BAB 2
TINJAUAN PUSTAKA

2.1 Landasan Teori

Landasan teori merupakan penjelasan berbagai konsep dan teori – teori yang berkaitan dalam pembangunan aplikasi Tutorial Pembelajaran Sistem Kontrol. Beberapa teori yang terkait dengan pembangunan aplikasi Tutorial Pembelajaran Sistem Kontrol adalah tutorial, konsep dasar *bluetooth*, android, eclipse, perangkat *inverted* pendulum dan UML.

2.1.1 Tutorial Pembelajaran

Dalam kamus Besar Bahasa Indonesia, tutorial didefinisikan sebagai: pembimbingan kelas oleh seorang pengajar (tutor) untuk seorang mahasiswa atau sekelompok kecil mahasiswa atau pengajaran tambahan melalui tutor, sedangkan tutor didefinisikan sebagai: orang yang memberi pelajaran kepada seseorang atau sejumlah kecil siswa (di rumah, bukan di sekolah) atau dosen yang membimbing sejumlah mahasiswa di pelajarannya. Berdasarkan definisi tersebut, tutorial berarti mengajar orang lain atau memberikan bantuan belajar kepada seseorang. Kegiatan tutorial melibatkan orang yang mengajar/memberi bantuan yang disebut tutor dan orang yang belajar atau yang diberi bantuan belajar, yang disebut tutee. Di antara tutor dan tutee terdapat bahan belajar yang merupakan sumber ilmu yang dikaji oleh tutee bersama tutor, dan diantara tutor dan tutee terjadi interaksi atau komunikasi yang merupakan inti dari tutorial tersebut [**6**]. Menurut UU Nomor 20 tahun 2003 tentang Sistem Pendidikan Nasional, Pembelajaran adalah proses interaksi peserta didik dengan pendidik dan sumber belajar pada suatu lingkungan belajar.

Berdasarkan definisi tutorial dan definisi Pembelajaran, secara bahasa dapat disebutkan bahwa Tutorial Pembelajaran adalah proses pengajaran suatu bahan belajar dan interaksi antara peserta didik dan pendidik dalam suatu lingkungan belajar.

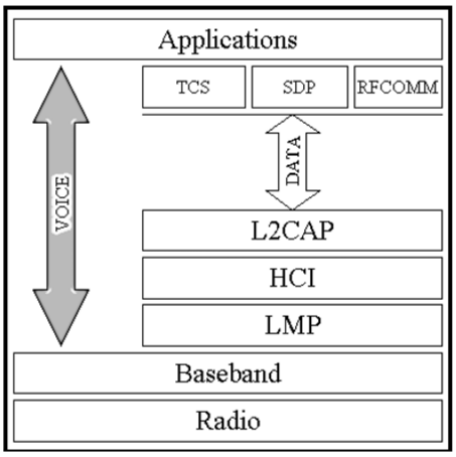
2.1.2 Konsep Dasar Bluetooth

Bluetooth adalah teknologi komunikasi data nirkabel(tanpa kabel) jarak pendek yang digunakan untuk menggantikan kabel sebagai media komunikasi data. Teknologi *bluetooth* memanfaatkan gelombang radio yang beroperasi pada kelompok ISM(*Industrial, Scientific, Medical*), bekerja pada frekuensi 2.4GHz dan sinyal *full duplex*. Pada umumnya, frekuensi 2.4GHz ISM tersedia dan tidak membutuhkan lisensi pada sebagian negara.

Keuntungan dari penggunaan teknologi *bluetooth* adalah memiliki daya yang kecil, dapat beroperasi lama, dan biaya implementasi yang relatif rendah, akan tetapi, teknologi *bluetooth* memiliki kekurangan, diantaranya, *bluetooth* dapat digunakan untuk membangun jaringan *adhoc* yang terbatas, hanya untuk delapan perangkat yang terdiri satu *Master* dan tujuh *active slave.* [**7**]

2.1.2.1 Protokol Bluetooth

Bluetooth melakukan pembagian tugas antara *host* dan sebuah *host controller*. Host bertanggung jawab untuk protokol lapisan seperti *Logical Link Control* dan Adaptasi Protokol (L2CAP) dan *Service Discovery Protocol*(SDP), yaitu fungsi yang dilakukan oleh perangkat komputasi seperti komputer, Sedangkan *host controller* bertanggung jawab untuk lapisan Radio, Baseband, dan *Link Manager Protocol*(LMP) yang dilaksanakan secara integrasi atau eksternal. *Host* dan *Host controller* saling mengirimkan informasi melalui *Host controller Interface*(HCI). Berdasarkan pengembangan *Bluetooth Interest Group,* penentuan protokol *bluetooth* terlihat seperti pada gambar 2.1. [**7**]



Gambar 2.1 Bluetooth Protocol dan hubungan antar layer

Fungsi dari masing-masing layer secara singkat dapat digambarkan sebagai berikut:

1. Layer radio adalah koneksi nirkabel fisik. Untuk menghindari gangguan dengan perangkat lain yang berkomunikasi pada pita ISM, modulasi ini didasarkan pada *fast frequency hopping*.
2. Layer baseband bertanggung jawab untuk mengendalikan dan mengirimkan paket data melalui radio link. Layer ini menyediakan saluran transmisi baik untuk data maupun suara. Lapisan baseband digunakan mempertahankan *Synchronous Connection-Oriented* (SCO) untuk link suara *dan Asynchronous Connectionless* (ACL) untuk data.
3. *Link Manager Protocol* (LMP) untuk menjalin hubungan dan mengelola piconet. Tanggung Jawab LMP juga termasuk otentikasi dan keamanan, dan pemantauan kualitas layanan.
4. Lapisan *Host Controller Interface* (HCI) adalah garis pemisah antara software dan hardware. Lapisan L2CAP dan lapisan di atasnya saat ini diimplementasikan dalam perangkat lunak, dan LMP dan lapisan di bawahnya berada di hardware. Lapisan HCI adalah driver interface untuk bus fisik yang menghubungkan dua komponen.

5. *Logical Link Control And Adaptation Protocol* (L2CAP) menerima data aplikasi dan menyesuaikan ke format *Bluetooth* dengan memperhatikan parameter *Quality of Service*.

2.1.2.2 Pendeteksian Posisi Pada Bluetooth

Berdasarkan pada parameter sinyal elektromagnetik yang dipilih untuk penentuan posisi, pendekatan dasar untuk sistem radiolokasi dapat dilakukan dengan berbagai cara pengukuran, yaitu:

1. Pengukuran Received Signal Strength Indicator (RSSI)
2. Pengukuran Link Quality (LQ) untuk menentukan lokasi dari bluetooth yang dipantau. Penggunaan link quality dapat pula dikombinasikan dengan menggunakan nilai RSSI, yang digunakan untuk menunjang keakuratan teknik radio lokasi.
3. Pengukuran Sudut Kedatangan (Angle Of Arrival – AOA)
4. Pengukuran berdasarkan Waktu (Time Based System)

2.1.2.3 Jaringan Piconet dan Scatternet

Piconet adalah sistem jaringan yang terdiri dari dua atau lebih (max 8) perangkat *Bluetooth* dalam jarak fisik yang dekat dan beroperasi pada channel yang sama menggunakan frekuensi yang sama urutan loncatannya. *Bluetooth* memungkinkan perangkat untuk membangun jaringan, baik *ad hoc* atau infrastruktur. Jaringan Infrastruktur menggunakan *Bluetooth Access Point* (AP) yang tetap, yang memfasilitasi komunikasi antar perangkat *Bluetooth*. Walaupun hanya satu perangkat yang berfungsi sebagai master untuk setiap *piconet*, *Time Division Multiplexing* (TDM) memungkinkan slave dalam satu piconet dapat juga bertindak sebagai master bagi *piconet* lain secara bersamaan, sehingga membentuk jaringan yang disebut *scatternet* yang lebih jauh jangkauannya. [7]

2.1.3 Bluetooth Serial Port Profile

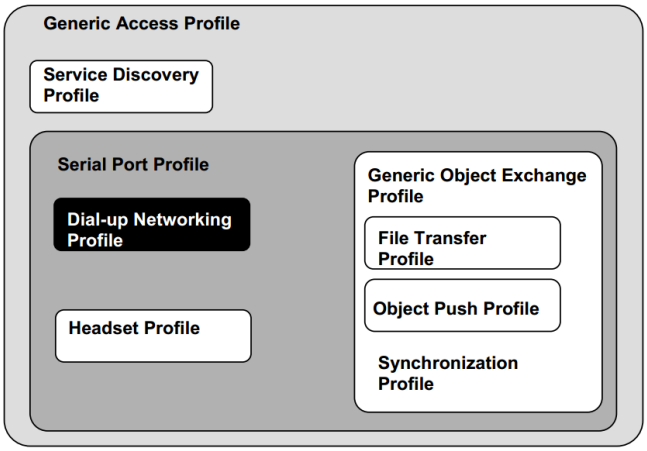
Profil ini mendefinisikan persyaratan untuk perangkat *Bluetooth* yang diperlukan untuk menyiapkan tiruan kabel data serial menggunakan RFCOMM antara dua rekan perangkat. Persyaratan dinyatakan dalam layanan yang disediakan untuk aplikasi, dan oleh mendefinisikan fitur dan prosedur yang diperlukan untuk interoperabilitas antara perangkat *Bluetooth*.

2.1.3.1 Ruang lingkup

Serial port profile yang mendefinisikan protokol dan prosedur yang akan digunakan oleh perangkat menggunakan *Bluetooth* untuk kabel serial emulasi RS232 (atau sejenisnya). Hal ini dapat digantikan oleh profil ini berkaitan dengan aplikasi-aplikasi legasi yang menggunakan *Bluetooth* sebagai kabel pengganti, melalui virtual serial port abstraksi (yang dalam dirinya sendiri beroperasi sistem ketergantungan).

2.1.3.2 Struktur Profil *Bluetooth*

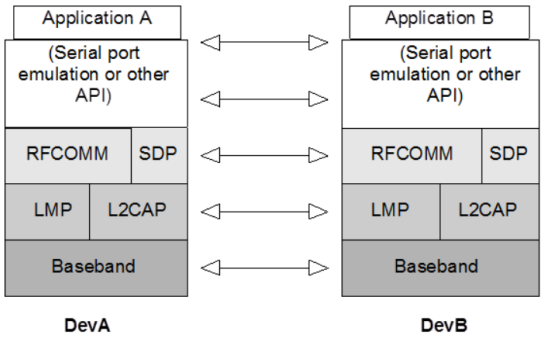
Pada Gambar 2.2, struktur profil *Bluetooth* dan dependensi profil digambarkan. Profil bergantung pada profil lain jika kembali menggunakan bagian profil, oleh secara implisit ataupun eksplisit merujuknya. Ketergantungan diilustrasikan pada gambar: memiliki profil ketergantungan pada profile(s) di mana itu terkandung-langsung dan tidak langsung



Gambar 2.2 Profil Bluetooth

2.1.3.3 *Profile Stack*

Pada gambar 2-3 menunjukkan protokol dan entitas yang digunakan dalam profil ini



Gambar 2.3 Model Protokol

Baseband LMP dan L2CAP adalah OSI layer 1 dan 2 protokol Bluetooth. RFCOMM adalah adaptasi Bluetooth GSM TS 07,10, menyediakan sebuah protokol transport emulasi serial port. SDP adalah *Bluetooth Service Discovery Protocol*. Aplikasi di kedua sisi yang biasanya aplikasi legasi, dapat dan bisa berkomunikasi melalui kabel serial. Namun demikian, aplikasi-aplikasi legasi tidak tahu tentang prosedur *Bluetooth* untuk mengatur tiruan kabel serial. Aplikasi non-legasi yang akan melakukan komunikasi serial *Bluetooth* harus juga mengikuti aturan yang ditentukan dalam profil ini. [**8**]

2.1.4 Android

Android adalah sebuah sistem operasi untuk perangkat mobile berbasis linux yang mencakup sistem operasi, middleware, dan aplikasi. Android menyediakan *platform* terbuka bagi para pengembang untuk membuat aplikasi mereka sendiri. Pada awalnya Google Inc. membeli Android Inc. yang berupakan sebuah perusahaan pendatang baru yang membuat perangkat lunak untuk ponsel. Kemudian untuk mengembangkan android, dibentuklah *Open Handset Alliance*, konsorsium dari 34 perusahaan perangkat keras, perangkat lunak, dan telekomunikasi, termasuk Google, HTC, Intel, Motorola, Qualcomm, T-Mobile, dan NVidia. [**9**]

2.1.4.1 The Dalvik Virtual Machine

Salah satu elemen kunci dari Android adalah *Dalvik Virtual Machine* (DVM). Android berjalan di dalam DVM bukan di *Java Virtual Machine* (JVM), sebenarnya banyak persamaannya dengan JVM seperti Java ME (*Java Mobile Edition*), tetapi Android menggunakan *Virtual Machine* sendiri yang dapat dikustomisasi dan dirancang untuk memastikan bahwa beberapa fitur – fitur berjalan lebih efisien pada perangkat *mobile*.

DVM adalah "Register based" sementara JVM adalah "stack based", DVM didisain dan ditulis oleh Dan Bornsten dan beberapa *engineers* Google alinnya. Jadi, bisa dikatakan "*Dalvik equals(Java) == false*". DVM menggunakan kernel Linux untuk menangani fungsionalitas tingkat rendah termasuk keamanan, *threading,* dan preses serta manajemen memori. Hal ini memungkinkan kita untuk menulis aplikasi C / C++ sama halnya seperti pada OS Linux kebanyakan. Meskipun dalam kenyataannya kita harus banyak memahami arsitektur dan proses sistem dari kernel Linux yang di gunakan Android tersebut.

Semua *hardware* yang berbasis Android dijalankan dengan menggunakan *Virtual Machine* untuk eksekusi aplikasi, pengembang tidak perlu khawatir tentang implementasi perangkat keras tertentu. DVM mengeksekusi *executable file,* sebuah format yang dioptimalkan untuk memastikan memori yang digunakan sangat kecil. *The executable file* diciptakan dengan mengubah kelas bahasa Java dan dikompilasi dengan menggunakan *tools* yang disediakan dalam SDK Android. [**9**]

2.1.4.2 Android SDK

Android SDK(*Software Development Kit*) adalah *tools* API(*Appplication Programing Interface*) yang diperlukan untuk memulai mengembangkan aplikasi pada platform Android menggunakan bahasa pemrograman Java. Android merupakan subset perangkat lunak untuk ponsel yang meliputi sistem operasi, *middleware* dan aplikasi kunci yang dirilis oleh Google. Saat ini disediakan Android SDK (*Software Development Kit*) sebagai alat bantu dan API untuk

mulai mengembangkan aplikasi pada *platform* Android menggunakan bahasa pemrograman Java. Sebagai *platform* aplikasi-netral, Android memberi kita kesempatan untuk membuat aplikasi yang kita butuhkan yang bukan merupakan bawaan ponsel. Beberapa fitur – fitur Android yang paling penting adalah: [**9**]

1. *Framework* aplikasi yang mendukung penggantian komponen dan *reuseable.*
2. DVM dioptimalkan untuk perangkat *mobile*.
3. *Integrated browser* berdasarkan *engine open source WebKit.*
4. Grafis yang dioptimalkan dan didukung oleg *libraries* grafis 2d, grafis 3d berdasarkan spesifikasi openGL ES 1,0 (Optional akselerasi *hardware*)
5. SQL Lite untuk menyimpan data
6. *Bluetooth*, EDGE, 3G, dan Wifi
7. *Media Support* yang mendukung audio, video, dan gambar (MPEG4, H.262, MP3, AAC, AMR, JPG, PNG, GIF) GSM telepon (tergantung *hardware*)
8. Kamera, GPS, kompas, dan accelerometer (tergantung Hardware)
9. Lingkungan *development* yang lengkap dan kaya termasuk perangkat emulator, *tools* untuk *debugging*, profil dan kinerja memori, dan *plugin* untuk IDE Eclipse.

2.1.4.3 Arsitektur Android

Secara garis besar arsitektur Android dapat dijelaskan dan digambarkan sebagai berikut: [**9**]

1. *Applications* dan *Widgets*

Application dan *Widgets* ini adalah layer dimana kita berhubungan dengan aplikasi saja, dimana biasanya kita *download* aplikasi kemudian kita lakukan instalasi dan jalankan aplikasi tersebut. Di *layer* terdapat aplikasi inti termasuk klien email, program SMS, kalender, peta, browser, kontak, dan lain-lain. Semua aplikasi ditulis menggunakan bahasa pemrograman Java.

2. *Application Frameworks*

Android adalah “Open Development Platform” yaitu Android menawarkan kepada pengembang atau memberi kemampuan kepada pengembang

untuk membangun aplikasi yang bagus dan inovatif. Pengembang bebas untuk mengakses perangkat keras, akses informasi *resources*, menjalankan *service background*, mengatur *alarm*, dan menambahkan status *notification*, dan sebagainya. Pengembang memiliki akses penuh menuju API *framework* seperti yang dilakukan oleh aplikasi yang kategori inti. Arsitektur aplikasi dirancang supaya dapat menggunakan kembali komponen yang sudah digunakan dengan mudah (*reuse*).

Sehingga dapat disimpulkan bahwa *Application frameworks* ini adalah *layer* dimana para pembuat aplikasi melakukan pengembangan/pembuatan aplikasi yang akan dijalankan di sistem operasi Android, karena pada *layer* inilah aplikasi dapat dirancang dan dibuat, seperti *content-providers* yang berupa SMS dan panggilan telepon. [**9**]

Komponen – komponen yang termasuk di dalam *Application Frameworks* adalah sebagai berikut:

a. *Views*
b. *Content Provider*
c. *Resource Manager*
d. *Notification Manager*
e. *Activity Manager*

3. *Libraries*

Libraries ini adalah *layer* dimana fitur – fitur Android berbeda, biasanya para pembuat aplikasi mengakses libraries untuk menjalankan aplikasinya. Berjalan di atas kernel, *layer* ini meliputi berbagai *library* C/C++ inti seperti Libc dan SSL, serta:

- *Libraries* media untuk pemutaran media audio dan video
- *Libraries* untuk manajemen tampilan
- *Libraries Graphics* mencakup SGL dan OpenGL untuk grafis 2D dan 3D
- *Libraries SQLite* untuk dukungan *database.*
- *Libraries* SSL dan Webkit terintegrasi dengan *web browser* dan *security*
- *Libraries LiveWebcore* mencakup modern *web browser* dengan *engine embeded web view*

- *Libraries* 3D yang mencakup implementasi OpenGL ES 1.0 API's

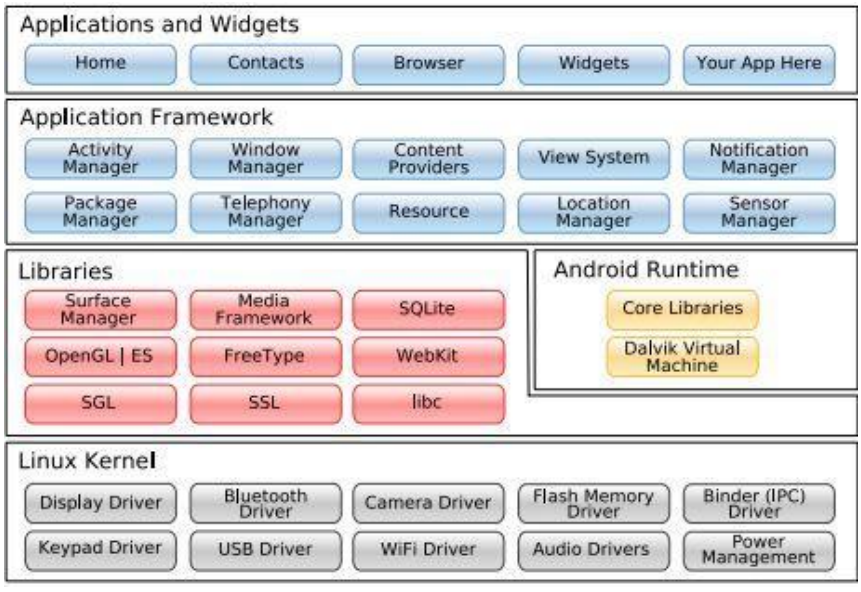
4. *Android Run Time*

Layer yang membuat aplikasi Android dapat dijalan kan dimana dalam prosesnya menggunakan implementasi Linux. *Dalvik Virtual Machine* (DVM) merupakan mesin yang membentuk dasar kerangka aplikasi Android. Di dalam Android *Run Time* dibagi menjadi dua bagian, yaitu:

- *Core Libraries*: Aplikasi Android dibangun dalambahasa Java, sementara *Dalvik* sebagai virtual mesinnya bukan virtual mesin Java, sehingga diperlukan *libraries* yang berfungsi untuk menterjemahkan bahasa Java/C yang ditangani oleh *Core Libraries.*
- *Dalvik Virtual Machine*: Virtual mesin berbasis register yang dioptimalkan untuk menjalankan fungsi – fungsi secara efisien, dimana merupakan pengembang yang mampu membuat Linux kernel untuk melakukan *threading* dan manajemen tingkat rendah.

5. Linux *Kernel*

Linux *Kernel* adalah *layer* dimana inti dari *operating system* dari Android itu berada. Berisi *file – file system* yang mengatur sistem *processing*, *memory*, *resource*, *drivers*, dan sistem – sistem operasi Android lainnya. Linux *Kernel* yang digunakan Android adalah Linux *kernel Release* 2.6. [**9**]



Gambar 2.4 Arsitektur Android

2.1.5 Inverted Pendulum

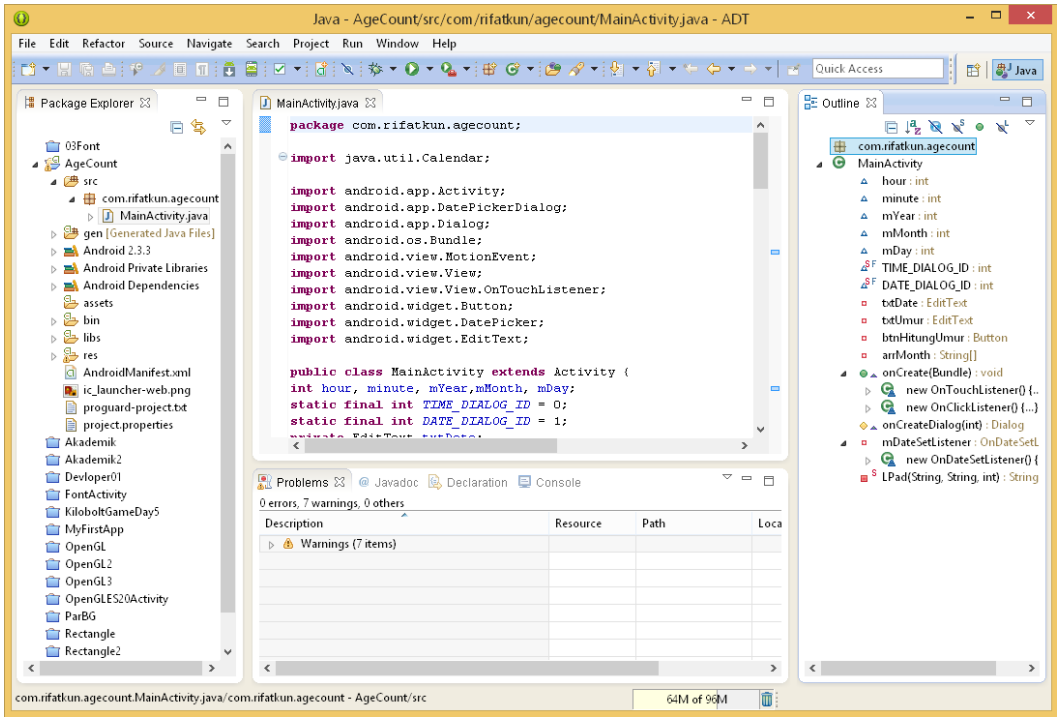
Inverted Pendulum adalah bandul yang memiliki massa yang berputar di atas titik. Hal ini sering dilakukan dengan berporos titik menjulang pada kereta yang dapat bergerak horisontal dan dapat disebut sebagai tiang dan kereta. Sedangkan biasa bandul stabil ketika gantung ke bawah, sebuah bandul terbalik adalah memiliki sifat tidak stabil, dan harus seimbang aktif untuk tetap tegak, baik oleh tenaga putaran (*tourqe*) yang berlaku pada persendian atau dengan menggerakan berporos horizontal titik sebagai bagian dari arus balik sistem [**10**].

2.1.6 Eclipse

Eclipse adalah sebuah IDE (Integrated Development Environment) untuk mengembangkan perangkat lunak dan dapat dijalankan di semua platform (platform-independent). Berikut ini adalah sifat dari Eclipse:

1. Multi-platform: Target sistem operasi Eclipse adalah Microsoft Windows, Linux, Solaris, AIX, HP-UX dan Mac OS X.
2. Mulit-language: Eclipse dikembangkan dengan bahasa pemrograman Java, akan tetapi Eclipse mendukung pengembangan aplikasi berbasis bahasa pemrograman lain seperti C/C++, Cobol, Python, Perl, PHP, dan lain sebagainya.
3. Multi-role: Selain sebagai IDE untuk pengembangan aplikasi. Eclipse pun bisa digunakan untuk aktivitas dalam siklus pengembangan perangkat lunak seperti dokumentasi, pengujian perangkat lunak, pengembangan web, dan lain sebagainya.

Pada saat ini, Eclipse merupakan salah satu IDE favorit karena gratis dan open source. Open source berarti setiap orang boleh melihat kode pemrograman perangkat lunak ini. Selain itu, kelebihan dari Eclipse yang membuatnya populer adalah kemampuannya untuk dapat dikembangkan oleh pengguna dengan membuat komponen yang disebut plug-in [**11**]. Lingkungan kerja atau antarmuka aplikasi Eclipse dapat dilihat pada gambar 2.5.



Gambar 2.5 Tampilan Antarmuka Eclipse

2.1.6.1 1. Sejarah

Eclipse awalnya dikembangkan oleh IBM untuk menggantikan perangkat lunak pengembangan IBM Visual Age for Java 4.0. Produk Eclipse ini diluncurkan oleh IBM pada tanggal 5 November 2001. IBM menginvestasikan US$ 40 juta untuk pengembangannya. Sejak 5 November 2001, konsorsium Eclipse Foundation mengambil alih pengembangan Eclipse lebih lanjut.

2.1.6.2 2. Arsitektur

Sejak versi 3.0, Eclipse pada dasarnya merupakan sebuah kernel. Apa yang dapat digunakan di dalam Eclipse sebenarnya adalah fungsi dari plug-in yang sudah dipasang (diinstal). Ini merupakan basis dari Eclipse yang dinamakan Rich Client Platform (RCP). Berikut ini adalah komponen yang membentuk RCP:

1. Core platform
2. OSGi
3. SWT (Standard Widget Toolkit)

4. JFace
5. Eclipse Workbench

Secara standar Eclipse selalu dilengkapi dengan JDT (Java Development Tools), plug-in yang membuat Eclipse kompatibel untuk mengembangkan program Java, dan PDE (Plug-in Development Environment) untuk mengembangkan plug-in baru. Eclipse beserta plug-in-nya diimplementasikan dalam bahasa pemrograman Java. Konsep Eclipse adalah IDE yang bersifat:

1. Terbuka (open),
2. Mudah diperluas (extensible) untuk apa saja
3. Tidak untuk sesuatu yang spesifik

Eclipse tidak saja untuk mengembangkan program Java, tetapi juga untuk berbagai macam keperluan. Perluasan apapun cukup dengan menginstal plug-in yang dibutuhkan. Apabila ingin mengembangkan program C/C++ maka telah terdapat plug-in CDT (C/C++ Development Tools) yang dapat dipasang di Eclipse untuk Eclipse menjadi perangkat untuk pengembangan C/C++.

Pengembangan secara visual bukan hal yang tidak mungkin oleh Eclipse, plug-in UML2 tersedia untuk membuat diagram UML. Dengan menggunakan PDE setiap orang bisa membuat plug-in sesuai dengan keinginannya.

2.1.6.3 3. Histori Eclipse

Sejak tahun 2006, Eclipse Foundation mengkoordinasikan peluncuran Eclipse secara rutin dan simultan yang dikenal dengan nama Simultaneous Release. Setiap versi peluncuran terdiri dari Eclipse Platform dan juga sejumlah proyek yang terlibat dalam proyek Eclipse. Tujuan sistem ini adalah untuk menyediakan distribusi Eclipse dengan fitur-fitur dan versi yang terstandarisasi. Hal ini juga dimaksudkan untuk mempermudah deployment dan maintenance untuk sistem enterprise, serta untuk kenyamanan. Peluncuran simultan dijadwalkan pada bulan Juni setiap tahunnya.

Tabel 2.1 Peluncuran Eclipse

Kode Peluncuran	Tanggal Peluncuran	Platform	Nama Proyek
Eclipse 3.0	28 Juni 2004	3.0	
Eclipse 3.1	28 Juni 2005	3.1	
Callisto	30 Juni 2006	3.2	Callisto projects
Europa	29 Juni 2007	3.3	Europa projects
Ganymede	25 Juni 2008	3.4	Ganymede projects
Galileo	24 Juni 2009	3.5	Galileo projects

2.1.7 UML

UML(*Unified Modeling Language*) merupakan bahasa visual untuk pemodelan dan komunikasi mengenai sebuah sistem dengan menggunakan diagram dan teks – teks pendukung untuk menspesifikasikan, menggambarkan, membangun, dan dokumentasi dari sistem perangkat lunak [**12**].

2.1.7.1 Sejarah UML

Bahasa pemrograman berorientasi objek yang pertama dikembangkan dikenal dengan nama Simula-67 yang dikembangkan pada tahun 1967. Bahasa pemrograman ini kurang berkembang dan dikembangkan lebih lanjut, namun dengan kemunculannya telah memberikan sumbangan yang besar pada developer pengembang bahasa pemrograman berorientasi objek selanjutnya.

Perkembangan aktif dari pemrograman berorientasi objek mulai menggeliat ketika berkembangnya bahasa pemrograman Smalltalk pada awal 1980-an yang kemudian diikuti dengan perkembangan bahasa pemrograman berorientasi objek yang lainnya seperti C objek, C++, Eiffel, dan CLOS. Secara aktual, penggunaan bahasa pemrograman berorientasi objek pada saat itu masih terbatas, namun telah banyak menarik perhatian di saat itu. Sekitar lima tahun setelah Smalltalk berkembang, maka berkembang pula metode pengembangan

berorientasi objek. Metode yang pertama diperkenalkan oleh Sally Shaler dan Stephen Mellor (Shaler-Mellor, 1988) dan Peter Coad dan Edward Yourdon (Coad-Yourdon, 1991), diikuti oleh Grady Booch (Booch, 1991), James R. Rumbaugh, Michael R. Blaha, William Lorensen, Frederick Eddy, William Premerlani (Rumbaugh-Blaha-Premerlani-Eddy-Lorensen, 1991), dan masih banyak lagi.

Buku terkenal yang juga berkembang selanjutnya adalah karangan Ivar Jacobson (Jacobson, 1992) yang menerangkan perbedaan pendekatan yang fokus pada *use case* dan proses pengembangan. Sekitar lima tahun kemudian muncul buku yang membahas mengenai metodologi berorientasi objek yang diikuti dengan buku – buku yang lainnya. Didalamnya juga membahas mengenai konsep, definisi, notasi, terminologi, dan proses mengenai metodologi berorientasi objek.

Karena banyaknya metodologi – metodologi yang berkembang pesat itu, maka muncullah ide untuk membuat sebuah bahasa yang dapat dimengerti semua orang. Usaha penyatuan ini banyak mengambil dari metodologi – metodologi yang berkembang saat itu. Maka dibuat bahasa yang merupakan gabungan dari beberapa konsep seperti konsep *Object Modeling Technique* (OMT) dari Rumbaugh dan Booch (1991), konsep *The Classes, Responsibilities, Collaborators* (CRC) dari Rebecca Wirfs-Brock (1990), konsep pemikiran Ivar Jacobson, dan beberapa konsep lainya dimana James R. Rumbaigh, Grandy Booch, dan Ivar Jacobson bergabung dalam sebuah perusahaan *Rational Software Corporation* menghasilkan bahasa yang disebut dengan *Unified Modeling Language* (UML) [**12**].

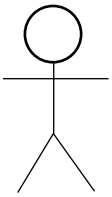
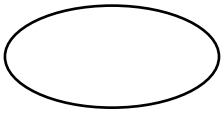
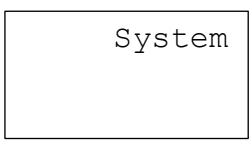
2.1.7.2 Diagram UML

Berikut akan dijelaskan diagram yang akan digunakan dalam pembangunan aplikasi Tutorial Pembelajaran Sistem Kontrol ini, yaitu *use case diagram, class diagram, sequence diagram,* dan *Activity diagram*.

1. *Use Case Diagram*

Use case diagram merupakan pemodelan untuk *behavior* aplikasi yang akan dibuat. *Use case* mendeskripsikan sebuah interaksi antara satu atau lebih aktor dengan aplikasi yang akan dibuat. Secara kasar, *use case* digunakan untuk mengetahui fungsi apa saja yang ada di dalam sebuah aplikasi dan siapa saja yang berhak mengguankan fungsi – fungsi itu. Penjelasan tentang simbol – simbol *use case* dapat dilihat pada tabel 2.2.

Tabel 2.2 Definisi Simbol Use Case Diagram

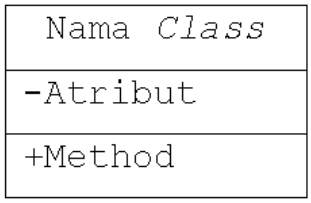
Simbol	Nama Simbol	Fungsi
	Aktor	Merupakan orang, proses, atau sistem lain yang berinteraksi dengan aplikasi yang akan dibuat di luar aplikasi yang akan dibuat itu sendiri, jadi walaupun simbol dari aktor adalah gambar orang, tapi aktor belum tentu orang
	*Use Case*	Merupakan orang, proses, atau sistem lain yang berinteraksi dengan aplikasi yang akan dibuat di luar aplikasi yang akan dibuat itu sendiri, jadi walaupun simbol dari aktor adalah gambar orang, tapi aktor belum tentu orang
	Association	Merelasikan Aktor dengan *use case*
	*System Boundary*	Menggambarkan batasan sistem terhadap lingkungannya

2. *Class Diagram*

Class diagram menggambarkan struktur sistem dari segi pendefinisian kelas – kelas yang akan dibuat untuk membangun sistem. Kelas memiliki apa yang disebut atribut dan metode atau operasi.

1. Atribut, merupakan variabel – variabel yang dimiliki oleh satu kelas.
2. Operasi, atau metode adalah fungsi – fungsi yang dimiliki oleh suatu kelas

Class diagram dibuat agar pembuat program atau *programmer* membuat kelas – kelas sesuai rancangan di dalam diagram kelas agar antara dokumentasi perancangan dan perangkat lunak sesuai. Contoh *class diagram* dapat dilihat pada gambar

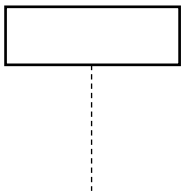
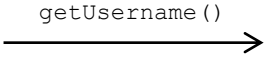


Gambar 2.6 Contoh *Class Diagram*

3. *Sequence Diagram*

Sequence diagram menggambarkan *behavior* objek pada *use case* dengan mendeskripsikan waktu hidup dan pesan yang dikirimkan dan diterima antar objek. Oleh karena itu untuk menggambar *sequence diagram* maka harus diketahui objek – objek yang terlibat dalam sebuah *use case* beserta metode – metode yang dimiliki kelas yang diinstansiasi menjadi objek itu. Membuat *sequence diagram* yang harus dibuat semakin banyak. Penjelasan tentang simbol – simbol *class diagram* dapat dilihat pada tabel 2.2

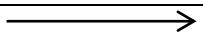
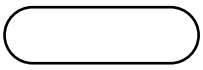
Tabel 2.3 Definisi Simbol *Sequence Diagram*

Simbol	Nama Simbol	Fungsi
	*Lifelines*	Representasi dari sebuah *class*
	*Message*	Representasi pemanggilan method oleh satu *class* kepada *class* yang lain

4. *Activity Diagram*

Activity diagram menggambarkan aliran kerja atau aktivitas dari sebuah sistem atau proses bisnis atau menu yang ada pada perangkat lunak. Yang perlu diperhatikan adalah *activity diagram* menggambarkan aktivitas sistem bukan apa yang dilakukan aktor. Berikut adalah penjelasan tentang simbol – simbol *activity diagram*:

Tabel 2.4 Definisi Simbol *Activity Diagram*

Simbol	Nama Simbol	Fungsi
	*Initial State*	Pernyataan awal dari sebuah aktivitas
	*Final State*	Pernyataan akhir dari sebuah aktivitas
	*Transisi*	Menggambarkan alur dari aksi – aksi yang ada pada aktivitas
	Aksi	Menggambarkan aksi yang ada pada aktivitas