BAB II

LANDASAN TEORI

2.1 Pengendalian Demam Berdarah Dengue

Penyakit Demam Berdarah Dengue (DBD) disebabkan virus dan ditularkan lewat nyamuk merupakan salah satu masalah kesehatan masyarakat di Indonesia, yang cenderung semakin luas penyebarannya sejalan dengan meningkatnya mobilitas dan kepadatan penduduk. Pengendalian penyakit demam berdarah dengue pada dasarnya dilakukan sesuai peraturan KEMENKES NOMOR 581/MENKES/SK/VII/1992 yang berisi, upaya pengendalian penyakit demam berdarah dengue (DBD) dilakukan melalui kegiatan pencegahan, penemuan, pelaporan penderita, pengamatan penyakit, dan penyelidikan epidemiologi, penanggulangan seperlunya, penanggulangan lain dan penyuluhan masyarakat. Pelaksanaan kegiatan pengendalian penyakit demam berdarah dengue dilakukan oleh pemerintah dan masyarakat dibawah koordinasi kepala wilayah/daerah (Depkes, 2011).

2.2 Sistem Informasi

Sistem informasi adalah suatu sistem di dalam suatu organisasi yang mempertemukan kebutuhan pengolahan transaksi harian yang mendukung fungsi operasi organisasi yang bersifat manajerial dengan kegiatan strategi dari suatu organisasi untuk dapat menyediakan kepada pihak luar tertentu dengan laporan-laporan yang diperlukan (Bocij, 2008).

2.2.1 Sistem

Sistem dapat didefinisikan sebagai kumpulan komponen yang saling terkait yang bekerjasama untuk mencapai tujuan bersama. Fungsi sistem adalah untuk menerima masukan dan mengubah ini menjadi output (Bocij, 2008).

2.2.2 Informasi

Seperti konsep data, ada beberapa definisi informasi yang umum digunakan, yaitu:

a. Data yang telah diolah sehingga mereka bermakna
b. Data yang telah diolah untuk tujuan
c. Data yang telah dipahami dan dimengerti oleh penerima

Tigahal penting dapat ditarik dari definisi ini pertama, ada proses yang jelas dan logis yang digunakan untuk menghasilkan informasi. Proses ini melibatkan pengumpulan data untuk sebuah proses transformasi dalam rangka menciptakan informasi. Kedua, informasi melibatkan dan menempatkan beberapa inisial data dalam bentuk konteks yang bermakna, sehingga mereka dapat dipahami dan ditindak lanjuti. Ketiga, informasi yang dihasilkan untuk suatu tujuan, untuk melayani kebutuhan informasid ari beberapa jenis (Bocij, 2008).

2.3 Demam Berdarah Dengue

Demam Berdarah Dengueadalah penyakit menular yang disebabkan oleh virus dari golongan *Arbovirosis* group A dan B. Di Indonesia penyakit akibat gigitan nyamuk yang paling bermasalah adalah Demam Berdarah Dengue (DBD), Chikungunya dan *Japanese Encephalitis* (JE). Penyakit Demam Berdarah Dengue (DBD) mulai dikenal di Indonesia sejak tahun 1968 di Surabaya dan Jakarta, dan

setelah itu jumlah kasus Demam Berdarah Dengue (DBD) terus bertambah seiring dengan meluasnya daerah *endemis* Demam Berdarah Dengue (DBD).Penyakit ini menimbulkan kejadian luar biasa (KLB) yangberdampak buruk pada segi sosial dan ekonomi. Kerugian sosial yang terjadi antara lain karena menimbulkan kepanikan dalam keluarga, kematian anggota keluarga, dan berkurangnya usia harapan penduduk. Pada tiga tahun terakhir (2008-2010) jumlah rata-rata kasus yang dilaporkan sebanyak 150.822 kasus dengan rata-rata kematian 1.321. Situasi kasus Demam Berdarah Dengue (DBD) tahun 2011 sampai dengan juni 2011 dilaporkan sebanyak 16.612 orang dengan kematian sebanyak 142 orang (Depkes, 2011). Dari jumlah kasus tersebut, proporsi penderita Demam Berdarah Dengue (DBD) pada perempuan sebesar 50,33% dan laki-laki sebesar 49,67%. Disisi lain angka kematian akibat DBD pada perempuan lebih tinggi dibanding laki-laki (Depkes, 2011).

2.4 Monitoring

Monitoring adalah kegiatan pemantauan atau pengamatan yang berlangsung selama kegiatan berjalan untuk memastikan dan mengendalikan keserasianpelaksanaan program dengan perencanaan yang telah ditetapkan. (Rinda Hedwig, 2007). Disini penderita DBD di *monitoring* sesuai dengan peraturan yang berlaku terdiri dari penyelidikan epidemiologi (PE) yang dilaporkan melalui dokumen kasus harian dan dilakukan penanggulangan seperlunya berdasarkan hasil penyelidikan tersebut. Penyelidikan Epidemologi sendiri adalah kegiatan pencarian penderita DBD disekitar tempat tinggal penderita termasuk tempat-tempat umum diradius sekurang-kurangnya 100m.

2.5 Evaluasi

Evaluasi adalah upaya menilai kualitas program dan hasil-hasilnya secara berkala dengan menggunakan pendekatan yang tepat. Evaluasi penelitian berarti upaya menggali informasi terhadap proses dan hasil penelitian untuk menilai kualitasnya dengan menggunakan pendekatan yang tepat (Hedwig, 2007).

2.5.1 Ukuran Epidemologi

Ukuran (parameter) frekuensi penyakit yang paling sederhana dan menghitung jumlah individu yang sakit pada suatu populasi. Frekuensi tersebut bermanfaat bagi petugas kesehatan di daerah untuk mengalokasi dana atau kegiatan penanggulangan. Ukuran-ukuran *epidemiologi* yang sering digunakan dalam kegiatan pengendalian DBD adalah *Insidence Rate* (IR) dan *Case Fatality Rate* (CFR).

a. Angka Kesakitan/*Insiden Rate* (IR)

IR adalah ukuran yang menunjukan kecepatan kejadian (baru) penyakit populasi, IR merupakan proporsi antara jumlah orang yang menderita penyakit DBD dan jumlah orang dalam resiko X lamanya penderita dalam resiko.

$$\text{IR} = \frac{\text{Jumlah Kasus Baru Penyakit}}{\text{Jumlah Orang Yang Beresiko}} X\ 100\%$$

b. Angka Kematian/*Case Fatality Rate* (CFR)

CFR adalah angka kematian yang di akibatkan dari suatu penyakit dalam suatu waktu tertentu dikalikan 100%

$$\text{CFR} = \frac{\text{Jumlah Kematian}}{\text{Jumlah Kasus}} X\ 100\%$$

2.5.2 Ukuran Survielans Nyamuk

Surveilans Jentik nyamuk DBD meliputi proses pengumpulan, pencatatan, pengolahan, analisis dan interpretasi data jentik serta penyebarluasan informasi secara sistematis dan terus menerus agar dapat memutus rantai penularan dan penyhebaran nyamuk. Berikut adalah ukuran yang dipakai untuk mengetahui angka bebas dari jentik *Aedes Aegypti* :

a. Angka Bebas Jentik (ABJ) :

Jumlah atau bangunan rumah yang tidak ditemukan jentik.

$$\text{ABJ} = \frac{\text{Jumlah rumah/bangunan yang tidak ditemukan jentik}}{\text{Jumlah rumah/bangunan yang diperiksa}} X\ 100\%$$

2.6 *Software Engineering Body of Knowledge* (SWEBOK)

SWEBOK menggambarkan pengetahuan secara umum tentang rekayasa perangkat lunak yang dibagi kedalam 10 area pengetahuan (*Knowledge Areas*) atau disebut Kas." *Software Engineering Body of Knowledge* (SWEBOK) adalah produk dari komite koordinasi rekayasa perangkat lunak disponsori oleh IEEE *Computer Society*. SWEBOK sendiri mempunyai panduan yang disebut *Guiede to the SWEBOK*, panduan ini dibuat untuk 5 tunjuan, yaitu :

a. Untuk memperlihatkan kesamaan pandangan tentang rekayasa perangkat lunak diseluruh dunia.
b. Untuk memperjelas tempat dan menetapkan batas dari rekayasa perangkat lunak dan hubungannya dengan disiplin ilmu lain seperti ilmu komouter, manajemen proyek, teknik komputer dan matematika.
c. Untuk memberi karakter isi dari disiplin ilmu rekayasa perangkat lunak.
d. Untuk meberikan akses topik ke SWEBOK

e. Untuk memberikan pengetahuan dasar bagi pengembangan kurikulum dan sertifikasi serta perizinan.

Berikut adalah penjabaran tentang ruang lingkup pengetahuan atau yang disebut juga *Knowledge Area* (KAs) yang digunakan sebagai panduan dalam mengembangkan aplikasi (England, 2004).

2.6.1 *Requirements*

Tahapan awal dalam membangun aplikasi, *Software Requirements* merupakan sebuah properti yang disajikan untuk memenuhi kebutuhan dalam menyelesaikan permasalahan yang ada akan diselesaikan oleh aplikasi tersebut. Menjabarkan bagaimana mengotomatiskan sebuah permasalahan sebuah tugas yang dihadapi oleh pengguna, membantu menganalisa proses bisnis perusahaan yang telah menggunakan aplikasi, menganalisa kekurangan yang ada, dan lainnya. Berikut penjabaran tentang beberapa tahapan yang ada pada *software requirement*:

a. Requirement Elicitation

Tahapan awal dalam pemenuhan *software requirements* makna dari kebutuhan mendatang ini berhubungan dengan darimana kebutuhan perangkat lunak itu sendiri dan bagaiman para pengembangan perangkat lunak dapat mengumpulkannya. Pada dasarnya, kegiatan yang dijabarkan adalah dari tiap individu dan tiap pemegang kendali sistem tersebut untuk membangun ketersinambungan antara pihak pengembang dan pengguna perangkat lunak itu sendiri.

b. Requirement Analysis

Tahapan ini mebahas tentang kegiatan menganalisa kegiatan manganalisa kebutuhan untuk :

1. Mendeteksi dan menyelesaikan ketidakcocokan yang ada pada tiap-tiap kebutuhan.
2. Menggali batasan yang ada pada perangkat lunak yang dikembangkan dan bagaimana perangkat lunak tersebut akan berinteraksi dengan sistem.
3. Menguraikan kebutuhan sistem yang akan digunakan sebagai kebutuhan perangkat lunak

c. Requirements spesification

Secara teknis pada kata "*specification*" mengacu pada banyaknya jumlah pekerjaan atau kemampuan perangkat lunak tersebut dalam mencapai tujuannya. Dalam sebuah istilah pengembangan perangkat lunak. "software requirements specification" secara khusus mengarah kepada hasil ketepatan, atau penyamaan elektronik, yang dapat ditinjau, dinilai, dan dibenarkan.

d. Requirement Verification and Validation

Beberapa dokumen requirements di atas menjadi bahan dari tahapan validasi dan verifikasi. Kebutuhan yang ada di validasi untuk menjamin bahwa pengembang dari perangkat lunak tersebut dapat memahami kebutuhan yang akan dicapai. Penyesuaian kebutuhan untuk standar perusahaan sangat penting untuk diperhatikan bahwa kebutuhan tersebut dimengerti, konsisten, dan lengkap.

2.6.2 Analisis

Tahap Analisis merupakan tahap identifikasi, seleksi, dan perencanaan sistem yang bertujuan untuk mendeteksi dan memberikan solusi antar kebutuhan serta mengetahui ruang lingkup perangkat lunak dan bagaimana perangkat lunak tersebut berinteraksi dengan lingkungan.

Tahapan analisis kebutuhan, menunjukkan tahapan-tahapan didalam analisis kebutuhan. Pada dasarnya, aktivitas analisis dibutuhkan dalam setiap proses dalam daur hidup pengembangan perangkat lunak. Dalam proses rekayasa kebutuhan, analisis pun dilakukan dalam setiap aktivitas-aktivitasnya. Aktivitas tersebut antara lain sebagai berikut :

1. *Domain Understanding* : Dalam tahapan ini, pengembang harus mengetahui bagaimana organisasi perusahaan beroperasi dan apa yang menjadi permasalahan pada sistem yang berjalan.
2. *Requirements Collection* : Tahapan ini merupakan tahapan pengumpulan kebutuhan akan sistem yang akan dibangun sehingga diperlukan adanya interaksi secara intensif dengan *stakeholder*.
3. *Classification* : Tahapan ini mengelompokkan hasil dari tahap kebutuhan sehingga menjadi lebih terstruktur untuk selanjutnya diorganisir kedalam kelompok-kelompok yang koheren.
4. *Conflict Resolution* : Tahapan ini berguna untuk menemukan dan menyelesaikan kebutuhan yang didalamnya terdapat konflik. Konflik tersebut dapat terjadi antara dua *stakeholder* yang saling terkait tetapi memiliki fasilitas yang tidak sesuai, atau dapat terjadi antara kebutuhan dan sumber daya.

5. *Prioritisation* : Tahap ini melakukan interaksi dengan *stakeholder* untuk mengidentifikasikan kebutuhan-kebutuhan prioritas dari masing-masing kebutuhan agar memenuhi sumber daya yang tersedia pada organisasi.
6. *Requirements Checking*: Menganalisis sekumpulan kebutuhan dari hasil tahapan sebelumnya untuk menverifikasi dan memvalidasi berdasarkan aspek kelengkapan, konsistensi, dan kebutuhan nyata.

Semua jenis kebutuhan yang telah diperoleh tersebut kemudian dituangkan dalam bentuk dokumen yang berisi tentang kebutuhan sistem secara keseluruhan. Dokumen ini menjelaskan secara rinci tentang kesepakatan antara pengembang dengan klien, desain perangkat lunak yang akan dibangun, segala resiko yang akan dihadapi dan jadwal pembuatan perangkat lunak. Dokumen ini sangat berguna bagi pihak yang ingin mengetahui tentang perangkat lunak yang akan dibangun namun tidak mengerti secara teknik karena dokumen ini menggunakan bahasa yang sederhana. Secara umum dokumen ini biasa disebut dengan *Software Requirements Spesification* (SRS).

Pada dokumen SRS akan dijelaskan juga mengenai kebutuhan fungsional dan non-fungsional dimana kebutuhan non-fungsional dibuat berdasarkan dokumen *IEEE standart* 803:1993. *IEEE* 803:1993 mengelompokkan kebutuhan non-fungsional kedalam sejumlah kategori kualitas dari suatu perangkat lunak. Kategori-kategori tersebut secara umum dibagi kedalam 2 kelompok, yaitu faktor kualitas eksternal dari perangkat lunak dan faktor kualitas internal perangkat lunak. Faktor kualitas eksternal merupakan kategori kualitas yang dapat diobservasi atau menjadi ketertarikan utama dari pelanggan. Kategori-kategori yang termasuk didalam kelompok ini antara lain :

a. Ketepatan (*correctness*),

b. *Robustness*,

c. Unjuk Kerja (*performance*),

d. Ketersediaan dan kualitas antar muka (*interface*),

e. Kehandalan (*reliability*), dan

f. Ketersediaan (*availability*)

Sedangkan kualitas faktor internal merupakan kategori kualitas yang dapat diobservasi atau menjadi ketertarikan utama dari pengembang. Seprerti :

a. Kemudahan membaca/memahami struktur perangkat lunak (*readibility*),

b. Kemampuan untuk dilakukan pengujian (*testability*),

c. Ketersediaan dan kualitas dokumentasi (*documentation*),

d. Kemudahan pemeliharaan (*maintainability*), dan

e. Adaptasi terhadap lingkungan berbeda (*portability*)

2.6.3 Desain

Desain adalah penggambaran, perencanaan dan pembuatan sketsa atau pengaturan dari beberapa elemen yang terpisah ke dalam satu kesatuan yang utuh dan berfungsi (Burch, 1986).

Analis sistem dapat mendesain model dari sistem informasi yang diusulkan dalam bentuk *physical system* dan *logical model*.Bagan alir sistem (*system flowchart*) merupakan alat yang tepat digunakan untuk menggambarkan *physical system*.

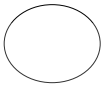
Logical model dari sistem informasi lebih menjelaskan kepada *user* bagaimana nantinya fungsi-fungsi di sistem informasi secara logika akan bekerja.

Logical model dapat digambarkan dengan menggunakan diagram arus data (*data flow diagram*) (Burch, 1986).

1. *Flowchart*

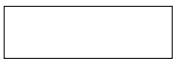
a. Flow Direction Symbol

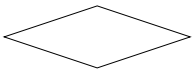
Tabel 2.1 *Flow Direction Symbols*

	Simbol arus / *flow*, yaitu menyatakan jalannya arus suatu proses
	Simbol *connector*, berfungsi menyatakan sambungan dari proses ke proses lainnya dalam halaman yang sama.
	Simbol *off-page connector*, menyatakan sambungan dari proses ke proses lainnya dalam halaman yang berbeda.

b. Processing Symbols

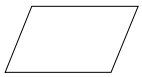
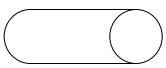
Tabel 2.2 *Processing Symbols*

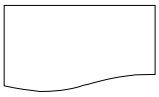
	Simbol *process*, yaitu menyatakan suatu tindakan (proses) yang dilakukan oleh computer
	Simbol *manual*, yaitu menyatakan suatu tindakan (proses) yang tidak dilakukan oleh computer

	Simbol *decision*, yaitu menunjukkan suatu kondisi tertentu yang akan menghasikan dua kemungkinan jawaban : ya / tidak.
	Simbol *preparation*, yaitu menyatakan penyediaan tempat penyimpanan suatu pengolahan untuk memberi harga awal
	Simbol *terminal*, yaitu menyatakan permulaan atau akhir suatu program.
	Simol *offline-storage*, menunjukkan bahwa data dalam simbol ini akan disimpan ke suatu media tertentu
	Simbol *manual-input*, memasukkan data secara manual dengan menggunakan online keyboard

c. Input / Output Symbol

Tabel 2.3 *Input / Output Symbol*

	Simbol *input-output* menyatakan proses *input* atau *output* tanpa tergantung jenis peralatannya
	Simbol *storage* menyatakan input berasal dari disk atau *output* disimpan ke disk

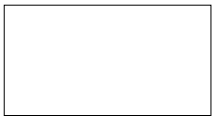
	Simbol *document* mencetak keluaran dalam bentuk dokumn (melalui printer)
	Simbol *display* mencetak keluaran dalam layar monitor.

2. Data Flow Diagram

Data Flow Diagram (DFD) adalah diagram yang menggunakan notasi-notasi ini untuk menggambarkan arus dari data sistem, sekarang di kenal dengan nama diagram arus data (*data flow diagram*). *Data Flow Diagram* (DFD) sering digunakan untuk menggambarkan suatu sistem yang telah ada atau sistem baru yang akan di kembangkan secara logika tanpa mempertibangkan lingkungan fisik dimana data tersebut mengalir.

a. External entity

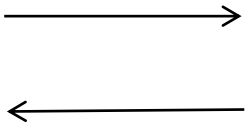
External entity merupakan kesatuan di lingkungan luar sistem yang dapat berupa orang, organisasi, atau sistem lainnya yang berada di lingkungan luarnya yang akan memberikan input atau menerima output dari sistem.



Gambar 2.1 Simbol Eksternal Entity

b. Data flow

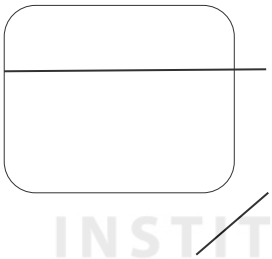
Data flow menunjukkan arus dari data yang berupa masukan untuk sistem atau hasil dari proses sistem dan dapat berbentuk sebagai berikut ini.



Gambar 2.2 Simbol Data flow

c. *Process*

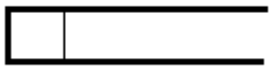
Process adalah kegiatan atau kerja yang dilakukan oleh orang, mesin atau komputer dari hasil suatu arus data yang masuk kedalam proses untuk dihasilkan arus data yang akan keluar dari proses.



Gambar 2.3 Simbol Process

d. *Data Store*

Data store adalah simpanan dari data yang berupa, suatu file database di sistem komputer, arsip atau catatan manual, dan suatu tabel acuan manual.



Gambar 2.4 Simbol Data source

3. *Entity Relationship Diagram*

Atribute adalah kolom di sebuah relasi.

Macam-macam *atribute* yaitu :

a. *Simple Atribute*

Atribute ini merupakan *atribute* yang unik dan tidak dimiliki oleh *atribute* lainnya, misalnya *entity* mahasiswa yang *atribute*-nya NIM.

b. *Composite Atribute*

Composite Atribute adalah *atribute* yang memiliki dua nilai harga, misalnya nama besar (nama keluarga) dan nama kecil (nama asli).

c. *Single Value Atribute*

Atribute yang hanya memiliki satu nilai harga, misalnya *entity* mahasiswa dengan *atribute*-nya umur (tanggal lahir).

d. *Multi Value Atribute*

Atribute yang banyak memiliki nilai harga, misalnya *entity* mahasiswa dengan *atribute*-nya pendidikan (SD, SMP, SMA).

e. *Null Value Atribute*

Atribute yang tidak memiliki nilai harga, misalnya *entity* tukang becak dengan *atribute*-nya pendidikan (tanpa memiliki ijazah).

ERD ini diperlukan agar dapat menggambarkan hubugan antar *entity* dengan jelas, dapat menggambarkan batasan jumlah *entity* dan partisipasi antar *entity*, mudah dimengerti pemakai dan mudah disajikan oleh perancang *database*. (Kadir, 2008)

Untuk itu ERD dibagi menjadi 2 jenis model, yaitu :

a. Conceptual Data Model (CDM)

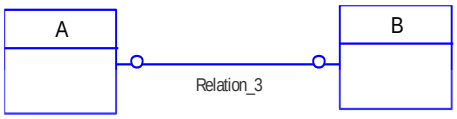
Merupakan jenis model data yang menggambarkan hubungan antar tabel secara konseptual.

b. Physical Data Model (PDM)

Merupakan jenis model data yang menggambarkan hubungan antar tabel secara fisikal.

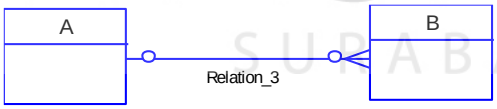
ERD mempunyai 4 jenis hubungan antara lain :

a. Hubungan *one–to–one* (1:1) menyatakan bahwa setiap entitas pada tipe entitas A paling banyak berpasangan dengan satu entitas pada tipe entitas B. Begitu pula sebaliknya. Contoh :



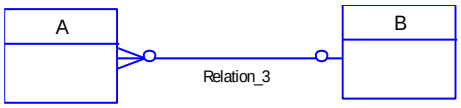
Gambar 2.5 Hubungan *one-to-one*

b. Hubungan *one–to–many* (1:M) menyatakan bahwa setiap entitas pada tipe entitas A bisa berpasangan dengan banyak entitas pada tipe entitas B, sedangkan setiap entitas pada B hanya bisa berpasangan dengan satu entitas pada tipe entitas B. Contoh :



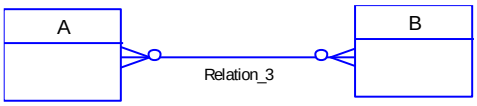
Gambar 2.6 Hubungan *one-to-many*

c. Hubungan *many–to–one* (M:1) menyatakan bahwa setiap entitas pada tipe entitas A paling banyak berpasangan dengan satu entitas pada tipe entitas B dan setiap entitas pada tipe entitas B bisa berpasangan dengan banyak entitas pada tipe entitas A. Contoh :



Gambar 2.7 Hubungan *many-to-one*

d. Hubungan *many–to–many* (M:N) Menyatakan bahwa setiap entitas pada suatu tipe entitas A bisa berpasangan dengan banyak entitas pada tipe entitas B dan begitu pula sebaliknya. Contoh :



Gambar 2.8 Hubungan *many-to-many*

e. Kardinalitas menggambar hubungan antara dua entitas dengan mengindentifikasi berapa banyak *instance* untuk setiap entitas yang nantinya dapat dihubungkan dengan setiap *instance* yang spesifik di entitas yang lain.

2.6.4 *Construction*

Software construction lebih di artikan sebagai pembuatan detail dari suatu pekerjaan, menciptakan satu software yang penting yang di kombinasikan dengan *code*, proses verifikasi, *testing unit*, dan testing yang terintegrasi, serta proses debuging. *Software construction* lebih sering di hubungkan dengan proses desain dan proses testing. Hal ini dikarenakan proses tersebut saling ketergantungan satu sama lain, dimana software construction merupakan keluaran dari desain software dan juga sebagai masukan dari software testing. Software construction bertipikal memproduksi volume konfigurasi item yang lebih tinggi dan juga di butuhkan dalam mengelola sebuah software proyek (file sumber, isi, test cases, dll). (England, 2004)

A. *Software Contsruction Fundamentals*

Pada tahap pertama, dilakukan pendefinisian dasar tetang prinsip-prinsip yang digunakan dalam proses implementasi seperti minimalisasi kompleksitas, mengantisipasi perubahan, dan standar yang digunakan.

B. *Managing Costruction*

Bagian ini mendefeinisikan tentang model implementasi yang digunakan, rencana implementasi, dan ukuran pencapaian dari implementasi tersebut.

C. *Practical Considerations*

Bagian ini membahas tentang desain implementasi yang digunakan, bahasa pemrograman yang digunakan, kualitas dari mplementasi yang dilakukan, proses pengetesan dan integritas.

Dalam proses pengimplementasia ini, digunakan beberapa aplikasi pendukung yaitu :

a. Adobe Dreamweaver

Adobe Dreamweaver merupakan aplikasi yang di gunakan sebagai HTML editor profesional untuk mendesain web secara visual.Yang intinya adalah anda tidak harus berurusan dengan tag-tag HTML untuk membuat sebuah site dan dapat melihat hasil desainnya secara langsung.

Kemampuan Dreamweaver untuk berinteraksi dengan beberapa bahasa pemrograman seperti PHP, ASP, JavaScript, dan yang lainnnya juga memberikan fasilitas maksimal kepada desainer web dengan menyertakan bahasa pemrograman di dalamnya. (Madcoms, 2011)

b. Bahasa Pemrograman PHP

Bahasa Pemrograman PHP adalah bahasa pemrograman yang bekerja dalam sebuah webserver.Script-script PHP harus tersimpan dalam sebuah server dan dieksekusi atau diproses dalam server tersebut. Dengan menggunakan program PHP, sebuah website akan lebih interaktif dan dinamis.(Madcoms, 2011)

c. Database MySQL

Database MySQL adalah jenis database yang sangat populer dan digunakan pada banyak website di internet sebagai bank data, selain itu Database MySQL juga dapat dijalankan dibeberapa platform, antara lain linux, windows, dan sebagainya.(Madcoms, 2011).

2.6.5 *Testing* dan Implementasi

Tahap ini mendemonstrasikan sistem perangkat lunak yang telah selesai dibuat untuk dijalankan, apakah telah sesuai dengan kebutuhan yang telah di spesifikasikan dan dapat diadaptasi pada lingkungan sistem yang baru. Thapan ini tertuang dalam suatu dokumen *Test Plan*, yang dimulai dari membuat *Software Testing fundamentals* yang berisi tentang penjelasan penting mengenai terminology tetsting, kemudian selanjutnya merancang *Test Levels* yang terbagi antara target pengetesan dan objektif dari pengetesan. Pada tahap berikutnya adalah mendefinisikan *Test Techniques*, yaitu tentang bagaiman teknik yang digunakan termasuk dasar-dasar pengetesan berdasarkan intuisi dan pengalaman serta teknik pengetesan secara teknik *coding*, teknik kesalahan, teknik penggunaan, dan teknik terkait lainnya.Tahap selanjutnya adalah mendefinisikan *Test – Related Measures*, yaitu ukuran-ukuran pencapaian testing yang telah dilakukan untuk kemudian dievaluasi kembali. Tahap terakhir adalah

mendefinisikan *test Process* yang berisi tentang aktivitas testing. (England, John Wiley & sons, 2004)

2.6.6 *Maintenance*

Pada tahap ini akan dilakukan pendeskripsian pekerjaan untuk mengoperasikan dan memelihara sistem informasi pada lingkungan pengguna termasuk implementasi akhir dan proses peninjauan kembali. Pemeliharaan sistem ini terdiri dari beberapa jenis yaitu:

a.) *Corrective*, yaitu memperbaiki desain dan *error* pada program.

b.) *Adaptive*, yaitu memodifikasi sistem untuk beradaptasi dengan perubahan lingkungan.

c.) *Perfective*, yaitu elibatkan sistem untuk menyelesaikan masalah baru atau mengambil kesempatan untuk penambahan fitur.

d.) *Preventive*, yaitu menjaga sistem dari kemungkinan masalah di masa yang akan datang.

Prosedur pemeliharaan tersebut disusun dalam beberapa tahapan.Tahap awal adlah menyusun *software maintenance fundamentals* yang berisi tentang dasar-dasar pemeliharaan, segalayang dibutuhkan untuk melakukan pemeliharaan, dan ketgori pemeliharaan. Selanjutnya adalah mendefinisikan *Key Issues in Software Maintenance*, yang berisi tentang teknik pemeliharaan, manajemen pemeliharaan dan biaya, serta ukuran pemeliharaan perangkat lunak. Tahap selanjutnya adalah mendefinisikan proses dan aktivitas pemeliharaan tersebut ke dalam *Maintenance Process*.(England, John Wiley & Sons, 2004).