BAB II
LANDASAN TEORI

II.1 *Stress*

II.1.1 Pengertian *Stress*

Stress dalam istilah biologi memiliki pengertian suatu konsekuensi dari kegagalan tubuh seorang manusia atau seekor hewan untuk merespon secara tepat terhadap ancaman yang datang baik fisik maupun mental, baik yang benar-benar terjadi maupun hanya imajinasi. Sedangkan suatu kejadian eksternal yang menyebabkan respon-respon *stress* internal, baik secara fisik maupun emosi tersebut disebut dengan *stressor*.

Kata *stress* pertama kali digunakan oleh seorang *endocrinologist* bernama Hans Selye pada tahun 1930 untuk mengidentifikasi respon fisiologis dalam laboratorium hewan. Hans Selye akhirnya mendefinisikan *stress* sebagai suatu cara yang dilakukan tubuh untuk merespon terhadap suatu perubahan. Persepsi terhadap perubahan tersebut menentukan jenis respon. *Stress* menjadi positif ketika seseorang merasa terstimulasi dan mampu untuk me-*manage* situasi (*eustress*), namun *stress* menjadi negatif ketika seseorang merasa terancam dan tidak bisa mengontrol situasi (*distress*).

Beberapa pengertian stres lain yang telah dikemukakan oleh beberapa ahli adalah sebagai berikut :

1. Lazarus (1984)

   Stress psikologi adalah hubungan spesifik antara individu dengan lingkungan yang dinilai oleh individu sebagai tuntutan atau melebihi sumber dayanya dan membahayakan keberadaannya (*well-being*).

2. Lazarus (1976)

   Stress muncul ketika ada tuntutan-tuntutan terhadap pribadi seseorang yang membebani atau melampaui batas sumber daya yang diperlukan atau dimiliki dalam menyesuaikan dirinya.

3. Walter Cannon (1932)

   Stress sebagai suatu keadaan keseimbangan yang terganggu, dan orang yang mengalami *stress* sebagai dibawah tekanan (*under stress*).

Dengan demikian, *stress* secara umum dapat didefinisikan sebagai suatu ketidakseimbangan dalam tubuh yang menimbulkan ketidaknyamanan yang merupakan akibat dari adanya tuntutan-tuntutan terhadap individu yang membebani atau melampaui batas sumber dayanya atau kemampuannya[4].

II.1.2 Dampak yang Ditimbulkan Akibat *Stress*

Sarafino menjabarkan tentang dua aspek utama dari dampak yang ditimbulkan akibat *stress* yang terjadi pada manusia, yaitu[5]:

1. Aspek Biologis

   Beberapa gejala fisik yang dirasakan ketika seseorang sedang mengalami *stress,* diantaranya adalah sakit kepala yang berlebihan, tidur menjadi tidak nyenyak, gangguan pencernaan, hilangnya nafsu makan, gangguan kulit, dan produksi keringat yang berlebihan di seluruh tubuh.

2. Aspek Psikologis

   Terdapat tiga gejala psikologis yang dirasakan ketika seseorang sedang mengalami *stress,* diantaranya :

   a. Gejala Kognisi

      Gejala-gejala yang muncul pada aspek kognisi seperti menurunnya daya ingat, perhatian dan konsentrasi yang berkurang sehingga seseorang tidak fokus dalam melakukan sesuatu hal.

   b. Gejala Emosi

      Gejala-gejala yang muncul pada aspek emosi seperti mudah marah, kecemasan yang berlebihan terhadap segala sesuatu, merasa sedih, dan depresi.

   c. Gejala Tingkah Laku

      Gejala-gejala yang muncul pada aspek tingkah laku seperti mudah menyalahkan dan mencari kesalahan orang lain, melanggar norma

karena tidak bisa mengontrol perbuatannya, bersikap tak acuh pada lingkungannya, serta suka melakukan penundaan pekerjaan.

II.1.3 Faktor-faktor yang Mempengaruhi *Stress*

Gunawan, Hartati, dan Listiara menjelaskan tentang faktor-faktor yang mempengaruhi *stress* pada mahasiswa yang sedang menyusun skripsi, diantaranya[5]:

1. Faktor Internal
   a. Jenis Kelamin

      Penelitian yang dilakukan di Amerika menyatakan bahwa wanita cenderung memiliki tingkat *stress* yang lebih tinggi 30 % dibandingkan pria.

   b. Status Sosial Ekonomi

      Seseorang yang memiliki status sosial ekonomi menengah ke bawah cenderung memiliki tingkat *stress* yang tinggi disebabkan adanya tekanan dalam hidup.

   c. Karakteristik Kepribadian

      Karakteristik kepribadian mahasiswa yang berbeda-beda menyebabkan adanya perbedaan reaksi terhadap sumber *stress* yang sama.

   d. *Coping Strategy*

      Coping Strategy yang digunakan oleh mahasiswa yang sedang menyusun skripsi berpengaruh pada tingkat *stress*-nya.

   e. Suku dan Kebudayaan

      Stuart dan Sundeen (1991) mencoba menjelaskan bahwa kebudayaan berpengaruh terhadap gangguan psikis seseorang dikarenakan setiap suku memiliki metode penyelesaian masalah yang berbeda.

   f. Intelegensi

      Mahasiswa yang mempunyai tingkat intelegensi yang lebih tinggi cenderung lebih adaptif terhadap perubahan lingkungannya.

2. Faktor Eksternal
   a. Tuntutan Tugas Akademik (Skripsi)
      Seseorang yang menganggap skripsi merupakan beban dan berpikir bahwa tugas tersebut tidak sesuai dengan kemampuan yang ada di dalam dirinya, maka mahasiswa tersebut cenderung mengalami *stress*.
   b. Hubungan Mahasiswa dengan Lingkungan Sosialnya
      Hubungan mahasiswa dengan lingkungan sosialnya meliputi dukungan dari orang tua, teman, dan para dosen. Dukungan sosial mempengaruhi motivasi mahasiswa dalam menyusun skripsi dan dapat mengurangi *stress* individual yang terjadi pada mahasiswa.

II.1.4 *Coping Strategy*

Stres yang negatif memiliki banyak pengaruh yang buruk terhadap kondisi tubuh baik secara fisik maupun mental sehingga diperlukan suatu cara untuk menanggulanginya. *Coping strategy* merupakan salah satu teknik untuk menanggulangi *stress.* Terdapat dua bentuk pendekatan dasar di dalam *coping strategy* diantaranya[12] *:*

1. *Problem-Focused Coping*
   Strategi *problem-focused coping* lebih diarahkan kepada menyelesaikan masalah yang menjadi sumber *stress* atau *stressor.*
2. *Emotion-Focused Coping*
   Strategi *emotion-focused coping* lebih diarahkan kepada mengelola emosi dengan cara mengidetifikasi perasaan yang ada, fokus terhadap perasaan yang telah diidentifikasi kemudian mengelola perasaan tersebut.

Secara umum, *problem-focused coping* dapat digunakan dengan baik ketika terdapat beberapa aksi konkret yang dapat dilakukan untuk mengurangi *stressor*. Di sisi lain, *emotion-focused coping* dapat membantu ketika sebuah situasi harus diterima secara sederhana atau pada saat emosi butuh untuk dikelola, sebelum akhirnya dapat berpikir dengan jernih kemudian beraksi secara rasional.

Namun terkadang, kedua bentuk *coping* tersebut juga dapat bekerja secara bersamaan dengan baik. Contohnya adalah ketika seseorang dipecat dari pekerjaannya, dia mungkin mulai mencari pekerjaan lain (*problem-focused*) tetapi dia tidak dapat fokus pada kegiatan tersebut karena sangat marah dan kebingungan setelah dipecat dari pekerjaannya. Dalam situasi tersebut, dia sebaiknya mencoba *emotion-focused coping* untuk membantu menenangkan diri dan berpikir lebih jernih. Dia mungkin dapat berolahraga, berbincang dengan sahabat, menulis jurnal, atau melakukan kegiatan yang dapat mengelola perasaannya.

Alternatif lain, dia mungkin mandi air panas, beristirahat, atau makan makanan bergizi. *Emotion-focused coping* yang seperti itu bukanlah sebuah *defense* (menjauhkan diri sepenuhnya dari masalah), tetapi lebih kepada memproses respon emosi sebelum semuanya berada diluar kendali dan merusak kesehatan. Kemudian, setelah dirasakan cukup tenang dan siap, dia dapat berkonsentrasi pada langkah apa yang harus dilakukan terhadap *stressor* dan kemudian menyelesaikannya.

II.2 *Game*

II.2.1 Teori Bermain

Kata *play* pada *game playing* mempunyai arti bermain. Terdapat banyak sekali teori yang dikemukakan para pakar mengenai kegiatan bermain yang kemudian dibagi kedalam dua kelompok besar sebagai berikut [19]:

1. Teori Klasik
   a. *Surplus Energy* (Schiller 1873 Spencer 1875)
      Bermain merupakan hasil dari kelebihan energi yang ada pada kaum muda yang dibebaskan dari persiapan diri melalui kegiatan orang tua mereka.
   b. *Relaxation/Recreation Theory* (Lazarus 1883 Patrick 1916)
      Bermain dipandang sebagai kegiatan yang berlawanan dengan kerja dan kesungguhan hidup, permainan merupakan imbangan antara kerja dan istirahat. Orang yang merasa penat akan bermain

untuk mengadakan pelepasan agar dapat mengembalikan kesegaran jasmani dan rohani.

c. *Pre-exercise Theory* (Groos 1898)

Bermain merupakan suatu latihan perilaku yang penting bagi kelangsungan hidup nantinya. Permainan kasar yang dilakukan oleh hewan atau anak-anak pada dasarnya merupakan latihan keterampilan yang nantinya akan membantu kelangsungan hidup mereka.

d. *Recapitulation Theory* (G'Stanley Hall 1906 Wundt 1913)

Bermain dipandang bukan sebagai kegiatan yang mengembangkan keterampilan insting masa depan, melainkan berfungsi untuk membersihkan organisme dari insting primitif dan tidak perlu dibawa oleh keturunannya.

e. *Growth Theories* (Appleton 1919)

Bermain merupakan sebuah respon untuk pertumbuhan di dalam organisme. Bermain berfungsi untuk memfasilitasi penguasaan keterampilan yang diperlukan untuk fungsi perilaku dewasa.

f. *Ego Expanding Theories* (Lange 1902 Claparde 1911)

Bermain merupakan cara alami untuk menyelesaikan ego yang ekspresif dari seluruh kepribadian, sebuah latihan yang mengembangkan keterampilan kognitif dan membantu munculnya keterampilan tambahan.

2. Teori Saat ini

a. *Infantile Dynamic* (Lewin)

Bermain terjadi karena ruang kehidupan kognitif anak masih tidak terstruktur, sehingga gagal untuk membedakan antara yang nyata dan tidak nyata. Anak masuk kedalam wilayah menyenangkan yang tidak nyata dimana hal-hal yang berubah sewenang-wenang.

b. *Cathartic Theory* (Freud 1908)

Bermain merupakan upaya untuk memenuhi kebutuhan atau menyelesaikan konflik ketika anak-anak benar-benar tidak memiliki sarana untuk melakukannya.

c. *Psychoanalytic Theory* (Buhler 1930 Anna Freud 1937)

Bermain tidak hanya merupakan sebuah keinginan untuk memenuhi suatu kecenderungan, tetapi juga penguasaan (upaya berulang untuk mengatasi situasi kecemasan besar yang memprovokasi. Bermain bersifat defensif dan adaptif dalam menghadapi kecemasan.

d. *Cognitive Theory* (Piaget 1962)

Bermain berasal dari anak yang bekerja dari dua karakteristik fundamental yakni pengalaman dan pengembangan. Upaya untuk mengintegrasikan pengalaman baru ke dalam jumlah motorik yang relative terbatas dan keterampilan kognitif yang tersedia pada usia masing-masing.

II.2.2 Pengertian *Game*

Game playing merupakan salah satu bidang AI yang sangat populer berupa permainan antara manusia melawan mesin yang memiliki intelektual untuk berpikir[3]. *Game* mempunyai banyak variasi jenis, dalam konteks *game story game* dibagi menjadi dua kategori diantaranya[2]:

1. *Game type* merupakan pengklasifikasian *game* berdasarkan deskripsi dari *game play* contohnya, *action, adventure, puzzle, role playing, simulation,* dan *strategy.*
2. *Game genre* merupakan pengklasifikasian *game* berdasarkan deskripsi dari konten yang bersifat naratif dan bergantung pada elemen *storytelling* dari suatu *game* contohnya, drama, kriminal, fantasi, horror, misteri, fiksi ilmiah, dan lain-lain.

Game memiliki berbagai fungsi selain fungsi utamanya sebagai media hiburan, fungsi tersebut bergantung pada tujuan untuk apa sebuah *game* dibangun

misalnya fungsi sebagai media pembelajaran yang bertujuan untuk membantu meningkatkan pengetahuan *user* atau sebagai media promosi komersial ataupun budaya yang bertujuan untuk membantu memperkenalkan suatu produk ataupun meningkatkan penjualan. Selain itu, g*ame* juga dapat digunakan untuk meringankan stres yang diklasifikasikan dengan kriteria sebagai berikut[18] :

1. Mudah untuk Dimenangkan

   Seseorang yang berada dibawah tekanan sangat sensitif terhadap kegagalan. Mudah untuk dimenangkan tidak serta merta menjadikan *game* menjadi tidak menarik, tetapi memiliki pengertian bahwa *game* dapat dimainkan sampai perasaan terpuaskan.
2. Desain yang Sederhana

   Kombinasi warna sangatlah penting dalam hal untuk tidak mendapatkan iritasi dan membuat suatu kenyamanan. Selain itu, *game* yang tidak terlalu cepat pergerakannya juga berkontribusi untuk membuat perasaan yang tenang dan bebas dari tekanan
3. Musik yang Membuat Nyaman

   Musik merupakan sebuah alat komunikasi, sebuah bahasa. Terlepas dari bahasa sehari-hari yang digunakan dan tempat dimana kita tinggal, kenyataannya adalah musik dapat dirasakan. Terlepas dari disadari atau tidak, musik sangat berpengaruh terutama limbik, sebuah bagian otak yang bertanggung jawab terhadap kondisi fisik tubuh seperti menjaga keseimbangan internal kehidupan.
4. Tanpa Agresi

   Beberapa *game* yang ditawarkan untuk meringankan stres adalah dengan kekerasan dan agresi, misalnya *game* yang memiliki misi untuk menghancurkan sesuatu. Kebanyakan orang berpendapat bahwa *game* yang seperti demikian tidaklah berbahaya, namun beberapa studi menunjukan *game* tersebut dapat meningkatkan reaksi amarah yang dapat memperburuk keadaan daripada membantu membuat perasaan nyaman.

5. Peraturan yang Sederhana

   Peraturan yang sederhana sangat penting untuk meringankan tekanan secara efektif. Terdapat banyak sekali *game* yang menarik namun membutuhkan waktu yang cukup lama untuk mempelajari seluruh peraturan sebelum memainkan *game.* Seringkali, ini tidak memungkinkan dan bahkan direkomendasikan untuk seseorang yang sepanjang hari ditugaskan untuk menyelesaikan masalah yang berat dalam pekerjaannya.

II.2.3 *Collision Detection*

Collision Detection merupakan komponen penting dari *game engines* yang dirancang untuk memberikan animasi interaksi objek yang realistis antara pemain dengan lingkungan permainan. Secara fisik, simulasi realistis dinamis seperti simulator penerbangan dan *simulator mobile robot* juga memerlukan algoritma *collision detection* yang efisien. Tes *intersection* merupakan tulang punggung dari algoritma *collision detection*. *Collision detection* yang terdiri dari beberapa objek membutuhkan suatu metode yang efisien untuk meminimalkan jumlah tes *intersection*.

Video *game* harus membagi waktu komputasinya yang sangat terbatas di antara beberapa *task*. Meskipun sumber daya dibatasi, penggunaan algoritma *collision detection* yang relatif primitif cukup dapat dipercaya untuk digunakan dalam *game.* Untuk waktu yang lama, video *game* memiliki jumlah objek yang sangat terbatas untuk digunakan, dan memeriksa semua pasangan tidaklah masalah.

Pada beberapa kasus dalam *game* 2D, perangkat keras mampu mendeteksi secara efisien *overlapping pixel* antar *sprite* di layar. Pada kasus lain, cukup melakukan *tilling* layar lalu *binding* setiap *sprite* ke dalam *tile* yang *overlap*, sementara itu untuk pemeriksaan berpasangan, *bounding* persegi panjang atau lingkaran yang disebut *hitboxes* telah digunakan dan dianggap cukup akurat.

Hampir semua *game* menggunakan *collision detection* dan suatu tabrakan sering diselesaikan dengan menggunakan aturan yang sangat sederhana. Misalnya,

jika sebuah karakter menabrak dinding, karakter tersebut mungkin hanya pindah kembali ke lokasi terakhir yang diketahui cukup baik. Beberapa *game* akan menghitung berapa jarak karakter dapat bergerak sebelum menabrak dinding dan hanya memungkinkan untuk bergerak sejauh itu.

Sebuah *simulator* yang baik adalah *simulator* yang bereaksi terhadap setiap *input* secara wajar. Misalnya, jika membayangkan sebuah *video game* mobil balap berkecepatan tinggi, dari satu langkah simulasi ke langkah berikutnya, dapat dibayangkan bahwa mobil akan maju sepanjang trek balap. Jika terdapat hambatan di trek, misalnya dinding bata tidak seharusnya mobil melompat di atasnya. *Big Rigs: Over the Road Racing* adalah contoh *game* yang tidak terkenal karena memiliki sistem *collision detection* yang gagal atau bahkan tidak memiliki sistem *collision detection*[13].

II.3 *Tools* yang digunakan

II.3.1 *Object Oriented Programming*

Object Oriented Programming (OOP) merupakan paradigma pemrograman yang berorientasikan kepada objek. Semua data dan fungsi di dalam paradigma ini dibungkus ke dalam kelas-kelas atau objek-objek. Kelebihan penggunaan model data berorientasi objek adalah memberikan fleksibilitas, kemudahan mengubah program, dan digunakan luas dalam teknik piranti lunak skala besar.

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.

Pemrograman berorientasi objek menekankan konsep berikut:

1. Kelas (*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 *behavior* atau perilaku 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.

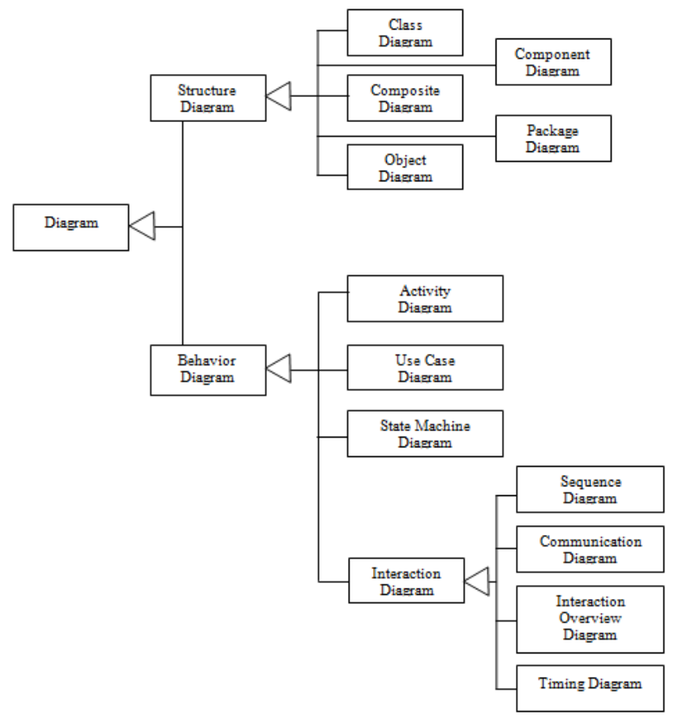
2. Objek (*Object*) - membungkus data dan fungsi bersama menjadi suatu unit dalam sebuah program komputer. Objek merupakan dasar dari modularitas dan struktur dalam sebuah program komputer berorientasi objek.
3. Abstraksi (*Abstract*) - Kemampuan sebuah program untuk melewati aspek informasi yang diproses olehnya, yaitu kemampuan untuk memfokus pada inti. Setiap objek dalam sistem melayani sebagai model dari "pelaku" abstrak yang dapat 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. Enkapsulasi (*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. Polimorfisme (*Polimorphism*) 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 pesan tersebut dikirim. Contohnya, bila sebuah burung menerima pesan "gerak cepat", dia akan menggerakan sayapnya dan terbang. Bila seekor singa menerima pesan yang sama, dia akan menggerakkan kakinya dan berlari. Keduanya menjawab sebuah pesan yang sama, namun yang sesuai dengan kemampuan hewan tersebut. Ini disebut polimorfisme karena sebuah variabel tungal dalam program dapat memegang berbagai jenis objek yang berbeda selagi program berjalan, dan teks program yang sama dapat memanggil beberapa metode yang berbeda di saat yang berbeda dalam pemanggilan yang sama. Hal ini berlawanan dengan bahasa fungsional yang mencapai polimorfisme melalui penggunaan fungsi kelas-pertama.
6. Inheritas (*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) *behavior* mereka tanpa harus mengimplementasi ulang *behavior* tersebut (bahasa berbasis-objek tidak selalu memiliki inheritas).[8]

II.3.2 *Unified Modeling Language*

Unified Modelling Language (UML) adalah keluarga notasi grafis yang didukung oleh meta-model tunggal, yang membantu pendeskripsian dan desain sistem perangkat lunak, khususnya sistem yang dibangun menggunakan pemograman berorientasi objek (OO).

UML 2 terdiri dari 13 jenis diagram resmi, klasisfikasi jenis diagram UML dapat dilihat pada Gambar II.1.



Gambar II.1 Klasifikasi Jenis Diagram UML[1]

Sedangkan pengertian dari masing-masing diagram adalah sebagai berikut:

1. *Class Diagram* mendeskripsikan jenis-jenis objek dan berbagai macam hubungan statis yang terdapat diantara mereka. Class diagram juga menunjukkan properti dan operasi sebuah class dan batasan-batasan yang terdapat dalam hubungan-hubungan objek tersebut.
2. *Component Diagram* digunakan saat akan memecah sistem menjadi komponen-komponen dan ingin menampilkan hubungan-hubungan mereka dengan antarmuka atau pemecahan komponen menjadi struktur yang lebih rendah.
3. *Composite Diagram* memiliki kemampuan untuk secara hirarkis memecah sebuah class menjadi struktur internal. Hal ini memungkinkan untuk mengambil sebuah objek kompleks dan

memecahnya menjadi beberapa bagian. *Composite Diagram* ini merupakan cara terbaik untuk membedakan antara paket-paket dengan kelompok yang sedang berjalan.

4. *Deployment Diagram* menunjukkan susunan fisik sebuah sistem, menunjukkan bagian perangkat lunak mana yang berjalan pada perangkat keras mana.
5. *Package Diagram* adalah sebuah bentuk pengelompokan yang memungkinkan untuk mengambil setiap bentuk di UML dan mengelompokkan elemen-elemennya dalam tingkatan unit yang lebih tinggi namun kegunaannya yang paling umum adalah untuk mengelompokkan class.
6. *Object Diagram* merupakan sebuah gambaran tentang objek-objek dalam sebuah sistem pada satu titik waktu. Karena lebih menonjolkan perintah-perintah daripada class, object diagram lebih sering disebut sebagai sebuah diagram perintah.
7. *Activity Diagram* adalah teknik untuk menggambarkan logika prosedural, proses bisnis, dan jalur kerja. Dalam beberapa hal, diagram ini memainkan peran mirip sebuah diagram alir, tetapi perbedaan prinsip antara diagram ini dan notasi diagram alir adalah diagram ini mendukung *behavior* paralel.
8. *Use Case* adalah teknik untuk merekam persyaratan fungsional sebuah sistem. Use Case mendeskripsikan interaksi tipikal antara para pengguna sistem dengan sistem itu sendiri, dengan member sebuah narasi tentang bagaimana sistem tersebut digunakan.
9. *State Machine Diagram* adalah teknik yang umum digunakan untuk menggambarkan *behavior* sebuah sistem.
10. *Sequence Diagram* menjabarkan sebuah skenario tunggal. Diagram tersebut menunjukkan sejumlah objek contoh dan pesan-pesan yang melewati objek-objek ini di dalam use case.

11. *Communication Diagram* merupakan macam dari *interaction diagram*, memberi tekanan pada hubungan-hubungan data antar partisipan yang berbeda dalam sebuah interaksi.
12. *Interaction Overview Diagram* merupakan hasil pencakokan dari *activity diagram* dan *sequence diagram.*
13. *Timing Diagram* merupakan bagian lain *interaction diagram*, di mana fokusnya adalah batasan waktu untuk sebuah objek tunggal ataupun, menjadi lebih berguna, untuk sekelompok objek.[1]

II.3.3 Actionscript 3.0

ActionScript merupakan bahasa pemrograman yang berjalan di lingkungan Adobe Flash Player and Adobe AIR. ActionScript memungkinkan interaksi, *data handling,* dan lain-lain pada Flash, Flex, serta AIR konten dan aplikasi. ActionScript dieksekusi oleh sebuah ActionScript Virtual Machine (AVM), yang merupakan bagian dari Flash Player dan AIR. Kode ActionScript secara khusus di-*compile* ke dalam *bytecode format* (suatu bahasa pemrograman yang ditulis dan dimengerti oleh komputer) oleh sebuah *compiler*, saling melengkapi seperti yang dibangun ke dalam Adobe Flash CS4 Professional atau Adobe Flex Builder, atau yang tersedia pada Adobe Flex SDK. *Bytecode* tertanam di dalam file SWF, yang mana dieksekusi oleh Flash Player and AIR.

ActionScript 3.0 melampaui kemampuan *scripting* dari versi yang sebelumnya. Hal ini dirancang untuk memfasilitasi pembuatan aplikasi yang sangat kompleks dengan set data yang besar dan berorientasi objek, basis kode yang dapat digunakan kembali. Selama ActionScript 3.0 tidak diperlukan untuk konten yang berjalan di Adobe Flash Player, ini akan membuka pintu untuk perbaikan kinerja yang hanya tersedia dengan AVM2, sebuah mesin virtual baru. ActionScript 3.0 dapat mengeksekusi kode hingga sepuluh kali lebih cepat dari kode ActionScript sebelumnya.

Versi yang lebih tua dari ActionScript Virtual Machine, AVM1, mengeksekusi kode ActionScript 1.0 dan ActionScript 2.0. AVM1 didukung oleh

Flash Player 9 dan 10 untuk kompatibilitas dengan konten yang sedang berjalan dan konten sebelumnya.

ActionScript 3.0 menawarkan sebuah model pemrograman kokoh yang akan familiar dengan *developer* yang memiliki pengetahuan dasar dengan konsep *object-oriented programming*. Beberapa kelebihan fitur yang dimiliki oleh ActionScript 3.0 dibandingkan versi sebelumnya adalah sebagai berikut :

1. Sebuah ActionScript Virtual Machine baru, dinamakan AVM2 yang menggunakan sebuah set instruksi *bytecode* baru, menyediakan perbaikan kinerja yang signifikan.
2. *Compiler* basis kode yang lebih modern, bekerja dengan optimisasi lebih dalam daripada *compiler* pada versi sebelumnya.
3. Perluasan dan perbaikan *application programming interface* (API), dengan *low-level control* dari objek dan sebuah model *object-oriented* yang benar.
4. Sebuah XML API berdasarkan pada ECMAScript untuk spesifikasi XML (E4X) (ECMA-357 edisi 2). E4X merupakan ekstensi bahasa untuk ECMAScript yang mengikutsertakan XML sebagai sebuah tipe data asli dari sebuah bahasa.
5. Sebuah model event yang berdasarkan pada *Document Object Model* (DOM) Level 3 *Events Specification.*[7]

II.3.4 Adobe Flash Profesional CS5.5

Adobe Flash merupakan program pembuat animasi yang diproduksi oleh perusahaan perangkat lunak dari Amerika Serikat, yaitu Adobe System Incorporated. Program ini sangat andal dan popular di kalangan animator. Berbagai fasilitas dan fitur terbaru telah disediakan untuk kemudahan dalam pengolahan animasi bagi para penggunanya. Program aplikasi terbaru Adobe adalah Adobe Flash CS 5.5 (Creative Suite 5.5), yang merupakan penyempurnaan dari versi sebelumnya.

Berikut adalah beberapa fitur-fitur terbaru untuk Flash Professional CS5.5:

1. Skala konten ketika mengubah ukuran *Stage*

   Ketika mengubah ukuran *Stage* melalui kotak dialog *Document Properties*, secara otomatis Anda dapat skala konten agar sesuai dengan ukuran *Stage* baru.

2. Ekspor sebagai *bitmap*

   Simbol berbasis vektor dapat diekspor sebagai *bitmap* ketika mem-*publish* sebuah file SWF sebagai cara untuk mengurangi tuntutan CPU selama *playback*. Fitur ini sangat membantu untuk *publishing* pada perangkat mobile dengan CPU yang kurang kuat.

3. *Convert* ke *bitmap*

   Fitur ini memungkinkan untuk membuat *bitmap* di *Library* dari contoh simbol. *Bitmap* yang baru dapat berguna dalam versi terpisah dari proyek untuk mobile atau perangkat dengan kinerja yang lebih rendah.

4. *Code Snippets panel* HUD

   Code Snippets panel heads-up display (HUD) yang baru memungkinkan untuk melihat kode ActionScript dan deskripsi masing-masing snippet sebelum memasukkannya serta dapat dilakukan *drag-and-drop* referensi contoh di layar HUD ke kode contoh pada *Stage*.

5. *Auto Recover* dan *Auto Save*

   Auto Recover memungkinkan Flash untuk secara berkala melakukan *snapshot* semua dokumen yang terbuka sehingga pengguna dapat melakukan *recover* dari peristiwa kehilangan data secara tiba-tiba. *Auto Save* membantu pengguna secara berkala menyimpan setiap dokumen.

6. Tambahan kompilasi

   Ketika menggunakan perintah *Publish*, Flash Pro sekarang melalukan *cache* versi terkompilasi dari aset dalam file FLA untuk meningkatkan

kinerja. Setiap kali membuat file SWF, hanya item berubah yang dikompilasi ulang.[10]

II.3.5 Adobe Photosop CS 4

Adobe Photoshop adalah program *editing* grafis yang dikembangkan dan diterbitkan oleh Adobe Systems Incorporated. Pada tahun 2003, Adobe *"Creative Suite" rebranding* menyebabkan penggantian nama Adobe Photoshop 8 untuk Adobe Photoshop CS. Dengan demikian, Adobe Photoshop CS4 merupakan rilis utama ke-12 dari Adobe Photoshop. Adobe Photoshop dirilis dalam dua edisi: Adobe Photoshop, dan Adobe Photoshop Extended, dengan Extended yang memiliki ekstra penciptaan gambar 3D, *motion graphics editing*, dan fitur *advanced image analysis*. Adobe Photoshop Extended sudah termasuk dalam semua penawaran Adobe Creative Suite kecuali Design Standard, yang termasuk dalam edisi Adobe Photoshop.

Fitur CS4 *panning* dan *zooming* yang lebih halus, memungkinkan *editing* gambar yang lebih cepat dengan perbesaran tinggi. Antarmuka ini lebih disederhanakan dengan antarmuka berbasis *tab* sehingga lebih nyaman ketika digunakan. Fitur Photoshop CS4 dengan 3D *engine* baru memungkinkan konversi *gradient maps* untuk objek 3D, menambahkan kedalaman lapisan dan teks, dan mendapatkan kualitas cetak *output* dengan *ray-tracing rendering engine* baru.

Mendukung format 3D umum, *Adjusment* baru dan *Mask Panel*, *Content-aware scaling* (*seam carving*), *Fluid Canvas Rotation* dan *File display option. Content-aware scaling* memungkinkan pengguna untuk secara cerdas mengukur dan skala gambar, dan alat *Canvas Rotation* membuatnya lebih mudah untuk dirotasi dan mengedit gambar dari sudut manapun.