BAB II
LANDASAN TEORI

Pada Bab II, Landasan Teori akan membahas tentang *game*, kubik Rubik, perangkat lunak pendukung software, pengembangan berorientasi objek dan UML.

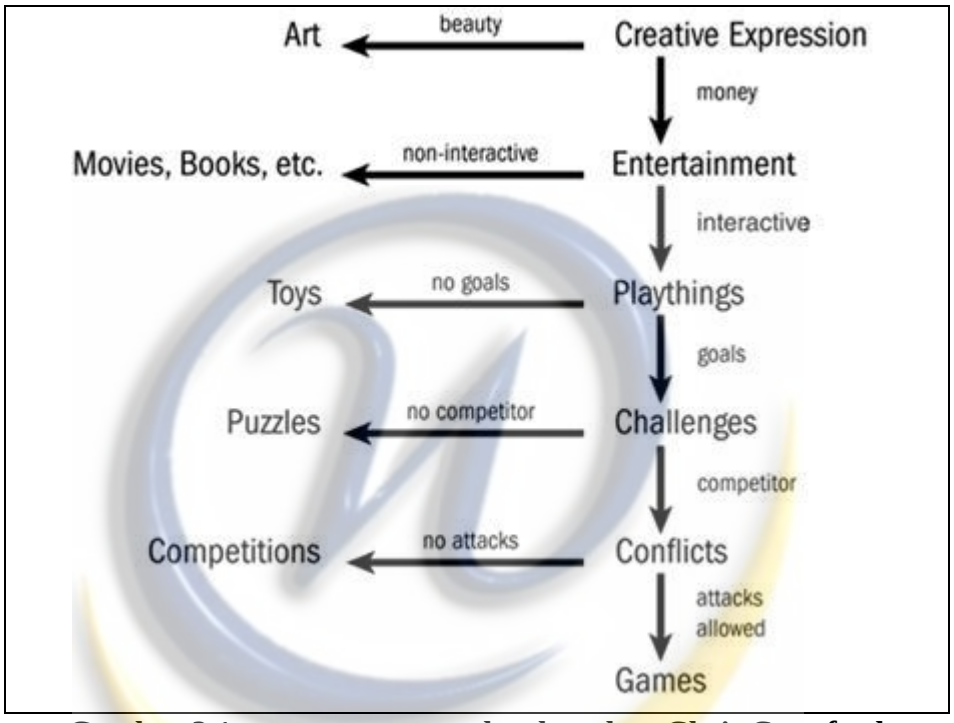
2.1 *Game* [1][2][3]

Kata *"**Game**"* tidak terlepas dari pengertian permainan. Kita mengenal permainan sejak usia balita sampai saat ini. Seiring dengan perkembangan jaman khususnya dalam bidang informasi mengakibatkan jenis permainan semakin berkembang puncaknya dapat kita lihat saat ini telah berkembang permainan menggunakan alat-alat elektronika seperti *GameBoy, Play Station, computer* bahkan alat komunikasi seperti *handphone*.

Chris Crawford (Desainer *game computer*) dalam *Chris Crawford on game design* mencoba mendefinisikan istilah *game* menggunakan runtunan dikotomi:

1. Hasil ekspresi kreatif atau seni, jika dibuat untuk menghasilkan keindahan, dan disebut *entertainment* jika dibuat dengan tujuan: untuk menghasilkan uang.
2. Sebuah hal yang dapat dimainkan (*plaything)*, jika interaktif. Film dan buku merupakan contoh entertainment yang termasuk dalam kategori *non-interactive entertainment*.
3. Jika tidak memiliki *goal* yang diasosiasikan sebagai *plaything*, hal itu disebut *toy* . Jika *plaything* memiliki *goal*, maka termasuk dalam kategori *challenge*.
4. Jika sebuah *challenge* tidak memiliki agen yang aktif yang bertindak sebagai lawan dalam berkompetisi, maka disebut sebagai *puzzle*. Jika memiliki agen yang aktif, disebut konflik.

5. Jika *player* (pemain) hanya dapat mengungguli lawan tanpa menyerangnya untuk mengganggu performa lawan, konflik ini disebut kompetisi. Jika penyerangan diizinkan maka konflik ini dapat dikategorikan sebagai sebuah *game*.



Gambar 2.1 perunutan *games* berdasarkan Chris Crawford

Berdasarkan pendefinisian *game* oleh Chris Crawford maka *game* dapat diuraikan sebagai : sebuah interaktif, berorientasi pada pencapaian *goal,* agen yang aktif sebagai lawan ataupun *player* lain yang bisa saling berinteraksi satu dengan yang lainnya. Selain pendefinisian di atas, Crawford juga memberikan beberapa alternatif pendefinisian tentang istilah *game*:

1. "Sebuah bentuk permainan dengan goal dan struktur".
2. "Sebuah bentuk seni yang memiliki partisipan yang diistilahkan sebahai player, yang membuat keputusan dengan tujuan memanaje sumber daya untuk mencapai sebuah goal".

3. “Sebuah aktifitas dengan beberapa aturan yang mengikat untuk mencapai suatu hasil”.

Beberapa *goal* yang umum berlaku pada *games* :

1. mengalahkan semua musuh
2. mencapai tempat keluar
3. mengalahkan waktu
4. mendapatkan *score*/nilai tertinggi
5. menyelamatkan putri
6. menangkap raja/bendera
7. membina hubungan
8. mendapatkan items yang banyak
9. menaikkan level
10. menyelesaikan sebuah *puzzle* atau permasalaan yang kompleks
11. mendapatkan, memiliki atau melakukan sesuatu
12. bertahan hidup
13. mencapai tempat keluar
14. mengoptimalkan strategi
15. mengalahkan orang lain

Definisi *computer games* menurut Jouni Smed dan Harri Hakonen “Merupakan sebuah *game* yang dimainkan dengan bantuan sebuah program komputer”. Sedangkan menurut Jacob Marner,B.Sc. dalam buku *Evaluating java for Game Development*, *computer games* adalah: sebuah perangkat lunak interaktif yang dibangun untuk menghibur *user* sebagai tujuan utama.

2.1.1 Karakteristik *Game* [1]

Ada 7 karakteristik / elemen yang menentukan sebuah *game* yang berkualitas atau tingkat "kecanduan" pada *game*, yaitu :

1. *The Environment*: *Environment* adalah *background*, *sprites*, tampilan atau grafik *game* dan elemen pendukung lainnya. Penampilan *game* haruslah dibuat semenarik mungkin dan seorisinal mungkin untuk menarik perhatian para *user*/*gamers*, namun juga harus diperhatikan kemampuan grafik *hardware*.
2. *Playability*: *Playability* adalah kemampuan "perasaan" didalam hal pengendalian *game* (skenario atau jalannya *game)* didalam mempengaruhi indera pemain.
3. *Quantity and Quality of interactions*: Hal ini berhubungan dengan interaksi *game* dengan *player*. Interaksi *game* dengan *player* diwujudkan dengan tampilan yang terjadi akibat interaksi *player – game* melalui media input.
4. *Interface Design*: *Interactive design* berhubungan dengan mekanisme yang meliputi tugas-tugas yang harus *player* lakukan didalam jeda interaksi. *Interface* haruslah dibuat simpel dan mudah untuk pengendaliannya.
5. *Levels of Play*: *The Levels of plays* merepresentasikan sebuah perkembangan pembelajaran didalam *game* serta penuh tantangan tehadap *player*. Dengan menetapkan tingkatan-tingkatan kesulitan serta menambahnya, para pemain harus meningkatkan kemampuan skill mereka untuk menyelesaikan *game*.
6. *Replayability*: *Replayability* adalah pengukuran kemampuan sebuah *game* didalam menyediakan pengalaman baru yang menarik setiap kali *game* dimainkan berulang-ulang.
7. *Character growth and development*: *Character growth and development* adalah adaptasi kemampuan pekembangan pengendalian pemain didalam memainkan *game*. Para pemain memiliki kemampuan adaptasi bahwa selama mereka memainkan *game*, kemampuan memainkan *game* mereka (skill) menjadi lebih baik dan juga karakter yang mereka kendalikanpun berkembang.

2.1.2 *Game Genres* / *Taxonomy* [4]

Banyak dimensi untuk menentukan atau mengelompokkan *game* menjadi beberapa *genre* atau tipe. Banyaknya *genre* tercipta dikarenakan tingginya perkembangan *game* serta tingginya inovasi kreasi didalam industri *game*. Pengelompokkan ini sebagian besar berpengaruh dari segi *gameplay* sebuah *game.* Beberapa jenis tipe atau genre *game* hingga saat ini:

a. Action

Unsur utama yang diangkat oleh *games* ini adalah daya refleks, koordinasi tangan yang mengendalikan karakter dan mata yang menangkap situasi di layar permainan. Genre ini adalah genre tertua dan merupakan basis dari genre-genre yang lain. Pada genre ini memiliki *gameplay* yang kuat di pertarungan. Ada beberapa *subgenre* didalam *action*, yaitu:

1. *Action-adventure*: adalah *game* yang berisikan (*gameplay*) aksi dan petualangan secara bersamaan. Ada 2 tipe didalam genre ini yaitu :
   - *Stealth*, cenderung memiliki elemen tambahan yaitu tampilan *third-person-shooter* dan memiliki tujuan yaitu: kegiatan mata-mata dan serangan presisi / akurat. Contoh *game*: Metal Gear (serial), Tom Clancy's Splinter Cell, dan Tenchu (serial).
   - *Survival horror*, memiliki fokus didalam kengerian dan berusaha menakuti para pemain dengan menggunakan elemen-elemen horor seperti darah, makhluk *undead*, kematian, suasana mencekam dan makhluk gaib. Contoh *game*: Resident Evil (serial), Alone in the Dark dan Silent Hill (serial).
2. *Beat 'em up and hack and slash*, memiliki *goal*(tujuan) mengalahkan semua musuh untuk membuka area baru. Merupakan *game side-scrolling*. Genre ini memiliki unsur murni kekerasan untuk menyelesaikan misi. Cenderung disamakan dengan genre *action adventure*. Contoh *game*: Double Dragon dan Final Fight (serial).

3. *Fighting*, merupakan *game* pertarungan satu lawan satu antara dua karakter, baik dengan pengendalian para pemain atau komputer (*artificial intelligence*). Pada saat ini *game* dengan genre ini tidak terfokus pada pertarungan satu lawan satu saja tapi bisa *multiplayer*. Contoh *game* ini yaitu Street Fighter (serial), Mortal Kombat (serial) dan Tekken (serial).
4. *Maze*, adalah *game* navigasi. Para pemain harus menemukan jalan keluar didalam lingkungan *Maze* dengan musuh-musuh dan rintangan yang menghadang. Contoh *game* yaitu Pac-Man dan M. Pac-Man.
5. *Pinball*, adalah *game* yang meniru penampilan dan "perasaan bermain" meja *Pinball* tradisional.
6. *Platform*, adalah *game* dimana *platforms* (*background* atau lingkungan *game*) sebagai rintangan. Untuk melewati rintangan ini dapat dilakukan dengan melompatinya, berlari, memanjat dan menggunakan tangga. Contoh *game* yaitu Donkey Kong, Sonic The Hedgehog (serial) dan Mario (serial).
7. *Shooter*, *game* menembak atau simulasi menembak. Dapat dikategorikan kembali menjadi:
   - *First-Person Shooter*, biasa disebut FPS. Contoh Game ini yaitu Halo (serial) dan Half-Life (serial). *Massively multiplayer online* FPS, adalah penggabungan *gameplay* FPS dengan dunia Virtual serta para pemain dengan jumlah besar melalui internet. Contoh *game* ini adalah World War II Online dan Half-Life Counter Strike 1.4 Online.
   - *Light gun shooter*, adalah *game shooter* dengan menggunakan *pointing device* dalam komputer dan *control device* dalam *arcade* atau *console.* Game ini menggunakan sensor pada *joystick* atau sinar yang diterima pada tabung monitor. Contoh *game* ini yaitu Time Crisis dan Duck Hunt.
   - *Shoot 'em up*, adalah jenis *game* menembak dimana pemain mengontrol karakter atau kendaraan kemudian menembak musuh dengan jumlah yang

banyak. Sering kali dikategorikan untuk *game non-3D shooter game*. Contoh *game* ini yaitu Star Fox (serial) dan Ikaruga.

- *Tactical shooter*, adalah variasi dari FPS *game* dengan menambahkan aspek taktik kedalam *gameplay*, seperti perencanaan dan *teamwork*. Contoh *game*: Tom Clancy's Ghost Recon (serial) dan SOCOM (serial).
- *Third-Person Shooter*, atau TPS. Menggunakan *Camera Perspective* orang ketiga yang memberikan area penglihatan yang lebih luas dibandingkan FPS. Contoh *game* ini: Grand Theft Auto.

b. *Adventure*

Didalam game ini aspek pertualangan bukan bedasarkan pada cerita atau isi *game* tersebut. Tetapi lebih menjelaskan perilaku *gameplay* tanpa tantangan rangsangan (refleks) atau aksi. Game ini memberikan *puzzle* untuk dipecahkan melalui interaksi pemain / karakter dengan lingkungan / *environtment game*, mayoritas tidak melalui jalur konfrontasi. Beberapa *sub-genre*:

1. *Text adventure / Interactive fiction*, jenis *game* ini menggunakan media *keyboard* untuk menginput/mengetikkan perintah seperti "*go west"* (gerak ke arah barat), *"get rope"* (ambil tali) dan lain-lain. Contoh *game* ini adalah Zork (1970 dan 1980).
2. *Graphic adventure, game* ini menggantikan penginputan teks melalui *keyboard* (*text adventure*) dengan *mouse* (*point and click adventure*). Contoh *game* ini adalah Dora: The explorer. Gameplay dalam *sub-genre* ini biasanya "*Escape the room*" / melarikan diri dari ruangan tertutup.
3. *Visual novel*, *game* ini menggunakan *static graphic* yaitu tampilan gambar tak bergerak dengan teks cerita dan mengikuti plot-plot seperti novel. Jenis *game* ini dipopulerkan di Jepang dengan mayoritas bertipe *"Dating Sims type"* / simulasi berpacaran, dimana *artwork*-nya bertipe "*anime style*" atau "*manga style"*. Contoh *game* ini adalah Tokimeki Memorial (serial).

4. *Interactive movie*, jenis *game* ini diciptakan ketika pertama kali pada media *Laserdisc*. *Gameplay* dalam *game* ini mengontrol karakter untuk bergerak ketika adegan tertentu, misalkan karakter tejebak dalam suatu keadaan terjepit, pemain menentukan kearah mana karakter menghindar. Dimana pilihan adegan telah ditentukan dalam *game*. *Gameplay game* ini biasanya disisipkan dalam *gameplay* genre *game* lain, seperti dalam Resident Evil 4. Contoh Game ini Night Trap dan Space Ace.

c. *Construction and management simulation* (CMSs)

Merupakan *game* bertipe simulasi dimana tujuan *game* ini adalah untuk membangun (*construct/build*), mengatur friksi didalam komunitas atau menyelesaikan tugas dengan *resource* terbatas.

1. *City-building*, pemain mengendalikan sebuah kota dengan *goal* / tujuan akhirnya adalah petumbuhan ekonomi yang baik dan kemakmuran. Contoh *game* ini adalah SimCity.
2. *Business simulation*, merupakan *game* simulasi dalam bidang bisnis. Dimana tujuan akhirnya adalah keuntungan bisnis (ekonomi). Dimana *game* me-simulasi-kan keadaan perekonomian dalam suatu daerah dan dalam suatu waktu.
3. *God games*, tidak seperti dalam *game* lainnya, *game* ini tidak memiliki tujuan akhir. Dalam *game* ini, pemain mengatur hidup karakter dalam *game* atau suatu populasi. Beberapa contoh *game* ini adalah SimEarth, SimAnt dan Populous.
4. *Government simulation*, merupakan *game* simulasi politik dimana tujuan / *goal game* ini adalah kekuasaan.

d. Life simulation

Game ini memiliki nama lain *Artificial Life Game*. Dimana dalam *game* ini pemain mengendalikan / mengatur kehidupan buatan dalam *game*, baik mengatur kehidupan sosial didalam populasi buatan atau ekosistem buatan.

1. *Biological simulation*, pemain dapat melakukan eksperimen dalam genetika, *survival* / evolusi dan pengamatan ekosistem.
2. *Pet-raising simulation*, dikenal dengan sebutan *Digital Pets*. Fokus dalam *game* adalah simulasi hewan peliharaan buatan (digital). Tamagotchi adalah *game* yang mempopulerkan jenis *game* ini.
3. *Social simulation*, dasar dalam permainan ini adalah interaksi antar kehidupan buatan (digital). *Game* yang mempopulerkan genre ini adalah The Sims.

e. *Role-playing* (RPG)

Keistemewaan dalam genre ini adalah *Turn Based System* dan *Active Time System*. Sistem dalam RPG berdasarkan sebuah *game* yaitu Dungeons and Dragons. Dimana pemain dapat memainkan beberapa karakter sekaligus. Karakter-karakter tersebut dapat dipilih dengan mengikuti *storyline game* tersebut, cenderung memiliki karakter tersembunyi, atau dengan hanya memilih karakter standar (awal). *Turn Based System* adalah sistem pertempuran dimana para karakter menyerang bergiliran dengan keahlian khusus (setiap karakter memiliki hanya satu atau lebih kekuatan khusus). Sedangkan *Active Time System* adalah sistem pertarungan dengan mengkalkulasikan setiap serangan dan penyerangan dengan setiap status para karakter, seperti level karakter, kekuatan karakter dan lain-lain.

1. *Computer and console role-playing*, jenis genre ini terpisahkan karena faktor kebudayaan dan tempat asal para perusahaan pengembangnya. *Computer role-playing* dipopulerkan di Amerika dan Eropa, memiliki "*non-linear*" *storyline* dimana pemain memiliki pilihan yang dapat dipilih untuk menentukan alur cerita dan pemain dapat membuat karakter mereka sendiri. Sedangkan *Console role-playing* (Japan), pemain mengatur *party* (grup karakter) dan memiliki *storyline* yang "*linear*". Contoh *game*: Ultima dan The Elder Scroll (*Computer* RPG), serta Dragon Quest dan Final Fantasy (*Console* RPG).

2. *Action role-playing*, atau *Action* RPG merupakan gabungan "*Action Games*" dan "*Action Adventures Games*". Dalam genre ini memiliki banyak aksi dan sering kali menyederhanakan atau menghilangkan atribut atau statistik *non-combat* yang ada dalam pengembangan karakter dalam *storyline.* Dalam pertarungan, presisi dan kecepatan serangan menentukan *damage* atau kekuatan serangan dibandingkan atribut karakter. Dalam *game*, menemukan harta karun lebih menentukan dibandingkan perjalanan cerita (*storyline*). Contoh *game* ini adalah Diablo dan Kingdom Hearth.
3. *Massively multiplayer online role-playing* (MMORPG), adalah genre yang populer pada saat ini dan menjadi pusat pengembangan dalam *game technologies.* Dimana *game* RPG dimainkan dengan menggunakan interaksi antar komputer baik LAN (jaringan) atau internet. World of Warcraft adalah contoh *game* yang terkenal saat ini.
4. *Roguelike*, subgenre ini menggunakan nama *game* yang memperkenalkan sistem gameplay ini yaitu *Rouge*. Didalam *game*, terdapat "*two-dimensional Dungeon Crawl*" (merangkak dalam menaiki jurang yang tergambar dua dimensi) dengan pengacakan(dengan kuat) dan pengembangan karakter secara statistik. Contoh *game*: Rogue, ADOM dan netHack.
5. *Tactical role-playing*, prinsip *game* ini adalah menggabungkan *gameplay* strategi untuk menggantikan sistem RPG sederhana. Seperti RPG sederhana, *game* ini mengontrol beberapa karakter, baik personal atau grup/*party*, untuk melawan musuh namun perlu diperhatikan secara *isometric* pergerakan karakter/grup dalam bertempur baik dari segi arah (*isometric grid*) dan dari segi pergerakan *strategical game* (*manual game).* Contoh *game*: Fire Emblem, Final Fantasy: the Tactic dan Age of Empire.

f. *Strategy*

Genre strategi lebih memfokuskan dalam *gameplay* yang menitikberatkan kehati-hatian, memeras otak dan perencanaan didalam menentukan *goal* / kemenangan. Mayoritas judul *game* bergenre strategi, pemain mengontrol karakter atau *party* / unit dalam pertempuran serta memiliki ciri khas "*godlike view of the game world*" (pandangan dari angkasa yang dapat melihat peta atau seluruh permainan dengan jelas). Game dengan genre ini merupakan penurunan dari *game* papan (catur, shogi dan lain-lain).

1. 4X, memiliki *goal* yaitu *eXplore* (menjelajah), *eXpand* (memperluas), *eXploit* (menggunakan sumber daya yang ada) dan *eXterminate* (menghancurkan). *Gameplay*-nya dapat berupa "*turn-based*" atau "*real-time*". Contoh *game*: Civilization.
2. *Artillery*, dapat disamakan dengan genre "*shooting game*" dikarenakan dalam *gameplay*-nya genre ini menggunakan senjata. Namun, karena memerlukan perhitungan yang tepat terhadap sudut kemiringan senjata dan kalkulasi layaknya perhitungan militer maka genre ini termasuk strategi. Contoh *game* ini adalah Goundbound, Scorched Earth dan Tanarus.
3. *Real-time strategy* (RTS), penamaan genre ini untuk mengklasifikasikan dalam genre *Computer Strategy Game* dengan perbedaannya: aksi yang kontinu dan pemain harus memutuskan dan melakukan aksi dalam perubahan status *storyline* atau *gameplay game*. Inti *game* ini adalah untuk menguasai sumber daya yang ada, bangunan-bangunan, penelitian teknologi dan memproduksi unit-unit. Contoh *game*: Comand & Conquer (serial) dan StarCraft.
4. *Real-time tactics*, dalam genre ini hanya memfokuskan aspek operasional dan pengontrolan peralatan perang. Tidak seperti dalam RTS, *resource* (sumber daya), manajemen ekonomi dan bangunan tidak ikut bagian dalam *gameplay* pertempuran. Contoh *game*: Warhammer: Dark Omen dan Close Combat (serial).

5. *Tower defense*, *gameplay* genre ini lebih sederhana karena hanya membangun dan menempatkan menara (*tower*) untuk membunuh dan menghalangi musuh (*monster*), yang biasa dinamakan *Creeps*. Contoh *game*: Desktop Tower Defense.
6. *Turn-based strategy* (TBS), *gameplay* genre ini untuk merunutkan didalam *Genre Computer Strategy Game* untuk membedakannya dengan *Real-time Computer Strategy Game*. Seorang pemain di dalam *turn-based* diberi waktu untuk menganalisa sebelum memberikan aksi dan dalam beberapa judul genre ini, diijinkan beberapa aksi atau langkah dilakukan dalam waktu yang bersamaan. Contoh *game*: Heroes of Might and Magic, Civilization dan The Nintendo Wars (serial).
7. *Turn-based tactics*, dalam *gameplay* genre ini dikarakteristikan dalam *goal*-nya yaitu pemain menyelesaikan semua tugas dengan kekuatan pasukan yang diberikan pada pemain tersebut dan biasanya penggambaran dalam pertempuran dan penggambaran taktik dibuat serealistik mungkin. Contoh *game*: Jagged Alliance dan X-COM.
8. *Wargames*, *gameplay* genre ini lebih kepada strategi peperangan berdasarkan peta (*map*). Fokus *game* ini sendiri lebih kepada *turn-based* atau *real-time* dan strategi militer atau taktik. Contoh *game*: Panzer General dan Romance of the Three Kingdoms (serial).

g. *Vehicle simulation*

Didalam *game* bergenre *vehicle simulation* ini tujuan utamanya adalah untuk memberikan suatu presentasi yang serealistik mungkin didalam mengoperasikan berbagai jenis kendaraan yang ada dalam dunia nyata.

1. *Flight*, genre ini biasanya untuk memberikan pengajaran kepada pemain bagaimana mengoperasikan pesawat senyata mungkin. *Combat Flight Simulation* adalah *sub-genre* yang paling popular untuk genre ini. Karena tidak hanya kerumitan pengoperasian sebuah pesawat saja yang diberikan tetapi juga simulasi tempur. Contoh *game*: Microsoft Flight Simulator, X-Plane, Falcon 4.0(*Combat)* dan IL-2 Sturmovik (*Combat*).
2. *Racing*, *genre* ini dapat dibagi kembali menjadi 2 *sub-genre*, yaitu *Imaginary Racing Games*, yang menambahkan unsur *action* dalam balapan(dikenal dengan nama *Arcade Racing Games*) dan *Simulation Racing Games*, yang berupaya menghadirkan pengalaman balapan nyata kedalam *game*. Contoh *game*: Mario Kart dan out Run (*imaginary*) serta Nascar Racing dan Gran Turismo (*simulation*).
3. *Space flight,* merupakan simulasi pesawat antariksa. Umumnya tidak menggambarkan secara nyata dengan keadaan sebenarnya pesawat antariksa sesungguhnya dan keadaan luar angkasa yang nyata serta sering memutarbalikkan perhitungan fisika. Namun sebagian kecil judul *game* ini cukup mesimulasikan keadaan nyata, contohnya Orbiter.
4. *Train*, *gameplay* genre ini memfokuskan dalam pengoperasian kereta api, keadaan sekitar dan sering kali memfokuskan kepada keadaan pengelolaan jalur kereta api. Contoh *game:* Microsoft Train Simulator, Trainz dan Rail Simulator.
5. *Vehicular combat,* fokus *game* genre ini lebih ke unsur *action* dimana pemain mengoperasikan kendaraan dengan tujuan menghancurkan lawan / musuh dengan mengkombinasikannya dengan unsur *shooting* seperti dalam Spy Hunter dan Rock ‘N Roll Racing. *Sub-Genre* dalam genre ini adalah *Mecha Combat* dimana kendaraan yang digunakan adalah robot besar menyerupai tank. *Mecha* diperkenalkan di Jepang untuk mendeskripsikan robot besar didalam *anime* (film animasi) atau *manga* (komik Jepang).

h. *Other Notable Genres*

1. *Music*, sering kali dikarakteristikan termasuk dalam *Arcade Games* yaitu memberi tantangan kepada pemain untuk mengikuti serangkaian pergerakan atau rangkaian nada. Beberapa *game* memerlukan pengendali di tangan (*game controller*), untuk dikendalikan dengan diinjak (*dance pad*) atau peralatan yang menyerupai alat musik seperti *drum set*. Sedangkan sebagian *game* genre ini dipasarkan untuk permainan keluarga (*home consoles*) yang menghindari *gameplay* berdasarkan alunan rangkaian nada (*rhythm*) dan memfokuskan kepada ketepatan, memori atau *Sandbox-style*. Contoh *game*: Guitar Hero dan Rock Star (Serial).
2. *Party*, adalah *genre* yang dikembangkan spesifik mungkin untuk dimainkan bersamaan oleh banyak orang (*multiplayer*). Umumnya, *party game* memiliki beberapa tipe jenis *game* kecil (mini) yaitu mengoleksi lebih banyak jenis benda dibandingkan pemain lain atau menyelesaikan sesuatu lebih cepat dibandingkan yang lain. *Gameplay versus* tidak dikategorikan ke dalam *party games*. Contoh *game*: Mario Party dan Sonic Shuffle.
3. *Programming*, adalah *game* yang dimainkan pada komputer dimana pemain tidak mempengaruhi langsung kedalam Jalannya *game*. Untuk memainkan *game*, pemain menuliskan *script* yang menggunakan bahasa pemrograman yang spesifik didalam mengontrol aksi karakter (biasanya robot-robot, tank-tank atau bakteria yang bertujuan menghancurkan satu sama yang lainnya). Contoh *game*: Cora War, Robocode dan Crobots.
4. *Puzzle*, dalam *game* ini pemain harus menyelesaikan *puzzle logical* atau menavigasi didalam daerah yang kompleks (*maze*). Didalam *arcade game*, varian-varian *game* Tetris seringkali diberi label *genre puzzle games*, meskipun dalam *gameplay*-nya memerlukan koordinasi tangan-mata dan refleks yang cepat dibandingkan pemikiran dan logik

5. *Sports*, merupakan simulasi dari permainan olahraga fisik trasisional. Beberapa *game* memberikan rasa seolah-olah dalam melakukan permainan olahraga tersebut, sedangkan yang lainnya lebih menekankan dalam strategi permainan olahraga tersebut (dibalik layar). Beberapa *game* lainnya menambahkan permainan olahraga dengan efek komik (seperti *Arch Rivals*). Pong merupakan *game* yang pertama kali diciptakan dalam sejarah *game*.
6. *Traditional*, merupakan *game* yang bertujuan me-komputerisasikan permainan-permainan papan (*board games*) dan permainan-permainan kartu (card *games*), dimana *Artificial Intelligence* komputer dapat membantu meningkatkan kemampuan seseorang dalam permainan tradisional.

i. *Made by Purpose*

1. *Adult*, sama halnya *adult movies* atau media lainnya, dikhususkan untuk pengguna dewasa. Umumnya, *adult games* lebih menyediakan hiburan erotik dibandingkan *gameplay*-nya. Objek daripada *adult game* dapat berbeda-beda bergantung pada *mainstream game* tersebut, hal tersebut dapat direpresentasikan dalam gambar (visual) telanjang, semi telanjang atau aktivitas sexual. Beberapa *game* memfokuskan kedalam komedi atau drama, atau hanya memiliki *gameplay* normal / standar dengan ditambahkan elemen-elemen dewasa.
2. *Advergame*, mayoritas *game* ini dapat ditemukan / dimainkan *online* dan kebanyakan merupakan sebuah *Flash games* yang simple dan murahan. *Advergame* merupakan produk periklanan yang bertujuan untuk meningkatkan popularitas sebuah perusahaan atau produknya.
3. *Casual*, merupakan *game* sederhana dengan peraturan yang simple atau bermaian secara teknik. Tujuan *game* ini lebih pada hiburan, namun dengan komitmen terhadap *game* yang sangat rendah.
4. *Faith / religion*, tujuan *game* ini adalah untuk mengajarkan pemain tentang ajaran agama.

5. *Educational*, tujuan *game* ini adalah untuk pendidikan. Dengan menargetkan kepada para pemain muda (usia sekolah) dan anak-anak pra-sekolah. Banyak sekali *sub-genre* untuk genre ini, tergantung pada mata pelajaran yang ada.
6. *Exergame*, bertujuan memberikan latihan. *Exergames* terbagi menjadi dua bagian. Bagian pertama diperuntukkan sebagai latihan *input Device* (sebagai contoh, *game Wii Fit* menggunakan *Wii Balance Board*) dan sebagian implementasi yang menggunakan sebuah genre dalam *game*.
7. *Serious*, memiliki tujuan untuk mendidik atau melatih pemain. *Game* ini cenderung bertujuan untuk mempromosikan pendidikan, ilmu pengetahuan, pelayanan kesehatan atau bahkan militer. Beberapa *game* ini tidak memiliki akhir (*goal*) yang spesifik. Esensi *game* berada pada bagaimana pemain mempelajari pelajaran kehidupan nyata dalam *game*.

j. *Game interfaces*

Klasifikasi *genre* berdasarkan media yang digunakan dalam *gameplay game* tersebut. Atau, media yang mayoritas dalam *gameplay*.

1. *Audio game*, menggunakan (mayoritas) media suara sebagai media input / output dalam *gameplay*.
2. *Browser game*, menggunakan media *browser* (*mouse* / *mice*).
3. *Text-based game*, menggunakan media tampilan teks dalam *gameplay*.
4. *Tile-based video games*, menggunakan visual grafik sederhana.
5. *Side-scrolling video game*, menggunakan visual grafik yang dapat melebar (atas-bawah atau kiri-kanan).

k. *Game platforms*

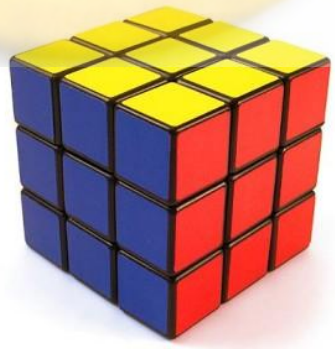
Klasifikasi *game* yang berdasarkan kepada mesin elektronik yang digunakan sebagai perantara untuk memainkan *game* ini.

1. *Arcade game*, *game* yang dimainkan dimainkan mesin *arcade*, merupakan mesin yang populer untuk memainkan *game*.

2. *Console game*, *console* merupakan mesin yang lebih ringan dan dipasarkan untuk masyarakat umum (alat elektronik khusus memainkan Game). Contoh: *Nintendo*, *Sega* dll.
3. *Handheld video game*, alat elektronik yang khusus untuk memainkan *game* dengan media input dengan pengontrolan dari tangan (*handheld* atau *joystick*).
4. *Massively multiplayer online game*, menggunakan jaringan antar komputer untuk memainkan *game*.
5. *Mobile game*, menggunakan *handphone* untuk memainkan *game*.
6. *Online game*, menggunakan akses internet untuk memainkan *game*.
7. *PC game*, menggunakan *Personal Computer* (PC) untuk memainkan *game*.

2.2 Kubik Rubik [5]

Rubik's Cube ditemukan oleh Erno Rubik dan telah dipatenkan atas namanya. Benda ini sekilas terlihat sebagai sebuah kubus yang terdiri atas 27 kubus kecil, padahal sebenarnya hanya terdapat 26 kubus kecil karena kubus kecil yang paling dalam tidak pernah tersentuh. Cara memainkan kubus ini adalah dengan mengacak sisi-sisinya lalu mengembalikannya ke keadaan tersusun, di mana keenam sisinya memiliki warna yang sama.



Gambar 2.2 Kubik Rubik Berdimensi 3x3x3

Pada tugas akhir ini, aplikasi *game* Kubik Rubik yang akan dibuat dapat dikategorikan ke dalam beberapa jenis *genre*. Dikategorikan *puzzle game*, dikarenakan sifat permainan Kubik Rubik yang memiliki *logical puzzle*. Aplikasi ini pun dapat dikategorikan sebagai *browser game*, *side-scrolling game* dan PC *game* dikarenakan media input dan ouput aplikasi Kubik Rubik yang akan dibuat ini.

Terminologi yang biasa dipakai untuk menjelaskan suatu algoritma Rubik's Cube sebagai berikut: Kubik Rubik terdiri atas enam buah sisi: depan (*front*), belakang (*back*), kiri (*left*), kanan (*right*), atas (*up*), dan bawah (*down*). Gerakan memutar sisi Kubik Rubik searah jarum jam disimbolkan dengan huruf pertama dari nama sisi tersebut dalam bahasa Inggris. Untuk yang berlawanan arah dengan jarum jam, ditambahkan tanda petik ('). Contohnya, F berarti memutar sisi depan (*front*) searah jarum jam, sedangkan L' berarti memutar sisi kiri (*left*) berlawanan arah dengan jarum jam.

Dalam tugas akhir ini ditambahkan beberapa gerakan tambahan untuk membantu dalam pencarian penyelesaian, yaitu :

1. Ls (2 gerakan memutar searah L: L dan R') dan Ls' (berlawanan arah dengan Ls: L' dan R).
2. La (2 gerakan berlawanan yaitu L dan R) dan La' (berlawanan arah dengan La: L' dan R').
3. Us (2 gerakan memutar searah U: U dan D') dan Us' (berlawanan arah dengan Ls: U' dan D).
4. Ua (2 gerakan berlawanan yaitu U dan D) dan Ua' (berlawanan arah dengan Ua: U' dan D').
5. Fs (2 gerakan memutar searah F: F dan B') dan Fs' (berlawanan arah dengan Fs: F' dan B).
6. Fa (2 gerakan berlawanan yaitu F dan B) dan La' (berlawanan arah dengan Fa: F' dan B').

2.3 Java [6][7]

Java yang dikembangkan di Sun Microsystem berawal dari gagasan untuk menciptakan suatu bahasa, perangkat lunak yang bebas dan tidak bergantung pada platform atau sistem operasi tertentu (tidak hanya bekerja pada sistem operasi tertentu). Tujuan awalnya adalah dengan menggunakan bahasa yang sudah ada, yaitu C++ namun seiring dengan kemajuan yang dicapai, para pencipta JAVA menyadari bahwa akan lebih baik bila mereka menemukan (menciptakan) bahasa mereka sendiri daripada mengembangkan C++.

Tidak seperti bahasa-bahasa compiler tradisional, yang mengubah kode (*source code*) menjadi perintah-perintah tingkat mesin (bahasa mesin), kompiler Java mengubah (menterjemahkan) kode-kode sumber Java menjadi perintah-perintah yang akan di- interpretasi (dibaca) oleh runtime Mesin Virtual Java (*Java Virtual Machine*). Java dapat digunakan untuk membuat dua jenis program, yaitu applet dan aplikasi mandiri (*stand alone application*). Secara sederhana, sebuah applet adalah bagian dari halaman web entah itu berupa animasi, gambar sederhana (image) atau hanya sebuah garis atau sekumpulan teks. Para pencipta Java di Sun Microsystem mendefinisikan Java sebagai bahasa yang sederhana, berorientasi object, terdistribusi, terinterpretasi, kokoh, aman, netral arsitektur, akrab, berkinerja tinggi, multi jalinan (*multithreaded*) dan dinamis.

2.3.1 Java 3D [8]

Java 3D API adalah sebuah interface program yang digunakan untuk menampilkan atau membuat grafik 3 dimensi dan melakukan penghalusan suara pada sebuah aplikasi. Java 3D merupakan pengembangan dari Java 2 JDK. Java 3D menyediakan sebuah bahasa programming high level untuk membuat dan memanipulasi geometri 3 dimensi dan struktur-struktur yang digunakan untuk melakukan penghalusan grafik.

Java 3D menyediakan fungsi untuk membuat gambar, visualisasi, animasi dan aplikasi 3 dimensi. Java 3D sangat berguna bagi pihak pengembang pembuat aplikasi untuk membuat dan memanipulasi bentuk geometri 3 dimensi dan untuk membuat struktur-struktur yang dapat digunakan untuk memperhalus gambar 3 dimensi tersebut. Pengembang aplikasi yang menggunakan Java 3D ini dapat menggambarkan dunia virtual yang sangat luas dan Java 3D menyediakan informasi-informasi yang cukup jelas untuk membuat dan memperhalus dunia virtual dengan sangat efisien.

Java 3D juga memiliki keunggulan yang dimiliki Java pada umumnya, yaitu aplikasi yang dibuat dapat dijalankan pada semua platform yang ada. Hal ini menambah keuntungan bagi pihak pengembang aplikasi yang mengutamakan pada grafik 3 dimensi. Java 3D merupakan bagian dari JavaMedia API, yang membuat Java 3D ini tersedia di berbagai platform. Java 3D juga dapat berintegrasi dengan internet karena aplikasi dan applet yang dibuat menggunakan Java 3D ini mempunyai akses pada semua class yang ada pada Java.

Java 3D menggunakan ide-ide yang ada pada application programming interface yang sudah ada. Java 3D pada level yang rendah membuat grafik dengan melakukan konstruksi dan analisa pada cara terbaik yang ditemukan di API level rendah seperti Direct3D, OpenGL, QuickDraw3D, dan XGL. Sedangkan pada Java 3D level yang lebih tinggi melakukan konstruksi dan analisa pada cara terbaik yang ditemukan pada sistem yang menggunakan *scene graph*.

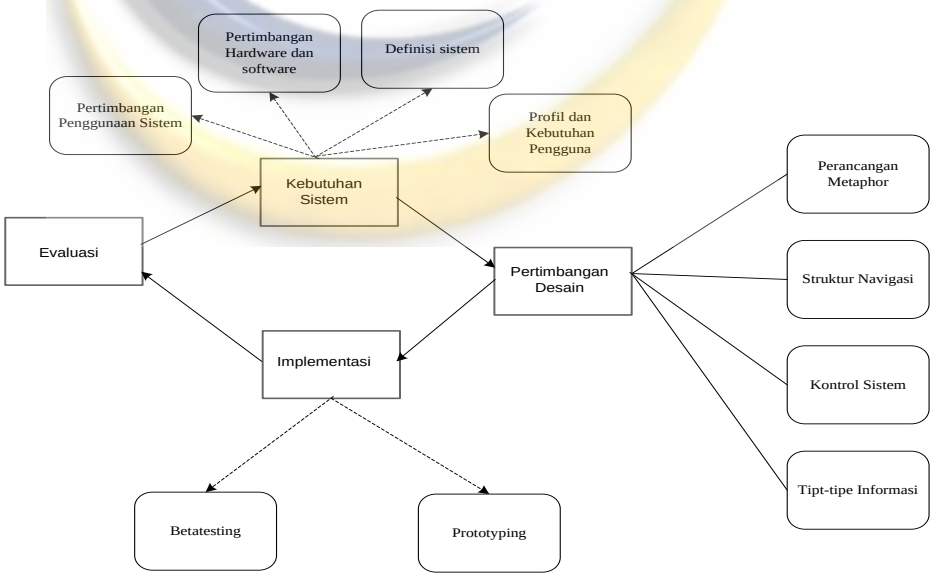
Java 3D memperkenalkan beberapa konsep yang tidak umum pada lingkungan pembuatan aplikasi yang berbasis grafik seperti Java 3D *spatial sound*. Java 3D sound mempunyai kemampuan untuk menyediakan kualitas suara yang lebih baik pada pengguna dari aplikasi yang menggunakan fasilitas ini. Java 3D ini memiliki tujuan utama yaitu aplikasi yang dibuat dengan Java 3D dapat memberikan hasil yang paling baik dan memuaskan untuk pengguna aplikasi. Selain itu alternatif

yang akan menguntungkan terutama pada proses eksekusi dipilih untuk mendapatkan hasil yang maksimal.

Java 3D memberikan banyak sekali dukungan kepada pihak pengembang aplikasi terutama yang menggunakan grafik 3 dimensi. Java 3D ini mempunyai lima package utama, yaitu audioengines, loaders, utils, j3d dan vecmath tetapi beberapa package utama ada yang merupakan kumpulan dari package-package lain. Package yang ada di dalam kumpulan package lain ada yang memiliki fungsi sama dengan package-package dari parent tetapi ada juga yang memiliki fungsi berbeda-beda.

2.4 Metodologi Pengembangan Perangkat Lunak *(Software)* [9]

Untuk menyelesaikan masalah aktual dalam sebuah rekayasa perangkat lunak diperlukannya strategi untuk pengembangan yang melengkapi lapisan proses dan metode. Model proses untuk rekayasa perangkat lunak dipilih berdasarkan sifat aplikasi dan proyeknya yaitu Perancangan dan Pembangunan Sistem Multimedia Interaktif.



Gambar 2.3 Siklus Perancangan dan Pembangunan Sistem Multimedia Interaktif Dastbaz

2.4.1 Tahap-tahap pada siklus IMSDD (Interactive Multimedia System Design & Development)

Tahap-tahap yang terdapat dalam siklus perancangan dan pengembangan IMSDD yaitu:

a. Kebutuhan Sistem

Tahap ini sama dengan tahap spesifikasi kebutuhan (*requirement specification*) yang terdapat dalam model *Waterfall* dan didalamnya terdapat elemen-elemen seperti *feasibility* dan *hardware selection* yang juga terdapat dalam model RMM (*The Relationhip Management Methodology*). Pada tahap ini mempunyai fungsi utama, diantaranya :

1. Untuk menyajikan definisi sistem yang mencakup garis besar dan tujuan dari sistem.
2. Untuk menjelaskan pengguna mana saja yang akan menggunakan sistem dan juga menjelaskan kebutuhan-kebutuhan khusus yang digunakan dalam pertimbangan. Sebagai contoh jika kita akan melakukan perancangan untuk mengajar bahasa isyarat bagi pengguna yang memiliki kekurangan dalam pendengaran dengan menggunakan *audio*, yang merupakan cara penyampaian informasi yang tepat. Oleh sebab itu kita harus memberikan perhatian khusus pada kegiatan mengelompokkan informasi yang akan digunakan dan pendekatan perancangan yang akan kita ambil untuk penyajian informasi.
3. Untuk mengevaluasi kebutuhan *hardware* dengan *platform-platform software* yang digunakan, sehingga dapat dibuat keputusan yang tepat.
4. Untuk mempertimbangkan dengan baik, *platform* yang dibutuhkan untuk sistem pada kenyataannya membangun sistem multimedia interaktif yang terdistribusi yang dapat dijalankan pada jaringan (LAN/WAN) membutuhkan pendekatan yang berbeda dibandingkan dengan tipe sistem CD-ROM yang *stand alone* terutama dibagian perancangan dan pembangunan.

b. Pertimbangan Desain

Tujuan dari tahap ini yaitu untuk menyusun pedoman mengenai rincian perancangan. Dalam hal ini, tahap ini sama dengan tahap-tahap perancangan arsitektual (*architectural design*) dan perincian perancangan (*detailed design*) pada model *waterfall* atau tahap Perancangan (*design*) pada siklus perancangan antarmuka pengguna (*pengguna interface design cycle*) yang dikemukakan oleh *Preece* (1993). Tahap ini bertujuan untuk mengemukakan hal-hal:

1. Perancangan Metafora (*design Metaphor*)

Melakukan pemilihan model yang sesuai dengan keadaan dilapangan (*real word mental mode*) yang akan digunakan sebagai solusi perancangan antarmuka bagi sistem (contoh : sebuah film, buku, permainan, dan lainnya.)

2. Tipe dan format Informasi (*Information types and formats)*

Untuk mendefinisikan tipe informasi yang ingin diintegrasikan ke dalam sistem (contoh: teks/tulisan, grafik, suara, video dan animasi), sebagai contoh sebuah sistem multimedia interaktif untuk film dan bioskop akan menunjukkan bahwa isi dari tipe video yang akan digunakan kemungkinan dibutuhkan dalam skala yang besar. Sedangkan sebuah sistem ensiklopedia akan membutuhkan campuran isi yang seimbang dengan memberikan penekanan pada tipe teks/isi dari informasi.

3. Struktur Navigasi (*Navigational Structures*)

Untuk menjelaskan strategi dari alat navigasi yang akan digunakan termasuk didalamnya struktur link dan fitur-fitur.

4. Kontrol Sistem (*System Control*)

Untuk menjelaskan fitur-fitur dan tipe dari control dan alat-alat yang dibutuhkan bagi sistem. Termasuk didalamnya alat-alat pencarian, suara, video, dan animasi *control*, fasilitas penanda buku, dan lain-lain.

c. Implementasi

Ketika fitur perancangan telah di definisikan, tahap implementasi pada sistem akan dimulai dengan menggunakan *multimedia-outhoring tools*. Tahap implementasi terdiri atas:

1. Membuat *prototype* sistem

Tahap ini adalah proses atau rancangan yang akan dibangun untuk pengembangan penelitian.

2. Melakukan *betatesting*

Pada tahap ini melakukan pengecekan pada *prototype* untuk mengetahui rancangan yang akan bisa digunakan dan *control* pada setiap permasalahan.

Tahap ini sama dengan tahap *coding, integration, unit testing* pada model *waterfall* atau tahap implementasi pada siklus perancangan antarmuka pengguna (*user interface design cycle*), tahap implementasi pada model perancangan OOHDM (*The Object Oriented Hypermedia Design Model*) dan tahap *construction* pada model perancangan RMM.

d. Evaluasi

Pada tahap ini sistem akan dinilai berdasarkan tujuan awal yang telah direncanakan. Terdapat dua jenis pendekatan yang bisa digunakan dalam evaluasi seperti *formative* atau *summative.*

2.4.2 Panduan Perancangan Antarmuka pada IMSDD

Berikut ini adalah panduan dalam merancang antarmuka pada IMSDD :

a. Menggunakan *metaphor* (bayangan/imaginasi) yang tepat.

Bayangan/imajinasi yang baik akan menciptakan suasana yang nyaman bagi pengguna sehingga dengan cepat dapat mempelajari atau mengenali sistem.

b. Kesederhanaan dan kenyamanan dalam penggunaan merupakan hal yang utama.

Antarmuka yang bagus dapat membuat pengguna langsung menjalankan sistem tanpa harus mempelajari petunjuk pemakaian terlebih dahulu.

c. Konsistensi dalam perancangan merupakan hal yang sangat penting.

Dengan adanya konsistensi dalam perancangan akan membuat pengguna merasa nyaman dalam menggunakan sistem. Penggunaaan *icon* dan fitur alat navigasi yang konsisten akan membantu mengurangi kompleksitas pada antarmuka sistem multimedia interaktif.

d. Kebutuhan akan panduan yang dapat membantu pengguna.

Penyediaan panduan informasi (seperti keterangan yang muncul ketika pengguna menggerakkan *mouse* pada sebuah *icon*) dan panduan manual akan membantu pengguna yang masih awam agar dapat mempelajari sistem lebih jauh lagi dan menguasainya secara mendalam.

e. Menyediakan mekanisme untuk menangani kesalahan yang mungkin dilakukan oleh pengguna.

Suatu hal penting yang harus diperhatikan oleh seorang *designer* dalam IMS yaitu adanya fitur control yang dapat membuat *pengguna* memperbaiki kesalahan yang telah dibuat dan mengulang kembali proses yang telah mereka jalani dengan kurang hati-hati.

2.5 Pengembangan Perangkat Lunak Berorientasi Objek [10][11]

Pengembangan perangkat lunak berorientasi objek berbeda dari pengembangan konvensional yang memandang perangkat lunak sebagai fungsi dan data yang terisolasi. Pandangan ini dapat dinyatakan dengan persamaan yang dikemukakan Niklaus Wirth sebagai berikut:

Algorithms + Data Structures = programs

Persamaan itu menyatakan bahwa program perangkat lunak adalah sekumpulan mekanisme yang melakukan aksi – aksi tertentu pada data tertentu.

Dengan demikian, pada pendekatan tersebut terdapat dua hal berbeda yang saling melengkapi dalam memandang pembangunan perangkat lunak, yaitu:

1. Fokus pada fungsi, atau
2. Fokus pada data

Pada pendekatan konvensional, kebanyakan berfokus pada fungsi. Namun juga terdapat pendekatan yang fokus pada data terutama pada kubu basisdata dan pemodelan informasi. Sementara itu pandangan berorientasi objek berpusat pada objek yang mengkombinasikan data dan fungsionalitas. Keduanya sekaligus, jadi malah bukan termasuk dalam salah satu dari dua kubu.

Coad – Yourdon mendefinisikan pendekatan berorientasi objek dengan persamaan berikut:

Berorientasi objek = objek + klasifikasi + pewarisan + komunikasi

Pendefinisian di persamaan ini dapat dipandang sebagai ekspesi sangat minimal yang harus dipenuhi metodologi berpredikat pendekatan berorientasi objek.

a. Kelas(*class*)

Kelas merupakan blok pembangun di pendekatan berorientasi objek. Kelas merupakan pengkapsulan nilai – nilai atribut dan layanan – layanan eksklusifnya. Kelas mendeskripsikan satu objek atau lebih dengan sekumpulan atribut dan layanan yang seragam, ermasuk deskripsi penciptaan objek baru di kelas itu. Perbedaan antara kelas dan objek adalah objek merupakan entitas konngkret yang ada secara ruang dan waktu, sedangkan kelas hanya merupakan representasi abstraksi.

Kelas memiliki properti – properti berikut:

1. Nama, yang membedakannya dengan kelas lain.
2. Atribut, yang mendeskripsikan satu cakupan nilai yang dapat dimiliki instan kelas/objek
3. Operasi/metode, mengimplementasikan layanan yand disediakan objek

4. Tanggungjawab, termanifestasi pada operasi dan atribut yang dimiliki.

b. Objek(*object*)

Objek adalah sesuatu yang berarti di dalam koteks suatu sistem. James Marting dan James J Odell mengemukakan objek adalah sesuatu yang dapat dikonsepkan yang diperlukan untuk memecahkan masalah. Objek dapat berupa konsep, abstraksi atau sesuatu dengan batas – batas tegas dan mempunyai arti untuk persoalan yang ditangani.

c. Pewarisan (*inheritance*)

Hampir selalu, perangkat lunak baru memperluas pengembangan dari perangkat lunak yang sebelumnya, cara terbaik untuk menciptakannya adalah dengan meniru, memperbaiki dan mengkombinasikan. Pewarisan merupakan sarana untuk meniru dan memperbaiki yang dilakukan secara jelas dan bersih.

d. Komposisi(*compotition*)

Kita mendefinisikan hubungan komposisi sebagai menghubungkan kelas rakitan dan kelas komponen. Rakitan dengan banyak jenis komponen berkorespondensi dengan banyak *link* komposisi kita mendefinisikan tiap pasangan sebagai satu komposisi.

e. Polymorphism

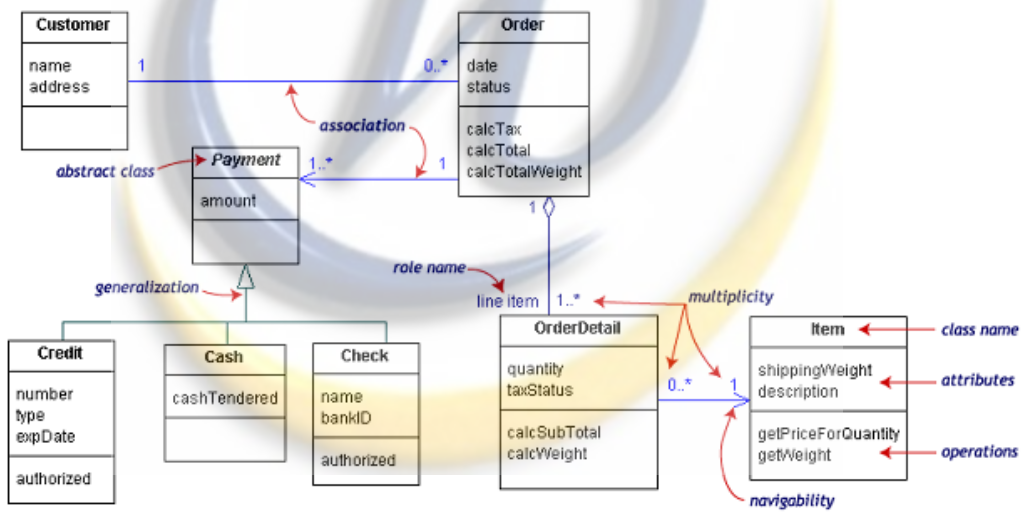
Adalah kemampuan dapat memperoleh beberapa bentuk. Pada pengembangan berorientasi objek yang dimaksud dapat memperoleh beberapa bentuk adalah satu variabel mempunyai kemampuan pda saat jalan dicantolkan ke objek – objek yang bertipe beda.

2.5.1 Pengembangan Perangkat Lunak Berorientasi Objek Menggunakan UML (*Unified Modelling Language*) [12]

UML adalah bahasa grafis untuk mendokumentasi, menspesifikasi, dan membangun sistem perangkat lunak. UML menjelaskan apa - apa saja yang harus sistem lakukan, bukan menjelaskan bagaimana sistem melakukan hal – hal tersebut. UML menyediakan sejumlah diagram untuk mengekspresikan pemodelan berorientasi objek yang dilakukan, yaitu:

1. Diagram kelas(*class diagram*)

Diagram ini menunjukkan sekumpulan kelas, *interface* dan kolaborasi dan keterhubungannya. Diagram kelas ditujukan untuk pandangan statik terhadap sistem. Contoh Diagram kelas:

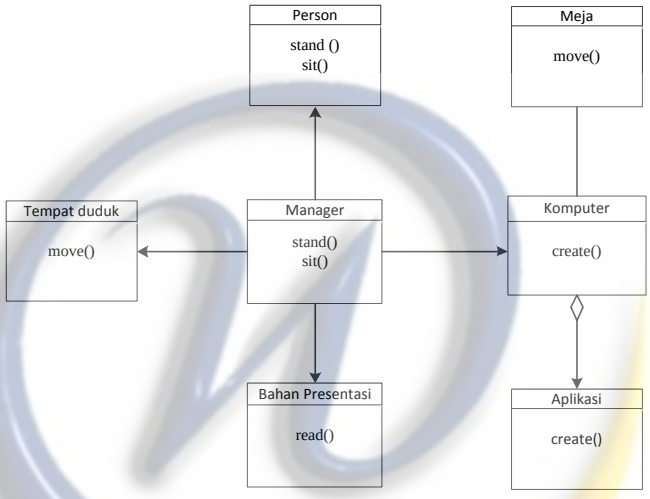


Gambar 2.4 Diagram class

2. Diagram objek (*object diagram*)

Diagram ini menunjukkan sekumpulan objek dan keterhubungannya dan menunjukkan potongan statik dari instan-instan yang ada di diagram kelas. Diagram ini untuk memperlihatkan satu prototipe atau kasus tertentu yang mungkin terjadi. Diagram objek menyediakan notasi grafis formal guna memodelkan objek, kelas dan

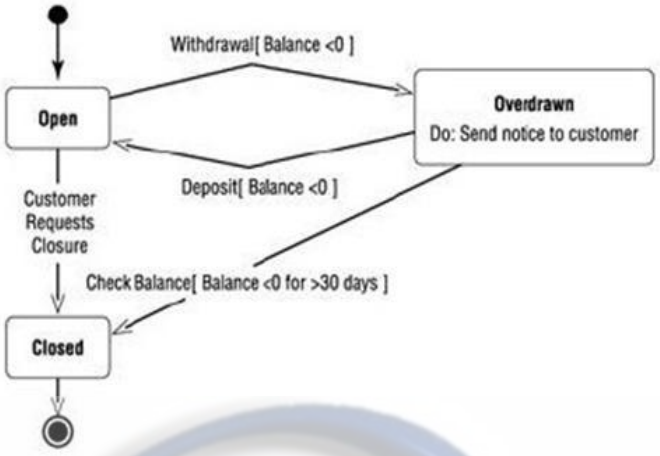
saling keterhubungan. Diagram objek berguna untuk *abstract modeling* dan perancangan program-program sesungguhnya. Pada pendekatan ini, bentuk dasar dari sistem perangkat lunak adalah objek atau kelas. Kelas adalah deskripsi dari objek-objek yang *common*. Setiap objek mempunyai identitas, *state* dan perilaku. Contoh Diagram objek :



Gambar 2.5 Diagram objek

3. Diagram komponen (*component diagram*)

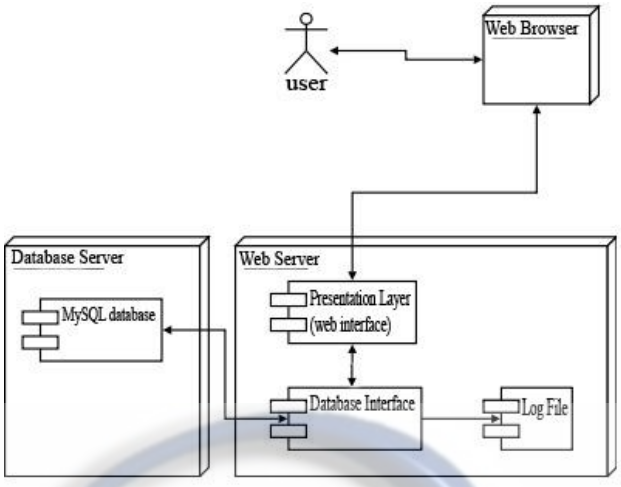
Diagram ini menunjukkan organisasi dan kebergantungan di antara sekumpulan komponen. Diagram ini merupakan pandangan statik terhadap implementasi sistem. Contoh Diagram komponen :



Gambar 2.6 Diagram komponen

4. Diagram deploymen (*deployment diagram*)

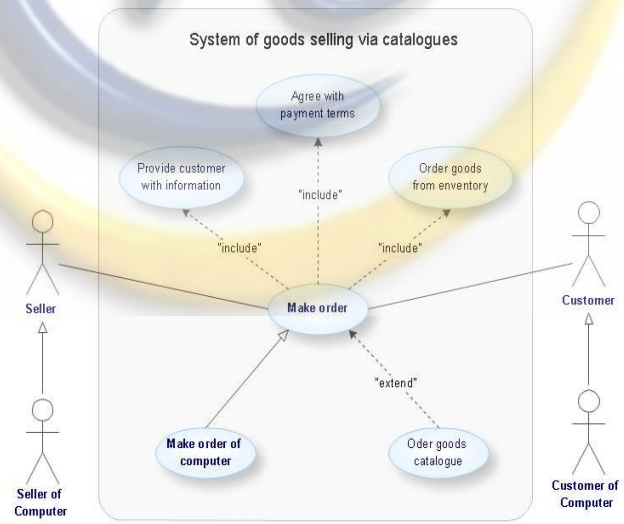
Diagram ini menunjukkan konfigurasi pemrosesan saat jalan dan komponen-komponen yang terdapat di dalamnya. Diagram ini merupakan pandangan statik dari arsitektur. Pilihan model dan diagram yang digunakan dipengaruhi oleh bagaimana persoalan ditangani dan bagaimana solusi dibentuk. Abstraksi, fokus pada yang relevan sambil mengabaikan rincian-rincian yang tidak relevan merupakan kuncinya. Karena itu, setiap sistem kompleks perlu didekati melalui sekumpulan pandangan model yang hampir idependen. Tidak ada satu pandangan pada level-level berbeda. Model-model yang terbaik dapat dikoneksikan ke kenyataan. Contoh Diagram *deployment* :



Gambar 2.7 Diagram *deployment*

5. Diagram *use-case*(*use-case diagram*)

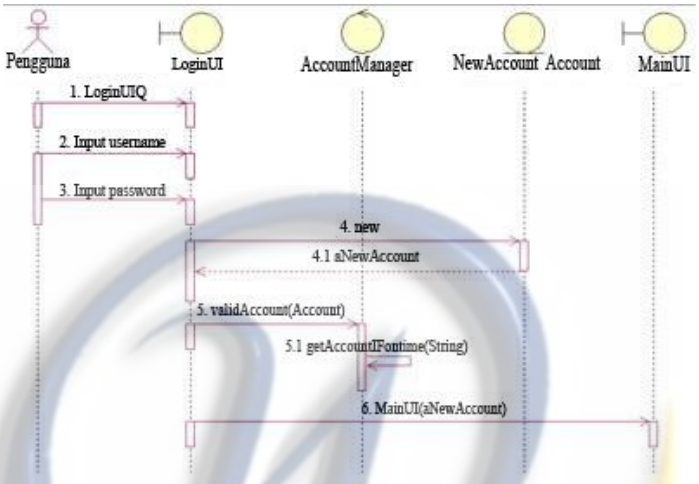
Diagram ini menunjukkan sekumpulan kasus fungsional dan aktor (jenis kelas khusus) dan keterhubungannya. Contoh Diagram *use case*:



Gambar 2.8 Diagram *use-case*

6. Diagram sekuen (*sequence diagram*)

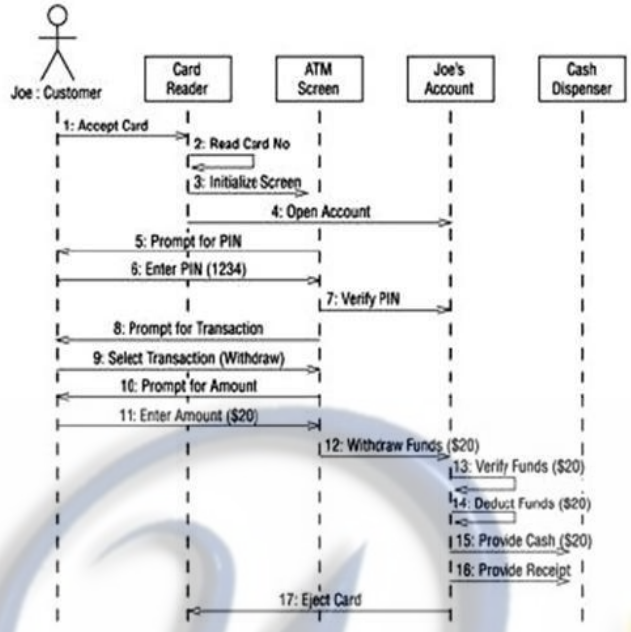
Diagram ini menunjukkan interaksi yang terjadi antar objek. Diagram ini merupakan pandangan dinamis terhadap sistem. Diagram ini menekankan pada basis keberurutan waktu dari pesan-pesan yang terjadi. Contoh Diagram sequen :



Gambar 2.9 Diagram sekuen

7. Diagram kolaborasi *(colaboration diagram*)

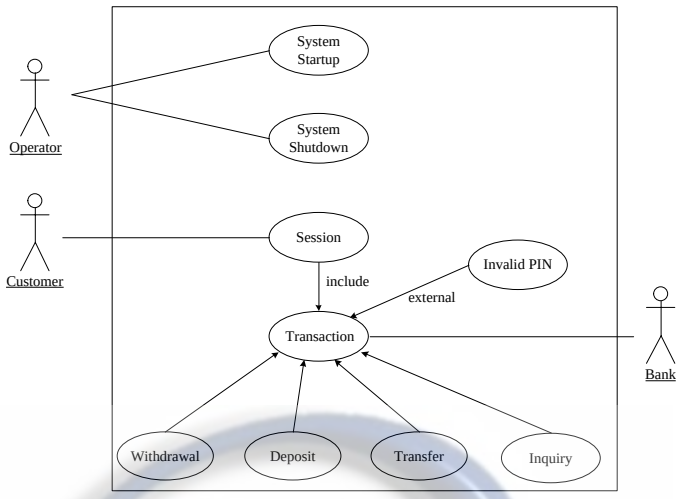
Diagram ini juga merupakan diagram interaksi. Diagram ini menekankan pada organisasi struktur dari objek-objek yang mengirim dan menerima pesan. Contoh Diagram kolaborasi :



Gambar 2.10 Diagram kolaborasi

8. Diagram *statechartt* (*statechart diagram*)

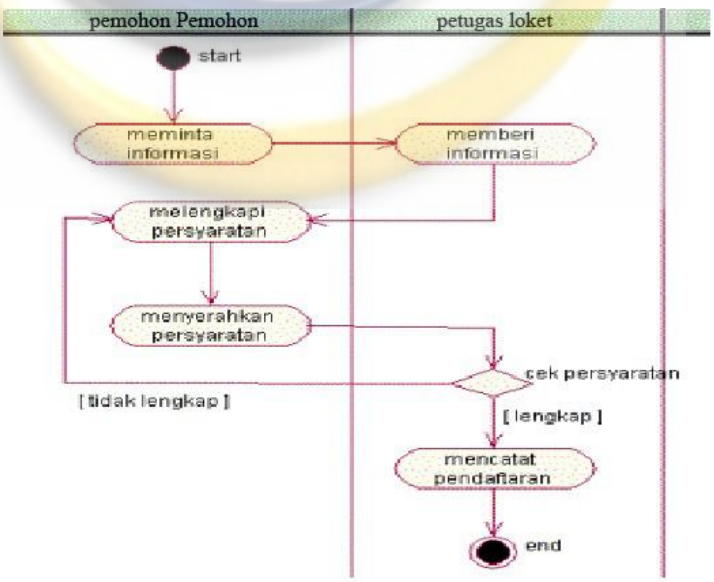
Diagram ini adalah *state-machine diagram,* berisi *state,* transisi, kejadian dan aktivitas. *Statechart* merupakan pandangan dinamis dari system. Diagram ini penting dalam memodelkan perilaku antarmuka, kelas, kolaborasi dan menekankan pada urutan kejadian. Penting untuk sistem reaktif yang dipicu kejadian di dunia nyata. Contoh Diagram bagian :



Gambar 2.11 Statechart Diagram

9. Diagram aktifitas (*activity diagram*)

Diagram ini untuk menunjukkan aliran aktivitas di sistem. Diagram ini adalah pandangan dinamis dari sistem. Diagram ini penting untuk memodelkan fungsi sistem dan menekankan pada aliran kendali di antara objek-objek. Contoh Diagram aktivitas :



Gambar 2.12 Diagram aktivitas