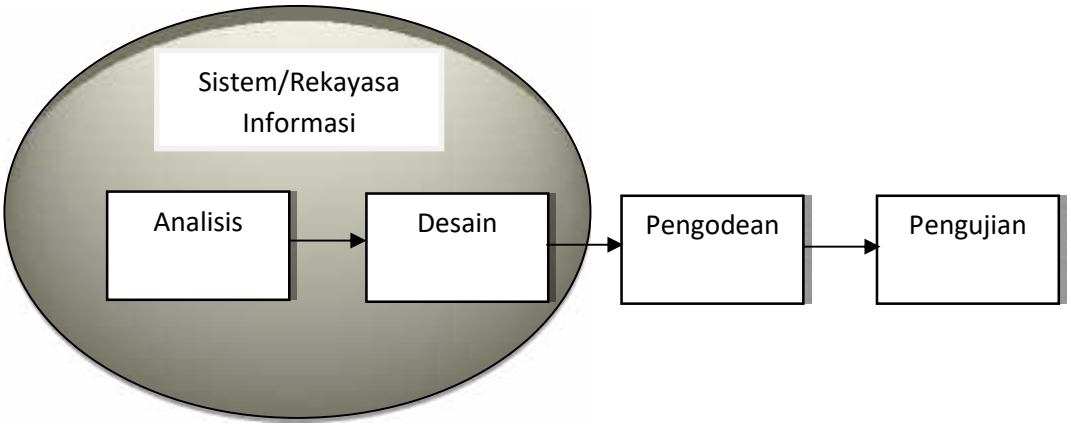
BAB II
LANDASAN TEORI

2.1. Tinjauan Pustaka

2.1.1 Konsep Dasar Pengembangan Sistem

Metode ini memberikan pendekatan-pendekatan sistematis dan berurutan bagi pengembangan sistem informasi. Berikut adalah gambaran dari model *waterfall*.



Sumber : Rosa dan Shalahuddin (2015:29)

Gambar II. I Ilustrasi model w*aterfall*

Penjelasan dari tahap-tahap waterfall model adalah sebagai berikut:

1. Analisis kebutuhan perangkat lunak

   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 nprogram perangkat lunak. Hasil dari tahap ini adalah program komputer sesuai dengan desain yang telah dibuat pada tahap desain.

4. Pengujian *(Testing)*

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.

5. Pendukung (*support)* atau pemeliharaan (*maintenance)*

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

analisis spesifikasi untuk perubahan perangkat lunak yang sudah ada, tapi tidak untuk membuat perangkat lunak baru.

2.1.2. Konsep Dasar Pemrograman Berorientasi Objek

Menurut Rosa dan Shalahuddin (2014 : 100) “metodologi berorientasi objek adalah suatu strategi pembangunan perangkat lunak yang mengorganisasikan perangkat lunak sebagai kumpulan objek yang berisi data dan operasi yang diberlakukan terhadapnya, serta suatu cara bagaimana sistem perangkat lunak dibangun melalui pendekatan objek secara sistematis”.

2.1.2 Konsep Dasar Berorientasi Objek

Menurut Rosa dan Shalahuddin (2014 : 103) “Pendekatan berorientasi objek merupakan suatu teknik atau cara pendekatan dalam melihat permasalahan dan sistem (sistem perangkat lunak, sistem informasi, atau sistem lainnya).”

Berikut ini adalah beberapa konsep dasar yang harus dipahami tentang metodologi berorientasi objek menurut Rosa dan Shalahuddin (2014 : 104) yaitu :

1. Kelas (*class*)

   Menurut Rosa dan Shalahuddin (2014 : 104) “Kelas adalah kumpulan objek-objek dengan karakteristik yang sama, statik dan himpunan objek yang sama yang mungkin lahir atau diciptakan dan kelas tersebut.”

2. Objek (*object*)

   Menurut Rosa dan Shalahuddin (2014 : 106) “Objek adalah abstraksi dan sesuatu yang mewakili dunia nyata seperti benda, manusia, satuan organisasi, tempat, kejadian, struktur, status, atau hal-hal lain yang bersifat abstrak.”

Menurut Rosa dan Shalahuddin (2014 : 106) “ Objek merupakan suatu entitas yang mampu menyimpan informasi (status) dan mempunyai operasi (kelakuan) yang dapat diterapkan atau dapat berpengaruh pada status objeknya.”

3. Metode (*method*)

   Menurut Rosa dan Shalahuddin (2014 : 107) “ Operasi atau metode merupakan fungsi atau transformasi yang dapat dilakukan terhadap objek atau dilakukan oleh objek.”

4. Atribut (*attribute*)

   Menurut Rosa dan Shalahuddin (2014 : 108) “ Atribut dari sebuah kelas adalah variable global yang dimiliki sebuah kelas.”

5. Enkapsulasi (*encapsulation*)

   Pembungkusan atribut data dan layanan (operasi-operasi) yang dipunyai onjek untuk menyembunyikan implementasi dan objek sehingga objek lain tidak mengetahui cara kerjanya (Rosa dan Shalahuddin, 2014 : 108).

6. Pewarisan (*inheritance*)

   Mekanisme yang mungkin satu objek mewarisi sebagian atau seluruh definisi dan objek lain sebagai bagian dan dirinya (Rosa dan Shalahuddin, 2014 : 109).

7. Antarmuka (*interface*)

   Antarmuka atau *interface* biasanya digunakan agar kelas yang lain tidak mengakses langsung ke suatu kelas, mengakses antarmukanya (Rosa dan Shalahuddin, 2014 : 109).

8. Polimorfisme (*polymorphism*)

   Menurut Rosa dan Shalahuddin (2014 : 110) “Kemampuan suatu objek untuk digunakan di banyak tujuan yang berbeda dengan nama yang sama sehingga menghemat baris program.”

9. *Package*

   Rosa dan Shalahuddin (2014 : 110) “*Package* adalah container atau kemasan yang dapat digunakan untuk mengelompokkan kelas-kelas sehingga memungkinkan beberapa kelas yang bernama sama disimpan dalam *package* yang berbeda.”

2.1.3. Peralatan Pendukung *(Tools System)*

Dalam kegiatan merancang system baru dibutuhkan suatu peralatan pendukung *(tools system)* yang merupakan alat yang dapat digunakan untuk menggambarkan bentuk *logical model* dari suatu sistem, dimana sumber-sumber, lambing-lambang, dan diagram-diagram secara tepat arti fisiknya.

2.1.4. UML (Unified Modeling Language)

UML (*Unified Modelling Language*) merupakan salah satu bentuk pemodelan atau sebuah bahasa standar yang banyak digunakan di dunia industri untuk mendefinisikan *requirement*, membuat analisis dan desain serta menggambarkan arsitektur dalam pemrograman berorientasi objek (Rosa dan Shalahuddin, 2013:133).

Dengan menggunakan UML kita dapat membuat model untuk semua jenis aplikasi piranti lunak, dimana aplikasi tersebuat dapat berjalan pada piranti keras, sistem operasi dan jaringan apapun, serta ditulis dalam bahasa pemrograman apapun. Tetapi karena UML juga menggunakan classdan operation dalam konsep

dasarnya, maka ia lebih cocok untuk penulisan piranti lunak dalam bahasa-bahasa berorientasi objek seperti C++, Java, C# atau VB.NET.

Seperti bahasa lainnya, *Unified Modeling Language* (UML) mendefinisikan notasi dan syntax/semantic. Notasi UML merupakan sekumpulan bentuk khusus untuk menggambarkan berbagai diagram piranti lunak. Setiap bentuk memiliki makna tertentu dan UML Syntax mendefinisikan bagaimana benuk-bentuk tersebut dapat dikombinasikan. Notasi UML terutama diturunkan dari 3 notasi yang telah ada sebelumnya: Grady Booch OOD (Object-Oriented Design), Jim Rumbaugh OMT (Object Modeling Technique), dan Ivar Jacobson OOSE (Object-Oriented Software Engineering).

UML mendefinisikan diagram-diagram sebagai berikut:

a. Use Case Diagram

Use case diagram merupakan pemodelan untuk menggambarkan kelakuan (*behavior*) dari sebuah sistem informasi yang dibuat dengan cara mendeskripsikan sebuah interaksi antara satu atau lebih aktor dengan sistem informasi. Dengan kata lain *use case diagram* dapat digunakan untuk mengetahui fungsi apa saja yang ada di dalam sebuah sistem informasi dan siapa saja yang berhak untuk dapat menggunakan fungsi-fungsi tersebut (Rosa dan Shalahuddin, 2013:155). Setiap *use case* dilengkapi dengan skenario yang merupakan alur jalannya proses *use case* dari sisi aktor dan sistem.

b. Activity Diagram

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

menggambarkan aktivitas atau kegiatan yang dapat dilakukan oleh sistem bukan apa yang dilakukan aktor (Rosa dan Shalahuddin, 2013:161). *Activity diagram* dibuat berdasarkan sebuah atau beberapa *use case* pada *use case* diagram.

c. Class Diagram

Class diagram bersifat statis. Diagram ini memperlihatkan himpunan kelas-kelas, antarmuka-antarmuka, kolaborasi-kolaborasi, serta relasi-relasi. Diagram ini umum dijumpai pada sistem pemodelan berorientasi objek. Meskipun bersifat statis, sering pula diagram kelas memuat kelas-kelas aktif.

d. Sequence Diagram

Sequence Diagram bersifat dinamis. Diagram urutan adalah diagram iteraksi yang menekankan pada pengiriman pesan salam suatu waktu tertentu.

e. Component Diagram

Diagram komponen merupakan salah satu macam UML yang dibuat untuk menunjukkan organisasi dan ketergantungan diantara kumpulan komponen dalam sebuah sistem. Diagram ini fokus pada komponen sistem yang dibutuhkan dan ada di dalam sistem. Diagram ini juga dapat digunakan untuk memodelkan *source code* program perangkat lunak (Rosa dan Shalahuddin, 2013:148).

Berikut penjelasan simbol-simbol yang ada pada *component diagram* menurut Rosa dan Shalahuddin (2013:149).

1. *Package*

   Package merupakan sebuah bungkusan dari satu atau lebih komponen.

2. Komponen

   Merupakan kumpulan komponen yang membentuk suatu sistem.

3. Kebergantungan (*dependency*)

   Menggambarkan kebergantungan antar komponen dan biasanya arah panah mengarah pada komponen yang dipakai.

4. Antarmuka (*interface*)

   Berfungsi sebagai antarmuka agar tidak mengakses komponen secara langsung.

5. *Link*

   Menggambarkan relasi antar komponen.

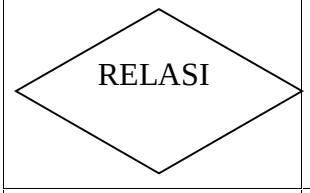
f. Deployment Diagram

Deployment atau *physical diagram* menggambarkan detail bagaimana komponen di-deploy dalam infrastruktur sistem, dimana komponen akan terletak (pada mesin, server atau piranti keras apa), bagaimana kemampuan jaringan pada lokasi tersebut, spesifikasi server, dan hal-hal lain yang bersifat fisikal. Sebuah node adalah server, workstation, atau piranti keras lain yang digunakan untuk men-deploy komponen dalam lingkungan sebenarnya. Hubungan antara node (misalnya TCP/IP) dan requirement dapat juga didefinisikan dalam diagram ini.

2.1.5. ERD *(Entity Relationship Diagram)*

ERD adalah model konseptual yang mendeskripsikan hubungan antara penyimpanan (dalam DFD). ERD digunakan untuk memodelkan struktur data dan hubungan antar data. Dengan ERD, model dapat diuji dengan mengabaikan proses yang dilakukan.

ERD pertama kali dideskripsikan oleh Peter Chen yang dibuat sebagai bagian dari perangkat lunak CASE. Notasi yang digunakan dalam ERD dapat dilihat pada Tabel di bawah ini :

Notasi	Keterangan
	**Entitas,** Adalah suatu objek yang dapat diidentifikasi dalam lingkungan pemakai
	**Relasi**, Menunjukan adanya hubungan di antara sejumlah entitas yang berbeda
	**Atribut**, berfungsi mendeskripsikan karakter entitas (atribut yang berfungsi sebagai key diberi garis bawah)
	Garis, sebagai penghubung antara relasi dengan entitas, relasi dan entitas dengan atribut.

Sumber : Fathansyah (2007 : 70-77)

Gambar II.2 Tabel Notasi ERD

Fathansyah (2007:70-77) "Pemakaian istilah 'Model Keterhubungan-*Entitas*' dalam dahasa Indonesia dapat digunakan sebagai pandangan dari istilah asing: *Entity Relationship Diagram (E-R Model).*" Akan tetapi, istilah Model *Entity Relationship* telah demikian popular/umum digunakan dalam berbagai pembahasan tentang analisis atau perancangan basis data."

Pada dasarnya ada tiga simbol yang digunakan, yaitu :

a. Entity

Entity merupakan objek yang mewakili sesuatu yang nyata dan dapat dibedakan dari sesuatu yang lain. Simbol dari entity ini biasanya digambarkan dengan persegi panjang.

b. Atribut

Setiap entitas pasti mempunyai elemen yang disebut atribut yang berfungsi untuk mendeskripsikan karakteristik dari entitas tersebut. Isi dari atribut mempunyai sesuatu yang dapat mengidentifikasikan isi elemen satu dengan yang lain. Gambar atribut diwakili oleh simbol elips.

c. Hubungan / Relasi

Relasi didefinisikan sebagai hubungan yang terjadi antar entiti. Representasi diagram relasi adalah sebuah garis lurus yang menghubungkan dua buah entity. Jenis-jenis atau hubungan yang biasa terjadi antar satu entity dengan entity lain dalam sebuah basis data, meliputi :

1. One to one/satu ke satu (1 : 1)

Hubungan relasi satu ke satu yaitu setiap entitas pada himpunan entitas A berhubungan paling banyak dengan satu entitas pada himpunan entitas B. Contoh hubungan antara entity Mahasiswa dengan Kartu Mahasiswa. Seorang mahasiswa hanya boleh memiliki satu kartu mahasiswa, satu kartu mahasiswa hanya dimiliki oleh satu orang mahasiswa.

2. One to Many/Satu ke Banyak (1 : *)

Setiap entitas pada himpunan entitas A dapat berhubungan dengan banyak entitas pada himpunan entitas B, tetapi setiap entitas pada entitas B dapat

berhubungan dengan satu entitas pada himpunan entitas A. Contohnya hubungan yang terjadi antara entiti konsumen dengan motor. Sedangkan satu motor hanya dapat dimiliki oleh seorang konsumen.

3. Many to Many/Banyak ke Banyak (* : *)

Setiap entitas pada himpunan entitas A dapat berhubungan dengan banyak entitas pada himpunan entitas B. Contohnya hubungan yang terjadi antara Mahasiswa dengan Mata Kuliah. Satu mahasiswa dapat mengikuti lebih dari satu mata kuliah dan satu mata kuliah dapat diikuti oleh lebih dari satu mahasiswa.

2.2. Penelitian Terkait

Menurut Gede Sastra Kurniawan, Made windu Antara Kesiman, dan Gede Mahendra Darmawiguna (2013 :2) dalam penelitiannya :
Universitas Pendidikan Ganesha (Undiksha) sebagai sebuah badan organisasi memiliki beberapa badan administrasi antara lain administrasi akademik, administrasi kemahasiswaan, administrasi kepegawaian dan administrasi keuangan. Dari keempat badan administrasi yang dimiliki, hanya dua yang sudah memanfaatkan teknologi sistem informasi dalam pengelolaannya yaitu administrasi akademik dan administrasi kemahasiswaan. Sistem administrasi akademik Undiksha sudah mampu menangani hal-hal sebagai berikut.

1. Penawaran Matakuliah
2. Perencanaan Kartu Rencana Studi (KRS)
3. Mencetak/menyediakan Daftar Peserta Kuliah (DPK)
4. Mencetak/menyediakan Daftar Peserta Kuliah dan nilai akhir (DPNA)
5. Mencetak Kartu Hasil Studi (KHS)
6. Mencetak Pengajuan Kutipan Daftar Nilai (KDN) oleh Fakultas untuk mahasiswa yang akan maju ujian
7. Mencetak Transkrip Nilai mahasiswa yang akan di wisuda/sudah lulus
8. Mencetak ijazah

Sedangkan sistem administrasi kemahasiswaan mampu menangani manajemen data mahasiswa, memvalidasi pembayaran spp serta penanganan mahasiswa cuti, nonaktif, pindah, keluar dan berhenti. Kedua sistem tersebut sudah berjalan dengan baik.

Menurut Keyko Riskian Perdana, Bambang Eka Purnama, dan M.Kom Siska Iriani dalam penelitiannya yaitu Perkembangan jaman juga menuntut efisiensi dalam pekerjaan. Sekarang dunia teknologi informasi pun juga merambah di dunia birokrasi, Indonesia. Untuk salah satunya adalah di

Kantor Dinas Koperasi Perindustrian dan Perdagangan (DISKOPERINDAG) Kabupaten Pacitan. Kantor pelayanan publik dituntut akan kecepatannya dalam mengolah data. Sehingga sistem pengolahan data sangatlah penting. Dinas koperasi Perindustrian dan Perdagangan Kabupaten Pacitan memiliki bidang, yaitu bidang sekertariat, bidang bina usaha, bidang perdagangan, bidang industri, bidang kelembagaan. Berdasarkan survei di DISKOPERINDAG Kabupaten Pacitan penulis menemukan beberapa proses kerja yang dianggap sangat tidak efisien dari beberapa aspek seperti waktu, tenaga kerja, dan hasil kerja yang tidak rapi. Disini juga ditemukan fakta, bahwa bentuk data kepegawaian di DISKOPERINDAG masih menggunakan pembukuan. Sehingga memperlambat kinerja. Semua itu berpengaruh kesemua bidang, sehingga akan menghambat kelancaran dalam mengelola data di semua bidang.

Untuk memenuhi kebutuhan akan informasi ini, maka sistem informasi kepegawaian pada Mikro Mandiri Green Ville Jakarta ini perlu dikembangkan suatu sistem informasi kepegawaian yang memadai sehingga dapat memberikan informasi yang lebih baik. Sistem informasi kepegawaian berbasis web ini diharapkan mampu menjawab permasalahan yang ada mengenai sistem informasi kepegawaian terkomputerisasi sehingga akan mempermudah karyawan dalam menghasilkan sistem informasi yang akurat, tepat dan relevan sehingga dapat digunakan dalam pengelolaan data kepegawaian.