BAB II

LANDASAN TEORI

2.1. Tinjauan Pustaka

1. Konsep Dasar Sistem Informasi

Menurut Japerson (2014:2) Sistem adalah "suatu jaringan kerja dari prosedur - prosedur yang saling berhubungan, terkumpul bersama-sama untuk melakukan suatu kegaiatan atau sasaran yang tertentu".

Sedangkan pendekatan yang merupakan jaringan kerja dari prosedur lebih menekankan urutan-urutan operasi di dalam sistem. Menurut Richard F. Neuschel dalam Japerson (2014:3) bahwa "Suatu prosedur adalah suatu urutan operasi klerikal (tulis-menulis), yang melibatkan beberapa orang didalam satu atau lebih departemen, yang diterapkan untuk menjamin penanganan yang seragam dari transaksi-transaksi bisnis yang terjadi.

Menurut Japerson (2014:9) mengatakan "Informasi adalah data yang diolah menjadi bentuk yang lebih berguna dan lebih berarti bagi yang penerimanya". Sedangkan menurut Gordon B. Davis dalam Japerson (2014:9) Informasi adalah "Data yang telah diolah menjadi suatu bentuk yang penting bagi si penerima dan mempunyai nilai nyata atau yang dapat dirasakan dalam keputusan-keputusan yang sekarang atau keputusan-keputusan yang akan datang.

Menurut Azahra sutanto dalam Puspitawati dkk. (2011:15) mengemukakan bahwa "sistem informasi merupakan komponen-komponen dari subsistem yang saling berhubungan dan bekerja sama secara harmonis untuk mencapai satu tujuan yaitu mengolah data menjadi informasi.".

Menurut Burch dan Grudnitski dalam Puspitawati dkk. (2011:20) mengemukakan bahwa sistem informasi terdiri dari komponen-komponen yang disebutnya dengan istilah blok bangunan (*building block*), yaitu:

1. Blok Masukan (*Input Block*)

   Input yang mewakili data yang masuk ke dalam sistem informasi. Input disini dapat termasuk metode-metode dan media untuk menangkap data yang akan dimasukkan, yang dapat berupa dokumen-dokumen dasar.

2. Blok Model (*Model Block*)

   Terdiri dari kombinasi prosedur, logika dan model matematik yang akan memanipulasi data input dan data yang tersimpan dalam basis data dengan cara yang sudah tertentu untuk menghasilkan keluaran yang diinginkan.

3. Blok Keluaran (*Output Block*)

   Produk dari sistem informasi adalah keluaran yang merupakan informasi yang berkualitas dan dokumentasi yang berguna untuk semua tingkatan manajemen serta semua pemakai sistem.

4. Blok Teknologi (*Technology Block*)

   Teknologi merupakan “kotak alat” dalam sistem informasi yang digunakan untuk menerima input, menjadikan model, menyimpan dan mengakses data, menghasilkan dan mengirim keluaran serta membantu pengendalian dari sistem secara keseluruhan. Teknologi terdiri dari 3 bagian utama, yaitu teknisi (*humanware* atau *brainware*), perangkat lunak (*software*) dan perangkat keras (*hardware*).

5. Blok Basis Data (*Database Block*)

   Merupakan kumpulan dari data yang saling berhubungan satu dengan yang lainnya, tersimpan diperangkat keras komputer dan digunakan perangkat lunak untuk memanipulasinya.

6. Blok Kendali (*Control Block*)

   Beberapa pengendalian perlu dirancang untuk mencegah kerusakan, kegagalan sistem yang mungkin terjadi.

2. Konsep Dasar Model Pengembangan Sistem

Menurut Rosa dan M. Shalahuddin (2013:28) menjelaskan bahwa "model SDLC air terjun (*waterfall*) sering juga disebut model sekuensial linier (*sequential linear*) atau alur hidup klasik (*classic life cycle*)". Ada 5 tahapan dalam metode ini antara lain: analisis, desain, pembuatan kode program, pengujian, dan pemeliharaan.

Berikut adalah tahapan dalam pengembangan perangkat lunak dengan metode *waterfall*:

1. Analisis

   Proses pengumpulan kebutuhan dilakukan secara intensif untuk mespesifikasikan kebutuhan perangkat lunak agar dapat dipahami 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 antarmuka, dan prosedur pengkodean. 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

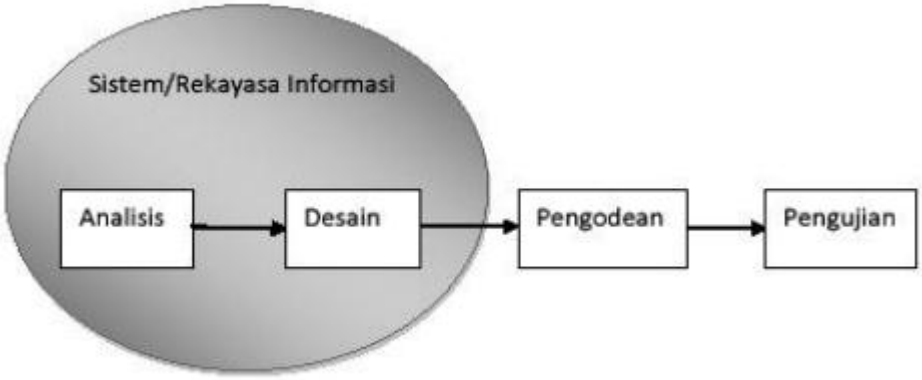
   Desain harus ditranslasikan ke dalam program perangkat lunak. Hasil dari tahap ini adalah program komputer sesuai dengan desain yang telah dibuat pada tahap desain.

4. Pengujian

   Pengujian fokus pada perangkat lunak secara 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.

5. Pemeliharaan (*Maintenance*)

   Tidak menutup kemungkinan sebuah perangkat lunak mengalami perubahan ketika sudah dikirimkan ke *user*. Perubahan bisa terjadi karena adanya kesalahan yang muncul dan tidak terdeteksi saat pengujian atau perangkat lunak harus beradaptasi dengan lingkungan baru. Tahap pendukung atau pemeliharaan dapat mengulangi proses pengembangan mulai dari tahap analisis spesifikasi untuk perubahan perangkat lunak baru.



Sumber: Rosa dan M. Shalahuddin (2013:29)

Gambar II.1.
Model *Waterfall*

Kemunculan model air terjun adalah untuk membantu mengatasi kerumitan yang terjadi dari proyek-proyek pengembangan perangkat lunak, sebuah model air terjun untuk memperinci apa yang seharusnya perangkat lunak lakukan (mengumpulkan dan menentukan kebutuhan sistem) sebelum sistem dikembangkan. Model ini memungkinkan pemecahan misi pengembangan yang rumit menjadi beberapa langkah logis dengan beberapa langkah yang pada akhirnya akan menjadi produk akhir yang siap pakai. (Simarmata, 2010:54).

3. Konsep Dasar Pemrograman

Menurut Sugiyono (2005:14) dijelaskan bahwa "Definisi pemrograman terstruktur adalah merupakan suatu tindakan atau metode untuk mengorganisasikan dan membuat kode-kode program supaya mudah untuk dimengerti, mudah di *test* dan mudah dimodifikasi".

Ada beberapa hal yang menjadi tujuan dilakukan pemrograman terstruktur, yaitu:

1. Meningkatkan kehandalan program
2. Program mudah dibaca dan dapat ditelusuri apabila ada kesalahan-kesalahan
3. Menyederhanakan dari kerumitan program
4. Menyederhakan penulisan program
5. Meningkatkan kualitas dan produktivitas program

Dalam pembuatan pemrograman terstruktur ada beberapa kriteria yang harus dipenuhi sehingga suatu program itu bisa disebut pemrograman terstruktur, diantaranya:

1. Ekspresifitas

Bahasa pemrograman yang baik harus jelas dalam menggambarkan algoritma yang dibuat.

2. Definitas

Bahasa pemrograman dapat didefinisikan dari adanya sintak dan semantik serta bahasa pemrograman yang baik haruslah konsisten dan tidak bermakna ganda

3. Tipe data dan strukturnya

Bahasa pemrograman yang baik harus berkemampuan dalam mendukung berbagai tipe data (*Integer, Real, String, Boolean* dan lain sebagainya) serta struktur data (*Array, Record, File* dan sebagainya).

4. Modularitas

Bahasa pemrograman yang baik harus mempunyai fasilitas subprogram, sehingga suatu program yang besar dapat dikerjakan oleh beberapa pemrogram

secara bersama-sama yang nantinya dengan mudah dapat digabungkan hanya menjadi sebuah modul saja.

5. Adanya *Input-Output*

Bahasa pemrograman yang baik harus dapat mendukung berbagai jenis model file seperti Sequensial, random, index dan sebagainya dalam proses masukan dan keluaran.

6. Portabilitas

Bahasa pemrograman yang dapat digunakan pada berbagai tipe mesin komputer yang berbeda-beda, sehingga dapat menjadi bersifat *machine independent.*

Adanya portabilitas suatu program ditentukan pada ketergantungan program tersebut terhadap mesin komputer atau sistem operasi, maka semakin banyak usaha yang diperlukan untuk memindahkan program tersebut, sehingga dengan adanya ketergantungan pada mesin akan sangat berdampak pada penggunaan piranti masukan dan keluaran, misalnya pada saat mengakses suatu berkas atau file.

7. Efisiensi

Bahasa pemrograman yang dapat mengatur banyaknya instruksi program dalam membatasi waktu tempuh pemrosesan, mengatur besar dan jenis input data yang digunakan serta jumlah memori yang digunakan program.

8. Interaktif

Bahasa pemrograman yang baik harus mudah dipelajari dan diajarkan pada *User* serta mudah dimengerti tentang proses yang sedang dilakukan oleh program.

9. Umum

Bahasa pemrograman yang baik harus memiliki jangkauan yang luas untuk berbagai aplikas pemrograman sehingga dapat bersifat bahasa serbaguna.

Pemrograman terstruktur harus mengikuti aturan dan teknik yang telah diberlakukan. Adanya beberapa cara atau teknik dalam pembuatan pemrograman terstruktur, diantaranya:

1. Pemrograman modular

Pemrograman modular biasanya menggunakan pernyataan *subroutine* yaitu kumpulan perintah yang melakukan tugas terbatas. Misalnya dalam bahasa Pascal dan menggunakan *procedure* dan *function.*

Kriteria program modular:

a. Program dipecah-pecah kedalam modul-modul
b. Setiap modul mempunyai tugas dan fungsi sendiri
c. Setiap modul ditulis secara terpisah dengan modul lainnya, shingga program mudah dicari kesalahannya.
d. Setiap program mempunyai modul program utama, untuk mengontrol semua proses submodul yang terjadi, termasuk mengirimkan kontrol program ke submodul untuk melakukan tugas dan fungsi tertentu.
e. Setiap submodul mengembalikan kontrol program ke modul utama apabila selesai melakukan tugasnya.

2. Pemrograman *top-down*

Mendefinisikan program modul utama yang pertama kali dijalankan, selanjutnya memanggil modul lainnya (submodul) untuk melakukan tugas dan

juga untuk mengakhiri proses program tersebut. Kriteria dari pemrograman *top-down* biasanya berbentuk diagram HIPO *(Hierarchy plus Input-Process-Output).*

4. *Unified Modelling Language* (UML)

Menurut Rosa (2013:13) "UML (*Unified Modeling Language)* adalah salah satu standar bahasa yang banyak digunakan di dunia industri untuk mendefinisikan *requirement,* membuat analisi dan design, serta menggambarkan arsitektur dalam pemrogramman berorientasi objek".

Abstraksi konsep dasar UML terdiri dari *structural classification, dynamic behaviour* dan *model management.*

Menurut Yulianta dan Mursanto (2010:115) "*Unified Modeling Language* (UML) digunakan sebagai notasi utama untuk mengambarkan dan mendokumentasikan sistem yang dibangun".

1. *Usecase* Diagram

Menurut Rosa (2013:155) "*Usecase* merupakan pemodelan untuk tingkah laku (*behavior*) sistem informasi yang akan dibuat". *Usecase* mendeskripsikan sebuah interaksi antara satu atau lebih aktor dengan sistem informasi yang akan dibuat. Secara kasar *usecase* digunakan untuk mengetahui fungsi apa saja yang ada di dalam sebuah sistem informasi dan siapa saja yang berhak menggunakan fungsi-fungsi itu.

2. *Activity* Diagram

Menurut Rosa (2013:161) "*Activity Diagram* menggambarkan *workflow* (aliran kerja) atau aktifitas dari sebuah sistem atau proses bisnis atau menu yang ada pada perangkat lunak".

Activity diagram mempunyai peran seperti halnya *flowchart,* akan tetapi perbedaan dengan *flowchart* adalah *activity diagram* bisa mendukung perilaku paralel sedangkan *flowchart* tidak bisa. *Activity* diagram merupakan *state diagram* khusus, dimana sebagian besar *state* sebelumnya (*internal processing).* Oleh karena itu *activity diagram* tidak menggambarkan *behavior internal* sebuah sistem dan interaksi antar subsistem secara eksak, tetapi lebih menggambarkan proses-proses dan jalur-jalur aktivitas dari *level* atas secara umum.

3. *Component* Diagram

Menurut Rosa (2013:148) "*Component* diagram dibuat untuk menunjukan organisasi dan ketergantungan diantara kumpulan komponen dalam sebuah sistem". Sebuah komponen bisa mengakses *service* tersebut disebut *export interface* sedangkan yang mengaksesnya disebut *import interface. Component Diagram* menggambarkan struktur dan hubungan antar komponen piranti lunak, termasuk ketergantungan (*dependency)* diantara komponen.

Komponen piranti lunak adalah model berisi kode, baik berisi *source code* maupun *binary code,* baik *library* maupun *executable,* baik yang muncul pada *compile time, link time,* maupun *run time.* Umumnya komponen terbentuk dari beberapa *class* dan atau *package*, tapi dapat juga dari komponen-komponen yang lebih kecil. Komponen dapat juga berupa *interface,* yaitu kumpulan layanan yang disediakan sebuah komponen untuk komponen lain.

4. *Deployment* Diagram

Menurut Rosa (2013:154) "*Deployment* diagram menunjukkan konfigurasi komponen dalam proses eksekusi aplikasi".

Deployment Diagram juga dapat digunakan untuk memodelkan hal-hal berikut:

a. Sistem tambahan (*embedded system)* yang menggambarkan rancangan *device node* dan *hardware.*
b. Sistem *client/server.*
c. Sistem terdistribusi.
d. Rekayasa ulang aplikasi.

Bagian *hardware* adalah *node* yaitu nama semua jenis sumber komputasi. Ada dua tipe note yaitu *processor* dan *device. Processor* adalah *node* yang bisa mengeksekusi sebuah komponen sedangkan *device* tidak. *Device* adalah perangkat keras seperti *printer*, *monitor* dan perangkat keras lainnya.

5. *Entity Relation Diagram* (ERD)

Entity Relation Diagram (ERD) atau yang dikenal juga dengan *Diagram Entity-Relationship* (Diagram E-R) merupakan suatu model yang menjelaskan hubungan antar data dalam basis data berdasarkan objek-objek dasar data yang mempunyai hubungan antar relasi.

> Menurut Fatansyah (2007:79) mengemukakan bahwa *Model Entity-Relationship* yang berisi komponen-komponen himpunan entitas dan himpunan relasi yang masing-masing dilengkapi dengan atribut-atribut yang mempresentasikan seluruh fakta dari dunia nyata yang kita tinjau, dapat digambarkan dengan lebih sistematis dengan menggunakan *Diagram Entity-Relationship*.

"Entity Relation Diagram adalah alat pemodelan data utama dan akan membantu mengorganisasi data dalam suatu proyek ke dalam entitas-entitas dan menentukan hubungan antar entitas" (Simarmata dan Janner, 2010:67).

Notasi-notasi simbolik di dalam *Entity Relation Diagram* yang dapat digunakan adalah:

1. Entitas

   Digambarkan dengan kotak persegi panjang dan digunakan untuk menunjukkan sekumpulan orang, tempat, objek atau konsep dan sebagainya yang menunjukkan dimana data dicatat atau disimpan.

2. Hubungan atau Relasi

   Digambarkan dengan kotak berbentuk *diamond* atau belah ketupat dengan garis yang menghubungkan ke entitas yang terkait. Maka *relationship* diberi nama dengan kata kerja. Hubungan atau relasi menunjukkan abstraksi dari sekumpulan hubungan yang mengaitkan antara entitas yang berbeda.

3. Atribut

   Digambarkan dengan bentuk elips. Atribut menunjukkan karakteristik dari tiap entitas atau sesuatu yang menjelaskan entitas atau hubungan. Sehingga atribut dikatakan elemn dari entitas dan relasi. Dari setiap atribut entitas terdapat satu atribut yang dijadikan sebagai kunci (*key*). Beberapa jeni kunci tersebut antara lain : *Primary key, Candidate key, Composite key, Secondary key, Alternate key* dan *Foreign key*.

4. Tingkat Hubungan (*Cardinality*)

   Entity Relation Diagram (ERD) juga menunjukkan tingkat hubungan yang terjadi, dilihat dari segi kejadian atau banyak tiidaknya hubungan antara entitas tersebut, diantaranya ada tiga kemungkinan hubungan yang ada, yaitu:

a. Satu ke satu (*one to one* (1:1))

Tingkat hubungan dinyatakan satu ke satu jika suatu kejadian pada entitas pertama hanya mempunyai satu hubungan dengan satu kejadian maupun satu hubungan dengan satu kejadian pada entitas kedua. Demikian juga sebaliknya, satu kejadian pada entitas yang kedua hanya bisa mempunyai satu hubungann dengan satu kejadian pada entitas yang pertama.

b. Satu ke banyak (*one to many* (1:M))

Tingkat hubungan satu ke banyak (1:M) adalah sama dengan banyak ke satu (M:1), tergantung dari arah mana hubungan-hubungan tersebut dilihat. Untuk satu kejadian pada entitas yang pertama dapat mempunyai banyak hubungan dengan kejadian yang kedua, sebaliknya banyak kejadian pada entitas yang kedua hanya bisa mempunyai satu hubungan dengan satu kejadian pada entitas yang pertama.

c. Banyak ke banyak (*many to many* (M:N))

Tingkat hubungan banyak ke banyak terjadi jika tiap kejadian pada sebuah entitas akan mempunyai hubungan dengan kejadian pada entitas lainnya, baik dilihat dari entitas yang pertama maupun dilihat dari entitas yang kedua.

6. *Logical Relationals Structures* (LRS) atau Model Relasional

Menurut Kusrini (2007:18), "Model relasional adalah kumpulan tabel-tabel untuk merepresentasikan data dan relasi antar data – data tersebut". Setiap tabel terdiri atas kolom-kolom dan setiap kolom mempunyai nama yang unik.

Logical Relationals Structures (LRS) dibentuk dengan nomor tipe *record*. Beberapa tipe record digambarkan oleh kotak empat persegi panjang dan dengan nama yang unik.

2.2. Penelitian Terkait

Dalam proses pengolahan data dan penyajian menu perlu digunakannya suatu alat bantu dalam hal ini berupa aplikasi komputerisasi yang dapat mempermudah dan mempercepat sistem informasi pemesanan menu pada CAKI CAKE Karawang. Berikut beberapa tinjauan penelitian terdahulu (jurnal) yang dapat memperkuat alasan dibuatnya sistem informasi pemesanan menu hidangan pada CAKI CAKE Karawang, diantaranya:

> Menurut Christanto dkk. (2012:39) dalam jurnalnya bahwa:
> Pada era modernisasi, teknologi informasi memegang peranan yang sangat penting dalam kemajuan suatu foodcourt. Untuk memacu kemajuan tersebut teknologi informasi dapat dimanfaatkan guna mendapatkan suatu informasi yang cepat, tepat dan akurat. Oleh karena itu dibutuhkan suatu perangkat untuk meningkatkan suatu efisiensi dan produktivitas, sehubungan dengan hal tersebut maka pemakaian handphone tidak dapat dihindarkan lagi, berkaitan dengan keakuratan dan kecepatan pengolahan data sehingga dapat dihasilkan sistem informasi yang bermanfaat bagi foodcourt maupun pelanggan.

> Menurut Jevri Setia Nugroho dkk. (2014:127) dalam jurnalnya bahwa :
> Dengan memanfaatkan teknologi smartphone atau tablet yang saat ini sedang menjadi trend teknologi, pelaksanaan pemesanan konsumen di restoran dapat menjadi lebih teratur dan lebih akurat, selain dapat menghemat kertas karena pemesanan menu dicatat secara digital.