonitor kinerja pengguna, berdasarkan hasil monitoring ini kesulitan kemudian disesuaikan.
   e. Pengguna dapat melihat progressnya setiap saat.
   f. Menyediakan hint dan instruksi untuk membantu pengguna.
   g. Bantuk pengguna menemukan *'flow state'* yaitu keseimbangan antara tantangan dengan kemampuan untuk menghadapi tantangan.
3. Sosialisasi
   a. Pemain dapat berkolaborasi bersama.
   b. Pemain dapat saling berkompetisi.
   c. Pemenang bisa lebih dari satu (*multiple winner*)
4. Pedagogi
   a. Menyebutkan secara eksplisit target umur dan menyesuaikan rancangan permainan sesuai umur tersebut.
   b. Aktivitas pembelajaran dilaksanaan bersamaan dengan permainan.
   c. Objektif pembelajaran jelas.
   d. Memberi kesempatan kepada guru dan orangtua untuk ikut berperan.
   e. Memberikan petunjuk yang jelas sehingga pemain dapat berkonsentrasi kepada isi permainan, bukan bagaimana cara menggunakan permainan.
   f. Menyediakan tutorial sehingga pemain tidak perlu membaca manual.
5. Teknologi
   a. Menggunakan teknologi yang tersedia di sekolah dan masyarakat umum (tidak membutuhkan hardware yang terlalu tinggi).
   b. Memberikan lisensi yang terjangkau sekolah.

c. Menggunakan antarmuka yang intuitif.

6. Pengguna anak-anak dan yang berkebutuhan khusus
   a. Memberikan petunjuk yang dibacakan selain petunjuk tertulis.
   b. Gambar, objek, layar tertata rapi.
   c. Permainannya sendiri merupakan aktivitas yang penting (*play for the sake of play*)
   d. Transformasi terlihat. Saat anak melakukan interaksi, ada instant *feedback.*
   e. Input dan output yang mudah terlihat.
   f. Tantangan diberikan secara gradual.
   g. Permainan menyenangkan untuk diulang, beberapa karakter, music mudah teringat oleh anak.
   h. Objek dan suara diambil dari kehidupan sehari-hari yang mudah dikenali.
   i. Menginspirasi anak, bahkan setelah komputer dimatikan.

II.4 Berorientasi Objek

Secara spesifik, pengertian berorientasi objek berarti bahwa mengorganisasi perangkat lunak sebagai kumpulan dari objek tertentu yang memiliki struktur data dan perilakunya. Hal ini yang membedakan dengan pemograman konvensional dimana struktur data dan perilaku hanya berhubungan secara terpisah. Terdapat beberapa cara untuk menentukan karateristik dalam pendekatan berorientasi objek, tetapi secara umum mencakup empat hal, yaitu identifikasi, klasifikasi, *polymorphism* (polimorfisme) dan *inheritance* (pewarisan) [10].

II.4.1 Konsep Dasar Berorientasi Objek

Pendekatan berorientasi objek merupakan suatu teknik atau cara pendekatan dalam melihat permasalahan dan sistem (sistem perangkat lunak, sistem informasi, atau sistem lainnya). Pendekatan berorientasi obejek akan memandang sistem yang akan dikembangkan sebagai suatu kumpulan objek yang berkorespondensi dengan objek-objek dunia nyata [10].

Berikut ini adalah beberapa konsep dasar yang harus dipahami tentang metodelogi berorientasi objek:

1. Kelas (*class*)

   Kelas adalah kumpulan objek-objek dengan karakteristik yang sama. Kelas merupakan definisi statis dan himpunan objek yang sama yang mungkin lahir atau tercipta dan kelas tersebut. Sebuah kelas akan mempunyai sifat (atribut), kelakuan (operasi/metode), hubungan (*relationship*) dan arti. Suatu kelas dapat diturunkan dan kelas yang lain, dimana atribut dan kelas semula dapat diwariskan ke kelas yang baru.

2. Objek (*object*)

   Objek adalah abstraksi dan sesuatu yang mewakilkan dunia nyata seperti benda, manusia, satuan organisasi, tempat, kejadian, struktur, status, atau hal-hal lain yang bersifat abstrak. Objek merupakan entitas yang mampu menyimpan informasi (status) dan mempunyai operasi (kelakuan) yang dapat diterapkan atau dapat berpengaruh pada status objeknya. Objek mempunyai siklus hidup yaitu diciptakan, dimanipulasi, dan dihancurkan.

3. Metode (*method*)

   Operasi atau metode pada sebuah kelas hampir sama dengan fungsi prosedur pada metodelogi structural. Sebuah kelas boleh memiliki lebih dari satu metode atau operasi yang berfungi untuk memanipulasi objek itu sendiri. Operasi atau metode merupakan fungsi atau transformasi yang dapat dilakukan terhadap objek atau dilakukan oleh objek.

4. Atribut (*attribute)*

   Atribut dari sebuah kelas adalah variable global yang dimiliki sebuah kelas. Atribut dapat berupa nilai atau elemen-elemen data yang dimiliki oleh objek dalam kelas objek. Atribut dipunyai secara individual oleh sebuah objek, misalnya berat, jenis, nama dan sebagainya.

5. Abstraksi (*abstraction)*

   Prinsip untuk merepresentasikan dunia nyata yang kompleks menjadi satu bentuk model yang sederhana dengan mengabaikan aspek-aspek lain yang tidak sesuai dengan permasalahan.

6. Enkapsulasi (*encapsulation*)

   Pembungkusan atribut data dan layanan (operasi-operasi)yang dipunyai objek untuk menyembunyikan implementasi dan objek sehingga objek lain tidak mengetahui cara kerjanya.

7. Pewarisan (*inheritance)*

   Mekanisme yang memungkinkan suatu objek mewarisi sebagian atau seluruh definisi dan objek lain sebagai bagian dari dirinya.

8. Antarmuka (*Interface*)

   Antarmuka (*Interface*) sangat mirip dengan kelas, tapi tanpa atribut kelas dan memiliki metode yang dideklarasikan tanpa isi. Deklarasi metode pada sebuah *interface* dapat diimplementasikan oleh kelas lain.

9. *Reusability*

   Pemanfaatan kembali objek yang sudah didefinisikan untuk suatu permasalah pada permasalahan lainnya yang melibatkan objek tersebut.

10. Generalisasi dan Spesialisasi

    Menunjukkan hubungan antara kelas dan objek yang umum dengan kelas dan objek yang khusus. Misalnya kelas yang lebih umum (generalisasi) adalah kendaraan darat dan kelas khususnya (spesialisasi) adalah mobil, motor, dan kereta.

11. Komunikasi Antar Objek

    Komunikasi Antar Objek dilakukan lewat pesan (*message*) yang dikirim dari satu objek ke objek yang lain.

12. Polimorfisme (*polymorphism*)

    Kemampuan suatu objek untuk digunakan di banyak tujuan yang berbeda dengan nama yang sama sehingga menghemat baris program.

13. *Package*

    Package adalah sebuah container atau kemasan yang dapat digunakan untuk mengelompokkan kelas-kelas sehingga memungkinkan beberapa kelas yang bernama sama disimpan dalam *package* yang berbeda.

II.4.2 Analisis Berorientasi Objek

Analisis berorientasi objek atau *Object Oriented Analysis* (OOA) adalah tahap untuk menganalisis spesifikasi atau kebutuhan akan *sistem* yang akan dibangun dengan konsep berorientasi objek, apakah benar kebutuhan yang ada dapat diimplementasikan menjadi sebuah *sistem* berorientasi objek.

OOA biasanya menggunakan kartu CRC (*Component, Responsibility, Collaborator*) untuk membangun kelas-kelas yang akan digunakan atau menggunakan UML (*Unified Modeling Language*) pada bagian diagram *use case*, diagram kelas, dan diagram objek.

II.4.3 Desain Berorientasi Objek

Desain berorientasi objek atau *Object Oriented Design* (OOD) adalah tahapan perantara untuk spesifikasi atau kebutuhan *sistem* yang akan dibangun dengan konsep berorientasi objek ke desain pemodelan agar lebih mudah diimplementasikan dengan pemograman berorientasi objek.

Pemodelan berorientasi objek biasanya dituangkan dalam dokumentasi perangkat lunak dengan menggunakan perangkat lunak dengan menggunakan perangkat pemodelan berorientasi objek, diantaranya adalah UML (*Unified Modeling Language*). Kendala dan permasalahan pembangunan sistem berorientasi objek biasanya dapat dikenali dalam tahap ini [10].

II.4.4 Pemodelan

Pemodelan adalah gambaran dari realita yang sederhana dan dituangkan dalam bentuk pemetaan denan aturan tertentu. Pemodelan dapat menggunakan bentuk yang sama dengan realitas. Pemodelan juga banyak digunakan untuk merencanakan suatu hal agar kegagalan dan resiko yang mungkin terjadi dapat diminimalisir.

Pada dunia pembangunan perangkat lunak juga diperlukan pemodelan. Pemodelan perangkat lunak digunakan untuk mempermudah langkah berikutnya dari pengembangan sebuah sistem sehingga lebih terencana. Pemodelan pada

pembangunan perangkat lunak digunakan untuk menvisualkan perangkat lunak yang akan dibuat [10].

II.5 UML (Unified Modelling Language)

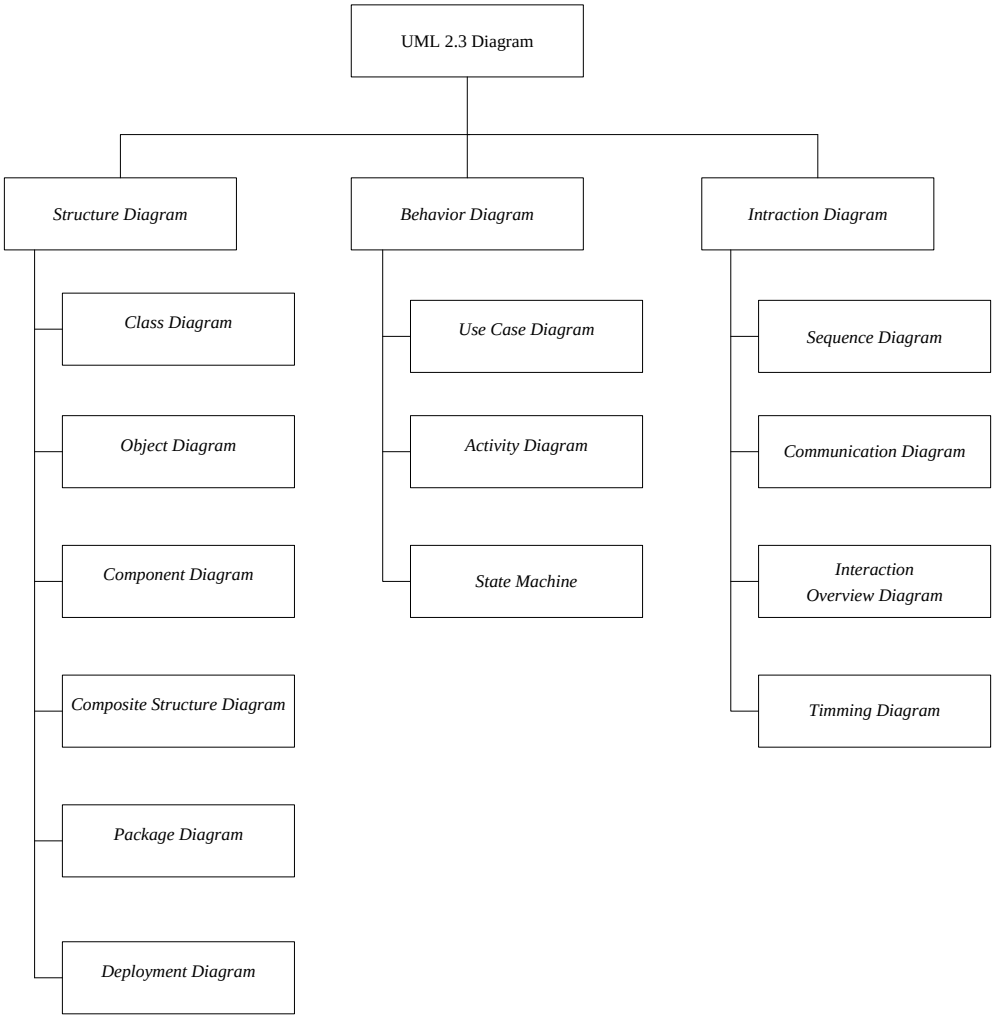
Unified Modelling Language (UML) adalah sebuah "bahasa" yang telah menjadi standar dalam industry visualisasi, merancang, dan mendokumentasikan sistem peranti lunak. UML menawarkan sebuah standar untuk merancang model sebuah sistem.

UML dapat digunakan untuk membuat model untuk semua jenis perangkat lunak, dimana perangkat lunak tersebut dapat berjalan pada piranti keras, sistem operasi dan jaringan apapun, serta ditulis dalam bahasa pemrograman apapun. Tetapi karena UML juga menggunakan *class* dan *operation* dalam konsep dasarnya, maka ia lebih cocok untuk penulisan piranti lunak dalam bahasa berorientasi objek.

Seperti bahasa-bahasa lainnya, UML mendefinisikan notasi dan *syntax*. Notasi UML merupakan sekumpulan bentuk khusus untuk menggambarkan berbagai diagram piranti lunak. Setiap bentuk memiliki makna tertentu, dan UML *syntax* mendefinisikan begaimana bentuk-bentuk tersebut dapat dikombinasikan. Notasi UML terutama diturunkan dari 3 notasi yang telah ada sebelumnya yaitu: Grady Booch OOD (*Object Oriented Design*), Jim Rumbaugh OMT (*Object Modeling Technique*) dan Ivar Jacobson OOSE (*Object Oriented Software Engineering*).

II.5.1 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.1 Diagram UML

Berikut ini penjelasan singkat dari pembagian katagori tersebut.

1. *Structure diagrams* yaitu kumpulan diagram yang digunakan untuk menggambarkan suatu struktur statis dari sistem yang dimodelkan.
2. *Behavior diagrams* yaitu kumpulan diagram yang digunakan untuk menggambarkan kelakuan sistem atau rangkain perubahan yang terjadi pada sebuah sistem.
3. *Interaction diagrams* yaitu kumpulan diagram yang digunakan untuk menggambarkan interaksi sistem dengan sistem lain maupun interaksi antarsubsistem pada suatu sistem.

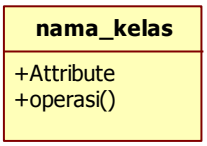
II.5.1.1 Class Diagram

Diagram kelas atau *class diagram* menggambarkan struktur sistem dari segi pendefinisian kelas-kelas yang akan dibuat untuk membangun sistem. Kelas mimiliki apa yang disebut atribut dan metode atau operasi.

- Atribut merupakan variable-variabel yang dimiliki oleh suatu kelas.
- Operasi atau metode adalah fungsi-fungsi yang dimiliki oleh suatu kelas.

Berikut adalah simbol-simbol yang ada pada diagram kelas:

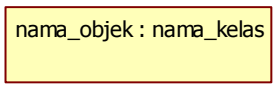
Tabel II.1 Simbol-simbol *Class Diagram*

Simbol	Deskripsi
Kelas 	Kelas pada struktur sistem
Antarmuka / *interface*  **nama_interface**	Sama dengan konsep *interface* dalam pemrograman berorientasi objek
Asosiasi / *association*	Relasi antar kelas dengan makna umum, asosiasi biasanya juga disertai dengan *multiplicity*
Asosiasi berarah / *directed association* 	Relasi antar kelas dengan makna kelas yang satu digunakan oleh kelas yang lain, asosiasi biasanya juga disertai dengan *multiplicity*
Generalisasi 	Relasi antar kelas dengan makna generalisasi-spesialisasi (umum-khusus)
Kebergantungan / *Dependency*	Relasi antar kelas dengan makna kebergantungan antar kelas
Agregasi / *aggregation*	Relasi antar kelas dengan makna semua bagian (*whole part*)

II.5.1.2 Objek Diagram

Diagram objek menggambarkan struktur sistem dari segi penamaan objek dan jalannya objek dalam sistem. Pada diagram objek harus dipastikan semua kelas yang sudah didefinisikan pada diagram kelas harus dipakai objeknya, karena jika tidak, pendefinisian kelas itu tidak dapat dipertanggungjawabkan.
Berikut adalah simbol-simbol yang ada pada diagram objek:

Tabel II.2 Simbol-simbol *Object Diagram*

Simbol	Deskripsi
Objek 	Objek dari kelas yang berjalan saat sistem berjalan
Link	Relasi antar objek

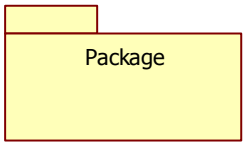
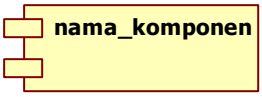
II.5.1.3 Component Diagram

Diagram komponen atau *component diagram* dibuat untuk menunjukkan organisasi dan ketergantungan di antara kumpulan komponen dalam sebuah sistem. Diagram komponen fokus pada komponen sistem yang dibutuhkan dan ada di dalam sistem. Diagram komponen juga dapat digunakan untuk memodelkan hal-hal berikut:

1. *Source code* program perangkat lunak.
2. Komponen *executable* yang dilepas ke *user*.
3. Basis data secara fisik.
4. Sistem yang harus beradaptasi dengan sistem lain.
5. *Framework* sistem, *framework* pada perangkat lunak merupakan kerangka kerja yang dibuat untuk memudahkan pengembangan dan pemeliharaan aplikasi.

Berikut adalah simbol-simbol yang ada pada diagram komponen:

Tabel II.3 Simbol-simbol *Component Diagram*

Simbol	Deskripsi
Package 	*Package* merupakan sebuah bungkusan dan satu atau lebih komponen
Komponen 	Komponen sistem
Kebergantungan / *Dependency* 	Kebergantungan antar komponen, arah panah mengarah pada komponen yang dipakai
Antarmuka / *interface* 	Sama dengan konsep *interface* pada pemograman berorientasi objek, yaitu sebagai antarmuka komponen agar tidak mengakses langsung komponen.
Link	Relasi antar komponen

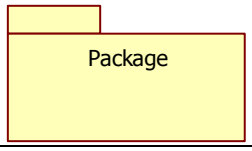
II.5.1.4 Composite Structure Diagram

Composite structure diagram baru mulai ada pada UML versi 2.0, pada versi 1.x diagram ini belum muncul. Diagram ini dapat digunakan untuk menggambarkan struktur dari bagian-bagian yang saling terhubung maupun mendeskripsikan struktur pada saat berjalan (*runtime*) dari instance yang saling terhubung. Contoh penggunaan diagram ini misalnya untuk menggambarkan deskripsi dari setiap bagian mesin yang saling terkait untuk menjalankan fungsi mesin tersebut, menggambarkan aliran data router pada jaringan computer, dll.

II.5.1.5 Package Diagram

Package diagram menyediakan cara mengumpulkan elemen-elemen yang saling terkait dalam diagram UML. Hampir semua diagram dalam UML dapat dikelompokkan menggunakan package diagram. Berikut ini simbol-simbol yang digunakan:

Tabel II.4 Simbol-simbol *Package Diagram*

Simbol	Deskripsi
Package 	*Package* merupakan sebuah bungkusan dari satu atau lebih kelas atau elemen diagram UML lainnya.

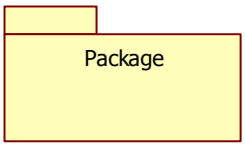
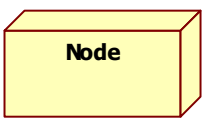
II.5.1.6 Deployment Diagram

Diagram *deployment* atau *deployment diagram* menunjukkan konfigurasi komponen dalam proses eksekusi aplikasi. Diagram deployment juga dapat digunakan untuk memodelkan hal-hal berikut:

1. Sistem tambahan (*embedded sistem*) yang menggambarkan rancangan *divice, node, dan hardware*.
2. Sistem *client/server*.
3. Sistem terdistribusi murni.
4. Rekayasa ulang aplikasi.

Berikut adalah simbol-simbol yang ada pada diagram *deployment*:

Tabel II.5 Simbol-simbol *Deployment Diagram*

Simbol	Deskripsi
Kebergantungan / *Dependency* 	Kebergantungan antar *node*, arah panah mengarah pada *node* yang dipakai
Link	Relasi antar *node*
Package 	*Package* merupakan sebuah bungkusan dari satu atau lebih *node*
Node 	Biasanya mengacu pada perangkat keras (*hardwar*e), perangkat lunak yang tidak dibuat sendiri (*software*), jika didalam node disertakan komponen untuk mengkonsistenkan rencana maka komponen yang diikutsertakan harus sesuai dengan komponen yang telah didefinisikan sebelumnya pada diagram komponen.

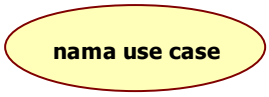
II.5.1.7 Use Case Diagram

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. Syarat pada *use case* adalah nama didefinisikan sesimpel mungkin dan dapat dipahami. Ada dua hal utama pada *use case* yaitu pendefinisian apa yang disebut aktor dan *use case*.

1. Aktor merupakan orang, proses, atau sistem lain yang berinteraksi dengan sistem informasi yang dapat dibuat di luar sistem informasi yang akan dibuat itu sendiri, jadi walaupun symbol dari aktor adalah gambar orang, tapi aktor belum tentu merupkan orang.
2. *Use case* merupakan fungsionalitas yang disediakan sistem sebagai unit-unit yang saling bertukar pesan antar unit aktor.

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

Tabel II.6 Simbol-simbol *Use case Diagram*

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 fase nama *use case*
Aktor / *actor* 	Orang, proses, atau sistem lain yang berinteraksi dengan sistem informasi yang akan dibuat di luar sistem informasi yang akan dibuat itu sendiri, jadi walaupun symbol atau aktor adalah gambar orang, tapi aktor belum tentu merupakan orang; biasanya dinyatakan menggunakan kata benda di awal fase nama aktor
Asosiasi / *association* 	Komunkasi antar aktor dan *use case* yang berpartisipasi pada *use case* atau *use case* memiliki interaksi dengan aktor

II.5.1.8 Activity Diagram

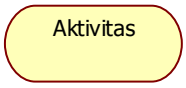
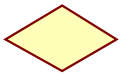
Diagram aktivitas atau *activity diagram* menggambarkan *workflow* (aliran kerja) atau aktivitas dari sebuah sistem atau proses bisnis. Yang perlu diperhatikan disini adalah bahwa diagram aktivitas menggambarkan aktivitas sistem bukan apa yang dilakukan aktor, jadi aktivitas yang dapat dilakukan oleh sistem.

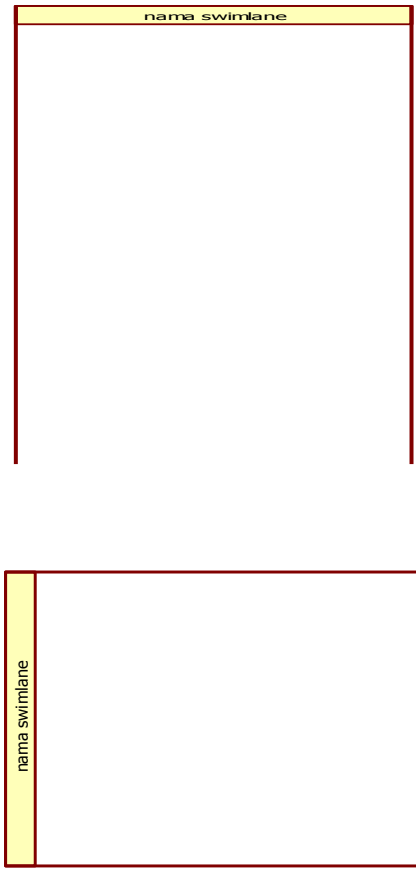
Diagram aktivitas juga banyak digunakan untuk mendefinisikan hal-hal beriukut:

1. Rancangan proses bisnis di mana setiap urutan aktivitas yang digambarkan merupakan proses bisnis sistem yang didefinisikan.
2. Urutan atau pengelompokan tampilan dari sistem /user interface di mana setiap aktivitas memiliki sebuah rancangan antarmuka tampilan.
3. Rancangan pengujian di mana setiap aktivitas memerlukan sebuah pengujian yang perlu didefinisikan kasus ujinya.

Berikut adalah simbol-simbol yang ada pada diagram aktivitas:

Tabel II.7 Simbol-simbol *Activity Diagram*

Simbol	Deskripsi
Status awal 	Status awal aktifitas 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

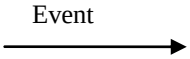
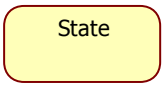
Simbol	Deskripsi
Swimlane 	Memisahkan organisasi bisnis yang bertanggung jawab terhadap aktivitas yang terjadi.

II.5.1.9 State Machine Diagram

State machine diagram atau dalam bahasa Indonesia disebut diagram mesin status digunakan untuk menggambarkan perubahan status atau transisi status dari sebuah mesin atau sistem. Perubahan tersebut digambarkan dalam suatu graf berarah. State machine diagram merupakan pengembangan dari diagram Finite State Automata dengan penambahan beberapa fitur dan konsep baru. Diagram *Finite State Automata* (FSA) ini biasanya diajarkan dalam mata kuliah Automata.

State machine diagram cocok digunakan untuk menggambarkan alur interaksi pengguna dengan sistem. Berikut ini komponen-komponen dasar yang ada dalam *state machine diagram*:

Tabel II.8 Simbol-simbol *State Machine Diagram*

Simbol	Deskripsi
Start (Initial State) 	*Start (Initial State)* adalah *state* atau keadaan awal pada saat sistem mulai hidup
End (Final State) 	*End (Final State)* adalah *state* keadaan akhir dari daur hidup sebuah sistem
Event 	*Event* adalah kegiatan yang menyebabkan berubahnya stastus mesin
State 	*State* atau status adalah keadaan sistem pada waktu

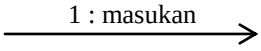
II.5.1.10Sequance Diagram

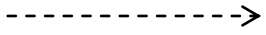
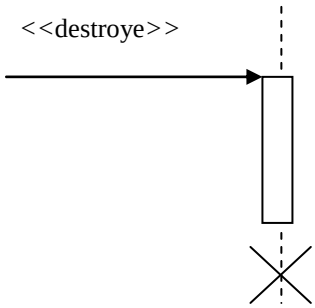
Diagram sekuen menggambarkan kelakuan objek pada *use case* dengan mendeskrifsikan waktu hidup objek dan message yang dikirimkan dan diterima antarobjek. Oleh karena itu untuk menggambar diagram sekuen maka harus diketahui objek-objek yang terlibat dalam sebuah *use case* beserta metode-metode yang dimiliki kelas yang diinstansiasi menjadi objek itu.

Banyaknya diagram sekuen yang harus digambar adalah sebanyak pendefinisian *use case* yang memiliki proses sendiri atau yang penting semua *use case* yang telah didefinisikan interaksi jalannya pesan sudah dicakup pada diagram sekuen sehinga semakin banyak *use case* yang didefinisikan maka diagram sekuen yang harus dibuat juga semakin banyak.

Berikut adalah simbol-simbol yang ada pada diagram sekuen:

Table II.9 Simbol-simbol *Sequence Diagram*

Simbol	Deskripsi
Aktor  **nama aktor** Atau Nama aktor Tanpa waktu aktif	Orang, proses, atau sistem lain yang berinteraksi dengan sistem informasi yang akan dibuat diluar sistem informasi yang akan dibuat itu sendiri, jadi walaupun symbol dari aktor adalah gambar orang, tapi aktor belum tentu merupakan orang; biasanya dinyatakan menggunakan kata benda di awal frase nama aktor
Garis hidup / *lifeline* 	Menyatakan kehidupan suatu objek
Objek nama_objek : nama_kelas	Menyatakan objek yang berinteraksi pesan
Waktu aktif 	Menyatakan objek dalam keadaan aktif dan berinteraksi pesan
Pesan tipe *create* <<create>> 	Menyatakan suatu objek membuat objek yang lain, arah panah mengarah pada objek yang dibuat
Pesan tipe *call* 1 : nama_metode () 	Menyatakan suatu objek memanggil operasi / metode yang ada pada objek lain atau dirinya sendiri Arah panah mengarah pada objek yang memiliki operasi/ metode, karena ini memanggil operasi/metode maka operasi/metode yang dipanggil harus ada pada diagram kelas sesuai dengan kelas objek yang berinteraksi.
pesan tipe *send* 1 : masukan 	Menyatakan bahwa suatu objek mengirimkan data / masukkan / informasi ke objek lainnya, arah panah mengarah pada objek yang dikirimi

Simbol	Deskripsi
Pesan tipe *return* 1 : keluaran 	Menyatakan bahwa suatu objek yang telah menjalankan suatu operasi atau metode menghasilkan suatu kembalian ke objek tertentu, arah panah mengarah pada objek yang menerima kembalian
Pesan tipe *destroy* 	Menyatakan suatu objek mengakhiri hidup objek yang lain, arah panah mengarah pada objek yang diakhiri, sebaiknya jika ada *create* maka ada *destroy*

II.5.1.11 Communication Diagram

Communication diagram atau diagram komunikasi pada UML versi 2.x adalah penyederhanaan dari diagram kolaborasi (collaboration diagram) pada UML versi 1.x. jadi *collaboration diagram* sudah tidak muncul lagi pada UML versi 2.x.

Diagram komunikasi menggambarkan interaksi antarobjek/bagian dalam bentuk urutan pengiriman pesan. Diagram komunikasi mempersentasikan informasi yang diperoleh dari Diagram Kelas, Diagram Sekuen, dan Diagram *Use case* untuk mendeskripsikan gabungan antara struktur statis dan tingkah laku dinamis dari suatu sistem.

Diagram komunikasi mengelompokkan message pada kumpulan diagram sekuen menjadi sebuah diagram. Dalam diagram komunikasi yang dituliskan adalah operasi/metode yang dijalankan antara objek yang satu dan objek lainnya secara keseluruhan, oleh karena itu dapat diambil dari jalannya interaksi pada semua diagram sekuen. Penomoran metode dapat dilakukan berdasarkan metode dijalankannya metode/operasi di antara objek yang satu dengan objek yang lainnya atau objek itu sendiri.

II.5.1.12Timing Diagram

Timing diagram merupakan diagram yang fokus pada penggambaran terkait batasan waktu. Timing diagram digunakkan untuk menggambarkan tingkah laku sistem dalam periode waktu tertentu. *Timing diagram* biasanya digunakan untuk mendeskripsikan operasi dari alat dijital karena penggambaran secara visual akan lebih mudah dipahami daripada dengan kata-kata.

II.5.1.13Interaction Overview Diagram

Interaction overview diagram mirip dengan diagram aktivitas yang berfungsi untuk menggambarkan sekumpulan urutan aktivitas. Interaction overview diagram adalah bentuk aktivitas diagram yang setiap titik merepersentasikan diagram interaksi. Interaksi diagram dapat meliputi diagram sekuen, diagram komunikasi, *interaction overview diagram*, dan *timing diagram*.

Hampir semua notasi pada interaction overview diagram sama dengan notasi pada diagram aktivitas. Sebagai contoh *initial, final, decision, merge, fork*, dan *join nodes* sama seperti pada diagram aktivitas. Tambahan pada *interaction overview diagram* adalah *interaction occurrence* dan *interaction element*.

II.6 Algoritma Fisher Yates Shuffle

Fisher-Yates Shuffle (dinamai berdasarkan penemunya, Ronald Fisher dan Frank Yates) digunakan untuk mengubah urutan masukan yang diberikan secara acak. Permutasi yang dihasilkan oleh algoritma ini muncul dengan probabilitas yang sama. Metode dasar yang diberikan untuk menghasilkan permutasi acak dari angka 1 - *N* berjalan sebagai berikut:

1. Tuliskan angka dari 1 sampai n
2. Isi nilai k dengan bilangan acak antara 0 hingga i+1 bulatkan kebawah
3. hitung dari low end, gantikan nilai k dan tuliskan di tempat lain
4. Ulangi dari langkah 2 sampai semua nomor digantikan
5. Urutan angka yang tertulis di langkah 3 sekarang permutasi acak dari nomor asli.

Algoritma ini dinyatakan bias karena permutasi yang dihasilkan oleh algoritma ini muncul dengan probabilitas yang sama.

II.7 Algoritma Minimax

Algoritma Minimax dikembangkan pada 1928 oleh John von Neuman dan prinsip-prinsip yang dimilikinya masih merupakan salah satu yang paling terkenal dan banyak digunakan dalam *game* sekarang. Prinsip-prinsip ini hanya berlaku untuk *game* yang memiliki dua pemain yang bermain secara bergantian, yaitu manusia melawan komputer seperti dalam permainan catur, dam, dan tic-tac-toe.

Minimax bekerja sangat baik bila seluruh keadaan langkah yang mungkin terjadi para pemain dapat diketahui seperti pada permainan catur, checker, dan tic-tac-toe. Algoritma minimax tidak bisa digunakan pada *game-game* poker karena komputer tidak bisa melihat semua langkah pemain(lawan) yang mungkin terjadi.

Untuk mengimplementasikan minimax diperlukan sebuah metode untuk mengukur seberapa baik sebuah posisi pada saat itu. Metode ini disebut sebagai fungsi evaluasi. Fungsi evaluasi merupakan sebuah heuristic, yang kenyataannya sangat sulit untuk menentukan beberapa nilai presisi pada sebuah posisi saat itu. Fungsi evaluasi yang baik akan memperhitungkan dari banyak aspek yang berbeda, seperti posisi papan, jumlah pieces relative terhadap lawan, control papan, control kunci tertentu pada papan, dan mobilitas. Mobilitas menunjukkan kebebasan setiap pemain dalam memainkan pieces dan seberapa banyak langkah setiap pieces yang dimiliki oleh setiap pemain [11].

Misalkan $f(n)$ adalah fungsi evaluasi dari posisi ke-n, maka:

a. $f(n) > 0$: adalah posisi n menang untuk komputer dan kalah bagi lawan.
b. $f(n) < 0$: adalah posisi n menang untuk lawan dan kalah bagi komputer.
c. $f(n) = 0$: adalah posisi n netral.

Contoh fungsi evaluasi pada permainan tic-tac-toe:

a. $f(n) = 1$ bila komputer menang.
b. $f(n) = -1$ bila lawan menang
c. $f(n) = 0$ bila keduanya bermain seri.

Contoh fungsi evaluasi pada catur:

$f(n) = w_1 f_1 + w_2 f_2 + \ldots + w_n f_n$

$w_1 = 1$, f_1 = jumlah pion komputer – jumlah pion lawan

$w_2 = 3$, f_2 = jumlah gajah komputer – jumlah gajah lawan

Dalam minimax, MAX adalah sebutan bagi pemain (komputer) yang bertujuan untuk mendapatkan nilai maksimal dan MIN adalah sebutan bagi (lawan) yang bertujuan untuk mendapatkan nilai minimal.

Prosedur minimax:

1. Tandai masing-masing *level* dari ruang pencarian sesuai dengan langkahnya pada level itu.
2. Mulai pada *node* daun (node paling bawah) dengan menggunakan fungsi evaluasi, berikan nilai pada masing-masing *node*.
3. Arah menjalar ke atas : Jika node orang tua adalah MAX, pilihlan nilai terbesar yang terdapat pada node anak dan berikan nilai ini pada MAX (*node* orang tua).
4. Arah menjalar ke atas : Jika node orang tua adalah MIN, pilihlan nilai terbesar yang terdapat pada node anak dan berikan nilai ini pada MIN (*node* orang tua).

II.8 Construct2

Construct2 adalah produk buatan Scirra, perusahaan yang berasal dari kota London, Inggris. Dengan menggunakan *game engine* ini, entah itu *developer* pemula maupun *expert* sangat dimanjakan dengan kemudahan pemakaian. Cukup membaca beberapa manual di situs resmi Scirra.

Game builder ini sebenarnya dirancang untuk *game* berbasis 2D. Dengan menggunakan Construct 2, pengembang permainan dapat mem-*publish*nya ke beberapa *platform* seperti:

1. HTML5 website
2. Google Chrome Webstore
3. Facebook
4. Phonegap (Android)

5. Windows Phone 8, bahkan Windows 8.

Untuk mempercantik *game*, disediakan pula 70 visual effect yang menggunakan *engine* WebGL. Construct 2 juga dilengkapi dengan 20 built-in *plugin* dan *behaviours* (perilaku objek) serta dapat menambah *sprite* animasi, objek teks, mengkoneksikan ke Facebook, menambah musik, manipulasi penyimpanan data *game*, dan penambahan efek-efek grafis layaknya Adobe Photoshop.

Pemanggilan fungsi-fungsi di Construct 2 hanya menggunakan pengaturan *Events* yang telah disediakan. *Events* merupakan pilihan-pilihan *Action* dan kondisi yang akan menjadi nyawa dalam *game* dan *game* dapat berjalan sesuai keinginan. Karena berbasis HTML5, *preview* saat ingin menjajal *game* pun hanya memanfaatkan *browser* (*localhost*).