BAB II

LANDASAN TEORI

2.1 Teori khusus

2.1.1 Manajemen Proyek

Pengertian manajemen proyek menurut (Schawalbe, 2004) manajemen proyek merupakan aplikasi dari ilmu pengetahuan, *skill*, *tools*, dan teknik untuk aktifitas suatu proyek dengan maksud memenuhi atau melampaui kebutuhan *stakeholder* dan harapan dari sebuah proyek.

Menurut (Olson, 2003) manajemen proyek adalah aplikasi sumber daya yang mencakup pengetahuan, peralatan, dan teknik untuk merancang aktivitas proyek dan kebutuhan proyek.

Menurut (Santoso, 2003) manajemen proyek adalah kegiatan merencanakan, mengorganisasikan, mengarahkan, dan mengendalikan sumber daya organisasi perusahaan untuk mencapai tujuan tertentu dalam waktu tertentu dengan sumber daya tertentu. Manajemen proyek mempergunakan personel perusahaan untuk ditempatkan pada tugas tertentu dalam proyek

2.1.2 Data Flow Diagram (DFD)

Data flow diagram adalah *diagram* yang menggunakan notasi untuk mengambarkan arus dari data sistem yang penggunanya sangat membantu untuk

memahami sistem secara logika, terstruktur dan jelas. DFD merupakan alat bantu dalam menggambarkan atau menjelaskan sistem yang sedang berjalan.

Menurut (Jogiyanto, 2005) ada beberapa simbol digunakan pada DFD untuk mewakili :

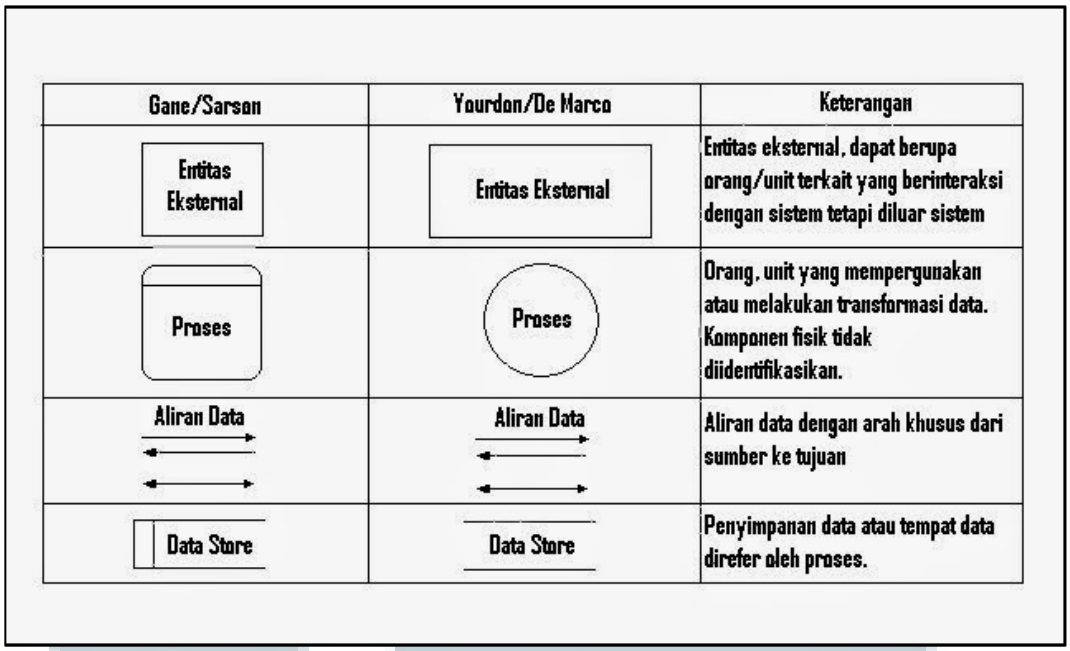
1. Kesatuan Luar (*External Entity*)

   Kesatuan luar (*external entity*) merupakan kesatuan (*entity*) di lingkungan luar sistem yang dapat berupa orang, organisasi, atau sistem lain yang berada pada lingkungan luarnya yang memberikan *input* atau menerima *output* dari sistem.
2. Arus Data (*Data Flow*)

   Arus Data (*data flow*) di DFD diberi simbol suatu panah. Arus data ini mengalir di antara proses, simpan data dan kesatuan luar. Arus data ini menunjukan arus dari data yang dapat berupa masukan untuk sistem atau hasil dari proses sistem.
3. Proses (*Process*)

   Proses (*process*) menunjukan pada bagian yang mengubah *input* menjadi *output*, yaitu menunjukan bagaimana satu atau lebih *input* diubah menjadi beberapa *output*. Setiap proses mempunyai nama, nama dari proses ini menunjukan apa yang dikerjakan proses.
4. Simpanan Data (*Data Store*)

   Data Store merupakan simpanan dari data yang dapat berupa suatu *file* atau *database* pada sistem komputer.



Gambar 2.1Simbol-simbol dalam DFD (Jogiyanto, 2005)

2.1.3 Entity Relationship Modeling

Menurut (Conolly, 2010) *entity-relationship modeling* merupakan permodelan yang berguna untuk memberikan pemahaman yang tepat terhadap data dan penggunaannya di dalam suatu perusahaan. Model ini menggunakan pendekatan *top-down* dalam perancangan basis data yang dimulai dengan mengidentifikasikan data penting yang disebut *entity* dan relasi antar data yang akan direpresentasikan ke dalam model. Kemudian ditambahkan *detail-detail* lebih seperti informasi yang akan dicari mengenai *entities* dan *relationship* yang disebut dengan atribut dan *constraints* pada *entity*, atribut, dan *relationship.*

a. Entity Type

Menurut (Conolly, 2010) *entity type* merupakan sekumpulan objek di dunia yang memiliki *property* yang sama. *Entity* direpresentasikan dalam bentuk diagram berupa persegi panjang berlabel nama dari *entity*.

b. Relationship Types

Menurut (Conolly, 2010) *relationship types* merupakan suatu hubungan antar *entity types. Relationship Types* direpresentasikan dalam bentuk diagram berubah garis lurus yang menghubungkan dua buat *entity types*, ditandai dengan nama dari relasi tersebut. Pada umumnya, relasi dinamai dengan kata kerja.

Menurut (Conolly, 2010) dalam *relationship types* terdapat *degree of relationship type.Degree of relationship type* merupakan jumlah tipe entitas yang terkait dalam *relationship.* Entitas yang terkait dalam *relationship* disebut dengan *participants.* Jadi, *degree* dari suatu *relationship* menunjukkan banyaknya entitas yang tergabung dalam suatu *relationship.* Terdapat 3 jenis *degree of relationship*, yaitu:

1. *Binary Relationship*

   Binary Relationship merupakan *relationship* yang mempunyai dua *degree.*
2. *Ternary Relationship*

   Ternary Relationship merupakan *relationship* yang mempunyai tiga *degree.*
3. *Quarternary Relationship*

   Quarternary Relationship merupakan *relationship* yang mempunyai empat *degree.*

c. Attributes

Menurut (Conolly, 2010) *attribute* adalah *property* suatu entitas atau jenis relasi. Atribut domain adalah himpunan nilai yang diperbolehkan untuk satu atau lebih atribut. Atribut dapat diklasifikasikan menjadi lima, yaitu:

1) *Simple Attribute*

   Simple attribute adalah sebuah atribut yang terdiri dari komponen tunggal yang mempunyai keberadaan bebas dan tidak dapat dibagi menjadi bagian yang lebih kecil dikenal dengan nama *atomic attribute.*

2) *Composite Attribute*

   Composite attribute adalah atribut yang terdiri dari beberapa komponen, dimana masing-masing komponen mempunyai keberadaan yang bebas.

3) *Single-valued Attribute*

   Single-valued attribute adalah atribut yang mempunyai nilai tunggal untuk setiap kejadian dari tipe *entity.*

4) *Multi-valued Attribute*

   Multi-valued attribute adalah atribut yang mempunyai beberapa nilai untuk setiap kejadian dari tipe *entity.*

5) *Derived Attribute*

   Derived attribute adalah atribut yang memiliki nilai yang dihasilkan dari satu atau sekelompok atribut yang berhubungan, dan tidak harus berasal dari satu entitas.

d. Key

Menurut (Conolly, 2010) *key* adalah sebuah *field* yang digunakan untuk mengidentifikasi satu atribut atau lebih secara unik mengidentifikasi setiap *record.*

Key yang sering digunakan yaitu:

1) *Candidate key,* merupakan kumpulan minimal dari atribut-atribut yang secara unik mengidentifikasikan suatu *entity.*
2) *PrimaryKey* merupakan *candidate key* yang dipilih untuk secara unik mengidentifikasikan suatu *entity.*
3) *Composite Key* merupakan *candidate key* yang terdiri atas dua atau lebih atribut.
4) *Alternate Key* merupakan *candidate key* yang tidak terpilih menjadi *primary key,* atau biasa disebut dengan *secondary key.*
5) *Foreign Key* merupakan sebuah *primary key* pada sebuah entitas yang digunakan pada entitas lainnya untuk mengidentifikasi sebuah *entity.*

e. Structural Constraints

Menurut (Conolly, 2010) memeriksa batasan tipe entitas yang mempunyai kesamaan dalam *relationship. Multiplicity* adalah jumlah atau *range* dari terjadinya yang mungkin dari suatu *entity* yang mungkin berhubungan dengan kejadian tunggal dari jenis entitas yang terkait melalui suatu hubungan tertentu.

Hubungan *Structural Constraints* dibagi menjadi 3 jenis, yaitu:

1) Hubungan *One-to-One* (1:1) adalah hubungan antara entitas yang satu dengan entitas lain mempunyai relasi hubungan satu entitas.

2) Hubungan *One-to-Many* (1:*) adalah hubungan antara entitas pertama yang mempunyai banyak relasi dengan entitas kedua yang mempunyai relasi satu entitas.
3) Hubungan *Many-to-Many* (*:*) adalah hubungan antara entitas pertama yang mempunyai relasi banyak dengan entitas kedua.

Cardinality adalah menjelaskan jumlah maksimum hubungan terjadinya yang mungkin untuk suatu entitas yang berpartisipasi dalam jenis hubungan tertentu.

Participation adalah menentukan semua atau hanya beberapa terjadinya entitas berpartisipasi dalam hubungan.

2.1.4 Flow Chart

Menurut (Jogiyanto, 2001)*Flowchart* merupakan gambaran secara grafik dari langkah-langkah yang merupakan suatu prosedur dari sebuah program. *Flowchart* menolong analis dan *programmer* memecahkan masalah masalah yang ditemukan yang kemudian dibagi ke dalam segmen-segmen lebih kecil.

2.2 Normalisasi

Merupakan serangkaian *test* dari sebuah relasi untuk menentukan relasi tersebut memuaskan atau mengganggu kebutuhan dari sebuah *form* yang diberikan (Conolly, 2002).

2.3 Proses Normalisasi

Teknik normalisasi menyangkut sebuah rangkaian aturan yang digunakan untuk memeriksa sebuah relasi, sehingga sebuah basis data dapat dinomalisai dalam beberapa tahap. Normalisasi bisasanya dijalankan seperti sekumpulan langkah secara bertahap, dimana relasi menjadi semakin kuat bila langka (*degree)* yang dikenakannya semakin tinggi.

2.3.1 Bentuk Normal Kesatu (1NF)

Mengidentifikasi dan membuang atribut yang berulang (*repeting group)* dan memiliki nilai yang lebih dari satu. Suatu hubungan dikatakan normal pertama jika :

a. Setiap dari basis dan kolom berisi atribut yang bernilai tunggal.
b. Kunci premier telah ditentukan
c. Atribut yang memiliki nilai banyak telah di hilangkan.

2.3.2 Bentuk Normal Kedua (2NF)

Suatu relasi memiliki *composite key* sebagai *primary key*-nya mempunyai kemungkinan untuk memiliki *partial functional dependency*, dimana atribut *primary key* merupakan fungsi pada salah satu sebagaian dari *primary key*. Dalam 2NF maka setiap atribut yang merupakan *functional dependency* harus dipisahkan ke relasi atau table yang baru dengan mengikut sertakan dereminannya. Bentuk 2NF diperoleh apabila setiap atribut bukan bagian dari *primary key* suatu tabel.

2.3.3 Bentuk Normal Ketiga (3NF)

Normalisasi 3NF dapar dilakukan bila terdapat kondisi dimana relasi memiliki dua atau lebih *composite key* dimana kandidat *key*-nya saling melengkapi, yang sedikitnya memiliki suatu atribut pada umumnya. Sebuah relasi dikatakan 3NF, jika dan hanya jika setiap *determinant* adalah *candidate key.*

2.3.4 Bentuk Normal Keempat (4NF)

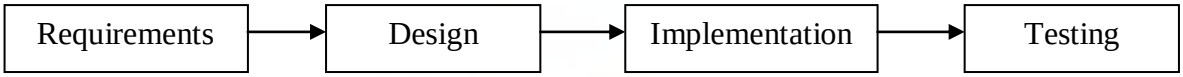
Sebuah relasi dikatakan memiliki *multi-value dependency* apabila dua atau lebih *one to many relationship* saling bebas terdapat pada relasi tersebut. *Multi-Value dependency* dapat dihilangkan dengan memisahkan masing-masing relasi *one to many* menjadi sebuah tabel baru dengan mengikutsertakan *determinant*nya. Bentuk 4NF diperoleh apabila relasi tersebut telah BCNF dan tidak terdapat *multi-value dependency.*

2.3.5 Bentuk Normal Kelima (5NF)

Relasi dikatakan telah 5NF apabila relasi tersebut tidak memiliki *joint dependency*.

2.4 Sistem Development Life Cycle (SDLC)

Menurut (Maryland, 2002) model umum dari SDLC menggambarkan tahapan siklus dan alur proses yang akan dilalui oleh sistem. Model dasar dari siklus pengembangan sistem terlihat pada gambar

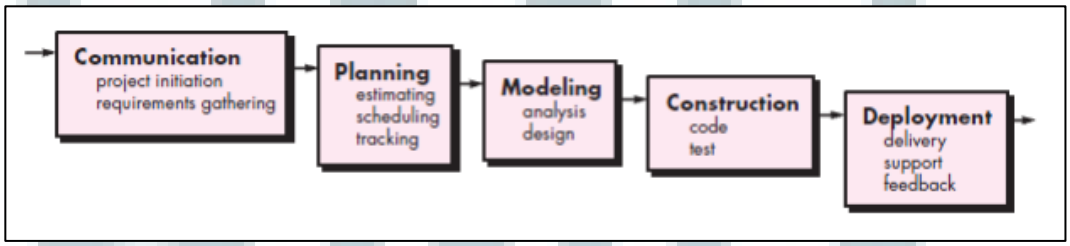


Gambar 2.2 Sistem Development Life Cycle

Berdasarkan gambar-gambar di atas, terlihat bahwa setiap proses menghasilkan penyampaian yang digunakan untuk tahapan selanjutnya. *Requirements* akan diterjemahkan ke dalam bentuk *design*, *design* akan diterjemahkan dalam bentuk kode program yang akan diimplementasikan dan terakhir adalah tahapan *testing* untuk melakukan pengujian dari sistem atau *software* yang dibuat.

2.4.1 Model *Waterfall*

Metode rekayasa peranti lunak yang digunakan peneliti adalah Metode *waterfall*. Menurut (Pressman, 2010) *waterfall* adalah model klasik yang bersifat sistematis, berurutan dalam membangun *software*. Berikut ini ada dua gambaran dari *waterfall* model. Fase-fase dalam model *waterfall* menurut referensi Pressman:



Gambar 2.3Waterfall Model (Pressman)

1. *Communication*

Langkah ini merupakan analisis terhadap kebutuhan *software*, dan tahap untuk mengadakan pengumpulan data dengan melakukan pertemuan dengan *customer*, maupun mengumpulkan data-data tambahan baik yang ada di jurnal, artikel, maupun dari internet.

2. *Planning* Proses

Planning merupakan lanjutan dari proses *communication* (*analysis requirement*). Tahapan ini akan menghasilkan dokumen *user requirement* atau bisa dikatakan sebagai data yang berhubungan dengan keinginan *user* dalam pembuatan *software*, termasuk rencana yang akan dilakukan.

3. *Modeling* Proses

Modeling ini akan menerjemahkan syarat kebutuhan ke sebuah perancangan *software* yang dapat diperkirakan sebelum dibuat *coding*. Proses ini berfokus pada rancangan struktur data, arsitektur *software*, representasi *interface*, dan *detail* (algoritma) prosedural. Tahapan ini akan menghasilkan dokumen yang disebut *software requirement.*

4. *Construction*

Construction merupakan proses membuat kode. *Coding* atau pengkodean merupakan penerjemahan desain dalam bahasa yang bisa dikenali oleh komputer. *Programmer* akan menerjemahkan transaksi yang diminta oleh *user*. Tahapan inilah yang merupakan tahapan secara nyata dalam mengerjakan suatu *software*, artinya penggunaan komputer akan dimaksimalkan dalam tahapan ini. Setelah pengkodean selesai maka akan dilakukan *testing* terhadap sistem yang telah dibuat

tadi. Tujuan *testing* adalah menemukan kesalahan-kesalahan terhadap sistem tersebut untuk kemudian bisa diperbaiki.

5. *Deployment*

Tahapan ini bisa dikatakan *final* dalam pembuatan sebuah *software* atau sistem. Setelah melakukan analisis, desain dan pengkodean maka sistem yang sudah jadi akan digunakan oleh *user*. Kemudian *software* yang telah dibuat harus dilakukan pemeliharaan secara berkala.

Kelebihan sistem *Waterfall* menurut Pressman :

1. Merupakan model pengembangan paling handal dan paling lama digunakan
2. Cocok untuk sistem software yang sudah jelas kebutuhannya dari awal.
3. Pengerjaan *project* sistem akan terjadwal dengan baik, dan mudah dikontrol.

Kekurangan sistem *Waterfall* menurut Pressman :

1. Persyaratan sistem harus digambarkan dengan jelas.
2. Rincian proses harus benar-benar jelas dan tidak boleh berubah rubah.
3. Sulit untuk mengadaptasikan jika terjadi perubahan spesifikasi pada suatu tahapan pengembangan.

2.5 User Acceptance Testing

User Acceptance Testing merupakan pengujian yang dilakukan oleh *end-user* dimana *user* tersebut adalah *staff* karyawan perusahaan yang langsung

berinteraksi dengan sistem dan dilakukan verifikasi apakah fungsi yang ada telah berjalan sesuai dengan kebutuhan/fungsinya (Perry, 2006).

Menurut (Black, 2002) *acceptance testing* biasanya berusaha menunjukkan bahwa sistem telah memenuhi persyaratan-persyaratan tertentu. Pada pengembangan *software* dan *hardware* komersial, *acceptance test* biasanya disebut juga "*alpha tests"* (yang dilakukan oleh pengguna *in-house)* dan "*beta tests"* (yang dilakukan oleh pengguna yang sedang menggunakan atau akan menggunakan sistem tersebut). *Alpha* dan *beta test* biasanya juga menunjukkan bahwa produk sudah siap untuk dijual atau dipasarkan. *Acceptance testing* mencakup data, *environment* dan skenario yang sama atau hampir sama pada saat *live* yang biasanya berfokus pada skenario penggunaan produk tertentu.

Dari definisi di atas, *user acceptance testing* adalah pengujian sistem yang dilakukan oleh pengguna dari sistem tersebut untuk memastikan fungsi-fungsi yang berada pada sistem berjalan dengan baik dan sesuai dengan kebutuhan pengguna.

2.6 User Interface

(Shneiderman, 2010) mengemukakan delapan aturan yang dapat digunakan sebagai petunjuk dasar yang baik untuk merancang suatu *user interface*. Delapan aturan ini disebut dengan *Eight Golden Rules of Interface Design*, yaitu :

1. **Konsistensi**

   Konsistensi dilakukan pada urutan tindakan, perintah, dan istilah yang digunakan pada *prompt*, *menu*, serta layar bantuan.

2. **Memungkinkan pengguna untuk menggunakan *shortcut***

   Ada kebutuhan dari pengguna yang sudah ahli untuk meningkatkan kecepatan interaksi, sehingga diperlukan singkatan, tombol fungsi, perintah tersembunyi, dan fasilitas makro.

3. **Memberikan umpan balik yang informatif**

   Untuk setiap tindakan operator, sebaiknya disertakan suatu sistem umpan balik. Untuk tindakan yang sering dilakukan dan tidak terlalu penting, dapat diberikan umpan balik yang sederhana. Tetapi ketika tindakan merupakan hal yang penting, maka umpan balik sebaiknya lebih substansial. Misalnya muncul suatu suara ketika salah menekan tombol pada waktu *input* data atau muncul pesan kesalahannya.

4. **Merancang dialog untuk menghasilkan suatu penutupan**

   Urutan tindakan sebaiknya diorganisir dalam suatu kelompok dengan bagian pembuka, isi, dan penutup. Umpan balik yang informatif akan memberikan indikasi bahwa cara yang dilakukan sudah benar dan dapat mempersiapkan kelompok tindakan berikutnya.\

5. **Memberikan penanganan kesalahan yang sederhana**

   Sedapat mungkin sistem dirancang sehingga pengguna tidak dapat melakukan kesalahan fatal. Jika kesalahan terjadi, sistem dapat mendeteksi kesalahan dengan cepat dan memberikan mekanisme yang sedehana dan mudah dipahami untuk penanganan kesalahan.

6. **Mudah kembali ke tindakan sebelumnya**

   Hal ini dapat mengurangi kekuatiran pengguna karena pengguna mengetahui kesalahan yang dilakukan dapat dibatalkan, sehingga pengguna tidak takut untuk mengekplorasi pilihan-pilihan lain yang belum biasa digunakan.

7. **Mendukung tempat pengendali internal (*internal locus of control*)**

   Pengguna ingin menjadi pengontrol sistem dan sistem akan merespon tindakan yang dilakukan pengguna daripada pengguna merasa bahwa sistem mengontrol pengguna. Sebaiknya sistem dirancang sedemikan rupa sehingga pengguna menjadi inisiator daripada responden.

8. **Mengurangi beban ingatan jangka pendek**

   Keterbatasan ingatan manusia membutuhkan tampilan yang sederhana atau banyak tampilan halaman yang sebaiknya disatukan, serta diberikan cukup waktu pelatihan untuk kode, mnemonic, dan urutan tindakan

2.7 Microsoft Project

Microsoft Project adalah sebuah aplikasi yang digunakan untuk mengelola proyek, dari proses dalam perusahaan sampai perencanaan budget. Aplikasi ini didesain untuk membantu pengguna dalam mengkolaborasikan sumber daya. Aplikasi ini memiliki kelebihan seperti bisa melakukan chat dengan sesama anggota tim tanpa harus mengganggu pekerjaan sehingga meningkatkan efektivitas kerja (Madcoms, 2013)

\