BAB II

LANDASAN TEORI

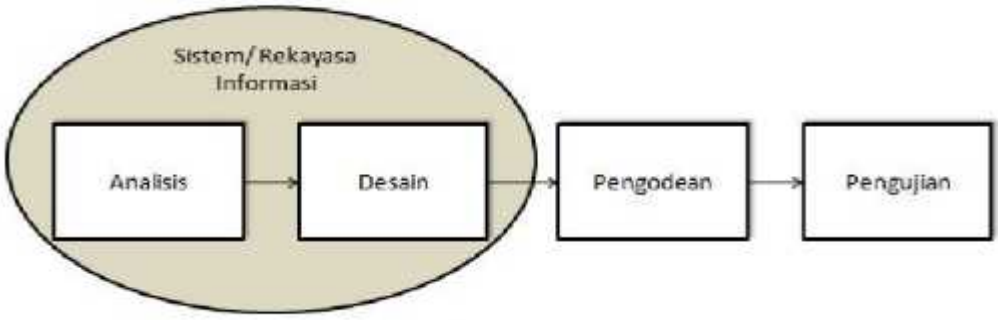
2.1. Tinjauan Pustaka

A. Teori Model *Waterfall*

> Menurut Sukamto dan Shalahuddin (2014:26), SDLC atau *Software Development Life Cycle* atau sering disebut juga *System Development Life Cycle* adalah proses mengembangkan atau mengubah suatu sistem perangkat lunak dengan menggunakan model-model dan metodologi yang digunakan orang untuk mengembangkan sistem-sistem perangkat lunak sebelumnya (berdasarkan *best practice* atau cara-cara yang sudah terjadi baik). Seperti halnya proses metamorfosis pada kupu-kupu yang indah akan dibutuhkan beberapa tahap untuk dilalui, sama halnya dengan membuat perangkat lunak, memiliki dasar tahapan yang dilalui agar menghasilkan perangkat lunak yang berkualitas.

SDLC memiliki beberapa model dalam penerapan tahapan prosesnya. Salah satunya adalah Model *Waterfall*. Menurut Sukamto dan Shalahuddin (2014:28) "model SDLC air terjun (*waterfall*) sering juga disebut model sekuensial linier (*sequential linier*) atau alur hidup klasik (*classic life cycle*)".

Model air terjun menyediakan pendekatan alur hidup perangkat lunak secara sekuensial atau terurut dimulai dari analisis, desain, pengodean, pengujian dan tahap pendukung (*support*). Berikut adalah gambar model air terjun.



Sumber : Sukamto dan Shalahuddin (2014:29)

Gambar II.1.
Ilustrasi Waterfall

1. Analisis kebutuhan perangkat lunak

   Proses pengumpulan kebutuhan dilakukan secara intensif untuk menspesifikasikan kebutuhan perangkat lunak seperti apa yang dibutuhkan oleh *user*. Spesifikasi kebutuhan perangkat lunak pada tahap ini perlu untuk didokumentasikan.

2. Desain

   Desain perangkat lunak adalah proses multi langkah yang fokus pada desain pembuatan program perangkat lunak termasuk struktur data, arsitektur perangkat lunak, representasi antar muka, dan prosedur pengodean. Tahap ini mentranslasi kebutuhan perangkat lunak dari tahap analisis kebutuhan ke representasi desain agar dapat diimplementasikan menjadi program pada tahap selanjutnya. Desain perangkat lunak yang dihasilkan pada tahap ini juga perlu didokumentasikan.

3. Pembuatan kode program

   Desain harus ditransalasi kedalam program perangkat lunak. Hasil dari tahap ini adalah program komputer sesuai dengan desain yang telah dibuat pada tahap desain.

4. Pengujian

   Pengujian fokus pada perangkat lunak secara dari segi lojik dan fungsional dan memastikan bahwa semua bagian sudah diuji. Hal ini dilakukan untuk meminimalisir kesalahan (*error*) dan memastikan keluaran yang dihasilkan sesuai dengan yang diinginkan.

Model *Waterfall* adalah model SDLC yang paling sederhana. Model ini hanya cocok untuk pengembangan perangkat lunak dengan spesifikasi yang tidak berubah-ubah.

B. Teori Metode Inferensi (*Inference Method*)

Menurut Kusrini (2008:8) “Inferensi merupakan proses untuk menghasilkan informasi dari fakta yang diketahui atau diasumsikan. Inferensi adalah konklusi logis (*logical conclusion*) atau implikasi berdasarkan informasi yang tersedia”.

Ada dua metode inferensi yang penting dalam sistem pakar yaitu:

1. Runut Maju (*Forward Chaining*)

Menurut Wilson dalam Kusrini (2008:8) mengemukakan bahwa “menggunakan himpunan aturan kondisi-aksi. Dalam metode ini, data digunakan untuk menentukan aturan mana yang akan dijalankan, kemudian aturan tersebut dijalankan”.

2. Runut Balik (*Backward Chaining*)

Menurut Giarattano dan Riley dalam Kusrini (2008:8) mengemukakan bahwa "Runut balik merupakan metode penalaran kebalikan dari runut maju. Dalam runut balik penalaran dimulai dengan tujuan kemudian balik ke jalur yang akan mengarah ke tujuan tersebut".

C. *Unified Modelling Language*

Menurut Sukamto dan Shalahuddin (2014:133) "UML *(Unifed Modeling Language)* adalah salah standar bahasa yang banyak digunakan di dunia industri untuk mendefinisikan *requirement*, membuat analisis, & desain, serta menggambarkan arsitektur dalam pemrograman berorientasi objek".

UML menyediakan 9 macam diagram untuk memodelkan aplikasi berorientasi objek, yaitu:

1. *Use Case Diagram*

Menurut Sukamto dan Shalahudin (2014:155) "*Use case* atau diagram *use case* merupakan pemodelan untuk kelakuan *(behavior)* sistem informasi yang akan dibuat. *Use case* mendeskripsikan sebuah interaksi antara satu atau lebih aktor dengan sistem informasi yang akan dibuat". Secara kasar, *use case* digunakan untuk mengetahui fungsi apa saja yang ada di dalam sebuah sistem informasi dan siapa saja yang berhak menggunakan fungsi-fungsi itu.

2. *Sequence Diagram*

Diagram sekuen menggambarkan kelakuan objek pada *use case* dengan mendeskripsikan waktu hidup objek dan *message* yang dikirimkan dan diterima antar objek. Oleh karena itu untuk menggambarkan diagram sekuaen maka harus diketahui

objek-objek yang terlibat dalam sebuah *use case* beserta metode-metode yang dimiliki kelas yang dinstansiasi menjadi objek itu. Membuat diagram sekuen juga dibutuhkan untuk melihat skenario yang ada pada *use case*.

3. *Collaboration Diagram*

Collaboration diagram dipakai untuk memodelkan interaksi antar objek di dalam sistem. Berbeda dengan *sequence diagram* yang lebih menonjolkan kronologis dari operasi-operasi yang dilakukan, *collaboration diagram* lebih fokus pada pemahaman atas keseluruhan operasi yang dilakukan oleh *object*.

4. *State Machine Diagram*

State machine diagram atau *statechart diagram* atau dalam bahasa Indonesia disebut diagram mesin status atau sering juga disebut diagram status yang digunakan untuk menggambarkan perubahan status atau transisi status dari sebuah mesin atau sistem atau objek. Jika diagram sekuen digunakan untuk interaksi antar objek maka diagram status digunakan untuk interaksi didalam sebuah objek. Perubahan tersebut digambarkan dalam suatu grafik berarah. *State machine diagram* merupakan pengembangan dari diagram *Finite State Automata* dengan penambahan beberapa fitur dan konsep baru. Diagram *Finite State Automata* (FSA) ini biasanya diajarkan dalam mata kuliah Automata. *State machine* diagram cocok digunakan untuk menggambarkan alur interaksi pengguna dengan sistem.

5. *Activity Diagram*

Diagram aktivitas atau *activity diagram* adalah menggambarkan *workflow* (aliran kerja) atau aktifitas dari sebuah sistem atau proses bisnis atau menu yang ada pada perangkat lunak. Yang perlu diperhatikan disini adalah bahwa diagram aktivitas

menggambarkan aktifitas sistem bukan apa yang dilakuakn oleh aktor. Diagram aktifitas juga banyak digunakan untuk mendefinisikan hal-hal sebagai berikut:

a. Rancangan proses bisnis dimana setiap urutan aktifitas yang digambarkan merupakan proses bisnis sistem yang didefinisikan.
b. Urutan atau pengelompokan tampilan dari sitem atau *user interface* dimana setiap aktifitas dianggap memiliki sebuah rancangan antarmuka tampilan.
c. Rancangan pengujian dimana setiap aktifitas dianggap memerlukan sebuah pengujian yang perlu didefinisikan kasus ujinya.
d. Rancangan menu yang ditampilkan pada perangkat lunak.

6. *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.

Atribut merupakan variabel-variabel yang dimiliki oleh suatu kelas. Metode atau operasi adalah fungsi-fungsi yang dimiliki oleh suatu kelas. Diagram kelas dibuat agar pembuat program atau *programmer* membuat kelas-kelas sesuai rancangan didalam diagram kelas agar antara dokumentasi perancangan dan perangkat lunak *sinkron*. Banyak berbagai kasus, perancangan kelas yang dibuat tidak sesuai dengan kelas-kelas yang dibuat pada perangkat lunak, sehingga tidaklah ada gunanya lagi sebuah perancangan karena apa yang dirancang dan hasilnya tidak sesuai.

7. *Object Diagram*

Diagram objek menggambarkan struktur sistem dari segi penamaan objek dan jalannya objek dalam sistem. Pada diagram objek harus dipastikan semua kelas yang sudah didefinisikan pada diagram kelas harus dipakai objeknya, karena jika tidak, pendefinisian kelas itu tidak dapat dipertanggungjawabkan. Diagram objek juga berfungsi untuk mendefinisikan contoh nilai atau isi dari atribut tiap kelas.

8. *Component Diagram*

Diagram komponen atau *component* diagram dibuat untuk menunjukan organisasi dan ketergantungan diantara kumpulan komponen dalam sebuah sistem.

9. *Deployment Diagram*

Diagram *Deployment* atau *deployment diagram* menunjukan konfigurasi komponen dalam proses eksekusi aplikasi. Diagram *deployment* juga dapat digunakan untuk memodelkan hal-hal sebagai berikut:

a. Sistem tambahan *(embedded system)* yang menggambarkan rancangan *device*, *node*, dan *hardware*nya.

b. Sistem terdistribusi murni.

c. Rekayasa ulang aplikasi.

D. *Entity Relationship Diagram*

Menurut Robi (2016:32) “ERD adalah Diagram untuk menggambarkan desain konseptual suatu basis data rasional. ERD juga merupakan gambaran yang merelasi antara objek yang satu dengan objek yang lain, dari objek dunia nyata yang sering dikenal dengan hubungan antar entitas”.

Menurut Robi (2016:32) ERD mempunyai 3 komponen utama, yaitu:

1. Entitas (*Entity*)

   Entitas adalah suatu objek di dunia nyata yang dapat dibedakan dengan objek lainnya. Objek tersebut dapat berupa orang, benda ataupun hal lainnya.

2. Atribut (*Attribute*)

   Atribut merupakan semua informasi yang berkaitan dengan entitas. Atribut sering dikenal sebagai properti dari suatu entitas atau objek.

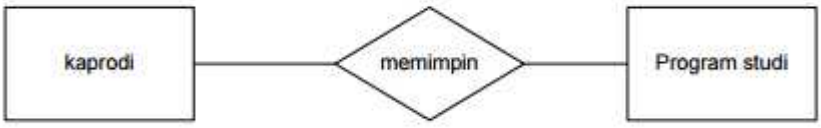
3. Relasi

   Gambar belah ketupat merupakan perlambangan relasi antar identitas atau sering disebut kerelasian.

Menurut Robi (2016:38) ada 3 macam relasi menurut derajatnya, yaitu:

1. Sejenis (*Binary*)

   Relasi yang menghubungkan entitas tidak sejenis.

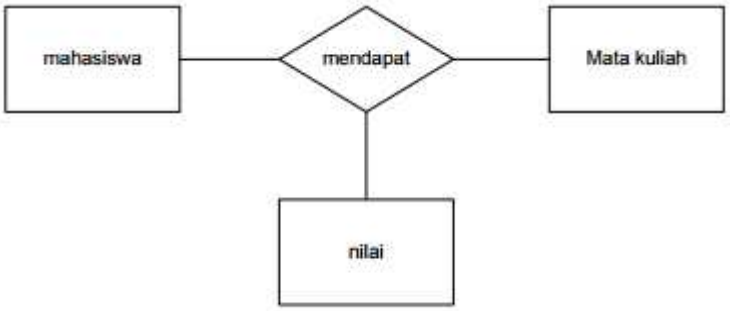


Sumber: Robi (2016:39)

Gambar II.2.
Derajat Relasi *Binary*

2. *Ternaty*

   Relasi yang menghubungkan lebih dari dua entitas yang tidak sejenis.

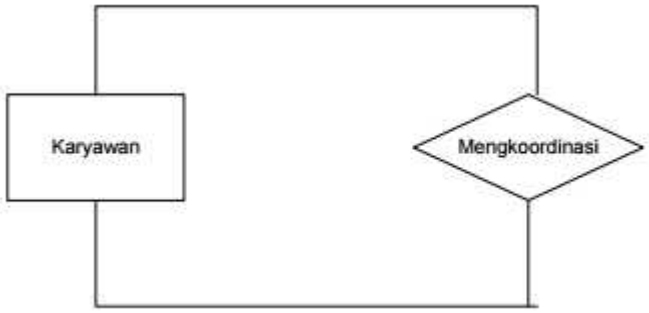


Sumber: Robi (2016:39)

Gambar II.3.
Derajat Relasi *Ternaty*

3. *Unary*

Entitas yang bekerja sama dengan entitas itu sendiri.



Sumber: Robi (2016:39)

Gambar II.4.
Derajat Relasi *Unary*

2.2. Penelitian Terkait

Menurut Saryoko dan Anggraheni Putri (2016:9) Penyakit kulit adalah jenis penyakit yang sering menginfeksi hewan peliharaan khususnya kucing. Kucing yang terinfeksi penyakit kulit, terkadang tampak baik-baik saja dan tidak terganggu sehingga pemilik kucing tidak mengambil serius tentang kesehatan kulit kucing peliharaannya. Namun akan berakibat fatal jika di

biarkan secara terus-menerus, karena dapat menyebabkan kematian bagi kucing peliharaannya.

Menurut Subroto dalam Paryati (2006) Walaupun kucing yang dipelihara oleh manusia selalu berada didalam rumah, tidak menutup kemungkinan kucing yang dipelihara tersebut terserang penyakit, baik itu penyakit biasa dilihat secara kasat mata maupun penyakit pada organ dalam. Penyebab dari penyakit yang menyerang kucing tersebut bermacam-macam, bisa karena parasit yang biasa menyerang kulit, protozoa, mikroba, dan faktor lainnya.