BAB 2 LANDASAN KEPUSTAKAAN

2.1 Kajian Pustaka

Kajian pustaka berisi perbandingan antara penelitian ini dengan penelitian yang telah dilakukan sebelumnya. Pada penelitian yang berkaitan dengan penelitian penulis adalah penelitian berjudul "*Penerapan Zachman Framework Dalam Merancang Sistem Pelaporan Kerusakan Komputer"* oleh Andika Agus Slameto, Ema Utami, dan Abas Ali Pangera. Pada penelitian tersebut membahas tentang penerapan *zachman framework* untuk memodelkan sistem informasi yang dipetakan dalam bentuk matrik *zachman* yang dilihat dari sudut pandang *Planner dan Owner.* Hasil akhir dari penelitian tersebut adalah sebuah *blueprint* rancangan sistem informasi.

Penelitian selanjutnya membahas tentang penerapan *zachman framework* yang berjudul "*Penerapan Zachman Framework Dalam Perancangan Arsitektur Sistem Manajemen Penyusunan Anggaran Keuangan Daerah"* oleh Antonius Wahyu Sudrajat. Pada penelitian tersebut menggunakan *zachman framework* dalam merancang sebuah sistem manajemen. Untuk baris pada kerangka kerja *zachman*, penelitian tersebut menggunakan *Scope (Contextual), Business Model (Conceptual), dan System Model (Logical)*.

Yang terakhir adalah penelitian yang berjudul "*Implementasi Enterprise Architecture Zachman Framework dan Metode Business Process Improvement pada Sistem Informasi Pemasaran"* oleh Arenzi Revelino Alwi. Pada penelitian tersebut membahas tentang perbaikan proses bisnis yang ada menggunakan metode *Business Process Imprvement* dan perancangan sistem informasi menggunakan *zachman framework.* Hasil akhir dari penelitian tersebut adalah sebuah rancangan perbaikan proses bisnis dan sebuah *prototype* sistem informasi yang dihasilkan dari analisis menggunakan *zachman framework.*

2.2 Sistem Informasi

Menurut Mustakini (2006) dalam bukunya yang berjudul *Analisis dan Desain Sistem Informasi*, sistem informasi adalah suatu sistem di dalam suatu organisasi yang mempertemukan kebutuhan pengolahan transaksi harian, mendukung operasi, bersifat manajerial dan kegiatan strategi dari suatu organisasi dan menyediakan pihak luar tertentu dengan laporan-laporan yang diperlukan.

Kegiatan sistem informasi diantaranya mencakup:

a. Input, berupa kegiatan yang bertujuan untuk menyediakan data yang nantinya akan diproses
b. Proses, berupa kegiatan yang menjelaskan bagaimana sebuah data diproses untuk menghasilkan informasi yang memiliki nilai tambah
c. Output, berupa suatu kegiatan yang bertujuan untuk menghasilkan laporan dari proses yang terjadi

d. Penyimpanan, suatu kegiatan yang bertujuan untuk menyimpan dan menyajikan data.

e. Kontrol, adalah suatu aktiitas yang bertujuan untuk menjamin bahwa sistem informasi tersebut sesuai yang diharapkan.

Dalam pengembangan sistem informasi akan terdapat Software Development Life Cycle (SDLC) yang berfungsi untuk mengetahui siklus pengembangan aplikasi.

2.3 Business Process Modelling Notation (BPMN)

Business process modelling notation adalah suatu standar yang dibuat oleh *Object Management Group (OMG)* untuk pemodelan proses bisnis. Dengan BPMN diharapkan pemodelan proses bisnis dapat dengan mudah dipahami oleh semua pengguna bisnis, termasuk juga yang bertanggung jawab dalam implementasi teknologi proses bisnis tersebut.

Sebuah *Business Process Diagram* terdiri dari elemen pemodelan berupa notasi grafis. Jenis dasar notasi BPMN adalah sebagai berikut (Camunda, n.d.):

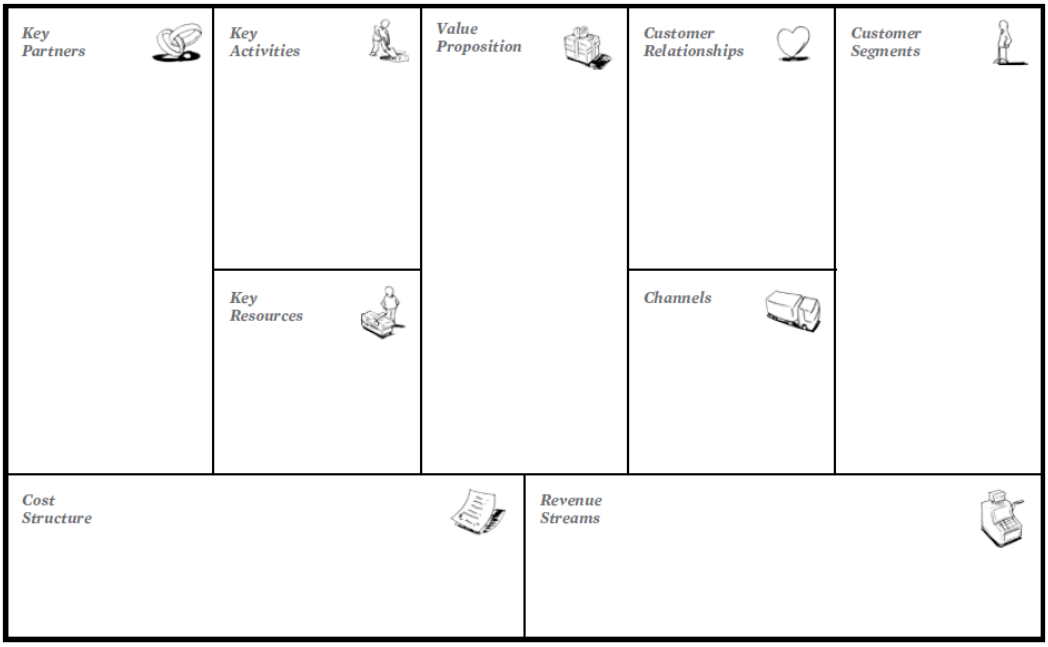
1. **Swimlane (Participants):** Swimlane adalah notasi grafis yang berbentuk persegi panjang yang digunakan untuk menyekat aktivitas satu dengan lainnya.
2. **Artifact:** *Artifact* digunakan untuk memberikan informasi mengenai objek atau elemen sebuah proses, tanpa memberi dampak apapun pada alur proses bisnis.
3. **Gateway:** *Gateway* digunakan untuk mengatur bagaimana alur proses berlangsung melalui *sequence flow,* dimana alur-alur tersebut dapat bertemu dan berpisah pada sebuah proses.
4. **Data:** Data digunakan untuk menjelaskan dan mengatur status data yang ada, seperti ditambahkan, dibaca, diperbarui, dihapus, dan sebagainya.
5. **Event:** *Event* adalah sesuatu yang terjadi didalam sebuah proses.
6. **Connecting object:** *Connecting object* berfungsi untuk membuat struktur proses bisnis beraturan dengan cara membuat alur bagi objek-objek yang ada pada proses.

2.4 Business Model Canvas

Business model canvas adalah bahasa yang sama untuk menggambarkan, memvisualisasikan, menilai, dan mengubah model bisnis. Model bisnis kanvas digambarkan dengan sembilan blok bangunan dasar yang menunjukkan logika bagaimana sebuah perusahaan bermaksud untuk menghasilkan uang. Kesembilan blok tersebut adalah (Osterwalder dan Pigneur, 2012):

1. **Customer segments:** mendefinisikan kelompok yang berbeda dari orang atau organisasi suatu perusahaan dengan tujuan menjangkau dan melayani.

2. **Value propositions:** menggambarkan bundle produk dan jasa yang menciptakan nilai bagi segmen pelanggan tertentu.
3. **Channels:** menggambarkan bagaimana sebuah perusahaan berkomunikasi dan menjangkau segmen pelanggan untuk memberikan *Value Proposition.*
4. **Customer Relationships:** menjelaskan jenis hubungan perusahaan dengan membentuk segmen pelanggan tertentu.
5. **Revenue streams:** merepresentasikan uang perusahaan menghasilkan dari setiap segmen pelanggan (biaya harus dikurangi pendapatan untuk memperoleh laba).
6. **Key resources:** menggambarkan asset yang paling penting yang dibutuhkan untuk membuat sebuah model bisnis bekerja.
7. **Key activities:** menggambarkan sesuatu yang paling penting yang perusahaan harus lakukan untuk membuat model bisnis bekerja.
8. **Key partnerships:** menggambarkan hubungan supplier dan partner yang membuat bisnis model bekerja.
9. **Cost structure:** menjelaskan semua biaya yang harus dikeluarkan untuk mengoperasikan model bisnis.



Gambar 2.1 Business Model Canvas

Sumber: Osterwalder dan Pigneur (2012)

2.5 Business Process Improvement

Business Process Improvement atau yang selanjutnya disebut BPI adalah sebuah metode yang dibangun untuk membantu sebuah organisasi dalam

membuat proses bisnis yang ada di dalam organisasi menjadi lebih baik. BPI memiliki tiga tujuan utama, yaitu (Harrington, 1991):

1. Membuat proses efektif: mengeluarkan hasil yang diinginkan.
2. Membuat proses efisien: meminimasi sumber daya.
3. Membuat proses adaptif: dapat beradaptasi terhadap perubahan kebutuhan pelanggan dan kebutuhan bisnis.

Terdapat 5 fase didalam *Business Process Improvement*, yaitu:

1. *Organizing for improvement* (mengorganisir perbaikan)

   Sasarannya adalah untuk mendapatkan suatu kesuksesan dengan membangun kepemimpinan, pemahaman, dan komitmen.

2. *Understanding the process* (pemahaman proses)

   Sasaran yang akan dicapai adalah pemahaman seluruh dimensi dari proses bisnis yang sedang berlangsung.

3. *Streamlining* (penyederhanaan proses)

   Sasaran yang dicapai adalah memperbaiki efisiensi, efektifitas, dan adaptibilitas dari proses bisnis.

4. *Measurement and controls* (pengukuran dan kontrol)

   Sasaran yang akan dicapai adalah mengimplementasikan suatu sistem untuk mengontrol jalannya proses perbaikan.

5. *Continuous improvement* (perbaikan berkelanjutan)

   Sasaran yang akan dicapai adalah untuk mengimplementasikan proses perbaikan selanjutnya.

Streamlining yaitu inisialisasi perbubahan proses kerja sehingga akan tercipta proses baru yang lebih sederhana dengan pencapaian tujuan yang sama. Harrington (1991) mengungkapkan ada beberapa cara yang bisa dilakukan untuk melakukan *streamlining,* yaitu:

1. *Bureaucracy elimination:* menghilangkan kegiatan administrasi yang tidak perlu.
2. *Duplication emlination:* menghilangkan kegiatan yang mirip atau serupa.
3. *Value-Added assessment:* evaluasi terhadap proses bisnis yang berkaitan dengan pelanggan.
4. *Simplification:* mengurangi kompleksitas proses bisnis.
5. *Process cycle time reduction:* mengurangi waktu siklus dan ongkos penyimpanan.
6. *Error proofing:* membuat sebuah kondisi agar sulit terjadinya kesalahan.
7. *Upgrading:* meningkatkan efektifitas dalam proses bisnis.

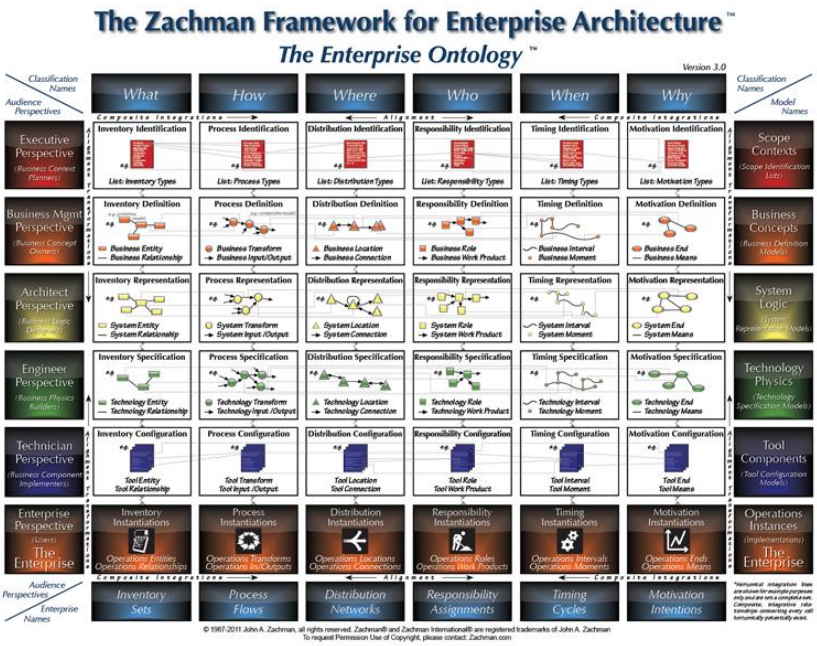
8. *Simple language:* mengurangi kompleksitas terhadap penulisan dan membuat dokumen yang lebih mudah dipahami.
9. *Standardization:* memilih standar baku dalam melakukan aktifitas proses bisnis.
10. *Supplier partnership:* meningkatkan kualitas input.
11. *Big picture improvement:* teknik yang digunakan jika kesepuluh teknik diatas tidak memberikan hasil yang diinginkan.
12. *Automation or / and mechanization:* penerapan peralatan dan komputer pada pekerjaan yang membosankan dan rutin, sehingga pegawai bisa lebih kreatif.

2.6 Enterprise Architecture

Enterprise Architecture adalah praktek yang didefinisikan dengan baik untuk melakukan analisis perusahaan, desain, perencanaan, dan pelaksanaan, menggunakan pendekatan holistik setiap saat, untuk keberhasilan pengembangan dan pelaksanaan strategi. Enterprise Architecture memberlakukan prinsip dan praktek arsitektur untuk memandu organisasi melalui bisnis, informasi, proses, dan perubahan teknologi yang diperlukan untuk menjalankan strategi mereka. Enterprise Architecture menganalisis kesenjangan antara keadaan masa depan dan keadaan saat ini, dan memberikan peta jalan yang mendukung evolusi dari perusahaan ke keadaan masa depan dengan menutup kesenjangan (The Federation of Enterprise Architecture Professional Organizations, 2013).

2.7 Zachman Framework

Zachman framework adalah sebuah struktur logis untuk mengklarifikasikan dan mengorganisir representasi deskriptif *Enterprise* yang signifikan terhadap pengelolaan *Enterprise* serta pengembangan sistem *Enterprise,* sistem manual serta sistem otomatis. Zachman framework mengkategorikan sudut pandang *stakeholder* yang berbeda menjadi satu perspektif yang mana semua orang bisa menemukan informasi yang tepat (Fatolahi, Somé, & Lethbridge, 2007).



Gambar 2.2 Zachman Framework

Sumber: Zachman (2008)

Zachman framework terdiri dari enam baris dan enam kolom. Baris merepresentasikan perbedaan sudut pandang *stakeholder* dalam membangun *Enterprise Architecture.* Berikut adalah enam perspektif zachman framework pada dimensi pertama:

1. ***The Planner's Perspective (Scope Context):*** menjelaskan penetapan objek dalam pembahasan, seperti latar belakang, lingkup, dan tujuan *enterprise.*
2. ***The Owner's Perspective (Business Concept):*** pemakai akhir dari produk atau jasa *enterprise.*
3. ***The Designer's Perspective (System Logic):*** perantara antara apa yang diinginkan (pemilik) dan apa yang dapat dicapai secara teknis dan fisik.
4. ***The Builder's Perspective (Technology Physic):*** pengawas atau pengatur dalam menghasilkan produk/jasa akhir.
5. ***The Sub-Contractor's Perspective (Component Assemblies):*** bertanggung jawab membangun dan merakit bagian-bagian dari produk/jasa akhir.
6. ***The Functioning Enterprise Perspective (Operation Classes):*** wujud nyata dari produk/jasa akhir.

Kolom merepresentasikan setiap isu perspektif yang membutuhkan cara yang berbeda untuk menjawab pertanyaan fundamental: *who, what, why, when, where,* dan *how.* Berikut adalah enam perspektif zachman framework pada dimensi kedua (Ertaul dan Rathod, n.d.):

1. **What (kolom data):** membahas pemahaman tentang data perusahaan.

2. **How (kolom fungsi):** menjelaskan berbagai proses yang terlibat dalam berurusan dengn kolom Data.
3. **Where (kolom jaringan):** menjelaskan lokasi geografis dan logistic antar entitas.
4. **Who (kolom orang):** menjelaskan orang-orang yang berpartisipasi dalam kegiatan organisasi.
5. **When (kolom waktu):** menjelaskan kapan "fungsi" harus dilakukan.
6. **Why (kolom tujuan):** menjelaskan akhir tujuan, kendala, aturan dan peraturan.

2.8 Analisis dan Perancangan Berorientasi Objek

Analisis dan perancangan berorientasi objek (*object-oriented analysis and design/OOAD)* adalah sebuah metode untuk menganalisis dan merancang sebuah sistem dengan menggunakan pendekatan *object-oriented.* Objek diartikan sebagai suatu entitas yang memiliki identitas, state, dan behavior (Bintarika, 2009).

Pada fase analisis menghasilkan kebutuhan perangkat lunak (*software requirements specification)* yang terdiri dari *business process diagrams, use-case diagrams,* dan *class diagram.* Pada fase perancangan mengaplikasikan pemodelan data ke dalam *class diagram* yang dihasilkan saat fase analisis. Perancangan meliputi *sequence diagram, collaboration diagram, activity diagram,* dan *statechart diagram* (ARPITA, 2011).

2.9 Unified Modeling Language (UML)

Unified Modelling Language (UML) adalah sebuah Bahasa grafis untuk memvisualisasikan, merancang, membangun, dan mendokumentasikan artefak dari sistem perangkat lunak. UML menawarkan suatu standar untuk menulis *blueprint* sistem, termasuk hal-hal konseptual seperti bisnis proses dan fungsi sistem serta hal-hal yang konkret seperti pertanyaan bahasa pemrograman, skema database, dan komponen perangkat lunak yang *reusable* (ARPITA, 2011).

Unified Modeling Language (UML) adalah bahasa pemodelan visual yang digunakan dalam menentukan, memvisualisasi, menciptakan, dan mendokumentasikan artefak dari sistem perangkat lunak. Terdapat beberapa tujuan dibalik pengembangan UML. Yang paling penting adalah supaya semua pemodel dapat menggunakannya. UML tidak dimaksudkan untuk menjadi suatu metode pengembangan yang lengkap, tetapi untuk dapat mendukung sebagian hingga semua dari proses pembangunan yang ada.

2.9.1 Use Case Diagram

Use case menggambarkan unit fungsi yang dihasilkan oleh suatu sistem. Tujuan dari use case diagram adalah untuk membantu proses development team dalam memvisualisasikan kebutuhan fungsional sistem, termasuk hubungan antar "aktor" (manusia atau sistem lain yang akan melakukan interaksi dengan sistem)

untuk melakukan suatu proses serta hubungan antara use case yang berbeda (Desanti, 2010).

Use case merupakan model fungsional sistem yang dipersepsikan sebagai pengguna dari luar yang disebut aktor. Use case dapat mengekspresikan transaksi antara actor dan sistem. Tujuan dari penggunaan use case adalah untuk mencatat daftar aktor dan use case, selain itu use case dapat menunjukkan partisipasi actor pada setiap use case.

Berikut merupakan elemen-elemen/komponen dari use case diagram:

1. **Aktor:** Menggambarkan tokoh atau sistem yang memperoleh keuntungan dan berada di luar dari sistem. Aktor diletakkan pada bagian luar subject boundary.
2. **Use Case:** Mewakili sebuah bagian dari fungsionalitas sistem dan ditempatkan dalam system boundary.
3. **Association Relationship:** Menghubungkan actor untuk berinteraksi dengan use case.
4. **Include Relationship:** Menunjukkan inclusion fungsionalitas dari sebuah use case dengan use case lainnya. Arah panah dari base use case ke included use case.
5. **Extend Relationship:** Menunjukkan keadaan yang hanya berjalan di kondisi tertentu seperti menggerakkan alarm
6. **Generalization Relationship:** Menunjukkan generalisasi dari use case khusus ke yang lebih umum

2.9.2 Sequence Diagram

Sequence diagram merepresentasikan interaksi antar objek di bagian dalam dan di sekitar sistem termasuk pengguna, display, dan sebagainya) berupa message yang direpresentasikan terhadap waktu. Sequence diagram terdiri antar dimensi vertical (waktu) dan dimensi horizontal (objek-objek yang terkait). Sequence diagram seringkali digunakan dalam menggambarkan skenario atau rangkaian langkah yang dilakukan sebagai respon dari sebuah event dalam menghasilkan output apa saja yang terjadi secara internal dan output apa yang dihasilkan.

Lifeline merepresentasikan partisipasi individu dalam sequence diagram. Lifeline bisaanya mempunyai kotak yang berisi nama objek. Kadang-kadang sequence diagram akan mempunyai lifeline dengan elemen actor. Elemen boundary, control dan entity dapat juga mempunyai lifelinenya sendiri. (Spark, 2014).

Berikut adalah elemen-elemen yang dimiliki oleh sequence diagram:

1. **Lifeline:** Objek entity, antarmuka yang saling berinteraksi.
2. **Aktor:** Digunakan untuk menggambarkan user/pengguna.
3. **Message:** Spesifikasi dari komunikasi antar objek yang memuat informasi-informasi tentang aktifitas yang terjadi.

4. **Boundary:** Digunakan untuk menggambarkan sebuah form.
5. **Control class:** Digunakan untuk menghubungkan boundary dengan table.
6. **Entity class:** Digunakan untuk menggambarkan hubungan kegiatan yang akan dilakukan.

2.9.3 Class Diagram

Class adalah sebuah spesifikasi yang jika di instansiasi akan menghasilkan sebuah objek dan merupakan inti dari pengembangan dan desain berorientasi objek. Class menggambarkan keadaan (atribut/property) dari suatu sistem, serta menawarkan layanan dalam memanipulasi keadaan tersebut (metode/fungsi).

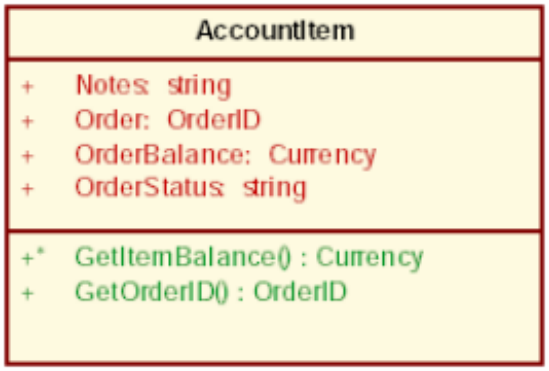
Class diagram membuat sebuah struktur dan deskripsi dari class, package serta hubungan antar komponen seperti containment, pewarisan, asosiasi, dan lain-lain (Sommerville, 2011).

Class memiliki tiga area pokok diantaranya:

1. Nama (Class Name)
2. Atribut
3. Metoda (Operations)

Atribut dan metoda dapat memiliki salah satu sifat sebagai berikut:

1. Private, tidak dapat dipanggil dari luar Class yang bersangkutan.
2. Protected, hanya dapat dipanggil oleh Class yang bersangkutan dan anak-anak mewarisinya.
3. Public, dapat dipanggil oleh siapa



Gambar 2.3 Notasi Class Diagram

Sumber: (Dharwiyanti dan Wahono, 2003)

Hubungan antar class dalam class diagram terdiri dari:

a. Asosiasi, yaitu hubungan statis antar class. Bisaanya menggambarkan suatu class yang memiliki atribut berupa class lain, atau class yang harus mengetahui keberadaan class lain. Panah navigability menunjukkan arah query antar class.
b. Agregasi, yaitu hubungan yang menyatakan bagian (“Terdiri atas.”).
c. Pewarisan, yaitu hubungan hirarkis antar class. Suatu class dapat diturunkan/inheritance dari class lain dan mewarisi semua atribut dan metoda

class induknya dan menambahkan sebuah fungsionalitas baru, sehingga ia disebut anak dari class yang diwarisinya. Kebalikan dari pewarisan adalah generalisasi.

d. Hubungan dinamis, yaitu rangkaian pesan (message) yang di-passing dari class kepada class lain.

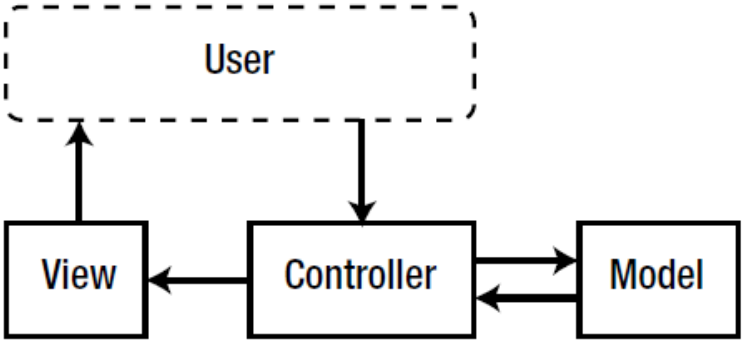
Berikut adalah elemen-elemen yang dimiliki oleh class diagram:

1. **Association:** yang menghubungkan objek satu dengan objek lainnya.
2. **Aggregation:** upaya untuk menghindari asosiasi dengan lebih dari 2 objek.
3. **Composite:** varian yang lebih kuat dari "*memiliki*" atau hubungan asosiasi, *composite* lebih spesifik dari agregasi.
4. **Class:** himpunan dari objek-objek yang berbagi atribut serta operasi yang sama.
5. **Collaboration:** deskripsi dari urutan aksi-aksi yang ditampilkan sistem.
6. **Realization:** operasi yang benar-benar dilakukan oleh suatu objek.
7. **Dependency:** hubungan dimana perubahan yang terjadi pada suatu elemen yang mandiri (*independent).*
8. **Boundary:** digunakan untuk model interaksi-interaksi antara sistem dan para aktornya.
9. **Control:** digunakan untuk *encapsulate control* yang dihubungkan dengan usecase yang spesifik.
10. **Entity class:** digunakan untuk model yang bertujuan sebagai informasi.

Pada penulisan skripsi ini penulis menggunakan MVC pattern. MVC pattern ini membagi class dalam 3 kategori yaitu: Boundary, controller, dan Entity class. Entity class adalah class yang memproses data atau business rules yang mengakses dan mengubah dari/ke database. Boundary class menterjemahkan isi dari entity class dan menampilkannya kepada pengguna. Controller class adalah yang bertanggung jawab untuk menterjemahkan interaksi antara boundary class dan memberikannya ke entity class. (Kama, 2011).

2.10 MVC (Model View Controller)

MVC (Model-View-Controller) adalah pola desain perangkat lunak yang dibangun di sekitar interkoneksi dari tiga jenis komponen utama, dalam bahasa pemrograman seperti PHP, sering dengan fokus yang kuat pada paradigma pemrograman berorientasi objek (OOP) perangkat lunak. Ketiga jenis komponen diistilahkan model, view, dan controller (Pitt, 2012).



Gambar 2.4 Komponen MVC

Sumber: Chris Pitt (2012)

2.10.1 Model

Model adalah di mana semua logika bisnis dari aplikasi disimpan. Logika bisnis bisa apa saja khusus untuk bagaimana sebuah aplikasi menyimpan data, atau menggunakan layanan pihak ketiga, dalam rangka untuk memenuhi kebutuhan bisnis. Jika aplikasi harus mengakses informasi dalam database, kode untuk melakukan itu akan disimpan dalam model. Jika diperlukan, misalnya, untuk mengambil data saham atau tweet tentang produk baru, kode juga akan disimpan dalam model (Pitt, 2012).

2.10.2 View

View adalah di mana semua elemen antarmuka pengguna dari aplikasi kita disimpan. Hal ini dapat mencakup markup kami HTML, CSS style sheet, dan file JavaScript. Apa pun yang pengguna lihat atau berinteraksi dapat disimpan dalam sebuah view, dan kadang-kadang apa yang pengguna lihat sebenarnya merupakan kombinasi dari banyak view yang berbeda dalam permintaan yang sama (Pitt, 2012).

2.10.3 Controller

Controller adalah adalah komponen yang menghubungkan model dan view bersama-sama. Controller mengisolasi logika bisnis model dari view elemen antarmuka pengguna, dan menangani bagaimana aplikasi akan merespon interaksi pengguna dalam view. Controller adalah titik pertama yang masuk ke trio komponen ini, karena permintaan pertama dilewatkan ke controller, yang kemudian akan instantiate model dan view yang diperlukan untuk memenuhi permintaan untuk aplikasi (Pitt, 2012).

2.11 Bizagi Modeler

Bizagi modeler adalah perangkat lunak yang digunakan untuk memodelkan proses bisnis dan mendokumentasikannya. Bizagi modeler memungkinkan untuk memvisualkan diagram, model, dan dokumen bisnis proses sesuai standard

BPMN. Bizagi modeler dapat diunduh secara gratis langsung dari internet dan digunakan desktop atau *personal computer.*

Hasil dokumentasi menggunakan bizagi modeler dapat dipublikasikan dengan kualitas tinggi menggunakan Word, PDF, Sharepoint, atau Wiki. Proses didalam bizagi juga dapat dengan mudah di impor dan di ekspor dari Visio atau XML, dan aplikasi lainnya (Bizagi, 2017).

2.11.1 Simulasi pada Bizagi Modeler

Menurut Bizagi (2017) Simulasi proses dapat membantu dalam proses evaluasi performansi sebuah model, dibawah konfigurasi yang berbeda dan dalam waktu real time, untuk dapat meminimalkan kegagalan yang terjadi dalam memenuhi spesifikasi dan mencegah kekurangan atau kelebihan sumber daya (termasuk sumber daya manusia dan biaya).

Simulasi pada Bizagi terdiri atas empat level. Setiap level berikutnya menggabungkan informasi tambahan yang lebih kompleks. Berikut merupakan keempat level tersebut (Bizagi, 2017):

1. Level validasi proses (*process validation*)

   Level validasi proses digunakan untuk memastikan semuua proses berjalan secara berurutan dan sesuai dengan yang diharapkan. Validasi proses melihat sejauh mana proses siap untuk dievaluasi, mengasumsikan sejauh mana terjadinya perubahan, dan melaporkan kesalahan yang ditemui.

2. Level analisis waktu (*time analysis*)

   Level analisis waktu digunakan untuk menampilkan perhitungan waktu yang dibutuhkan diagram proses bisnis. Menampilkan waktu mulai (*start event*) dan waktu distribusi pelayanan dan waktu penyelesaian *task* atau *service*.

3. Level analisis sumber daya (*resource analysis*)

   Simulasi yang dijalankan pada level analisis sumber daya digunakan untuk memberikan informasi ketika proses dieksekusi oleh sumber daya di organisasi, mendefinisikan sumber daya yang terlibat dan biaya yang diperlukan.

4. Level analisis kalender (*calendar analysis*)

   Level analisis kalender dilakukan dengan memasukkan informasi pada kalender untuk menampilkan performa proses bisnis dalam jangka waktu tertentu. Menentukan hari apa dan berapa kali sumber daya tersedia untuk bekerja dalam sehari, seminggu atau sebulan.

2.12 Pengujian

Pengujian adalah suatu proses pelaksanaan suatu program dengan tujuan menemukan suatu kesalahan. Suatu kasus test yang baik adalah apabila test tersebut mempunyai kemungkinan menemukan sebuah kesalahan yang tidak

terungkap. Suatu test yang sukses adalah bila test tersebut membongkar suatu kesalahan yang awalnya tidak ditemukan (Mustaqbal, 2015).

Beberapa orang yang terlibat dalam pengujian adalah penguji perangkat lunak, pengembang perangkat lunak, manajer proyek, dan pengguna terakhir. Dokumentasi uji melibatkan dokumentasi artefak yang harus dikembangkan sebelum atau selama pengujian software. Dokumen umum dalam pengujian perangkat lunak, yaitu *test plan, test scenario, test case,* dan *traceability matrix.*

2.12.1 Pengujian Black-box

Pengujian black-box berfokus pada kebutuhan fungsional perangkat lunak dan tanpa memperhatikan *source code* program. Pengujian black box bukanlah solusi alternatif dari pengujian white box, tapi lebih merupakan pelengkap untuk menguji hal-hal yang tidak dicakup oleh pengujian white box. Pengujian black box digunakan untuk menemukan hal-hal berikut (Mustaqbal, 2015):

1. Fungsi-fungsi yang tidak benar atau tidak ada
2. Kesalahan antarmuka (*interface errors)*
3. Kesalahan pada struktur data dan akses basis data
4. Kesalahan performansi (*performance errors)*
5. Kesalahan inisialisasi dan terminasi

Berikut ini merupakan beberapa teknik pengujian *black-box,* yaitu (Nidhra and Dondetti, 2012):

1. *Equivalence class partitioning*
2. *Boundary value analysis*
3. *Decision tables*
4. *State transition diagrams*
5. *Orthogonal arrays*
6. *All pair technique*

2.12.2 Pengujian White-box

White Box Testing adalah salah satu cara untuk menguji suatu aplikasi atau software dengan cara melihat modul untuk dapat meneliti dan menganalisis kode dari program yang di buat ada yang salah atau tidak. Kalau modul yang telah dan sudah di hasilkan berupa output yang tidak sesuai dengan yang di harapkan maka akan dikompilasi ulang dan di cek kembali kode-kode tersebut hingga sesuai dengan yang diharapkan. Berikut ini merupakan beberapa teknik pengujian *white-box,* yaitu (Nidhra and Dondetti, 2012):

1. *Static white box testing*
    a. *Desk checking*
    b. *Code walkthrough*

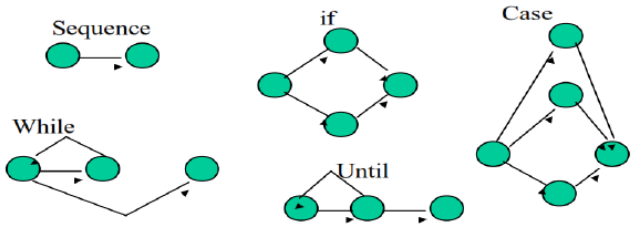
c. *Formal inspections*

2. *Structural white box testing*
   a. *Control flow/ coverage testing*
   b. *Basic path testing*
   c. *Loop testing*
   d. *Data flow testing*

Basic path testing – flow graph notation

Control flow graph (CFG) adalah grafik terarah yang terdiri dari 2 jenis, yaitu *node* dan *control flow. Node* dinotasikan dengan lingkaran berlabel, merepresentasikan satu atau lebih pernyataan, keputusan, kondisi, dan prosedur dari program. *Control flow* dinyatakan dengan garis, bisa disebut *edge,* merepresentasikan aliran control program. Berikut beberapa aturan yang terdapat dalam *flow graph* (Nidhra and Dondetti, 2012):

1. Dalam *flow graph* symbol panah disebut sebagai *Edge* yang merepresentasikan aliran control.
2. Lingkaran disebut sebagai *node* yang merepresentasikan satu atau lebih action.
3. Area yang dibatasi oleh *edges* dan *nodes* disebut *region.*
4. Predikat node adalah *node* yang mengandung sebuah kondisi.



Gambar 2.5 Notasi pada *flow graph*

Sumber: Nidhra and Dondetti (2012)