BAB II
TEORI PENUNJANG

2.1 Sistem E-voting

Pengertian dari *electronic voting* (*E-voting*) secara umum adalah penggunaan teknologi komputer pada pelaksanaan *voting*, menjelaskan secara umum sejarah, jenis *electronic voting*, keuntungan dan kerugian dalam penggunaannya. Pilihan teknologi yang digunakan dalam implementasi dari *E-voting* sangat bervariasi, seperti penggunaan *smart card* untuk otentikasi pemilih, penggunaan internet sebagai pemungutan suara, penggunaan *touch screen* sebagai pengganti kartu suara, dan masih banyak variasi teknologi yang digunakan.

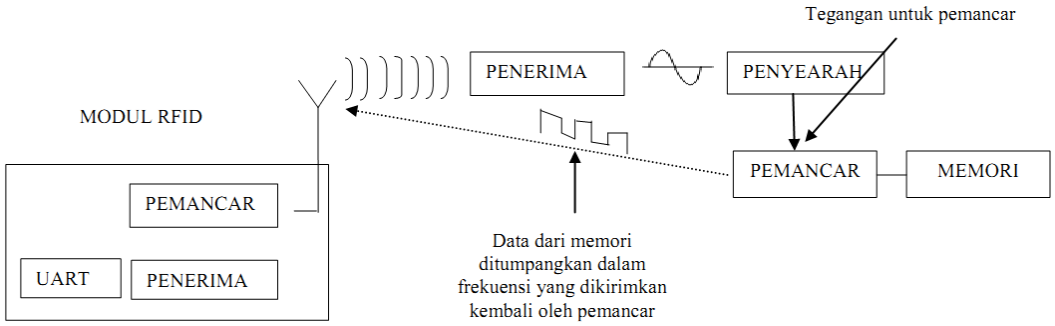
Penerapan dan teknologi *E-voting* terus berubah seiring perkembangan teknologi informasi yang sangat cepat. Kendala-kendala *E-voting* yang pernah terjadi di berbagai negara yang pernah dan sedang menerapkannya menjadi penyempurnaan *E-voting* selanjutnya. Salah satu segi positif dari penerapan *E-voting* saat ini adalah makin murahnya perangkat keras yang digunakan dan makin terbukanya perangkat lunak yang digunakan sehingga biaya pelaksanaan *E-voting* makin murah dari waktu ke waktu dan untuk perangkat lunak makin terbuka untuk diaudit secara bersama.

Secara garis besar, manfaat yang dapat diambil dengan adanya sistem *E-voting*, antara lain:

1. Proses dalam kalkulasi hasil *voting* akan lebih cepat.
2. Meningkatkan kenyamanan bagi para *user* (pemilih).
3. Hasil lebih akurat karena kesalahan manusia dikecualikan.
4. Berpotensi menghemat biaya dalam jangka panjang karna sistem dapat digunakan kembali.

2.2 Radio Freqeuncy Identification (RFID)

Radio Frequency Identification (RFID) merupakan suatu teknik identifikasi obyek yang dilakukan dengan menggunakan pancaran gelombang radio. Modul rfid akan memancarkan frekuensi ke kartu ataupun gantungan kunci (*key chain*) yang dalam hal ini berfungsi sebagai *transponder*. Frekuensi tersebut akan menimbulkan radiasi energi yang diubah menjadi sumber daya listrik yang akan memberikan tegangan pada rangkaian pemancar pada bagian *transponder* untuk memancarkan kembali identifikasinya ke modul rfid.



Gambar II.1 Teknik Rfid

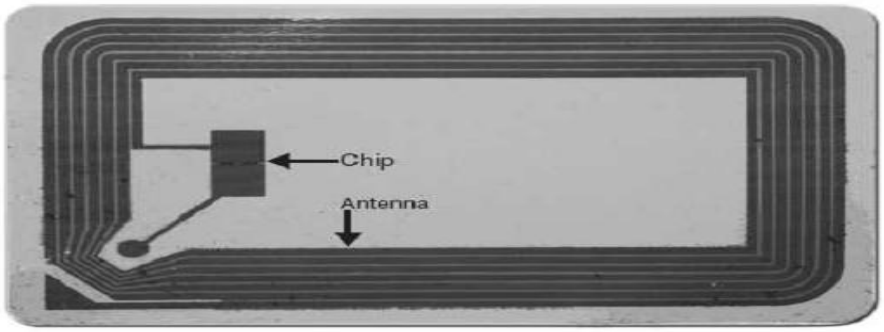
Teknik diatas adalah teknik rfid yang digunakan pada *passive tag* dimana *tag/transponder* harus menerima radiasi sinyal untuk membangkitkan sumber tegangan. Secara garis besar sebuah sistem rfid terdiri dari atas tiga komponen utama, yaitu *tag*, *reader* dan basis data. Secara ringkas, mekanisme kerja yang terjadi dalam sebuah sistem rfid adalah bahwa sebuah *reader* frekuensi radio melakukan *scanning* terhadap data yang tersimpan dalam *tag*, kemudian mengirimkan informasi tersebut ke sebuah basis data yang menyimpan data yang terdapat dalam *tag* tersebut.

2.2.1 Tag RFID

Sistem rfid merupakan suatu sistem identifikasi otomatis yang bertujuan untuk memungkinkan data ditransmisikan oleh peralatan *portabel* yang disebut *tag*. Kemudian *tag* tersebut dibaca oleh *reader* rfid dan diproses menurut kebutuhan dari aplikasi tertentu. Data yang ditransmisikan oleh *tag* dapat menyediakan informasi identifikasi atau lokasi, atau hal-hal khusus tentang produk-produk ber-*tag*, seperti harga, warna, tanggal pembelian dan lain-lain.

Sebuah *tag* rfid atau *transponder,* terdiri atas sebuah *microchip* dan sebuah antena. *Microchip* itu sendiri dapat berukuran sekecil butiran pasir, sekitar 0.4mm. *Chip* tersebut menyimpan nomor seri yang unik atau informasi lainnya tergantung kepada tipe memorinya. Tipe memori itu sendiri dapat *read-only, read-write,* atau *write-onceread-many*. Antena yang terpasang pada *microchip* mengirimkan informasi ke *reader* rfid. Biasanya rentang pembacaan diindikasikan dengan besarnya antena. Antena yang lebih besar mengindikasikan rentang pembacaan yang lebih jauh. *Tag* tersebut terpasang atau tertanam dalam obyek yang akan diidentifikasi. *Tag* dapat di-*scan* dengan *reader* rfid bergerak maupun stasioner.

Rfid *tag* dapat digolongkan menjadi, tak aktif dan *tag* pasif. *Tag* rfid yang digunakan pada pembuatan sistem *E-voting* ini menggunakan *tag passive*, karena paling sederhana, yaitu *tag* yang tidak memiliki catu daya sendiri serta tidak dapat minginisiasi komunikasi dengan *reader*.



Gambar II.2 Tag Rfid atau Transponder

2.2.2 Reader RFID

Untuk berfungsinya sistem rfid, maka diperlukan sebuah *reader* atau alat *scanning* yang dapat membaca *tag* dengan benar dan mengkomunikasikan hasilnya ke suatu *database* yang ada. Sebuah *reader* menggunakan antenanya sendiri untuk berkomunikasi dengan *tag*. Ketika *reader* memancarkan gelombang radio, seluruh *tag* yang dirancang pada frekuensi tersebut serta berada pada rentang bacanya akan memberikan respon. Sebuah *reader* juga dapat berkomunikasi dengan *tag* tanpa *line of sight* langsung, tergantung kepada frekuensi radion dan tipe *tag* (aktif, pasif atau semipasif) yang digunakan.



Gambar II.3 RFID Reader Starter Kit

Rfid *Reader* ID-12 mempunyai spesifikasi:

1. Tegangan pada kaki 11 adalah 2,8 Volt hingga 5 Volt.
2. Frekuensi yang digunakan adalah 125 KHz
3. Keseluruhan data digital dapat berupa *form*at ASCII ataupun *form*at wiegand pada kaki 8 dan kaki 9.
4. Hanya dapat Menangkap data dari RFID *Tag Card* yang berjenis EM 4001.

2.3 Teori Komunikasi Data

Komunikasi data merupakan sebuah proses pengiriman dan penerimaan data atau informasi dari dua atau lebih *device* seperti laptop, printer, *smartphone* dan jenis komunikasi yang digunakan, apakah menggunakan kabel, *infrared* atau menggunakan frekuensi.

Secara umum *diagram* blok komunikasi data sederhana seperti gambar di bawah ini.



Gambar II.4 Blok Diagram Komunikasi Data Sederhana.

Berikut penjelasan dari blok *diagram* pada Gambar II.4 blok *diagram* komunikasi data sederhana:

1. Sumber: Masukan data atau informasi yang akan dikirim ke tujuan
2. Transmiter: Jenis komunikasi yang digunakan dalam melakukan komunikasi data.
3. Media transmisi (transmitter) Jalur transmisi yang menghubungkan antara sumber dengan tujuan.
4. Receiver : Penerima sinyal yang dikirimkan melalui media transmisi untuk kemudian dikirimkan ke tujuan.
5. Tujuan : Menampilkan hasil data yang dikirim oleh sumber.

Dalam Komunikasi data jenis komunikasinya ada 3 macam yakni:

1. *Simplex*, yaitu komunikasi data hanya dengan satu arah dari tujuan sumber ke tujuan.
2. *Half Duplex*, yaitu komunikasi data dengan dua arah tetapi tidak bisa melakukan komunikasi secara bersamaaan.

Full Duplex, yaitu komunikasi data dengan dua arah dan bisa melakukan komunikasi secara bersamaan.

2.3.1 Jaringan

Jaringan (*Network*) adalah jaringan dari sistem komunikasi data yang melibatkan sebuah atau lebih sistem komputer yang dihubungkan dengan jalur transmisi alat komunikasi membentuk satu sistem. *Network* merupakan cara yang sangat berguna untuk mengintegrasikan sistem informasi dan menyalurkan arus informasi dari satu area ke area lainnya. Untuk membentuk suatu sistem *network* dibutuhkan suatu *software communication* yang khusus yaitu *protocol*.

Berdasarkan luas areanya atau letak geografisnya, jaringan area komputer dibedakan menjadi 3 bagian yaitu, *Local Area Network (*LAN), *Metropolitan Area Network* (MAN), *Wide Area Network* (WAN). *Local Area Network* (LAN) merupakan suatu jenis jaringan komputer dengan mencakup wilayah local. Dengan menggunakan kabel UTP (*Unshielded Twisted-Pair*), *Hub, Switch, Router,* dan lain sebagainya.

2.3.2 Client Server

Arsitektur jaringan *client server* merupakan pengembangan dari arsitektur *file server*. Arsitektur ini adalah modal konektivas pada jaringan yang mengenal adanya *server* dan *client*, dimana masing-masing memiliki fungsi yang berbeda satu sama lain. *Server* dapat berbagi pakai data, aplikasi dan *peripheral* seperti *harddisk, printer, modem* dan lain-lain. Prinsip kerjanya sangat sederhana, dimana *server* akan menunggu permintaan dari *client*, memproses dan memberikan hasilnya kepada *client*. Sedangkan *client* akan mengirimkan permintaan ke *server*, menunggu proses dan melihat visualisasi hasil prosesnya.

Sistem *Client Server* yang digunakan penulis menggunakan protocol TCP/IP (*Transmission Control Protocol/Internet Protocol*). Unix dan Windows NT merupakan contoh yang baik dari sistem operasi jaringan *client server*.

2.3.3 Komponen Dasar Client Server

1. *Client*

Client merupakan terminal yang digunakan oleh pengguna untuk menerima layanan tertentu yang dibutuhkan. Terminal *client* dapat berupa PC, ponsel, komunikator, robot, televise dan peralatan lain yang membutuhkan informasi.

2. Middleware

Middleware merupakan komponen perantara yang memungkinkan *client* dan *server* untuk saling terhubung dan berkomunikasi satu sama lain. *Middleware* ini dapat berupa *Transaction Monitor/*TP. *Remote Procedure Cell* atau *Object Request Broker/*ORB.

3. Server

Server merupakan komputer khusus yang bertugas melayani aplikasi-aplikasi jaringan/pihak yang menyediakan layanan. *Server* ini akan dapat berupa basis data SQL, Monitor TP, *server groupware*, *serve*r objek dan *web*. Secara umum, *server* berperan menerima pesan permintaan layanan dari *client*, memproses permintaan tersebut dan mengirimkan hasil permintaan kepada *client*.

2.3.4 Web Server

Web server atau *server web* dapat merujuk baik pada perangkat keras ataupun perangkat lunak yang menyediakan layanan akses kepada pengguna melalui *Protocol* komunikasi HTTP atau HTTPS atas berkas-berkas yang terdapat pada suatu situs *web* dalam layanan ke pengguna dengan menggunakan aplikasi tertentu.

Fungsi utama sebuah *web server* adalah untuk mentrasfer berkas atas permintaan pengguna melalui *Protocol* komunikasi yang telah ditentukan. Pemanfaatan *web server* berfungsi untuk mentrasfer seluruh aspek pemberkasan dalam sebuah halaman *web* yang terkait, termasuk di dalamnya teks, gambar, video, atau lainnya.

Dalam rekayasa perangkat lunak, suatu *web application* sering disingkat *webapp* adalah suatu aplikasi yang diakses oleh *user* menggunakan penjelajah *web* seperti Fire Fox, Internet Explorer dan Google Crome, melalui sebuah jaringan internet atau intranet. *Web application* juga merupakan suatu aplikasi perangkat lunak komputer yang dikodekan dalam bahasa yang didukung penjelajah *web* (seperti HTML, JavaScript, PHP, AJAX, Java, dan sebagainya) yang bergantung pada penjelajah tersebut untuk menampilkan aplikasi.

Web Application menjadi populer karena kemudahan akan tersedianya aplikasi klien untuk mengaksesnya. Kemampuan untuk memperbaharui dan memelihara aplikasi *web* tanpa harus mendistribusikan dan menginstalasi perangkat lunak pada ribuan komputer klien merupakan kunci popularitasnya. Aplikasi *web* yang umum misalnya *webmail, wiki, weblog*, serta MMORPG.

Arsiterktur *web application* meliputi klien, *web server*, *middleware* dan *database*. Klien berinteraksi dengan *web server*. Secara internal, *web server* berkomunikasi dengan *middleware* dan *middleware* yang berkomunikasi dengan basis data. Contoh *middleware* adalah PHP dan ASP. Pada mekanisme aplikasi *web* dinamis, terjadi tambahan proses yaitu *server* menerjemahkan kode PHP menjadi kode HTML. PHP yang diterjemahkan oleh mesin PHP yang diterima oleh *client.*

2.3.5 Web Service

Web service atau layanan *web* merupakan aplikasi *web* yang menyediakan layanan pengolahan data dan sebagainya. Layanan *web* sepenuhnya berdasarkan standar *web* dan xml. *Web service* dapat membantu perantara pada integrasi *platform* sepanjang eksekusi mesin virtual. Selain itu integrasi dapat dilakukan antara *web* dan *objek oriented middleware*, serta intergrasi dari aliran kerja terisolasi dan *service-service* (*Web* Services Flow Language – WSFL).

Web service juga dapat digunakan sebagai pertukaran data pada aplikasi yang berbeda-beda dan memegang peran utama dalam standarisasi *web application*. Kemampuan aplikasi, fungsi dan operasi yang di ekspos untuk program lain melalui standar yang terbuka dan interoperable. Pengiriman data dari *web application* ke *Web service* melalui http atau internet protokol terbuka lainya. Data pada *web service* dapat diakses dari berbagai bahasa pemograman, *platform hardware* atau *system operasi*.

2.3.6 Keuntungan Web Service

Dalam penggunaanya *web* service juga memberikan keuntungan seperti *form*at penggunaan terbuka untuk semua *platform*, mudah di mengerti dan men-debug, terbuka untuk semua bahasa pemograman berbasis teks, dukungan *interface* yang stabil, dan tidak mahal untuk diimplementasikan.

Selain itu keutungan *web service* juga mudah untuk menengahi pesan-pesan proses dan menambahkan nilai, serta kenyamanan dalam melakukan *routing*, pengiriman data, *security*, *managemen monitoring*, *schema service design* dan implementasi yang *incremental*.

2.3.7 Hypertext Transport Protocol (HTTP)

HTTP merupakan singkatan dari *Hypertext Transport Protocol* yang menjadi standar komunikasi dalam pentransferan halaman-halaman atau dokumen *web* melalui Internet.

HTTP menentukan bagaimana sebuah pesan disusun dan ditransmisikan, serta aksi apa yang harus dilakukan oleh *Web Server* (tempat menyimpan situs *web*) dan *browser* (aplikasi piranti lunak untuk mengakses *web*) dalam merespon sebagai perintah.

2.4 Unified Modeling Language (UML)

UML (Unified Modeling Languange) adalah metode pemodelan (tools/model) secara visual sebagai sarana untuk merancang atau membuat *software* berorientasi objek dan memberikan standar penulisan sebuah sistem untuk pengembangan sebuah software yang dapat menyampaikan beberapa informasi untuk proses implementasi pengembangan software. UML adalah metodologi untuk mengembangkan sistem OOP dan sekelompok perangkat *tools* untuk mendukung pengembangan sistem tersebut. Ada beberapa *diagram* yang disediakan dalam UML antara lain:

1. *Diagram use case* (*use case diagram*)
2. *Diagram* aktivitas (*activity diagram*)
3. *Diagram* sekensial (*sequence diagram*)
4. *Diagram* kolaborasi (*collaboration diagram*)
5. *Diagram* kelas (*class diagram*)
6. *Diagram statechart* (*statechart diagram*)
7. *Diagram* komponen (*component diagram*)
8. *Diagram deployment* (*deployment diagram*)

Diagram-diagram yang digunakan dalam pembuatan perangkat lunak ini adalah *use case diagram, class diagram, sequence diagram* dan *activity diagram*. Dalam pembuatan UML penulis menggunakan *software* StarUML versi 5.0.

2.4.1 Use Case Diagram

Kebutuhan fungsional akan digambarkan melalui sebuah *diagram* yang dinamakan *diagram use case*. *Use Case Diagram* atau *diagram use case* merupakan pemodelan untuk menggambarkan kelakuan (*behavior*) sistem yang akan dibuat. *Diagram use case* mendeskripsikan sebuah interaksi antara satu atau lebih aktor dengan sistem yang akan dibuat. Dengan pengertian yang cepat, *diagram use case* digunakan untuk mengetahui fungsi apa saja yang ada di dalam sebuah sistem dan siapa saja yang berhak menggunakan fngsi-fungsi tersebut. Terdapat beberapa simbol dalam menggambarkan *diagram use case*, yaitu *use case*, aktor dan relasi. Hal yang perlu diingat mengenai *diagram use case* adalah *diagram use case* bukan menggambarkan tampian antar muka (*user interface*), *arsitektur* dari sistem, kebutuhan nonfungsional, dan tujuan per*form*ansi. Sedangkan untuk penamaan *use case* adalah nama didefinisikan sesimpel mungkin, dapat dipahami dan menggunakan kata kerja.

2.4.2 Class Diagram

Diagram kelas atau *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.

a. Atribut merupakan variabel-variabel yang dimiliki oleh suatu kelas.
b. Atribut mendeskripsikan properti dengan sebaris teks didalam kotak kelas tersebut.
c. Operasi atau metode adalah fungsi-fungsi yang dimiliki oleh suatu kelas.

Diagram kelas mendeskripsikan jenis-jenis objek dalam sistem dan berbagai hubungan statis yang terdapat di antara mereka. *Diagram* kelas juga menunjukan properti dan operasi sebuah kelas dan batasan-batasan yang terdapat dalam hubungan-hubungan objek tersebut.

Diagram kelas menggambarkan struktur dan deskripsi *class*, *package* dan objek beserta hubungan satu sama lain seperti *containment*, pewarisan, asosiasi, dan lain-lain. Kelas memiliki tiga bagian pokok yaitu:

1. Nama
2. Atribut
3. Operasi

2.4.3 Sequence Diagram

Diagram sequence menggambarkan kelakuan/perilaku objek pada *use case* dengan mendeskripsikan waktu hidup objek dengan *message* yang dikirimkan dan diterima antar objek. Oleh karena itu untuk menggambarkan *diagram sequence* maka harus diketahui objek-objek yang terlibat dalam sebuah *use case* beserta metode-metode yang dimiliki kelas yang diinstansiasi menjadi objek.

Banyaknya *diagram sequence* yang harus digambar adalah sebanyak pendefinisian *use case* yang dimiliki proses sendiri atau yang penting semua *use case* yang telah didefinisikan interaksi jalannya pesan sudah dicakup pada *diagram sequence* sehingga semakin banyak *use case* yang didefinisikan maka *diagram* sekensial yang harus dibuat juga semakin banyak.

Penomoran pesan berdasarkan urutan interaksi pesan. Penggambaran letak pesan harus berurutan, pesan yang lebih atas dari lainnya adalah pesan yang berjalan terlebih dahulu.

Diagram sequence memiliki ciri yang berbeda dengan *diagram* interaksi pada *diagram* kolaborasi sebagai berikut:

1. Pada *diagram sequence* terdapat garis hidup objek. Garis hidup objek adalah garis tegas vertikal yang mencerminkan eksistensi sebuah objek sepanjang periode waktu. Sebagian besar objek-objek yang tercakup dalam *diagram* interaksi akan eksis sepanjang durasi tertentu dari interaksi, sehingga objek-objek itu diletakan dibagian atas *diagram* dengan garis hidup tergambar dari atas hingga bawah *diagram*. Suatu objek lain dapat saja diciptakan, dalam hal ini garis hidup dimulai saat pesan *create* diterima suatu objek. Selain itu suatu objek juga dapat dimusnahkan dengan pesan *destroy*, jika kasus ini terjadi, maka garis hidupnya juga berakhir.
2. Terdapat fokus kendali (*Focus of Control*), berupa empat persegi panjang ramping dan tinggi yang menampilkan aksi objek secara langsung atau sepanjang sub ordinat. Puncak dari empat persegi panjang adalah permulaan aksi, bagian dasar adalah akhir sari suatu aksi. Pada *diagram* ini mungkin juga memperlihatkan penyaringan (*nesting*) dan

fokus kendali yang disebabkan oleh proses rekursif dengan menumpuk fokus kendali yang lain pada induknya.

Sequence diagram mendeskripsikan bagaimana entitas dalam system berinteraksi, termasuk pesan yang digunakan saat interaksi. Semua pesan dideskripsikan dalam urutan dari eksekusi. *Sequence diagram* berhubungan erat dengan *Use Case diagram*, dimana 1 *Use Case* akan menjadi 1 *Sequence Diagram*.

Sequence Diagram dibuat dengan banyak lifeline. Setiap lifeline mendapatkan tempat sendiri – sendiri. Beberapa jenis lifeline antara lain:

1. Aktor

   Actor merepresentasikan entitas yang berada di luar system. Mereka bisa berupa manusia, perangkat keras atau system lain.

2. *General*

   Merepresentasikan entitas tunggal dalam *sequence diagram*, digambarkan dengan kotak. Entitas ini memiliki nama, stereotype atau berupa instance (menggunakan instance:class).

3. *Boundary*

   Control element mengatur aliran dari informasi untuk sebuah scenario. Perilaku dan perilaku bisnis umumnya diatur oleh objek ini.

4. *Entity*

   Entity biasanya elemen yang bertanggung jawab menyimpan data atau informasi. Ini dapat berupa beans atau model object

2.4.4 Activity Diagram

Diagram aktivitas atau *activity diagram* menggambarkan *workflow* (aliran kerja) atau aktifitas dari sebuah sistem atau proses bisnis. Yang perlu diperhatikan disini adalah bahwa *diagram* aktivitas menggambarkan aktivitas sistem bukan apa yang dilakukan aktor, jadi aktivitas yang dapat dilakukan oleh sistem. *Diagram* aktivitas mendukung perilaku paralel.

Diagram aktivitas juga banyak digunakan untuk mendefinisikan hal-hal berikut:

a. Rancangan proses bisnis dimana setiap urutan aktivitas yang digambarkan merupakan proses bisnis sistem yang didefinisikan.

b. Urutan atau pengelompokan tampilan dari sistem/*user interface* dimana setiap aktivitas dianggap memerlukan sebuah pengujian yang perlu didefinisikan kasus ujiannya.
c. Rancangan pengujian dimana setiap aktivitas dianggap memerlukan sebuah pengujian yang perlu didefinisikan kasus ujiannya.

Activity diagrams menggambarkan berbagai alir aktivitas dalam sistem yang sedang dirancang, bagaimana masing-masing alir berawal, *decision* yang mungkin terjadi, dan bagaimana mereka berakhir. *Activity diagram* juga dapat menggambarkan proses paralel yang mungkin terjadi pada beberapa eksekusi.
Activity diagrams merupakan *state diagram* khusus, dimana sebagian besar *state* adalah *action* dan sebagian besar transisi di-trigger oleh selesainya *state* sebelumnya (*internal prosesing*). Oleh karena *activity diagram* tidak menggambarkan *behavior* internal sebuah sistem (dan interaksi antar subsistem) secara eksak, tetapi lebih menggambarkan proses-proses dan jalur-jalur aktivitas dari level atas secara umum.

2.4.5 Statechart Diagram

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

2.4.6 Collaboration Diagram

Collaborattion diagram juga menggambarkan interaksi antarobjek seperti *sequence diagram*, tetapi lebih menekankan pada peran masing-masing objek dan bukan pada waktu penyampaian *massage*. Setiap *message* memiliki *sequence number*, di mana *massage* dari level tertinggi memiliki nomor1. *Massage* dari level yang sama memiliki prefiks yang sama.

2.4.7 Component Diagram

Component diagram atau *diagram* komponen dibuat untuk menunjukkan organisasi dan ketergantungan diantara kumpulan komponen dalam sebuah sistem. *Diagram* komponen focus pada komponen sistem yang dibutuhkan dan

ada di dalam sistem. *Diagram* komponen juga dapat digunakan untuk memodelkan hal-hal berikut:

1. *Source code* program perangkat lunak
2. Komponen *executabel* yang dilepas ke *user*.
3. Basis data fisik.
4. Sistem yang harus beradaptasi dengan sistem lain.
5. *Framework* sistem, *framework* pada perangkat lunak merupakan kerangka kerja yang dibuat untuk memudahkan pengembangan dan pemeliharaaan aplikasi, contohnya seperti struts dari apache yang menggunakan prinsip desain Model-View Controller (MVC) dimana *source code* program dikelompokkan berdasarkan fungsinya.

2.4.8 Deployment Diagram

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

2.5 Teori Tools Interface

Tools adalah alat. Dalam pengembangan perangkat lunak (sistem) dapat diartikan sebagai proses membuat suatu perangkat lunak (sistem) baru untuk menggantikan perangkat lunak (sistem) lama secara keseluruhan atau memperbaiki perangkat lunak yang telah ada. Dalam membangun sistem *E-voting* ini penulis menggunakan beberapa *tools* yaitu, Delphi 7, MsSql, Macromedia Dreamweaver 8, PHP, dan HTML.

2.5.1 Delphi 7

Delphi adalah *compile*/penterjemah bahasa Delphi (awalnya dari pascal) yang merupakan bahasa tingkat tinggi sekelas dengan *Basic* dan *C*. Bahasa Pemrograman di Delphi disebut bahasa prosedural artinya bahasa / sintaknya mengikuti urutan tertentu/prosedur. Ada jenis pemrograman non-prosedural seperti pemrograman untuk kecerdasan buatan seperti bahasa Prolog. Delphi termasuk Keluarga *Visual* sekelas *Visual Basic*, *Visual C*, artinya perintah-

perintah untuk membuat objek dapat dilakukan secara visual. Pemrogram tinggal memilih objek apa yang ingin dimasukkan kedalam *Form* / Window, lalu tingkah laku objek tersebut saat menerima *event* / aksi tinggal dibuat programnya. Delphi merupakan bahasa berorentasi objek, artinya nama objek, properti dan *method*/ *procedure* dikemas menjadi satu kemasan (*encapsulate*). Pengunaan *tools* Delphi sendiri karna merupakan program berbasiskan *windows* yang memiliki bahas mirip dengan *pascal*. Perbedaan mendasar antara Delphi dengan *pascal* adalah bahwa Delphi memiliki oop (*object orientation program*) atau program berorientasi ke objek. Dengan demikian Delphi lebih mudah digunakan dibandingkan *pascal*, dan tujuan digunakan Delphi untuk merancang aplikasi *interface* pada sistem *E-voting*.

2.5.2 Microsoft SQL Server

MS SQL Server adalah salah satu produk *Relational Database Management System* (RDBMS). Fungsi utamanya adalah sebagai database *server* yang mengatur semua proses penyimpanan data dan transaksi suatu aplikasi. Dalam DBMS seperti MS SQL Server biasanya tersedia paket bahasa yang digunakan untuk mengorganisasi basis data yang ada, yaitu *Data Definition Language* (DDL) dan *Data Manipulation Language* (DML).

Data Definition Language (DDL) adalah satu paket bahasa DBMS yang berguna untuk melakukkan spesifikasi terhadap skema basis data. Hasil kompilasi dari DDL adalah satu set tabel yang disimpan dalam file khusus yang disebut Data Directory/Dictionary. Sercara umum perintah-perintah DDL berhubungan dengan operasi-operasi dasar seperti membuat basis data baru, menghapus basis data, membuat tabel baru, menghapus tabel, membuat indeks, mengatur struktur tabel. Contoh perintah DDL misalnya, *create tabel, create index, Alter,* dan *drop database*.

Data Manipulation Language (DML) adalah satu paket DBMS yang memperbolehkan pemakai untuk mengakses atau memanipulasi data sebagaimana yang telah diorganisasikan sebelumnya dalam model data yang tepat. Dangan DML dapat dilakukan kegiatan: mengambil informasi yang tersimpan dalam basis

data. Menyisispkan informasi baru dalam basis data. Menghapus informasi dari tabel.

2.5.3 Macromedia Dreamweaver 8

Macromedia Dreamweaver 8 merupakan sebuah HTML *edition* professional yang berfungsi untuk mendesain *web* secara *visual*, melakukan editing, dan mengelola sistus atau halaman *web*.

Pada dreamweaver 8 terdapat beberapa kemampuan seperti ruang kerja dreamweaver beserta *tools* yang sudah tersedia, sehingga dapat digunakan dengan sangat mudah dan cepat. Dalam membangun sebuah *website* menggunakan dreamweaver 8 kita tidak melakukan penulisan *coding*.

Fungsi *coding* dreamweaver tidak hanya mendukung coding HTML tetapi juga CSS (*Cascoding Style Sheet*), *javascript, coldfusion,* ASP (*Active Server Sheet*), dan dreamweaver juga memungkinkan membangun *website* dengan *server* berbahas CFML (*Colfusion Markup Language*), ASP.net dan PHP.

2.5.4 PHP

PHP (*PHP: Hypertext Preprosesor*) adalah sebuah *general-purpose scripting language* bersifat *open source* yang banyak digunakan khususnya dalam pengembangan *web* dan dapat diintegrasikan dengan HTML.

Sintaksnya mirip dengan bahasa C, Java, dan Perl sehingga mudah untuk dipelajari. Tujuan utama dari bahasa ini memungkinkan para pengembang *web* dalam menulis halaman-halaman *web* dinamis. PHP juga mendukung untuk berkomunikasi dengan layanan lain menggunakan *protocol* IMAP, SNMP, NNTP, POP3, HTTP dan lainnya. Pemrogram juga dapat membuka soket jaringan secara mentah dan berinteraksi dengan menggunakan *protocol* lainnya.

Script PHP disisipkan langsung dalam tubuh *file* HTML yang ditandai dengan *tag* pembuka dan penutup. Sebagaimana yang telah diketahui, ada dua cara pasang *tag* PHP yang dapat digunakan yaitu: *<?php..>* dan *<script language=”php”>...</script>*.

2.5.5 HTML

HTML (*HyperText Markup Language*) adalah suatu *form*at data yang digunakan untuk membuat dokumen *hypertext* yang dapat dibaca dari satu *platform* ke *platform* lainnya tanpa melakukan perubahan apapun. Dokumen HTML sebenarnya adalah suatu dokumen teks biasa, sehingga di *platform* apapun dokumen tersebut dapat dibaca. Dokumen HTML disebut juga *Markup Language* karena mengandung tanda-tanda tertentu yang digunakan untuk menentukan tampilan teks dan tingkat kepentingan dari teks tersebut dalam suatu dokumen.

Dalam dokumen HTML dibagi menjadi dua bagian besar, yaitu *header* (bagian atas) dan *body* (tubuh dokumen). Bagian *head* berisikan judul dokumen dan informasi-informasi dasar lain, sedangkan bagian *body* adalah data dokumennya.