BAB II
LANDASAN TEORI

II.1 Pelestarian Permainan Tradisional

Saat ini dengan perkembangan teknologi yang semakin maju permainan tradisional kurang diminati. Belum ada penelitian yang pasti mengenai penyebabnya, namun diduga perkembangan games yang semakin canggih membuat anak-anak lebih tertarik memainkan games dibandingkan permainan tradisional. Menurut Zaini Alif pendiri komunitas Hong pusat kajian permainan rakyat mengenai kurangnya popularitas permainan tradisional di kalangan anak-anak masa kini adalah orang Indonesia itu mengambang maksudnya tidak bisa berakar pada budayanya dan tidak akan berkembang sedikitpun terhadap kemajuannya. Jadi budaya-budaya luar masuk dengan mudahnya dan membuat kebudayaan lokal terlupakan karena orang Indonesia tidak punya pegangan terhadap ilmu-ilmu tradisi sehingga tidak mencengkram alam ranah budayanya [7].

Padahal permainan tradisional memiliki nilai-nilai kearifan lokal sebagai pandangan hidup yang menjadi acuan berperilaku suku bangsa tersebut sehingga mulai terlupakannya permainan tradisional tentunya berpengaruh besar terhadap nilai moral yang dimiliki generasi penerus [8].

Selain itu terdapat aspek positif yang dapat mempengaruhi psikologi anak. Misalnya, permainan yang menggunakan alat dapat memupuk anak untuk berpikir kreatif serta peka terhadap lingkungan dengan memanfaatkan segala yang ada di lingkungan untuk dibuat mainan, lalu permainan tanpa menggunakan alat dan dimainkan secara berkelompok mengajari nilai-nilai kebersamaan, kejujuran, tanggung jawab, sikap lapang dada dan mau menerima kekalahan, dorongan berprestasi, dan taat pada aturan [2].

II.2 Perceptual Computing

Perceptual computing atau dikenal sebagai komputasi perseptual adalah sebuah metodologi yang membantu kita dalam melakukan penilaian subjektif

dimana *perceptual computer* (*Per-C*) merupakan instansiasi dari perceptual computing itu sendiri. Lotfi Zadeh (1996, 1999, 2008), ayah dari logika fuzzy, menciptakan istilah "*Computing With Words*" (*CWW*) yang artinya komputasi melalui kata-kata sehingga objek dari komputasinya merupakan kata-kata yang diambil dari *natural language* atau bahasa natural. Ini terisnspirasi oleh kemampuan manusia yang luar biasa untuk melakukan berbagai tugas-tugas fisik dan mental tanpa pengukuran dan perhitungan. *CWW* mungkin memiliki pengaruh penting terhadap bagaimana manusia membuat keputusan yang rasional berdasarkan persepsi-dalam lingkungan ketidaktepatan, ketidakpastian, dan kebenaran parsial. [9]

Dengan kata lain *perceptual computing* mengubah cara interaksi manusia dengan komputer menggunakan bahasa natural seperti kata-kata, karena *perceptual computer* (*Per-C*) mampu membuat keputusan yang rasional berdasarkan persepsi selayaknya manusia. *Natural language* atau bahasa natural yang kita kenal tidak hanya melalui kata-kata, bahasa tubuh atau *gesture*, maupun bahasa isyarat juga termasuk didalam *natural language* maka *perceptual computer* (*Per-C*) juga harus dapat mengenali bahasa tubuh, serta bahasa isyarat tersebut.

II.3 Intel Perceptual Computing

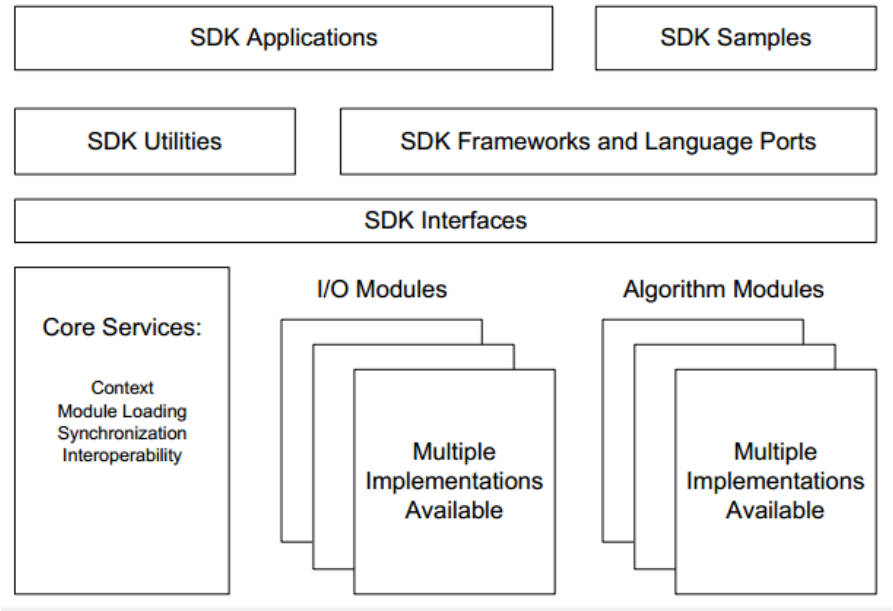
Intel Perceptual Computing merupakan produk Intel yang mengubah cara interaksi manusia dengan komputer dengan memanfaatkan pengenalan suara, pengenalan gerakan tangan, pengenalan wajah, bahkan augmented reality. Dengan kamera Intel Perceptual Computing, dan Intel Perceptual Computing SDK memungkinkan developer membangun perangkat lunak dengan fitur-fitur tersebut.

Kamera Intel Perceptual Computing merupakan kamera eksternal yang dihubungkan melalui USB layaknya webcam eksternal, namun disertai sensor yang dapat melakukan pengenalan suara, gerakan tangan, pengenalan wajah, hingga augmented reality.



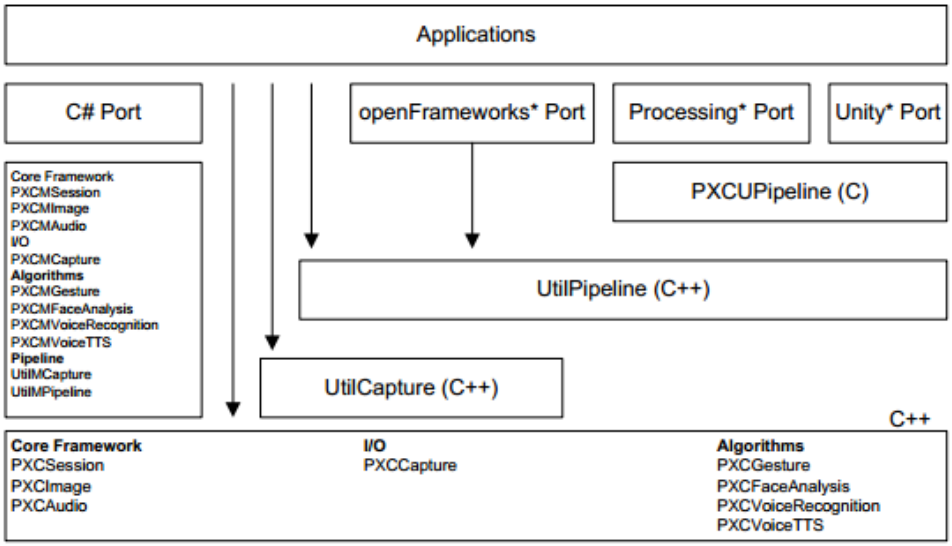
Gambar II.1 Kamera Intel Perceptual Computing

Intel Perceptual SDK adalah library *pattern detection* dan dan penerapan *recognition algorithm* yang terpapar melalui antarmuka yang sudah distandarkan. Tujuan library ini adalah untuk mengurangi hambatan pengembang untuk menggunakan algoritma ini dan mengalihkan fokus pengembang dari kode yang rinci menjadi lebih kearah berinovasi pada penggunaan algoritma ini untuk interaksi manusia dan komputer generasi berikutnya. [4]



Gambar II.2 Arsitektur Library Perceptual SDK

Arsitektur dari library SDK seperti yang tertera pada gambar II.2 terdiri dari beberapa lapisan komponen. Esensi dari fungsionalitas SDK terdapat pada modul I/O, dan modul algoritma. I/O modul berperan sebagai sumber *retrieve data* dari *input device* atau *sink* yang mengkonsumsi data ke *output device.* Modul algoritma termasuk didalamnya berbagai *pattern detection* dan algoritma *recognition algorithm* yang merupakan unsur penting dalam inovasi *human computer experience,* seperti *face recognition, gesture recognition, voice recognition,* dan *text to speech*.



Gambar II.3 Rangkaian Dasar Perceptual SDK

Seperti yang terlihat pada gambar II.3 diatas, rangkai dasar yang membangun SDK adalah serangkaian dari *interface* C++, yang mendefinisikan fungsionalitas dari *core framework*, modul I/O, dan modul algoritma. Interface ini menyediakan *raw funcionalites*. Contohnya, *interface* PXCCapture mengambil sampel suara (PXCAudio), atau sampel gambar (PXCImage) dari sensor input. *Interface* PXCCapture melakukan *finger tracking,* dan *gesture recognition* sesuai dengan tipe input gambar sampel.

SDK menyediakan beberapa kelas *utility* untuk menyederhanakan pemrograman aplikasi. Kelas UtilCapture merupakan ekstensi dari *interface* PXCCapture untuk membantu menghubungkan modul algoritma dengan *input*

device, dan melakukan *stream* data. Kelas UtilPipeline menyediakan *interface* sederhana untuk penggunaan yang terbatas pada *gesture, face,* dan *voice algorithm*.

II.3.1 Modul Intel Perceptual Computing

Kendali dalam permainan merupakan terapan dari modul-modul Intel Perceptual Computing, seperti *hand tracking, gesture recognition,* dan *face tracking.* Berikut penjelasan dari setiap modulnya.

1. Hand Tracking

Sebelum membahas *hand tracking,* penting untuk memahami struktur dari *GeoNode* dan konsep pelabelan. Label mendefinisikan bagaimana *body-related objects* harus diidentifikasi dalam *scene* dan dibagi menjadi dua bidang utama, yaitu *full-body label* yang menyediakan informasi berkaitan dengan posisi tulang yang Nampak, dan *hand-detailed label* menyediakan informasi rinci dari objek tangan. Struktur *GeoNode* menjelaskan node geometris yang memberikan data relevan terkait objek yang sedang dilacak. Di baris kode di bawah ini, dengan asumsi bahwa *pipeline* telah dibuat dan diinisialisasikan. Kita menciptakan variabel *body* dan *hand label* diperlukan yang menentukan objek untuk dilacak. Dalam hal ini kita mencoba untuk melakukan *recognize* terhadap bagian tengah dari tangan.

Dalam fungsi *Update,* kita mencoba untuk melakukan pengolahan *acquire frame,* untuk memastikan kita akan langsung keluar jika tidak ada *acquired frame*. Jika sebuah *acquire frame* berhasil, kita memanggil fungsi QueryGeoNode. Fungsi QueryGeoNode mengembalikan rincian dari struktur *GeoNode.* Panggilan membutuhkan dua parameter, yaitu referensi untuk GeoNode yang struktur dimana hasil data akan disimpan, dan label tangan OR, yang menetapkan bahwa kita mencari bagian tengah dari tangan. Ketika permintaan untuk pusat-posisi tangan kembali benar, kita memiliki data yang valid disimpan dalam *instance* dari struktur *GeoNode.* Menggunakan data dari struktur *GeoNode,* kita dapat mulai menentukan interaksi pengguna dengan mendapatkan posisi tangan pengguna, kemudian menemukan dan memetakan aksi yang terjadi pada sistem yang kita buat. Setelah proses memperoleh hasil, kita diharuskan untuk melakukan *release frame* dan *resources.*

```
        private PXCMGesture.GeoNode ndata;
        private PXCMGesture.GeoNode.Label bodyLabel =
        PXCMGesture.GeoNode.Label.LABEL_BODY_HAND_PRIMARY;
        private PXCMGesture.GeoNode.Label handLabel =
        PXCMGesture.GeoNode.Label.LABEL_HAND_MIDDLE;
        void Update()
        {
                if (!pp.AcquireFrame(false))
                return;
                if (pp.QueryGeoNode(bodyLabel | handLabel, out ndata))
        {
                //Get the standard hand position
                HandPosition.x = ndata.positionWorld.x;
                HandPosition.y = ndata.positionWorld.y;
                HandPosition.z = ndata.positionWorld.z;
        }
                pp.ReleaseFrame();
}
```

Gambar II.4 Contoh *Hand Tracking*

2. Gesture Recognition

Modul Gesture memanfaatkan fungsi *QueryGesture* dalam banyak cara yang sama seperti fungsi *QueryGeoNode. QueryGeoNode* menngambil dua parameter, *reference* terhadap *PXCMGesture.Gesture struct,* dan label yang menyatakan *gesture* untuk dilakukan *recognize*. Kode di bawah memanfaatkan LABEL_ANY yang menyatakan bahwa kita sedang mencari *gesture* apapun. Hal ini dilakukan agar jika gerakan setiap di *recognize*, kita dapat bergerak dan menentukan *gesture* mana yang telah diidentifikasi dan bagaimana harus ditangani. QueryGesture mengembalikan *true* ketika struct *gesture* menangkap *gesture* yang valid atau *pose data,* selain itu akan mengembalikan *false*. Contoh di bawah ini menggunakan pose "*Big 5*" untuk menghentikan permainan.

```
PXCMGesture.Gesture gdata;
if (pp.QueryGesture(PXCMGesture.GeoNode.Label.LABEL_ANY, out
gdata))
        {
if(gdata.label == PXCMGesture.Gesture.Label.LABEL_POSE_BIG5)
gamePause = !gamePause;
        }
```

Gambar II.5 Contoh *Gesture Recognition*

3. Face Tracking

Dengan *face location data* aplikasi dapat menggunakan fungsi QueryData *PXCFaceAnalysis :: Detection* untuk mengambil data lokasi wajah. Aplikasi dapat menggunakan fungsi QueryFace untuk mengambil pengenal wajah-wajah yang tersedia di gambar. Berikut contoh *retrieve face location data.*

```
PXCFaceAnalysis::Detection *detector=face□DynamicCast
<PXCFaceAnalysis::Detection>();
for (int i=0;;i++) {
pxcUID fid; pxcU64 ts;
if (face□QueryFace(i,&fid,&ts)<PXC_STATUS_NO_ERROR) break;
PXCFaceAnalysis::Detection::Data data;
detector□QueryData(fid,&data);
// Process data
                }
```

Gambar II.6 Contoh *Retrieve Face Location Data*

II.4 Unity3D

Unity3D adalah *game engine* yang menyediakan *tools* yang cukup lengkap dalam pembuatan *game* seperti membuat karakter, dan objek-objek *game* yang dapat berinteraksi dengan kode yang dibuat. Bahasa pemrograman yang *support* terhadap Unity3D diantaranya adalah: C#, Javascript, dan Booscript. Semua aset *game* hidup dalam direktori proyek sebagai file atau entitas yang dapat dikontrol

secara terpisah melalui kode. Unity3D menyediakan *scene files, 3D models, particle systems* beserta *tools* untuk membuat dan mengubah ini semua. [10]

Library yang ada pada Intel Perceptual Computing SDK dibuat berdasarkan bahasa *C#*, oleh karena itu *plugins* beserta *library* yang ada pada SDK tersebut memungkinkan untuk diimplementasikan kedalam lingkungan Unity3D.

II.4.1 Penggunaan Intel Perceptual Computing SDK Pada Unity3D

Prosedur umum pemrograman pada dasarnya sama untuk setiap *framework*, dan *game engine* yang didukung oleh Intel Perceptual Computing SDK, karena bersumber dari *library* yang sama. Prosedur tersebut diantaranya adalah *initialization, data processing*, dan *closing down*.

1. Initialization

Aplikasi menggunakan fungsi *Init* untuk melakukan initialize terhadap *pipeline*, yang dapat menjadi kombinasi warna, wajah, serta *gesture processing.*

2. Data Processing

Fungsi *AcquireFrame* menunggu pengolahan frame untuk menyelesaikan, dan mengunci hasil pengolahan. Fungsi *ReleaseFrame* untuk melepaskan, mengunci, dan mempersiapkan diri untuk pengolahan frame berikutnya. Di antara fungsi *AcquireFrame* dan fungsi *ReleaseFrame*, aplikasi dapat memanggil *Query series* dari fungsi untuk mengambil warna, wajah, dan hasil *gesture processing*.

3. Closing Down

Aplikasi menggunakan fungsi *Close* untuk menutup *pipeline*, serta membebaskan semua *resource*.

II.4.2 Contoh Penggunaan Intel Perceptual Computing SDK Pada Unity3D

Tatacara penggunaan Intel Perceptual Computing SDK pada Unity adalah sebagai berikut. Dalam folder *Assets* buat sebuah direktori dengan nama *Plugins*. Setelah itu taruh semua file yang dibutuhkan dari *SDK* seperti *libpxcupipeline.dll*, *pxcm-structures.cs*, dan *pxcupipeline.cs* ke dalam direktori *Plugins*. Berikut

merupakan contoh penggunaan Intel Perceptual Computing SDK pada lingkungan Unity 3D.

```
using UnityEngine;
using System;
using System.Runtime.InteropServices;
public class TexturePlayback : MonoBehaviour {
	private Color[] px;
	private Texture2D tt;
	private PXCUPipeline pp;
	void Start () {
		pp=new PXCUPipeline();
		pp.Init(PXCUPipeline.Mode.GESTURE);
		int[] size=new int[2];
		pp.QueryLabelMapSize(ref size);
		tt=new
	Texture2D(size[0],size[1],TextureFormat.ARGB32,false);
		renderer.material.mainTexture=tt;
	}
	void OnDisable() {
		pp.Close();
	}
	void Update () {
		if (!pp.AcquireFrame(false)) return;
		if (pp.QueryLabelMapAsImage(tt)) tt.Apply();
		pp.ReleaseFrame();
	}
}
```

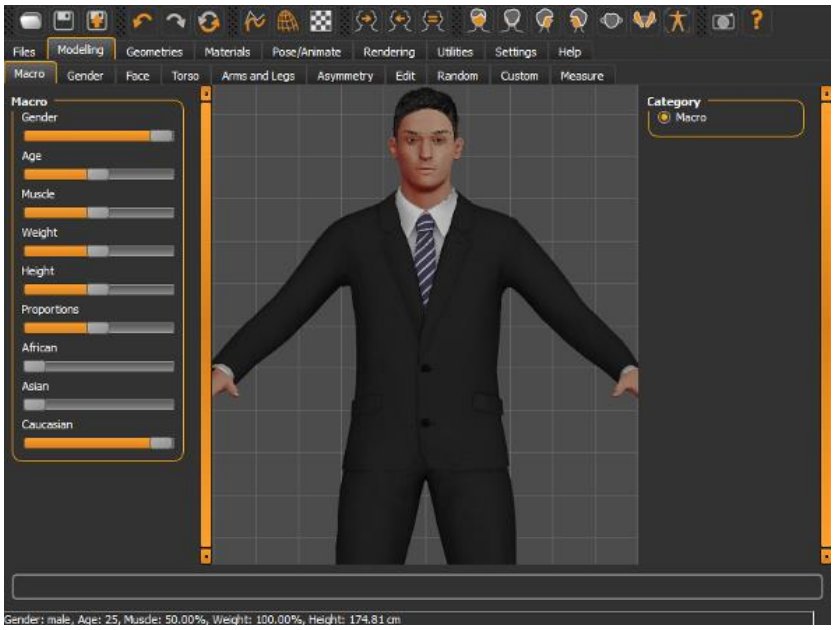
Gambar II.7 Contoh pengggunaan Intel Perceptual Computing SDK

II.5 Blender

Blender adalah open source *3D creation suite*. Mendukung keseluruhan dari *3D pipeline-modelling*, *rigging*, animasi, simulasi, *rendering*, *compositing*, dan *motion tracking*, bahkan *video editing* dan pembuatan *game*.

II.6 Make Human

MakeHuman adalah *tool* yang didesain untuk menyederhanakan pembuatan *virtual human* dengan memanfaatkan *Graphical User Interface* (GUI). *Virtual human* merupakan cabang yang lebih spesifik di bidang *3D Modelling*. Tim MakeHuman berfokus terhadap cabang tersebut untuk mencapai tingkat kualitas serta kemudahan dalam memodelkan *virtual human*. Kegunaan utamanya adalah menghasilkan *virtual human* yang realistis secara cepat dengan hanya beberapa klik pada *mouse*, serta dapat melakukan *render* dan *export* untuk digunakan dalam projek lainnya.



Gambar II.8 Make Human

Virtual human diciptakan melalui manipulasi dari kontrol yang memungkinkan untuk dilakukan *blending* terhadap *human attribute* yang berbeda untuk membuat karakter manusia *3D* yang unik. Kontrol dimaksudkan untuk menyediakan cara sederhana terhadap pengguna dalam memanipulasi *virtual human* yang dibuat untuk mencapai jangkauan terluas dari bentuk manusia. Atribut kontrol yang dibagi menjadi dua bagian, yaitu makro, dan detail. Makro menangani karakteristik manusia secara keseluruhan, seperti jenis kelamin, usia, tinggi, berat,

dan etnis. Detail memungkinkan karakter untuk lebih disempurnakan dengan menangani hal-hal rinci seperti bentuk mata, panjang jari, dan lain sebagainya. [11]

II.7 Alat Pemodelan

Alat pemodelan digunakan untuk memodelkan konsep-konsep dan perancangan yang nantinya diimplementasikan. Alat pemodelan yang digunakan dalam pembangunan game ini adalah MDA dan UML.

II.7.1 MDA

Untuk menentukan desain permainan yang akan digunakan dalam pembuatan game digunakan metode MDA (Mechanics, Dynamics, and Aesthetics) untuk memaksimalkan pembangunan aplikasi terkait [6]. Adapun model-model yang ada pada metode ini antaralain :

1. Model *Aesthetic*

Model ini digunakan sebagai landasan untuk mendeskripsikan mekanisme dari sebuah game. Dalam hal ini dilakukan penelitian serta analisis mengenai hal-hal apa saja yang menarik dan tidak menarik minat pengguna untuk memainkan games, sebab *aesthetic* berkaitan dengan perasaan "*fun*" atau kesenangan yang diperoleh selama memainkan *game*. Setelah dilakukan analisis lebih lanjut untuk ketahap selanjutnya yaitu membuat model *Dynamic* permainan.

2. Model *Dynamyc*

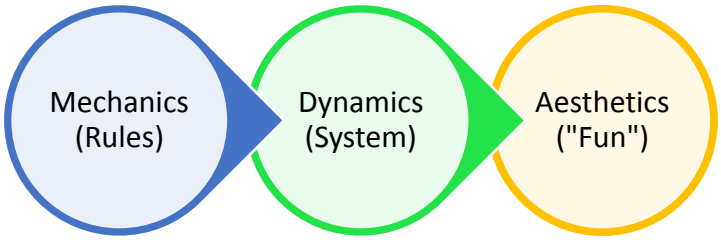
Pada model ini diciptakan kebutuhan-kebutuhan dari permasalahan yang didapatkan sebelumnya dari hasil analisis pada model *aesthetic*. Hasil dari analisa kemudian akan dilakukan penciptaan kebutuhan yang harus dibangun dan diimplementasikan untuk pemecahan masalah tersebut. Penciptaan tersebut dapat dibangun terkait dengan *challenge, expression, dramatical tension,* serta *fellowship* dimana setiap komponen ini akan memperlihatkan umpan balik dari pengguna nantinya. Model *dynamic* memperlihatkan hal dinamis yang berubah sepanjang alur permainan, sehingga dapat pula diartikan sebagai sistem yang ada pada *gameplay*.

3. *Mechanic*

Setelah selesai menganalisa apa saja yang akan dibangun pada permainan nantinya kemudian dilakukan penciptaan mekanisme secara lebih mendetail. Secara garis besar mechanic dalam sebuah *gameplay* merupakan aturan-aturan yang menyusun alur permainan. Pada game kompilasi permainan tradisional akan dilakukan penciptaan mekanisme permainan pada setiap *gameplay* seperti gesture yang akan digunakan. Pada model ini semua perancangan akan memperlihatkan mekanisme permainan secara lebih detail karena perancangan yang dilakukan akan berkaitan dengan pembangunan *gameplay* nantinya.

4. *Tuning*

Pada model ini, hasil perancangan mekanisme *gameplay* akan dianalisis ulang sampai tahap menemukan bahwa tujuan yang dihadapi dari masalah-masalah sebelumnya dapat tercapai. Model ini akan dilakukan tiga tahapan, pertama pada saat perancangan sistem, unit testing, serta evaluasi permainan.



Gambar II.9 MDA Model

II.7.2 UML

UML (*Unified Modeling Language*) adalah bahasa standar yang digunakan untuk memudahkan pengembang dalam menspesifikasikan, memvisualisasikan, dan mendokumentasikan model.

1. *Usecase* Diagram

Diagram yang bertujuan untuk memvisualisasikan persyaratan fungsional dari sistem, termasuk hubungan "aktor"(manusia yang akan berinteraksi dengan

sistem) untuk proses penting, serta hubungan antara *usecase* yang berbeda, biasanya mendefinisikan interaksi antara peran (dikenal dalam UML sebagai "aktor") dengan sistem untuk mencapai tujuan.

2. *Activity* Diagram

Aliran prosedural kontrol antara dua atau lebih kelas objek saat memproses suatu kegiatan dapat ditunjukkan dengan diagram aktivitas.

3. *Class* Diagram

Class diagram mendeskripsikan tipe-tipe objek yang digunkan dalam sistem dan keterhubungan antar kelas. Kelas merepresentasikan sesuatu yang ditangani oleh sistem. Kelas dapat berhubungan dengan yang lain melalui berbagai cara: *associated* (terhubung satu sama lain), *dependent* (tergantung/ menggunakan kelas yang lain), *specialed* (satu kelas merupakan spesialisasi dari kelas lainnya), atau *package* (group bersama sebagai satu unit).

4. Sequence Diagram

Diagram *sequence* menunjukkan aliran rinci untuk *usecase* tertentu atau bahkan hanya sebagian dari *usecase* tertentu. Ini menunjukkan panggilan antara objek berbeda sesuai urutan mereka dan dapat menunjukkan pada tingkat rinci, panggilan yang berbeda untuk objek yang berbeda.