BAB II

LANDASAN TEORI

Dalam bab ini penulis akan menjelaskan secara teoritis tentang perancangan sistem, dan segala sesuatu definisi tertulis yang berhubungan dengan judul yang ditulis mengenai **"PERANGKAT AJAR INTERAKTIF PEMBELAJARAN FISIKA BERBASIS MULTIMEDIA UNTUK SISWA SEKOLAH LANJUT TINGKAT PERTAMA (SLTP)"**.

2.1. Teori Tentang Permasalahan

Pada bab ini penulis akan mencoba untuk menjelaskan secara teori tentang permasalahan yang dijadikan judul penulisan skripsi yang dibuat oleh penulis. Sesuai dengan apa yang menjadi permasalahan yang ada secara garis besar kita harus dapat memahami pengertian dari penulisan judul tersebut, maka dari itu penulis akan menjelaskan secara garis besar tentang Perangkat Ajar Interaktif Pembelajaran Fisika Berbasis Multimedia.

2.1.1. Pengertian Perangkat Ajar

Pengertian perangkat ajar menurut Budi Atmoko adalah Perangkat ajar merupakan perangkat lunak pengajaran berbantuan komputer. (3)

2.1.2. Pengertian Interktif

Pengertian interaktif menurut *Encarta encyclopedia*, yaitu **:**

1. *Communicating or collaborating : involving the communication or collaboration*

2. *Comput with user-machine communication: allowing or involving the exchange of information or instructions between a person and a machine such as a computer or a television*
3. *Comput operator-controlled: operating on instructions entered by somebody at a keyboard or other input device.* (4)

Berdasarkan pengertian di atas, dapat disimpulkan bahwa interaktif adalah

1. Berkomunikasi atau bekerja sama : menyertakan kerja sama atau kolaborasi atau komunikasi hal-hal atau orang-orang.
2. Komunikasi : membiarkan atau menyertakan pertukaran instruksi atau informasi antara seseorang dan suatu mesin seperti suatu komputer atau suatu televisi.
3. Komputer kontrol : beroperasi instruksi yang dimasukkan oleh seseorang pada suatu papan tombol atau alat masukan lain.

Sedangkan menurut Kamus Besar Bahasa Indonesia pengertia interaktif adalah bersifat saling melakukan aksi; antar hubungan; saling aktif. Komputer berkaitan dengan dialog antar komputer dan terminal atau komputer dan komputer. (2)

2.1.3. Pengertian Fisika

Pengertian fisika menurut wikipedia.org**,** yaitu Fisika (Bahasa Yunani: *φυσικός (physikos)*, ‘alamiah’, dan *φύσις (physis)*, ’Alam’) adalah *sains* atau

ilmu tentang alam dalam makna yang terluas. Fisika mempelajari gejala alam yang tidak hidup atau materi dalam lingkup ruang dan waktu. (10)

2.1.4. Pengertian Multimedia

Pengertian multimedia menurut wikipedia.org adalah *multimedia is the use of several different media to convey information (text, audio, graphics, animation, video, and interactivity). Multimedia also refers to computer data storage devices, especially those used to store multimediantent.* (9)

Berdasarkan pengertian di atas, dapat disimpulkan pengertian multimedia adalah penggunaan beberapa media yang berbeda untuk menggabungkan dan menyampaikan informasi dalam bentuk *text*, *audio*, grafik, animasi, dan *video*.

Pengertian multimedia menurut total.or.id adalah berbagai media. Istilah bagi transmisi data dan manipulasi semua bentuk informasi, baik berbentuk kata-kata, gambar, *video*, musik, angka, atau tulisan tangan. (8)

Pengertian multimedia menurut Ariesto Hadi Sutopo adalah multimedia merupakan Perangkat lunak yang digunakan dalam pengembangaan dengan lebih dari satu cara untuk menyampaikan informasi yang menyajikan teks dengan diagram kepada pengguna. (1:251).

2.1.4.1. Komponen Multimedia

Ariesto Hadi Sutopo dalam bukunya menerangkan bahwa multimedia terdiri dari beberapa komponen yaitu :

a. Teks

Teks merupakan dasar dari pengolahan kata dan informasi yang berbasis multimedia. Dengan adanya multimedia maka penyajian informasi kepada pembaca teks dapat disajikan dengan cepat dan menarik karena tidak diperlukan membaca secara rinci dan teliti.

b. Grafik

Secara grafik adalah gambar garis (*Line Drawing*) yang merupakan komponen penting dalam multimedia, dan gambar merupakan salah satu sarana yang sangat baik untuk menyajikan informasi.

c. Image

Secara umum *Image* adalah gambar *raster* (h*alftone drawing*) yang merupakan salah satu komponen penting dalam multimedia, seperti *foto*, basisdata karyawan dengan atribut nama, alamat, dan lainnya supaya lebih efektif bila foto data karyawan tersebut bisa ditampilkan. Demikian juga foto-foto seperti gedung dan lain-lain sangat memerlukan penyimpanan yang besar. Hal inilah yang menyebabkan aplikasi multimedia disimpan dalam penyimpanan yang cukup besar kapasitasnya seperti CD ROM.

d. Animasi

Animasi digunakan untuk menggambarkan objek yang bergerak agar kelihatan hidup. Untuk membuat satu durasi animasi memerlukan jumlah

gambar (*Frame*) yang cukup banyak. Jumlah *frame* tiap detik (*Frame per second* / *fps*) merupakan satuan yang akan menghasilkan kuaalitas animasi. Makin banyak *frame* per detik maka makin baik kualitas animasi yang dihasilkannya.

e. Suara

Suara digunakan untuk lebih memperjelas pengertian yang ditampilkan dengan cara lain. Contohnya, narasi merupakan kelengkapan dari penjelasan yang dilihat melalui dari *video,* selain itu suara dapat memberi kejelasan karakteristik suatu gambar misalnya musik dan suara efek (*sound effect*).

f. Interactive link

Intractive link diperlukan apabila menunjuk pada suatu objek pada *screen* seperti *Button* atau teks supaya bisa mengakses program tertentu dan bisa juga digunakan sebagai alat untuk menggabungkan beberapa elemen multimedia sehingga menjadi sebuah informasi yang terpadu.

2.2. Metode Yang Digunakan

Metodologi yang digunakan adalah *Object Oriented*. dengan menggunakan *Unified Modeling Language (UML)*, Sehingga diharapkan hasil yang akan dikeluarkan tidak jauh dari apa yang telah dirancang.

2.2.1. Konsep Dasar Orientasi Objek (7)

Romi Satria Wahono menjelaskan pengertian *Object Orientation* sebagai berikut :

1. Teknologi baru yang didasarkan pada objek-objek dan kelas–kelas.
2. Cara berfikir untuk mengorganisir *software* sebagai kumpulan objek yang bekerjasama dengan struktur data dan kelakuan *(behaviour)*.
3. Sebuah abstraksi dari dunia nyata berdasarkan objek dan keterkaitan mereka dengan objek lain.

Pengertian Objek itu sendiri menurut Romi harus mencakup :

1. Objek adalah benda atau konsep. Bisa benda dunia nyata atau konsep atau abstraksi dari benda atau konsep dijelaskan sebagai representasi dari *software*.
2. Objek memiliki atribut dan kelakuan (*behavior*).

Sedangkan Kelas sendiri adalah :

1. Kelas adalah deskripsi dari koleksi objek objek dengan kesamaan atribut dan kelakuan.
2. Secara singkatnya, pengertian atau spesifikasi dari kelas mencakup pengertian dari atribut *comprising the state, methode* mengimplementasikan kelakuan (*behavior*), dan bagaimana menangani pembuatan dan penghancuran dari sebuah objek.

2.2.2. Konsep Dasar *Unified Modeling Language* (UML)

Sejarah UML sendiri cukup panjang 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 di*release 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.

Sedangkan Sri Dharwiyanti dan Romi Satria Wahono menjelaskan *Unified Modelling Language* (UML) adalah sebuah "bahasa" yang telah menjadi standar dalam industri untuk visualisasi, merancang dan mendokumentasikan sistem piranti lunak. UML menawarkan sebuah standar untuk merancang model sebuah sistem.

Diagram–diagram yang digunakan untuk medefinisikan UML adalah sebagai berikut :

a. 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 *test case* untuk semua *feature* yang 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 *use case* 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 *common*.

Sebuah *use case* juga dapat meng-*extend use case* lain dengan *behavior*-nya sendiri. Sementara hubungan generalisasi antar *use case* menunjukkan bahwa *use case* yang satu merupakan spesialisasi dari yang lain.

b. Class Dagram

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 (dan *stereotype*).
2. Atribut.
3. Metoda.

Atribut dan metoda dapat memiliki salah satu sifat berikut :

a. *Private*, tidak dapat dipanggil dari luar *class* yang bersangkutan.

b. *Protected*, hanya dapat dipanggil oleh *class* yang bersangkutan dan anak-anak yang mewarisinya.

c. *Public*, dapat dipanggil oleh *class* mana saja.

Class dapat merupakan implementasi dari sebuah *interface*, yaitu *class* abstrak yang hanya memiliki metoda. *Interface* tidak dapat langsung diinstansiasikan, tetapi harus diimplementasikan dahulu menjadi sebuah *class.* Dengan demikian *interface* mendukung resolusi metoda pada saat *run-time*.

Sesuai dengan perkembangan *class* model, *class* dapat dikelompokkan menjadi *package.* Kita juga dapat membuat diagram yang terdiri atas *package*.

Adapun hubungan antar *Class* yang dapat terbentuk antara lain:

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.

d. Statechart Diagram

Statechart diagram menggambarkan transisi dan perubahan keadaan (dari satu *state* ke *state* lainnya) suatu objek pada sistem sebagai akibat dari *stimuli* yang diterima. Pada umumnya *statechart diagram* menggambarkan *class* tertentu (satu *class* dapat memiliki lebih dari satu *statechart diagram*).

Dalam UML, *state* digambarkan berbentuk segiempat dengan sudut membulat dan memiliki nama sesuai kondisinya saat itu. Transisi antar *state* umumnya memiliki kondisi *guard* yang merupakan syarat terjadinya transisi yang bersangkutan, dituliskan dalam kurung siku. *Action* yang dilakukan sebagai akibat dari *event* tertentu dituliskan dengan diawali garis miring. Titik awal dan akhir digambarkan berbentuk lingkaran berwarna penuh dan berwarna setengah.

e. Activity Diagram

Activity diagrams menggambarkan berbagai alir aktivitas dalam sistem yang sedang dirancang, bagaimana masing-masing alir berawal, *decision* yang mungkin terjadi, dan bagaimana mereka berakhir. *Activity diagram* juga dapat menggambarkan proses paralel yang mungkin terjadi pada beberapa eksekusi.

Activity diagram merupakan *state diagram* khusus, dimana sebagian besar *state* adalah *action* dan sebagian besar transisi di-*trigger* oleh selesainya *state* sebelumnya (*internal processing*). Oleh karena itu *activity diagram* tidak menggambarkan *behaviour internal* sebuah sistem (dan interaksi antar subsistem) secara eksak, tetapi lebih menggambarkan proses-proses dan jalur-jalur aktivitas dari level atas secara umum.

Sebuah aktivitas dapat direalisasikan oleh satu *use case* atau lebih. Aktivitas menggambarkan proses yang berjalan, sementara *use case* menggambarkan bagaimana aktor menggunakan sistem untuk melakukan aktivitas.

Sama seperti *state*, standar UML menggunakan segiempat dengan sudut membulat untuk menggambarkan aktivitas. *Decision* digunakan untuk menggambarkan behaviour pada kondisi tertentu. Untuk mengilustrasikan proses-proses paralel (*fork* dan *join*) digunakan titik sinkronisasi yang dapat berupa titik, garis horizontal atau vertikal.

Activity diagram dapat dibagi menjadi beberapa *object swimlane* untuk menggambarkan objek mana yang bertanggung jawab untuk aktivitas tertentu.

f. 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.

Untuk objek-objek yang memiliki sifat khusus, standar UML mendefinisikan *icon* khusus untuk objek *boundary, controller* dan *persistent entity*.

g. Collaboration Diagram

Collaboration diagram juga menggambarkan interaksi antar objek seperti *sequence diagram*, tetapi lebih menekankan pada peran masing-masing objek dan bukan pada waktu penyampaian *message*. Setiap *message* memiliki *sequence number*, dimana *message* dari level tertinggi memiliki nomor 1. Messages dari level yang sama memiliki *prefiks* yang sama.

h. Component Diagram

Component diagram menggambarkan struktur dan hubungan antar komponen piranti lunak, termasuk ketergantungan (*dependency*) di antaranya. Komponen piranti lunak adalah modul berisi *code*, baik berisi *source code* maupun *binary code*, baik *library* maupun *executable*, baik yang muncul pada *compile time, link time*, maupun *run time*. Umumnya komponen terbentuk dari beberapa *class* dan/atau *package*, tapi dapat juga dari komponen-komponen yang lebih kecil. Komponen dapat juga berupa *interface*, yaitu kumpulan layanan yang disediakan sebuah komponen untuk komponen lain.

i. Deployment Diagram

Deployment/physical diagram menggambarkan detail bagaimana komponen di-*deploy* dalam infrastruktur sistem, di mana komponen akan terletak (pada mesin, *server* atau piranti keras), bagaimana kemampuan jaringan pada lokasi tersebut, spesifikasi *server*, dan hal-hal lain yang bersifat fisikal.

Sebuah *node* adalah *server*, *workstation*, atau piranti keras lain yang digunakan untuk men-*deploy* komponen dalam lingkungan sebenarnya. Hubungan antar *node* (misalnya TCP/IP) dan *requirement* dapat juga didefinisikan dalam diagram ini.

2.2.3. Macromedia Flash

Pada saat ini, animasi sudah menjadi elemen pentinh yang mendukung tampilan halaman *Web,* presentasi, dan yang lainnya. Hal ini dikarenakan animasi dapat memfokuskan pandangan auden pada tampilannya. Sebagai contoh, di dalam suatu halaman *Web* yang disisipkan animasi logo yang berputar, atau iklan dengan teks yang diberikan animasi, tentunya dapat menarik pandangan mata audien pada objek yang dianimasi tersebut.

Program *Macromedia flash MX* adalah suatu program yang sangat ideal untuk pembuatan animasi. Program ini memberikan fasilitas dan tool yang efektif dan fleksibel untuk menghasilkan animasi yang menarik.

Movie Macromedia Flash MX (untuk selanjutnya disebut dengan movie Flash) terdiri atas grafik, teks, animasi dan aplikasi untuk *Web site. Movie Flash* lebih cenderung berisi grafik vektor, namun demikian tidak menutup kemungkinan untuk disisipkan *image bitmap*, *video* maupun *sound*.

Pada umumnya, para *Web designer* menggunakan *Flash* untuk membuat kontrol navigasi, animasi logo, maupun animasi lainnya yang disinkronisasikan dengan *sound*. Selain itu, banyak juga *Web designer* yang menggunakan *Flash* untuk membuat Web site secara keseluruhan, dalam arti total pembuatannya hanya menggunakan program *Macromedia Flash*.

Dokumen *Flash* mempunyai ekstensi file.*fla*. Dokumen *Flash* berisi semua informasi yang dibutuhkan untuk membuat suatu desain. Dokumen *Flash* tidak ditampilkan di dalam *Flash Player*. Hasil pembuatan suatu desain di dalam dokumen Flash harus di-publish menjadi *movie Flash* (dengan ekstensi file.swf) agar bisa ditampilkan di dalam *Flash Player*.