

BAB II

LANDASAN TEORI

2.1 Pengertian Perancangan

Perancangan adalah gambaran perencanaan dan pembuatan sketsa atau pengaturan dari beberapa *wlwmwn* yang terpisah kedalam satu kesatuan yang utuh dan berfungsi perancangan *system* dapat dirancang dalam bentuk bagan alir *system* (*system flowchart*), yang merupakan alat bentuk grafik yang dapat digunakan untuk menunjukkan urutan-urutan proses dari *system* (Syifaun nafiah, 2003 : 2)

2.2 Pengertian Sistem

Sistem adalah sekelompok elemen-elemen yang teintegrasi dengan maksud yang sama untuk mencapai suatu tujuan, ada juga definisi lain yang menyebutkan bahwa sistem adalah suatu kumpulan atau himpunan dari unsur atau *variable-variable* yang saling terorganisir, saling berinteraksi, dan saling tergantung satu sama lain. yang bisa disimpulkan kumpulan dari *variable* yang berhubungan dengan membentuk suatu jaringan untuk mencapai tujuan tertentu.

2.3 Pengertian Informasi

Informasi adalah salah satu jenis sumber daya yang tersedia untuk membantu proses pengambilan keputusan, ada juga definisi lain yang menyebutkan bahwa informasi adalah data yang telah diolah menjadi sebuah bentuk yang berarti bagi penerimanya dan bermanfaat dalam pengambilan keputusan yang akan datang. Terdapat empat elemen yang Menyusun suatu DFD, yaitu:

1. Proses

Aktivitas atau fungsi yang dilakukan untuk alasan tertentu yang spesifik bisa manual atau terkomputerisasi

2. Data *flow*

Satu data tunggal atau kumpulan logis suatu data selalu diawali atau berakhir pada suatu proses.

3. Data Store

Kumpulan data yang disimpan dengan cara tertentu. Data yang mengalir disimpan dalam data store

4. External entity

Orang organisasi, atau sistem yang berada diluar sistem tetapi berinteraksi dengan sistem

2.4 Pengertian Nilai

Nilai ditujukan dalam menentukan pilihan. Selain itu nilai dapat diartikan sebagai patokan normative yang mempengaruhi manusia dalam menentukan pilihannya diantara cara-cara tindakan alternative. Nilai sama dengan sesuatu yang menyenangkan kita, nilai identik dengan apa yang diinginkan, nilai merupakan sarana pelatihan kita.

2.5 Pengertian Rapor

Rapor merupakan salah satu pertanggung jawaban sekolah terhadap masyarakat tentang kemampuan yang telah dimiliki siswa yang berupa sekumpulan hasil penilaian. Rapor adalah berupa buku yang berisi keterangan nilai murid disekolah, yang biasanya dipakai sebagai laporan guru kepada orang tua siswa/wali murid.

2.6 Pengertian Pengolahan Data Nilai Siswa Berbasis Web

Merupakan suatu sistem yang memberikan informasi laporan keaktifan siswa secara online yang berupa laporan nilai serta informasi siswa yang bersangkutan dengan berbasiskan web sehingga membantu kecepatan dan kualitas dalam penyampaian informasi.

2.7 Pengertian Website

Website menurut Gregorius Agung (2000 : 30), merupakan kumpulan halaman *web* yang saling terhubung dan file-filenya saling terkait. *Web* terdiri dari *page* atau halaman, dan kumpulan halaman yang dinamakan *homepage*. Anda bisa temukan *homepage* di posisi teratas. *Homepage* ini memiliki halaman-halaman yang terkait di posisi bawahnya. Pada umumnya, setiap halaman yang ada di bawah *homepage* disebut dengan *child page*, dimana berisi *hyperlink* ke halaman lain pada *web*.

2.8 Pengertian Hypertext Preprocessor (PHP)

PHP sering dipakai para programmer untuk membuat situs web yang bersifat dinamis karena gratis dan berguna dalam merancang aplikasi web. Supono dan Putratama (2016:3) mengemukakan bahwa "*PHP (PHP: Hypertext Preprocessor)* adalah suatu bahasa pemrograman yang digunakan untuk menerjemahkan baris kode program menjadi kode mesin yang dapat dimengerti oleh komputer yang berbasis *server-side* yang dapat ditambahkan ke dalam *HTML*".

Sedangkan, menurut Ahmad Solichin (2016:11) mengemukakan bahwa "*PHP* merupakan salah satu bahasa pemrograman berbasis web yang ditulis oleh dan untuk pengembang web". *PHP* merupakan bahasa (*script*) pemrograman yang sering digunakan pada sisi server sebuah web.

Kumpulan kutipan diatas menerangkan bahwa *hypertext preprocessor (PHP)* merupakan bahasa pemrograman untuk membuat/mengembangkan aplikasi berbasis web dan bersifat *open source* dan ditanamkan ke dalam *script HTML*.

2.9 Pengertian Framework

Framework merupakan perangkat lunak yang mulai menjadi pilihan untuk membuat suatu aplikasi (Andresta Ramadhan, 2008). Kemudahan-kemudahan yang diberikan menarik orang-orang untuk menggunakannya. Hal ini tidak terlepas

dari tingkat efektifitas dan efisiensinya yang lebih baik dalam proses pengembangan suatu perangkat lunak. Sedangkan *framework* menurut (Jhonson, 2009), adalah suatu aplikasi yang dapat digunakan ulang untuk membuat bermacam-macam aplikasi.

2.10 Pengertian CodeIgniter

CodeIgniter merupakan aplikasi *open source* berupa *framework PHP* dengan model *MVC (Model, View, Controller)* untuk membangun aplikasi web dinamis dengan cepat dan mudah. *CodeIgniter* memiliki desain dan struktur file yang sederhana, didukung dengan dokumentasi yang lengkap sehingga *framework* ini lebih mudah dipelajari.

CodeIgniter ini memungkinkan para pengembang untuk menggunakan *framework* secara parsial atau secara keseluruhan. Artinya bahwa *CodeIgniter* masih memberi kebebasan kepada para pengembang untuk menulis bagian-bagian kode tertentu di dalam aplikasi menggunakan cara konvensional atau dengan *syntax* umum didalam *PHP*, tidak harus menggunakan aturan penulisan kode di *CodeIgniter*. (Septian, 2011).

2.11 Pengertian Basis data

Database adalah kumpulan file-file atau tabel-tabel yang saling berelaborasi atau berhubungan dengan yang lain, Relasi tersebut ditunjukkan adanya kunci dari tiap file atau tabel yang ada.

2.12 Maria DB

MariaDB adalah *relational database management system (DBMS)* *open source* yang merupakan pengganti yang *kompatibel*. *MariaDB* adalah pengganti *MySQL*. Dengan kata lain, *MariaDB* merupakan pengganti *MySQL* yang ditingkatkan dan *drop-in*.

Pengganti *drop-in* berarti bahwa Anda dapat mengganti server *MySQL* standar dengan versi analog dari server *MariaDB* dan memanfaatkan sepenuhnya perbaikan *MariaDB* tanpa perlu mengubah kode aplikasi Anda. *MariaDB* mendukung, mendukung, dan kuat, lebih banyak mesin penyimpanan daripada *MySQL*


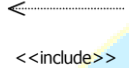


2.13 Pengertian UML


Unified Modeling Language (UML) adalah salah satu standar bahasa yang banyak digunakan di dunia industri untuk mendefinisikan *requirement*, membuat analisis dan desain, serta menggambarkan arsitektur dalam pemrograman berorientasi objek. *UML* merupakan bahasa visual untuk pemodelan dan komunikasi mengenai sebuah sistem dengan menggunakan diagram dan teks-teks pendukung.

UML muncul karena adanya kebutuhan pemodelan visual untuk menspesifikasikan, menggambarkan, membangun, dan dokumentasi dari sistem perangkat lunak. *UML* hanya berfungsi untuk melakukan pemodelan. Jadi penggunaan *UML* tidak terbatas pada metodologi tertentu, meskipun pada kenyataannya *UML* paling banyak digunakan pada metodologi berorientasi objek.

2.14 Usecase Diagram

Rosa dan M. Shalahudin (2014:155), mengemukakan *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. Berikut adalah simbol-simbol yang ada pada diagram *use case* :






NO	GAMBAR	NAMA	KETERANGAN
1		<i>Actor</i>	Seseorang atau sesuatu yang berinteraksi dengan sistem yang sedang kita kembangkan.
2		<i>Dependency</i>	Umumnya penggunaan <i>dependency</i> digunakan untuk menunjukkan operasi pada suatu <i>class</i> yang menggunakan <i>class</i> yang lain.
3		<i>Generalization</i>	Relasi <i>generalization</i> sepadan dengan sebuah relasi <i>inheritance</i> pada konsep berorientasi objek.
4		<i>Include</i>	Relasi cakupan memungkinkan suatu <i>use case</i> untuk menggunakan fungsionalitas yang disediakan oleh <i>use case</i> yang lainnya.
5		<i>Extend</i>	Memungkinkan suatu <i>use case</i> memiliki kemungkinan untuk memperluas fungsional yang disediakan <i>use case</i> yang lainnya.
6		<i>Association</i>	Melambangkan tipe-tipe <i>relationship</i> dan juga dapat menampilkan hukum-hukum multiplisitas pada sebuah <i>relationship</i> (Contoh: <i>One-to-one</i> , <i>one-to-many</i> , <i>many-to-many</i>).
7		<i>System</i>	Menspesifikasikan paket yang menampilkan sistem secara terbatas.
8		<i>Use Case</i>	Peringkat Tertinggi dari fungsional yang dimiliki sistem.
9		<i>Collaboration</i>	Interaksi aturan-aturan dan elemen lain yang bekerja sama untuk menyediakan perilaku yang lebih besar dari jumlah dan elemen-elemennya (sinergi).

10		Note	Elemen fisik yang eksis saat aplikasi dijalankan dan mencerminkan suatu sumber daya komputasi.
----	---	------	--

Tabel 2.1 Simbol *usecase diagram*

2.15 Activity Diagram



Diagram aktivitas atau *activity diagram* menggambarkan *workflow* (aliran kerja) atau aktivitas dari sebuah sistem atau proses bisnis atau menu yang ada pada perangkat lunak. Yang perlu diperhatikan disini adalah bahwa diagram aktivitas menggambarkan aktivitas sistem bukan apa yang dilakukan aktor, jadi aktivitas yang dapat dilakukan oleh sistem. Berikut adalah simbol-simbol yang ada pada diagram aktivitas :

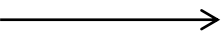

NO	GAMBAR	NAMA	KETERANGAN
1		Activity	memperlihatkan bagaimana masing-masing kelas antarmuka saling berinteraksi satu sama lain.
2		Action	State dari sistem yang mencerminkan eksekusi dari suatu aksi.
3		Initial Node	bagaimana objek dibentuk atau di awali
4		Activity Final Node	bagaimana objek dibentuk dan dihancurkan.
5		Fork Node	suatu aliran yang pada tahap tertentu berubah menjadi beberapa aliran.

Tabel 2.2 Simbol *activity diagram*

2.16 Sequence Diagram

Sequence diagram menggambarkan kelakuan objek pada *use case* dengan mendeskripsikan waktu hidup objek dan *message* yang dikirimkan dan diterima antar objek. Untuk menggambarkan *sequence diagram* harus diketahui objek-objek yang terlibat dalam sebuah *use case* beserta metode-metode yang dimiliki kelas yang diinstansiasi menjadi objek itu, Berikut adalah simbol-simbol yang ada pada *sequence diagram* :



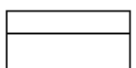


SIMBOL	KETERANGAN
 Aktor nama aktor <u>Nama aktor</u>	orang, proses, atau sistem lain yang berinteraksi dengan sistem informasi yang akan dibuat diluar sistem informasi yang akan dibuat itu sendiri, jadi walaupun simbol dari aktor adalah gambar orang , tapi aktor belum tentu merupakan orang, biasanya dinyatakan menggunakan kata benda di awal fase nama aktor.
garis hidup/lifeline 	menyatakan kehidupan suatu objek.
Objek <u>Nama objek : nama kelas</u>	menyatakan objek yang berinteraksi pesan.
Waktu aktif	menyatakan objek yang dalam keadaan aktif dan berinteraksi pesan.
pesan tipe <i>call</i> 1 : nama_metode() 	menyatakan suatu objek memanggil operasi/ metode yang ada pada objek lain atau dirinya sendiri, 1: nama_metode() arah panah mengarah pada objek yang memiliki operasi/metode, karena ini memanggil operasi/metode yang dipanggil harus ada pada diagram kelas sesuai dengan kelas objek yang berinteraksi.

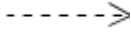

Pesan tipe <i>send</i> 1 : masukan 	menyatakan bahwa suatu objek mengirimkan data/masukan/informasi ke objek lainnya, arah panah mengarah pada objek yang dikirim.
Pesan tipe <i>return</i> 1 : keluaran 	menyatakan bahwa suatu objek yang telah menjalankan suatu operasi atau metode menghasilkan suatu operasi atau metode menghasilkan suatu kembalian ke objek tertentu, arah panah mengarah pada objek yang menerima kembalian.

Tabel 2.3 Simbol *sequence diagram*

2.17 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. Kelas-kelas yang ada pada struktur sistem harus melakukan fungsi-fungsi sesuai dengan kebutuhan sistem, Berikut adalah simbol-simbol yang ada pada *class diagram* :

NO	GAMBAR	NAMA	KETERANGAN
1		<i>Generalization</i>	hubungan dimana objek anak (<i>descendent</i>) berbabagi perilaku dan struktur data dari objek yang ada di atasnya.
2		<i>Nary Association</i>	upaya untuk menghindari asosiasi dengan lebih 2 objek.
3		<i>Class</i>	himpunan dari objek-objek yang berbagi atribut serta operasi yang sama.
4		<i>Colaboration</i>	deskripsi dari urutan aksi-aksi yang ditampilkan sistem yang menghasilkan suatu hasil yang terukur bagi suatu aktor.
5		<i>Realization</i>	operasi yang benar-benar dilakukan oleh suatu objek.

6		<i>Dependency</i>	hubungan dimana perubahan yang terjadi pada suatu elemen mandiri (<i>independent</i>) akan mempengaruhi elemen yang bergantung padanya elemen yang tidak mandiri.
7		<i>Association</i>	apa yang menghubungkan antara objek satu dengan objek lainnya.

Tabel 2.4 Simbol *class diagram*

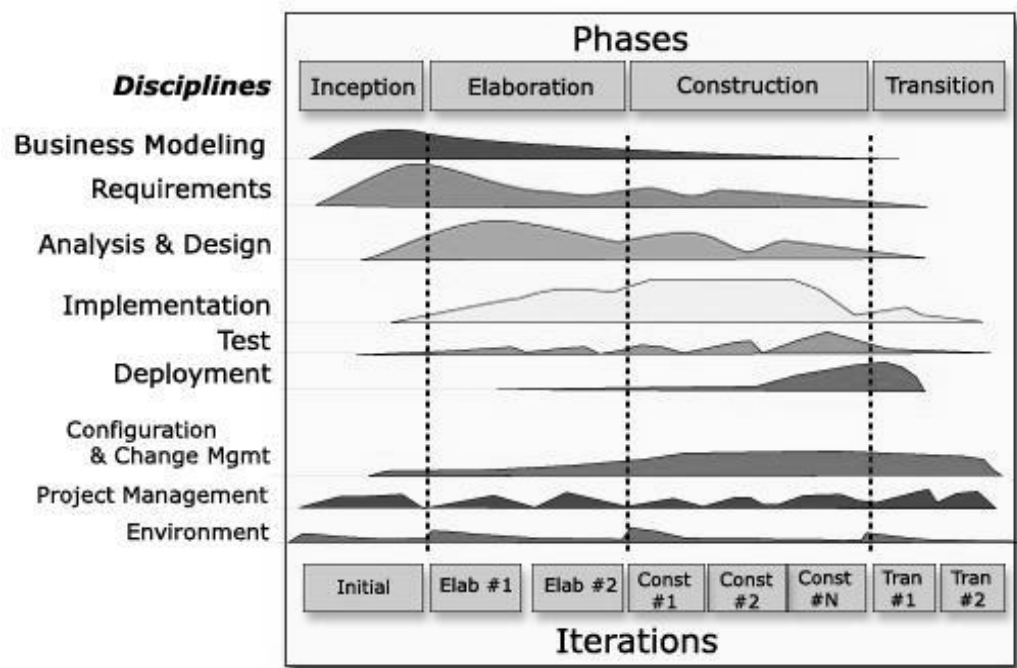
2.18 RUP (*Rational Unified Process*)

Rational Unified Process (RUP) merupakan suatu metode rekayasa perangkat lunak yang dikembangkan dengan mengumpulkan berbagai *best practises* yang terdapat dalam industri pengembangan perangkat lunak. Ciri utama metode ini adalah menggunakan *use-case driven* dan pendekatan interatif untuk siklus pengembangan perangkat lunak. Gambar dibawah menunjukkan secara keseluruhan arsitektur yang dimiliki *RUP*.

RUP menggunakan konsep *object oriented*, dengan aktifitas yang berfokus pada pengembangan model dengan menggunakan *Unified Model Language (UML)*. Melalui gambar dibawah dapat dilihat bahwa *RUP* memiliki, yaitu:

- Dimensi pertama digambarkan secara horizontal. Dimensi ini mewakili aspek-aspek dinamis dari pengembangan perangkat lunak. Aspek ini dijabarkan dalam tahapan pengembangan atau fase. Setiap fase akan memiliki suatu *major milestone* yang menandakan akhir dari awal dari phase selanjutnya. Setiap phase dapat berdiri dari satu beberapa iterasi. Dimensi ini terdiri atas *Inception, Elaboration, Construction, dan Transition*.
- Dimensi kedua digambarkan secara vertikal. Dimensi ini mewakili aspek-aspek statis dari proses pengembangan perangkat lunak yang dikelompokkan ke dalam beberapa disiplin. Proses pengembangan perangkat lunak yang dijelaskan kedalam beberapa disiplin terdiri dari empat elemen penting, yakni *who is doing, what, how* dan *when*. Dimensi ini terdiri atas *Business Modeling*,

Requirement, Analysis and Design, Implementation, Test, Deployment, Configuration dan Change Management, Project Management, Environment.



Gambar 2.1 Arsitektur *Rational Unified Process*

Pada penggunaan kedua standard tersebut diatas yang berorientasi obyek (*object oriented*) memiliki manfaat yakni:

- *Improve productivity*

Standard ini dapat memanfaatkan kembali komponen-komponen yang telah tersedia/dibuat sehingga dapat meningkatkan produktifitas

- *Deliver high quality system*

Kualitas sistem informasi dapat ditingkatkan sebagai sistem yang dibuat pada komponenkomponen yang telah teruji (*well-tested* dan *well-proven*) sehingga dapat mempercepat *delivery* sistem informasi yang dibuat dengan kualitas yang tinggi.

- *Lower maintenance cost*

Standard ini dapat membantu untuk menyakinkan dampak perubahan yang terlokalisasi dan masalah dapat dengan mudah terdeteksi sehingga hasilnya biaya pemeliharaan dapat dioptimalkan atau lebih rendah dengan pengembangan informasi tanpa standard yang jelas.

- *Facilitate reuse*

Standard ini memiliki kemampuan yang mengembangkan komponen-komponen yang dapat digunakan kembali untuk pengembangan aplikasi yang lainnya.

- *Manage complexity*

Standard ini mudah untuk mengatur dan memonitor semua proses dari semua tahapan yang ada sehingga suatu pengembangan sistem informasi yang amat kompleks dapat dilakukan dengan aman dan sesuai dengan harapan semua manajer proyek IT/IS yakni *deliver good quality software within cost and schedule time and the users accepted.*

Fase RUP

- *Inception/insepsi*
- *Elaboration/elaborasi*
- *Construction/konstruksi*
- *Transition/transisi*
- *Inception*
 - Menentukan Ruang lingkup proyek
 - Membuat 'Business Case'
 - Menjawab pertanyaan "apakah yang dikerjakan dapat menciptakan 'good business sense' sehingga proyek dapat dilanjutkan
- *Elaboration*
 - Menganalisa berbagai persyaratan dan resiko
 - Menetapkan 'base line'
 - Merencanakan fase berikutnya yaitu *construction*
- *Construction*
 - Melakukan sederetan iterasi
 - Pada setiap iterasi akan melibatkan proses berikut: analisa desain, implementasi dan testing
- *Transistion*
 - Membuat apa yang sudah dimodelkan menjadi suatu produk jadi

- Dalam fase ini dilakukan:
 - Beta dan performance testing
 - Membuat dokumentasi tambahan seperti; training, *user guides* dan sales kit
 - Membuat rencana peluncuran produk ke komunitas pengguna

Peran *Use Case* Pada *Setiap Fase*

- *Inception*
 - Menolong mengembangkan scope proyek
 - Menolong menetapkan penjadwalan dan anggaran
- *Elaboration*
 - Menolong dalam melakukan analisa resiko
 - Menolong mempersiapkan fase berikutnya yaitu konstruksi
- *Construction*
 - Melakukan sederetan iterasi
 - Pada setiap iterasi akan akan melibatkan proses berikut: analisa desain, implementasi dan testing
- *Transision*
 - Membuat apa yang sudah dimodelkan menjadi suatu produk jadi
 - Dalam fase ini dilakukan:
 - Beta dan performance testing
 - Membuat dokumentasi tambahan seperti; training, *user guides* dan sales kit
 - Membuat rencana peluncuran produk ke komunitas pengguna.