BAB II

LANDASAN TEORI

2.1. Siklus *Learning*

Siklus pengembangan *Learning* memiliki fase-fase (Siemens, 2005):

Tahap 1 : *Scope*

Tahap ini meliputi perencanaan dan analisis batasan dari perancangan *learning*, pendefinisisan tipe dan tujuan *learning* itu sendiri, menentukan metode *learning* yang digunakan, sifat *learning,* domain *learning,* serta teknologi yang digunakan. Pada umumnya tujuan *learning* ini digunakan untuk menuntun pengajar, bukan sebagai pedoman siswa.

Tahap 2: *Creation*

Pada tahap ini perancangan membuat desain, mengembangkan dan melakukan proses *learning.* Hal ini yang dipertimbangkan pada tahap ini adalah sifat dari materi *learning* serta perancangan proses interaksi yang diharapkan terjadi saat *learning* berlangsung. Ditahap ini perancangan proses interaksi yang diharapkan terjadi saat *learning* berlangsung. Di tahap ini perancang mendesain sasaran *learning.* Teknologi, pengembangan interaksi, skill yang dibutuhkan, taks dari pengajar serta *feedback* yang diharapkan.

Tahap 3 : *User Experience*

User experience adalah proses yang penting dalam menentukan

penggunaan sumber daya *learning* selama tahap 1 dan 2. Tahap ini akan menjawab apakah desain *learning* berguna dan bernilai, bagaimana reaksi siswa terhadap isi, presentasi dan interaksi *learning*.

Tahap 4 : *Meta-Evaluation*

Meta-evaluation adalah proses mengevaluasi efektifitas proses perancangan *learning.*

Eksplorasi terhadap keberhasilan dan kegagalan selama proses *learning* berlangsung dilakukan untuk melakukan perbaikan perancangan dan implementasi *learning* selanjutnya.

Tahap 5 : *Evaluation*

Tahap ini dilakukan untuk mengevaluasi *learning,* dapat dilakukan dengan pemberian ujian atau tugas secara berkelompok. Sifat evaluasi ini dapat berupa *formative* (dilakukan saat proses *learning* berlangsung) atau *summative (*dilakukan setelah seluruh materi disampaikan). Evaluasi juga dilakukan siswa dengan memberikan *feedback* atas proses *learning* yang telah dilakukan baik (dari sisi kualitas sumberdaya *learning*, instruksi yang diberikan, relevansi dan format *learning*) dengan cara langsung pada pengajar, atau melalui questioner dari intitusi.

2.1.1. Domain Penelitian

Penelitian pada area *collaborative Learning* dapat di pandang dari tujuh domain yang berbeda yaitu (Vive Kumar, 1996) :

1. *Control of collaborative learning*

   Kendali dan interaksi kolaborasi mengacu pada cara penyampaian dalam sistem di lingkungan kolaborasi. Sistem *collaborative learning* dapat menjadi bagian yang menganalisis dan mengendalikan kolaborasi atau hanya bertindak sebagai alat pengantar kolaborasi. Sistem *collaborative learning* pada dimensi ini dapat diklasifikasikan sebagai sistem yang aktif, pasif atau apapun yang ada dalam batasan keduanya.

2. *Task of collaborative leaning*

   Dalam lingkungan *collaborative learning,* kolaborator dapat memiliki berbagai jenis *task. Task* yang umumnya ditemukan dalam lingkungan *collaborative learning* adalah :

   a. *Collaborative concept_learning task*

   b. *Collaborative problem_solving task*

   c. *Collaborative design task*

3. *Theories of learning and cognition in collaboration*

   Dillenbourg(Vive Kumar, 1996) mengidentifikasikan 3 teori learning yang dapat digunakan dalam sistem *collaborative learning* yaitu : *socio-constructivist, socio-cultural* dan *shared cognition.*

4. *Design of collaborative learning context*

   Dasar utama dari kolaborasi adalah *willingness* dari partisipan untuk berpesan dan berkolaborasi dalam sense yang konstuktif. Slavin (Vive Kumar, 1996) melaporkan hasil studi yang dilakukan oleh kuhn(1972) yang menemukan

bahwa perbedaan yang kecil pada level kemampuan kognitif antara partisipan kolaborasi ternyata lebih mengembangkan pertumbuhan kognitif, dibandingkan dengan partisipan yang memiliki perbedaan yang jauh kemampuan kognitifnya. Hal ini menunjukan bahwa partisipan kolaborasi seharusnya mempunyai tingkat pengetahuan yang hampir sama untuk membangun kolaborasi yang konstruktif.

Kombinasi taks dan jumlah kolaborator yang terlibat dalam *learning* ditentukan oleh subjek domain, teori *learning* yang digunakan dan kemampuan sistem. Berikut ini adalah jenis-jenis rancangan lingkungan kolaborasi :

a. Dua atau lebih partisipan berkolaborasi satu sama lain menggunakan komputer sebagai media. Sistem tidak melakukan apapun selain menyediakan channel komunikasi untuk kolaborasi tanpa memainkan peran yang aktif.
b. Dua atau lebih partisipan berkolaborasi satu sama lain menggunakan seorang tutor aktif yang mengendalikan dan mengarahkan interaksi kolaborasi.
c. Dua partisipan atau lebih partisipan berkerjasama menangani masalah pada tempat kerja yang sama, menggunakan tutor sedemikian rupa sehingga sama dengan jika partisipan bekerja sendiri.
d. Dua atau lebih partisipan bekerjasama dengan sedikitnya satu partisipan menggunakan simulasi sistem. Partisipan dapat menentukan aksinya secara langsung atau atas permintaan partisipan lain.

5. *Roles of the peers*

Pada lingkungan *collaborative learning,* tujuan akan dibagi menjadi *subtasks* yang akan dilakukan oleh tiap partisipan . Hal ini juga menunjukan bahwa partisipan diberikan peran secara natural dan *applicable* pada domain yang diberikan. Blave (Vive Kumar, 1996) menggambarkan dua peran yaitu eksekutor, orang yang menangani masalah dan reflektor, orang yang meneliti dan mengomentari atas penyelesaian masalah.

Secara umum lingkungan *collaborative learning* dapat memiliki sekumpulan peran sebagai :

a. *Decomposing,* mengacu pada pekerjaan membagi penanganan masalah kedalam task, tiap task dibagi ke dalam sejumlah tujuan. Tujuan akan menjadi objek learning bagi pembelajarannya.
b. *Defining,* mengusulkan sebuah tujuan dari sebuah task.
c. *Critiquing,* hipotesa yang diusulkan seorang partisipan berita alternatifnya.
d. *Convicing,* aksi membandingkan sejumlah hipotesa dan mendukung salah satunya.
e. *Reviewing,* pekerjaan yang menjamin agar interaksi kolaborasi mengacu pada proses *learning* yang konstruktif.
f. *Referencing,* pekerjaan yang menyediakan fakta dan material terkait yang diminta oleh partisipan lain.

6. *Domain of collaboration*

   Secara umum *collaborative learning* efektif pada domain dimana partisipan berada pada pekerjaan *skill acquisition, join planing, categorization* dan *memory task.*

7. *Teaching methodologies*

   Sejumlah metodologi pengajaran yang diidentifikasi mendukung *collaborative learning* adalah :

   a. *Practice*, partisipan diminta untuk mengunakan sebuah tujuan *learning* pada sebuah masalah spesifik.

   b. *Learning by teaching*, metodologi ini mendukung *learning* dengan memiliki sistem sebagai *learnig* tools.

   c. *Situated learning*, siswa menjadi partisipan dalam sebuah latihan *sociocultural,* kemampuan belajar dan kemampuan sosial berjalan bersama.

   d. *Negotiated learning*, partisipan dan sistem bernegosiasi untuk mencapai tujuan belajar.

   e. *Discovery learning*, siswa mengekplorasi sebuah lingkungan untuk proses *learning*.

2.1.2. Definisi *Collaborative Learning*

Beberapa definisi *Collaborative learning* (CL)

1. *Umbrella term* untuk berbagai jenis pendekatan edukasi yang melibatkan

usaha bekerjasama secara intelektual antar siswa atau antar siswa dan pengajar (B.L.Smith dan J.T.MacGregor).

2. *An instruction method in wich students work in groups toward a common academic goal* (Suatu metode instruksi dimana para siswa bekerja dalam suatu kelompok untuk mencapai tujuan akademik tertentu) (A.Anuradha dan Gokhale, 1995).

3. Menurut Johnson (S.Gupta dan Dr.Robert P.Bostrom, 2004):

   ”*Collaborative learning (CL) refers to instructional methods that encourage students to work together to accomplish shared goals, beneficial to all. It involves social (interpersonal) proceses where participants help each other to understand as well as encourage each other to work hand to promote learning.”*

Tujuan utama penggunaan *collaborative learning* (Henry, 1997)

1. Fokus pada belajar yang aktif

   Usaha dalam the *Calculus Reform Movement* (Henry, 1997) mengindikasikan bahwa kelas menjadi tempat yang paling efektif saat para siswanya ikut serta atau terlibat dalam materi yang disampaikan. Format pengajar memungkinkan seorang instruktur meringkas sejumlah materi dengan efesien namun hal ini tidak otomatis mengefektifkan proses penyampaian materi pada siswa.

2. Membangun *skill* menulis dan komunikasi lisan

   Colaborative learning mendorong siswa untuk berkomunikasi satu sama lain,

menyatakan respon pada pertanyaan, bekerja dalam lontaran pendapat yang berbeda-beda dan menuliskan kesimpulan dengan jelas.

3. Memberikan tanggungjawab belajar secara eksplisit

   Jika dalam kelas lebih terfokus pada kerja secara kelompok, dan bukan pada instruktur, maka siswa akan menyadari bahwa kelompok mereka tidak dapat mengikuti pelajaran sehari-hari bila tanpa persiapan. Hal ini akan mendorong kegiatan membaca dan penyelesaian pekerjaan lain pada suatu hari untuk disampaikan pada kelompok mereka dikemudian hari.

4. Memperjelas peran pengajar sebagai fasilitator dan mentor

   Seorang instruktur akan semakin melepaskan control terhadap kelas apabila format kelas menekankan pada aktivitas kelompok, sehingga peranya akan menjadi pemberi tanggapan jika ada pertanyaan, sebagai pelatih kelompok secara individual, mengatasi kesulitan-kesulitan yang umum terjadi dan menyarankan suatu pendekatan baru.

5. Dapat mencakup materi lebih banyak atau lebih baik (untuk materi yang sama)

   Dengan semakin aktifnya siswa dalam kelas dan semakin reponsif atas proses belajar maka ditemukan bahwa kelas dapat mengalami percepatan sebanyak 20% (Henry, 1997), sehingga disemester selanjutnya materi dapat diperbanyak.

6. Membangun rasa percaya diri dan mandiri pada siswa

   Sebuah kelas yang melibatkan siswa dan partisipasi kelompok, akan

mengurangi ketergantungan siswa pada pengajar dan siswa belajar bagaimana cara belajar, Siswa menjadi cakap saat membaca dan berlatih, dan mereka akan membuat strategi dalam menguasai pengetahuan yang baru.

7. Memiliki pengalaman bekerja secara kelompok

   Saat kelompok kerja melakukan perancangan dan membuat sebuah program, anggota kelompok harus menentukan bagaimana struktur solusi yang diberikan, task apa yang akan ditangani suatu modul dan bagaimana bentuk *interface* dari modul tersebut kemudiah source code dibuat, kelompok akan mengetahui apa yang berjalan dan yang mana yang tidak. Aktivitas seperti ini akan menekankan prinsip-prinsip *software engineering,* misalnya seperti kebutuhan akan spesifikasi yang baik.

8. Mendukung *Peer Review*

   Saat bekerja pada proyek pemrograman sebagai bagian dari suatu kelompok, secara alami siswa akan melihat sumber kelompok lain untuk mengetahui pendekatan yang digunakan, menganalisa efisiensinya dan memperkirakan kesalahan yang mungkin terjadi.

2.2. *Internet* dan Aplikasi *Web*

2.2.1. *Internet*

Internet adalah suatu jaringan komputer global yang terbentuk dari jaringan-jaringan komputer lokal dan regional yang memungkinkan komunikasi data antar komputer yang terhubung ke jaringan tersebut. Internet awalnya

merupakan rencana dari Departemen Pertahanan Amerika Serikat (*US Departement Of Defense*) pada sekitar tahun 1960. Dimulai dari suatu proyek yang dinamakan ARPANET atau *Advanced Research Project Agency Network.* Beberapa universitas di Amerika Serikat diantaranya UCLA, Stanford, UC Santa Barbara dan University of Utah, diminta bantuan dalam mengerjakan proyek ini dan awalnya telah berhasil menghubungkan empat komputer di lokasi universitas yang berbeda tersebut. Perkembangan ARPANET ini cukup pesat jika dilihat dari perkembangan komputer pada saat itu. Karena perkembanganya sangat pesat, jaringan komputer ini tidak dapat lagi disebut sebagai APRANET karena semakin banyak komputer dan jaringan-jaringan regional yang terhubung. Konsep ini kemudian berkembang dan dikenal sebagai konsep *Internetworking* (*Jaringan antar jaringan*). Oleh karena itu istilah internet menjadi semakin popular, dan orang menyebutnya jaringan besar komputer tersebut dengan istilah internet (Budhi Irawan, 2005).

2.2.2. Aplikasi *Web*

Pada awalnya aplikasi *Web* dibangun hanya dengan menggunakan bahasa yang disebut HTML (*HyperText Markup Language)*. Pada perkembangan berikutnya, sejumlah skrip dan objek dikembangkan untuk memperluas kemampuan HTML. Pada saat ini, banyak skrip seperti itu, antara lain yaitu PHP dan ASP, sedangkan contoh yang berupa objek adalah *applet*. Aplikasi *Web* itu dapat dibagi menjadi *Web* statis dan *Web* dinamis. *Web* statis dibentuk dengan menggunakan HTML saja. Kekurangan aplikasi seperti ini terletak pada

keharusan untuk memelihara program secara terus-menerus untuk mengikuti setiap perubahan yang terjadi. Kelemahan ini diatasi dengan model *Web* dinamis. Dengan menggunakan pendekatan *Web* dinamis, dimungkinkan untuk membentuk sistem informasi berbasis *web*. Dari sisi teknologi yang digunakan untuk membentuk *web* dinamis terdapat dua pengelompokan, yaitu teknologi pada sisi *client* dan teknologi pada sisi *server*.

Teknologi *Web* pada sisi *client* diimplementasikan dengan mengirimkan kode perluasan HTML atau program tersendiri dan HTML ke *client*. *Client*lah yang bertanggung jawab dalam melakukan proses terhadap seluruh kode yang diterima. Kelemahan pendekatan seperi ini adalah terdapat kemungkinan bahwa *browser* pada *client* tidak mendukung fitur kode perluasan HTML. Kelebihan teknologi pada sisi *client*, yaitu memungkinkan penampilan yang bersifat dinamis. Contoh teknologi pada sisi *client*, yaitu Kontrol ActiveX, Java Applet, dan Skrip sisi-*client*. Teknologi *Web* pada sisi server memungkinkan pemrosesan kode di dalam *server* sehingga kode yang sampai pada pemakai berbeda dengan kode asli pada server. Contoh teknologi yang berjalan di *server*, yaitu CGI, ASP, JSP, PHP dan lain sebagainya. Keuntungan penggunaan teknologi pada sisi *server* adalah sebagai berikut:

1. Mengurangi lalu lintas jaringan dengan cara menghindari percakapan bolak-balik antara *client* dan server.
2. Mengurangi waktu pemuatan kode, mengingat *client* hanya mengambil kode HTML saja.
3. Mencegah masalah ketidakkompatibelan *browser*.

4. *Client* dapat berinteraksi dengan data yang ada pada *server*.
5. Mencegah *client* mengetahui rahasia kode (mengingat kode yang diberikan ke *client* berbeda dengan kode asli pada *server*) (Nugroho, 2004).

2.3. *Web* 2.0

2.3.1. Definisi *Web* 2.0

Istilah tentang *web* 2.0 dikeluarkan pada tahun 2004 oleh Dale Dougherty pada sebuah konfrensi mengenai aplikasi *web* (Paul Graham, 2005). Setelah melalui berbagai pembahasan dan perdebatan akhirnya disepakatilah bahwa *web* 2.0 bukanlah sebuah hipotesa atau teori atau paradigma ataupun metodologi dalam membangun aplikasi *web*.

Web 2.0 adalah istilah untuk suatu aplikasi *web* yang berorientasi proses bisinis dan arsitektur layananya mengedepankan kontribusi dari setiap penggunanya serta memberikan fitur-fitur yang mempermudah pengguna untuk mempersonalisasi kebutuhanya (Zibriel dan Supangkat, 2008).

2.3.2. Kelebihan dari *Web 2.0*

Dalam perkembanganya, aplikasi *web* yang dibangun dengan menggunakan orientasi *web* 2.0 ternyata dirasakan mempunyai beberapa nilai positif (Zibriel dan Supangkat, 2008). Nilai positifnya adalah sebagai berikut :

1. *Web* 2.0 berhasil menyajikan sebuah layanan yang komprehensif pada *platform* apapun. Cukup menggunakan sebuah *browser* dan melakukan koneksi dengan *server* maka setiap orang sudah dapat menggunakanya.

2. Dalam penggunaanya, web 2.0 lebih mudah digunakan karena aplikasinya berjalan secara terpusat di *server*, pengguna tidak perlu repot lagi untuk memperbaharui aplikasi mereka secara mandiri.
3. Dalam segi pemrograman , *web* 2.0 memiliki teknik pemrograman *front-end* yang relatif ringan hal ini dikarenakan *web* 2.0 adalah sebuah aplikasi yang berjalan di sebuah *browser*. Sehingga mudah untuk digunakan kembali (*reuse*).
4. Kelebihan orientasi *web* 2.0 dibandingkan dengan *web* 1.0 (aplikasi yang layanannya hanya berorientasi pada pemenuhan tujuan bisnis) adalah lebih cepat dan lebih mudah mengumpulkan data karena kontributornya berasal dari berbagai sumber, tingkatan dan bidang keahlian.

2.3.3. Kriteria *Web 2.0*

Biasanya terdapat tiga kriteria yang harus dipenuhi sebuah aplikasi *web* agar dapat dinilai sebagai aplikasi *web* 2.0 (Zibriel dan Supangkat, 2008).

1. Menggunakan SOA *(Service Oriented Architecture*)

   Web 2.0 umunya menggunakan SOA , dalam melaksanakan fungsinya sebagai penyedia layanan. SOA adalah sebuah konsep arsitektur sistem komputer yang membuat dan menggunakan langkah-langkah proses bisnis dalam bentuk paket layanan. Bentuk paket layanan yang dimaksud oleh SOA untuk berusaha membungkus kerumitan yang terjadi dari sudut pandang pengguna sistem. Penggunaan SOA memungkinkan perancang sistem untuk menghubungkan

berbagai aplikasi yang berlainan jenis tanpa perlu disadari oleh penggunanya. Oleh karena itu implementasi SOA biasanya menggunakan GUI (*Graphic User Interface*) untuk membungkus cara kerja aplikasi yang sebenarnya. Karakter utama dari aplikasi SOA adalah layanan yang menunggu secara terus-menerus untuk digunakan.

2. Menggunakan RIA (*Rich Internet Aplication*)

   RIA adalah aplikasi web yang dapat memberikan fitur apapun fungsi aplikasi desktop kepada para penggunannya. Artinya beberapa keunggulan atau kemudahan pada saat menggunakan aplikasi yang berjalan di atas *desktop* dapat dilakukan juga oleh aplikasi web RIA yang berjalan di suatu *server* serta diakses oleh pengguna sistem hanya dengan menggunakan bantuan sebuah *browser.* Contoh dari kemudahan aplikasi *desktop* yang telah beradaptasi oleh aplikasi *web* RIA adalah fitur *drag-and-drop* fitur *shortcut*, fitur *recovery*. Pada umumnya aplikasi *web* RIA hanya mengirimkan sejumlah data yang diperlukan klien tetapi tetap menyimpan seluruh data utama (seperti status pengguna) pada sisi *server* aplikasi. Contoh teknologi yang digunakan untuk mewujudkan RIA adalah Flash dan Ajax.

3. Menggunakan pendekatan *Social Web*

   Kriteria yang terakhir yang sekaligus merupakan daya tarik dari aplikasi web 2.0 adalah menggunakan pendekatan *Social Web* dalam memperkaya layanan yang diberikan. Dalam konsep *Social Web* setiap pengguna aplikasi *web* (terdaftar atau tidak) diminta untuk saling berkolaborasi untuk menambah, menghapus, menyunting ataupun mengkategorikan konten dari sebuah

layanan sehingga kualitas dan kegunaan layanan benar-benar ditentukan oleh kontribusi dari setiap pengguna layaknya sebuah komunitas dalam dunia nyata.

2.4. Metode Perancangan Sistem

2.4.1. Bagan Alir Dokumen *(Document Flowmap)*

Bagan alir dokumen menggambarkan aliran dokumen dan informasi antar *area* pertanggungjawaban didalam sebuah organisasi. Bagan alir ini menelusuri sebuah dokumen dari asalnya sampai tujuannya. Secara rinci bagan alir ini menunjukkan dari mana dokumen tersebut berasal, distribusinya, tujuan digunakan-nya dokumen tersebut dan lain-lain. Bagan alir ini bermanfaat untuk menganalisis kecukupan prosedur pengawasan dalam sebuah sistem. Bagan alir dokumen disebut juga bagan alir formulir yang merupakan yang menunjukkan arus dari laporan dan formulir termasuk tembusannya.

2.4.2. Diagram Konteks

Diagram konteks menggambarkan hubungan antara sistem dengan entitas luarnya. Diagram konteks berfungsi sebagai transformasi dari satu proses yang melakukan transformasi data *input* menjadi data *output*. Entitas yang dimaksud adalah entitas yang mempunyai hubungan langsung dengan sistem.

Suatu diagram konteks selalu mengandung satu dan hanya satu proses saja. Proses ini mewakili proses dari seluruh sistem. Diagram konteks ini menggambarkan hubungan *input* atau *output* antara sistem dengan dunia luarnya.

2.4.3. *Data Flow Diagram*

Data Flow Diagram (DFD –DAD/Diagram Alir Data) memperlihatkan hubungan fungsional dari nilai yang dihitung oleh sistem, termasuk nilai masukan, nilai keluaran, serta tempat penyimpanan internal. DAD adalah gambaran grafis yang memperlihatkan aliran data dari sumbernya dalam objek kemudian melewati proses yang mentransformasinya ke tujuan yang lain, yang ada pada objek lain. DAD sering digunakan untuk menggambarkan suatu sistem yang telah ada atau sistem baru yang akan dikembangkan secara logika tanpa mempertimbangan lingkungan fisik dimana data tersebut mengalir. DFD merupakan alat yang digunakan pada metodologi pengembangan sistem yang terstruktur (*structured analysis and design*). DFD merupakan alat yang cukup populer sekarang ini, karena dapat menggambarkan arus data di dalam sistem dengan terstruktur jelas.

Beberapa simbol yang digunakan dalam *Data Flow Diagram* (DFD) antara lain:

1. *External Entity* (kesatuan luar) atau *boundary* (batas sistem)

   Setiap sistem pasti mempunyai batas sistem (*boundary*) yang memisahkan suatu sistem dengan lingkungan luarnya. Sistem akan menerima input dan menghasilkan output kepada lingkungan luarnya. Kesatuan luar (*external entity*) merupakan kesatuan (*entity*) di lingkungan luar sistem yang dapat berupa orang, organisasi atau sistem lainnya yang berada di lingkungan luarnya yang akan memberikan *input* atau menerima *output* dari sistem.

2. *Data Flow* (arus data)

   Arus data (*data flow*) di DFD diberi simbol suatu panah. Arus data ini

mengalir diantara proses (*process*), simpanan data (*data strore*) dan kesatuan luar (*external entity*). Arus data ini menunjukkan arus dari data yang dapat berupa masukan untuk sistem atau hasil dari proses sistem.

3. *Process* (proses)

   Suatu proses adalah kegiatan atau kerja yang dilakukan oleh orang , mesin atau kompuiter dari hasil suatu arus data yang masuk ke dalam proses untuk dihasilkan arus data yang akan keluar dari proses. Untuk *physical data flow diagram* (PDFD), proses dapat dilakukan oleh orang, mesin atua komputer, sedangkan untuk *logical data flow diagram* (LDFD), suatu proses hanya menunjukkan proses dari komputer. Setiap proses harus diberi penjelasan yang lengkap meliputu identifikasi proses, nama proses dan pemroses.

4. *Data Store* (simpanan luar)

   Simpanan data (*data store*) merupakan simpanan dari data yang dapat berupa, yaitu suatu file atau *database* di sistem komputer, suatu arsip atau catatan manual, suatu kotak tempat data di meja seseorang, suatu tabel acuan manual, dan suatu agenda atau buku.

2.4.4. Diagram Nol (*Ovetview Diagram*)

Diagram nol adalah diagram yang menggambarkan proses dari dataflow diagram. Diagram nol memberikan pandangan secara menyeluruh mengenai sistem yang ditangani, menunjukkan tentang fungsi-fungsi utama atau proses yang ada, aliran data, dan eksternal entity. Pada level ini sudah dimungkinkan untuk digambarkan level selanjutnya, simbol ‘*’ atau ‘P’ (*functional primitive*) dapat

ditambahkan pada akhir nomor proses. Keseimbangan input dan output (*balancing*) antara diagram nol dengan diagram konteks harus terpelihara.

2.4.5. Diagam Rinci (*Level Diagram*)

Diagram rinci adalah diagram yang menguraikan proses apa yang ada dalam diagram zero atau diagram level diatasnya.

2.4.6. Penomoran Level pada DFD

Didalam satu level seharusnya tidak terdapat lebih dari 7 buah proses dan maksimal 9, bila lebih maka harus dilakukan dekomposisi.

Tabel 2.1. Tabel Penomoran Level Pada DFD

Nama Level	Nama Diagram	Nomor Process
0	Context	
1	Diagram 0	1.0,2.0,3.0,…
2	Diagram 1.0	1.1,1.2,1.3,…
2	Diagram 2.0	2.1,2.2,2.3,…
2	Diagram 3.0	3.1,3.2,3.3,…
3	Diagram 1.1	1.1.1,1.1.2,…
3	Diagram 1.2	1.2.1,1.2.2,…
3	Diagram 1.3	1.3.1,1.3.2,…

2.4.7. *Entity Relationship Diagram*

ERD hanya berfokus pada data, dengan menunjukkan "jaringan data" yang ada untuk suatu sistem yang diberikan. ERD sangat berguna bagi aplikasi di mana data dan hubungan yang mengatur data sangatlah kompleks. ERD pada mulanya diusulkan oleh Peter Chen untuk desain sistem *database* relasional dan telah dikembangkan oleh yang lainnya. Serangkaian komponen utama diidentifikasikan untuk ERD : objek data, atribut, hubungan dan berbagai tipe indikator. Tujuan utama dari ERD adalah untuk mewakili objek data dan hubungan mereka.

Kardinalitas model data harus dapat merepresentsikan jumlah peristiwa dari objek di dalam hubungan yang diberikan. Tillmann mendefinisikan kardinalitas dari *object-relationship pair* dengan cara sebagai berikut: kardinalitas merupakan spesifikasi dari sejumlah peristiwa dari satu [objek] yang dapat dihubungkan ke sejumlah peristiwa dari [objek] yang lain. Dengan mempertimbangkan semua kombinasi dari 'satu' dan 'banyak', dua [objek] dapat dihubungkan sebagai:

1. Satu-ke-satu (1:1)
2. Satu-ke-banyak (1:N)
3. Banyak-ke-satu (N:1)
4. Banyak-ke-banyak (M:N)

2.4.8. Kamus Data

Kamus data (KD) atau *data dictionary* (DD) atau disebut juga dengan istilah *system data dictionary* adalah katalog fakta tentang data dan kebutuhan-

kebutuhan informasi dari suatu sistem informasi. Dengan menggunakan kamus data, analisis sistem dapat mendefinisikan data yang mengalir di sistem dengan lengkap. Kamus data dibuat pada tahap analisis sistem dan digunakan baik pada tahap analisis maupun pada tahap perancangan sistem.

Pada tahap analisis dan perancangan, kamus data dapat digunakan sebagai alat komunikasi antara analsisis sistem dengan pemakai sistem tentang data yang mengalir di sistem, yaitu tentang data yang masuk ke sistem data tentang informasi yang dibutuhkan oleh pemakai sistem.

Pada tahap perancangan, kamus data digunakan untuk merancang *input*, merancang laporan-laporan dan *database*. Kamus data dibuat berdasarkan arus data yang ada di DFD. Arus data di DFD sifatnya global, hanya ditunjukkan nama arus datanya saja. Keterangan lebih lanjut tentang struktur data dari arus data di DFD secara lebih rinci dapat dilihat di kamus data. Kamus data harus dapat mencerminkan keterangan yang jelas tentang data yang dicatatnya, maka kamus data harus memuat nama arus data, alias, bentuk data, arus data, penjelasan, periode, volume, dan struktur data (Jogiyanto, 2005).

2.5. Basis Data

Basis data terdiri dari dua kata, yaitu Basis dan Data. Basis kurang lebih dapat diartikan sebagai markas atau gudang, tempat bersarang atau berkumpul. Sedangkan Data adalah representasi fakta dunia nyata yang mewakili suatu objek seperti manusia (pegawai, siswa, pembeli, pelanggan), barang, hewan, peristiwa, konsep, keadaan, dan sebagainya, yang direkam dalam bentuk angka, huruf,

simbol, teks, gambar, bunyi, atau kombinasinya.

Menurut fathansyah (1999) Basis data dapat didefinisikan dari sejumlah sudut pandang seperti:

a. Himpunan kelompok data (arsip) yang saling berhubungan yang diorganisasi sedemikian rupa agar kelak dapat dimanfaatkan kembali dengan cepat dan mudah.
b. Kumpulan data yang saling berhubungan yang disimpan secara bersama sedemikian rupa dan tanpa pengulangan (redudansi) yang tidak perlu, untuk memenuhi berbagai kebutuhan.
c. Kumpulan file atau tabel atau arsip yang saling berhubungan yang disimpan dalam media penyimpanan elektronik.

Prinsip utama Basis data adalah pengaturan data atau arsip. Dan tujuannya adalah kemudahan dan kecepatan dalam pengambilan kembali data atau arsip.

2.5.1. Manajemen Sistem Basis Data (*Database Managgement System)*

Manajemen Sistem Basis Data (*Database Managgement System*) adalah perangkat lunak yang didesain untuk membantu dalam hal pemeliharaan dan utilitas kumpulan data dalam jumlah besar. DBMS dapat menjadi alternatif penggunaan secara khusus untuk aplikasi, semisal penyimpanan data dalam file dan menulis kode aplikasi yang spesifik untuk pengaturannya.

Penggunaan DBMS untuk suatu aplikasi tergantung pada kemampuan dan dukungan DBMS yang beroperasi secara efisien. Sehingga agar bisa menggunakan DBMS dengan baik, perlu diketahui cara kerja dari DBMS tersebut.

Pendekatan yang dilakukan untuk menggunakan DBMS secara baik meliputi implementasi DBMS dan arsitektur secara mendetail untuk dapat memahami desain dari suatu basis data.

2.5.2. Normalisasi

Normalisasi merupakan tahapan perancangan dalam membangun basis data relasional yang tidak secara langsung berkaitan dengan model data, tetapi dengan menerapkan sejumlah aturan dan kriteria standar untuk menghasilkan struktur tabel yang normal. Adapun langkah-langkah untuk menghasilkan struktur tabel yang normal, diantaranya yaitu :

a Bentuk *Unnormallized*

Bentuk *unnormal* adalah bentuk tabel dengan mencantumkan semua *field* data yang ada.

b Bentuk Normal Pertama atau 1 NF (*First Normal Form*)

Bentuk normal pertama terpenuhi jika sebuah tabel tidak memiliki atribut bernilai banyak (*Multivalued Atribute)* atau lebih dari satu atribut dengan domain nilai yang sama.

c Bentuk Normal Kedua atau 2 NF (*Second Normal Form*)

Bentuk normal kedua terpenuhi jika pada sebuah tabel semua atribut yang tidak termasuk dalam *key primer* memiliki ketergantungan fungsional pada *key primer* secara utuh.

d Bentuk Normal Ketiga atau 3 NF (*Third Normal Form*)

Bentuk normal ketiga terpenuhi jika dan hanya jika semua atribut bukan kunci

memiliki *dependensi* transitif terhadap kunci primer.

e *Boyce-Codd Normal Form* (BCNF)

Suatu tabel dikatakan berada pada BCNF jika dan hanya jika semua ketergantungan fungsional dengan notasi X → Y, maka X harus merupakan *superkey* pada tabel tersebut.

f Bentuk Normal Keempat atau 4 NF (*Fourth Normal Form*)

Suatu tabel dikatakan berada pada normal keempat jila tidak mengandung dua atribut atau lebih yang bernilai banyak.

g Bentuk Normal Kelima atau 5 NF (*Fifth Normal Form*)

Bentuk normal kelima berkaitan dengan ketergantungan relasi antar tabel (*Join Dependency*).

2.6. *Hypertext Markup Language* (HTML)

Hypertext Markup Language (HTML) adalah bahasa yang digunakan untuk membuat dokumen pada *World Wide Web* (WWW). HTML adalah pengembangan dari SGML (*Standard Generelize Markup Language*), dapat dibuat dengan menggunakan berbagai *tools* dari peng-*edit* teks yang sederhana. HTML menggunakan *tag-tag* tertentu untuk menyusun sebuah dokumen, seperti <body> dan </body> untuk membuat isi dari suatu dokumen yang dibuat ([W3C], 2005).

2.7. PHP dan MySQL

PHP merupakan bahasa pemrograman untuk *script web server-side*. Bahasa pemrograman PHP diciptakan pertama kali oleh Rasmus Lerdorf, seorang pemrogram C yang sangat handal. Semula PHP hanya digunakan untuk mencatat seberapa jumlah pengunjung pada *hompage*-nya. Rasmus adalah salah seorang pendukung *open source*. Karen itulah ia mengeluarkan *Personal Home Page Tools* versi 1.0 secara gratis atau *freeware* pada tahun 1995. Setelah mempelajari YACC dan GNU Bison, Rasmus menambah kemampuan pada PHP 1.0 dan menertbitkan PHP 2.0 sebagai pengembangan dari PHP 1.0 yang telah ada sebelumnya.

PHP 2.0 mampu berhubungan dengan *database* dan dapat diintegrasikan dengan HTML. Pada tahun 1996, PHP telah digunkana oleh banyak *website* di dunia. Sebuah kelompok pengembang *software* yang terdiri dari Rasmus, Zeew Suraski, Andi Gutman, Stig Bakken, Shane Carveo dan Jim Winstead bekerja selama tujuh bulan untuk melakukan penyempurnan terhadap PHP 2.0, akhirnya pada tanggal 6 juni 1998, PHP 3.0 resmi dikeluarkan ke dunia pemrograman. Pada pertengahan tahun 1999, Zend merilis interpreter PHP baru dan rilis tersebut dikenal dengan PHP 4.0. PHP 4.0 adalah versi PHP yang paling banyak dipakai. Versi ini banyak dipakai sebab versi ini mampu dipakai untuk membangun aplikasi *web* kompleks tetapi tetap memiliki kecepatan proses dan stabilitas yang tinggi. PHP versi 4.2 telah diterbitkan pada tanggal 22 April 2002 dengan log kelompok fungsi, sampai dengan versi 4.3.7 tercatat 125 kelompok fungsi yang dimiliki oleh PHP. Pada juni 2004, Zend merilis PHP 5.0. Versi ini adalah versi

yang paling mutakhir dari PHP.

Kelebihan PHP

PHP memiliki beberapa kelebihan yang tidak dimiliki oleh bahasa-bahasa sejenisnya, yaitu:

1. Bahasa pemrograman PHP adalah sebuah bahasa *script* yang tidak melakukan sebuah kompilasi dalam penggunaanya.
2. *Web Server* yang mendukung PHP dapat ditemukan dimana - mana dari mulai IIS sampai dengan apache, dengan configurasi yang relatif mudah.
3. Dalam sisi pengembangan lebih mudah, karena banyaknya milis - milis dan *developer* yang siap membantu dalam pengembangan.
4. Dalam sisi pemahamanan, PHP adalah bahasa *scripting* yang paling mudah karena referensi yang banyak.
5. PHP adalah bahasa *open source* yang dapat digunakan di berbagai mesin (*linux, unix, windows*) dan dapat dijalankan secara *runtime* melalui *console* serta juga dapat menjalankan perintah-perintah sistem.

MySQL (*My Structure Query Language*) adalah sebuah program pembuat *database* yang bersifat *open source*. MySQL sebenarnya produk yang berjalan pada *platform Linux*. Karen sifatnya *open source*, maka MySQL dapat dijalankan pada semua *platform* baik *Windows* maupun *Linux*. Selain itu, MySQL juga merupakan program pengakses *database* yang bersifat jaringan sehingga dapat digunakan untuk aplikasi *Multi User* (Banyak Pengguna). Saat ini *database* MySQL telah digunakan hampir oleh semua *programer database*, apalagi dalam pemrograman *web*.

Kelebihan lain dari MySQL adalah menggunakan bahasa *Query* standar yang dimiliki SQL (*Structure Query Language*). SQL adalah suatu bahasa permintaan yang terstruktur yang telah distandarkan untuk semua program pengakses *database* seperti Oracle, Posgres SQL, SQL Server, dan lain-lain.

Sebagai sebuah program penghasil *database*, MySQL tidak dapat berjalan sendiri tanpa adanya sebuah aplikasi lain (*interface*). MySQL dapat didukung oleh hampir semua program aplikasi baik yang *open source* seperti PHP maupun yang tidak, yang ada pada *platform Windows* seperti Visual Basic, Delphi, dan lainnya (Nugroho, 2004).

2.8. Rekayasa *Web* (*Web Engineering*)

Aplikasi dan sistem berbasis *web* (*WebApp*) memberi suatu susunan *content* yang kompleks kepada *end users*, dimana rekayasa *web* merupakan proses yang digunakan untuk menghasilkan aplikasi *web* yang berkualitas tinggi (Pressman, 2001). Rekayasa *web* bukan merupakan hasil penggandaan sempurna dari rekayasa perangkat lunak, melainkan lebih kepada mengambil beberapa fundamental dari konsep dan prinsip rekayasa perangkat lunak yang menekankan pada teknik dan aktivitas manajemen yang sama.

Ada beberapa karakteristik yang perlu diperhatikan dari aplikasi *web*:

1 Kesiapan (*Immediacy*)

Aplikasi *web* memiliki kesiapan yang tidak akan ditemukan pada tipe perangkat lunak lain. Artinya waktu memasarkan halaman *web* secara lengkap

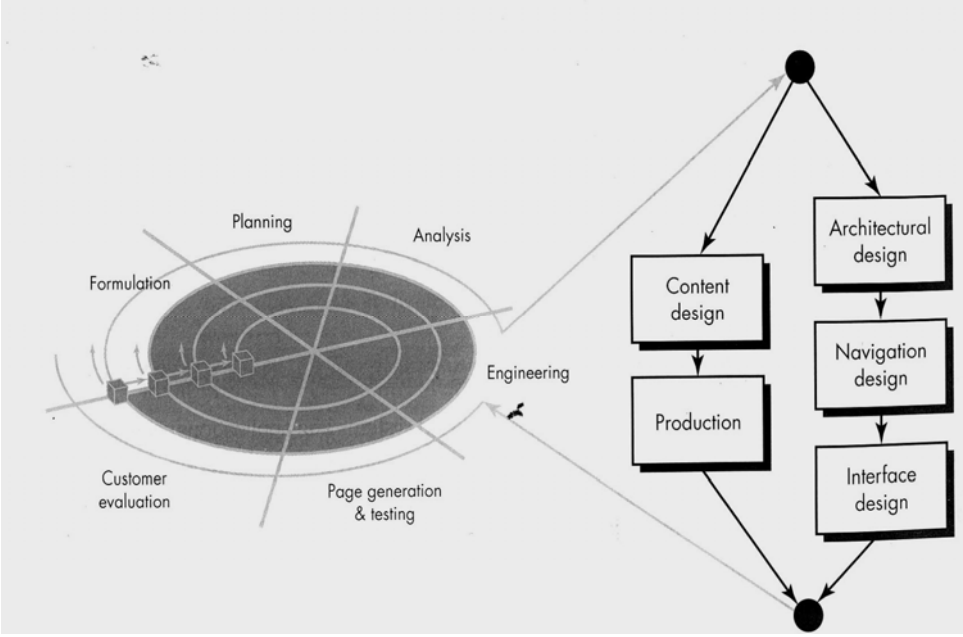
dapat dilakukan dalam beberapa hari atau beberapa minggu. Pengembangan harus menggunakan metode untuk perancangan, analisis, desain, implementasi, dan pengujian yang disesuaikan dengan jadwal yang dibutuhkan dalam mengembangkan aplikasi *web*.

2 Keamanan (*Security*)

Aplikasi *web* tersedia pada suatu akses jaringan, yang berarti sulit untuk membatasi populasi dari *end user* yang mengakses aplikasi. Untuk menjaga *content* yang sensitif dan menyediakan cara yang aman untuk mentransmisikan data, maka perlu diterapkan suatu aturan keamanan yang kuat secara menyeluruh pada infrastruktur yang mendukung suatu aplikasi *web* dan dalam aplikasi itu sendiri.

3 Estetika (*Aesthetics*)

Bagian yang tidak terbantahkan dari daya tarik aplikasi *web* adalah rasa dan tampilan aplikasi itu sendiri. Ketika suatu aplikasi didesain untuk memasarkan atau menjual suatu produk atau pemikiran, estetis mungkin memiliki tingkat kesuksesan yang sama dengan desain tekniknya. Tahapan proses rekayasa web dapat dilihat pada Gambar 2.1.



Gambar 2.1. Tahapan proses pada rekayasa perangkat *web* (Pressman, 2001)

Tahapan-tahapan dalam proses rekayasa perangkat *web* (Pressman, 2001):

1 Formulasi (*Formulation*)

Tahap formulasi merupakan tahap untuk melakukan pengidentifikasian tujuan pembuatan aplikasi *web* dan batasan pengembangan sistem, penganalisaan model sesuai dengan spesifikasi kebutuhan sistem, serta penentuan sarana yang akan digunakan dengan tujuan untuk mendapatkan hasil keluaran yang baik.

2 Perencanaan (*Planning*)

Tahap perencanaan merupakan tahap untuk melakukan perkiraan biaya secara keseluruhan, mengevaluasi resiko yang mungkin terjadi, dan mendefinisikan jadwal pengembangan aplikasi.

3 Analisis (*Analysis*)

Tahap analisis merupakan tahap untuk mengidentifikasikan *content* yang akan ditampilkan dalam aplikasi dan menentukan kebutuhan untuk estetika pada desain.

4 Perancangan (*Engineering*)

Proses perancangan pada proses rekayasa *web* disebut *engineering*, yang terdiri dari beberapa tahap, yaitu:

a. Perancangan Isi (*Content Design*)

Pada tahap ini merancang *content* dari aplikasi *web* dan merancang basis data yang digunakan berdasarkan kebutuhan informasi yang telah diidentifikasi pada tahap analisis.

b. Perancangan Arsitektur (*Architectural Design*)

Tahap ini difokuskan pada struktur *hypermedia* secara keseluruhan dari aplikasi *web.* Struktur arsitektur berkaitan erat dengan tujuan dari pengembangan situs, *content* yang disediakan, dan pengguna yang mengunjungi situs.

c. Perancangan Keluaran (*Production*)

Pada tahap ini merancang keluaran yang akan dihasilkan dari aplikasi *web*. Informasi yang dihasilkan harus memenuhi kebutuhan dari pengguna. Informasi terdiri dari informasi umum sampai informasi yang merupakan hasil *query* pada sistem.

d. Perancangan Navigasi (*Navigation Design*)

Pada tahap ini merancang navigasi untuk masing-masing pengguna yang bertujuan untuk mangatur hak otorisasi.

b. Perancangan Antarmuka (*Interface Design*)

Perancangan antarmuka sebelumnya dibuat dalam bentuk *tag* HTML yang kemudian dipecah dan disimpan dalam bentuk ekstensi PHP untuk mempermudah proses pengkodean dan penggabungan, sehingga hampir seluruh *file* yang dieksekusi berupa *file* PHP.

5 Pembuatan Halaman dan Pengujian (*Page Generation and Testing*)

Tahap ini merupakan pembuatan halaman *web* dalam bentuk HTML, yang didasarkan atas hasil proses perancangan isi. Sedangkan tahap pengujian dilakukan untuk mengetahui kemungkinan terjadinya kesalahan pada *script* atau *form* agar kemudian dapat dilakukan perbaikan sehingga aplikasi dapat berjalan dengan baik dan benar.

6 Evaluasi (*Customer Evaluation*)

Tahap evaluasi dilakukan untuk mengetahui kualitas suatu aplikasi yang mengandung faktor-faktor sebagai berikut:

a. *Usability*

Evaluasi diantaranya didasarkan pada nilai estetis dan pemahaman isi situs.

b. *Functionality*

Evaluasi didasarkan pada kemampuan proses pencarian data, proses navigasi, dan *browsing*.

c. *Reliability*

Evaluasi didasarkan pada ketepatan proses *link* dan validasi dari *input* yang diberikan.

d. *Efficiency*

Evaluasi didasarkan pada kecepatan dalam menampilkan gambar dan menampilkan halaman baru.

e. *Maintainability*

Evaluasi didasarkan pada kemudahan untuk memperbaiki dan kemampuan untuk beradaptasi dengan perangkat lunak yang digunakan.

2.9. Struktur Arsitektur *Web*

Desain arsitektur untuk sistem aplikasi berbasis *web* terfokus pada keseluruhan struktur *hypermedia*, pola desain aplikasi dan *template* yang terstruktur. Menurut Pressman (2001) terdapat empat struktur yang berbeda yang dapat digunakan jika hendak mengembangkan desain untuk aplikasi *web* yang tipikal, yaitu struktur linier, struktur *grid*, struktur hirarki dan struktur *web* murni (*networked*).

1. Struktur Linier.

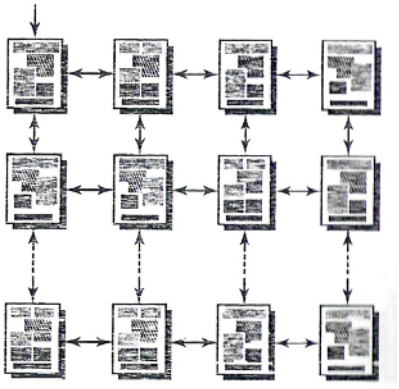
Digunakan ketika urutan interaksi yang diperkirakan dengan bentuk variasi yang umum. Sebuah contoh klasik dapat menjadi presentasi tutorial dimana informasi beserta grafik, video pendek atau audio yang bersangkutan dipresentasikan setelah informasi yang dibutuhkan telah didapat. Urutan presentasi isi didefinisikan sebelumnya dan linier secara umum (Gambar 2.2).



Gambar 2.2. Struktur Linier (Pressman, 2001)

2. Struktur *Grid*.

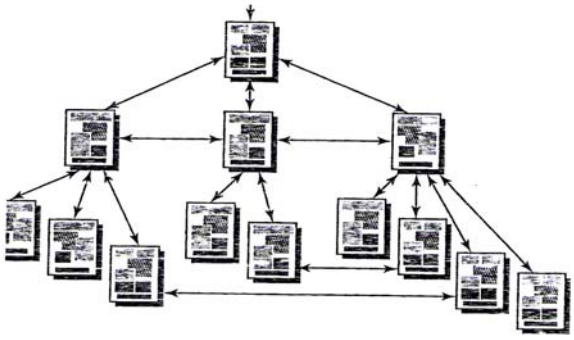
Merupakan pilihan arsitektur yang dapat diaplikasikan ketika isi dari *WebApp* disusun berdasarkan kategorinya didalam dua atau lebih dimensi. Arsitektur *web* seperti ini berguna hanya jika isi yang didapatkannya bersifat regular (Gambar 2.3).



Gambar 2.3. Struktur *grid* (Pressman, 2001)

3. Struktur Hirarki.

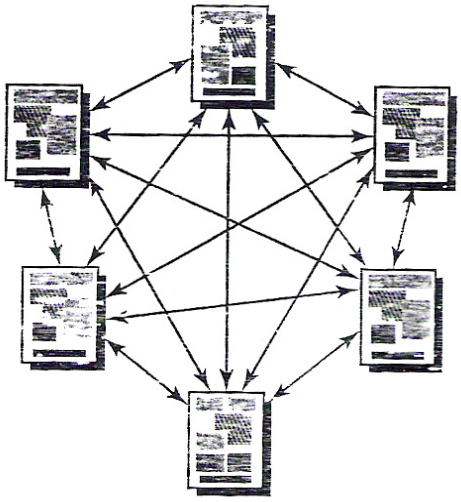
Merupakan struktur arsitektur *WebApp* yang paling umum. Struktur *WebApp* yang hirarki dapat didesain dalam perlakuan yang memperbolehkan kontrol *flow* secara horisontal, melalui cabang-cabang vertikal dari struktur. Harus diingat bahwa meskipun tipe ranting seperti itu mengijinkan navigasi yang baik melalui isi *WebApp* namun dapat menyebabkan kebingungan bagi sebagian penguna (Gambar 2.4).



Gambar 2.4. Struktur Hirarki (Pressman, 2001)

4. Struktur *Web* Murni (*Networked).*

Pada struktur ini serupa dengan cara-cara biasa terhadap arsitektur yang dikembangkan untuk sistem berorientasi objek. Komponen-komponen arsitektural (dalam kasus ini, halaman-halaman *web*) didesain sedemikian rupa sehingga mereka dapat melewatkan kontrol (melalui *hypertext link*) kepada setiap komponen lain di dalam sistem secara virtual.(Gambar 2.5).



Gambar 2.5. Struktur *Networked* (Pressman, 2001)