BAB II
TINJAUAN PUSTAKA

1.1. Penelitian Terdahulu

Penelitian terdahulu yang digunakan dalam penelitian ini adalah :

Tabel 2.1. Penelitian Terdahulu

Judul	Penulis	Tahun
Sistem Pendukung Keputusan Untuk Pemilihan Supplier Fasilitas Rumah Sakit Menggunakan Metode Profile Matching	Tiya Nur Cahya, Suaidah	2021
Penerapan Metode Simple Additive Weighting pada Sistem Pendukung Keputusan Penerimaan Debitur Anggota Koperasi	Putu Adi Wiryawan, I Gede Suardika, I Ketut Putu Suniantara	2020
Sistem Pendukung Keputusan Pemberian Kredit Menggunakan Metode Simple Additive Weighting (SAW) (Studi Kasus Koperasi Bengkawas Jaya)	Kiki Yasdomi, Detri Amelia Chandra	2017
Sistem Penunjang Keputusan Penilaian Kesehatan Organisasi Koperasi Menggunakan Metode Simple Additive Weighting (SAW)	Nadya Oktavina, Dedih, Yessy Yanitasari	2019
Sistem Pendukung Keputusan Pemilihan Koperasi Berprestasi Menggunakan Metode *Simple Additive*	Mirajul Rifqi, Dona	2018

Weighting Pada Kabupaten Rokan Hulu		

1.1.1. Tinjauan pada literature 1

(Nur Cahya and Suaidah, 2021). Sistem Pendukung Keputusan Untuk Pemilihan Supplier Fasilitas Rumah Sakit Menggunakan Metode Profile Matching. Supplier merupakan salah satu bagian terpenting dalam suatu perusahaan penyedia jasa fasilitas rumah sakit. Untuk mendapatkan hasil yang terbaik, dibutuhkan pula supplier yang terbaik dan berkualitas. Salah satu upaya untuk mendapatkan supplier tersebut adalah dengan melakukan pemilihan supplier, oleh karena itu pemilihan supplier sangat diperlukan untuk perusahaan penyedia jasa fasilitas rumah sakit. Proses penilaian supplier yang masih bersifat manual mengakibatkan sering terjadi kehilangan dan kerusakan dokumen, pencarian dan perubahan data memerlukan waktu yang cukup lama, serta jadwal yang terlewat dalam hal pengiriman barang oleh supplier ke pihak perusahaan. Hal ini mengakibatkan pihak perusahaan mengalami banyak kesulitan dalam proses penilaian supplier, serta pihak perusahaan kesulitan dalam melihat hasil penilaian daripersyaratan penilaian supplier yang dihitung untuk dijadikan bahan evaluasi kinerja supplier.

Sebuah Sistem Pendukung Keputusan (SPK) untuk memberikan rekomendasi terhadap kesesuaian alternatif atau pilihan yang dapat dibangun dengan menggunakan metode Profile Matching. Penggunaan metode profile matching untuk kasus yang mengganggap bahwa nilai tertinggi adalah nilai terbaik mengharuskan nilai ideal yang digunakan adalah nilai maksimum agar tidak terjadi ekspektasi yang melebihi nilai ideal. Aspek yang akan digunakan pada penelitian ini adalah izin perusahaan, izin edar barang, peredaran barang, spesifikasi barang, harga barang, dan garansi. Hasil pengujian sistem menggunakan blackbox testing mendapatkan hasil 100% sesuai dengan pengujian fungsionalitas sistem, dan hasil pengujian ISO 25010 mendapatkan hasil 85,47% maka ISO 25010 prototype Sangat Baik untuk sistem pendukung keputusan untuk pemilihan supplier.

1.1.2. Tinjauan pada literature 2

(Putu Adi Wiryawan, I Gede Suardika and Suniantara, 2020). Penerapan Metode Simple Additive Weighting pada Sistem Pendukung Keputusan Penerimaan Debitur Anggota Koperasi. Koperasi simpan pinjam Graha Computindo merupakan salah satu lembaga keuangan masyarakat

untuk melakukan penyimpanan dan memberikan pinjaman/kredit pada setiap anggota dan dibentuk untuk menyejahterahkan anggotaanggotanya. Setiap anggota koperasi yang ingin meminjam kredit harus mengajukan permohonan pengajuan pinjaman. Analisa pinjaman di koperasi tersebut berdasarkan pada umur, penjamin, jenis usaha dan prinsip 5C, yaitu Characteristic, Capital, Collateral, Condition dan Capacity yang dilakukan secara manual. Untuk mempercepet proses pemberian kredit dilakukan dengan sistem pendukung keputusan, dengan Metode Simple Additive Weighting (SAW) sebagai dasar perhitungan keputusan karena menggunakan kriteria dan membandingkan nilai antar satu calon dengan calon yang lain. Hasil penelitian ini berbentuk sebuah sistem pendukung keputusan yang mengolah data debitur yang dapat digunakan dalam proses pemberian kredit. Sistem ini dapat memberikan perangkingan sesuai data masing – masing debitur dan mempermudah pengambil keputusan dalam pemberian kredit.

1.1.3. Tinjauan pada literature 3

(Yasdomi and Amelia Chandra, 2017). Sistem Pendukung Keputusan Pemberian Kredit Menggunakan Metode Simple Additive Weighting (SAW) (Studi Kasus Koperasi Bengkawas Jaya). Koperasi Bengkawas Jaya merupakan salah satu usaha di bidang pelayanan simpan pinjam yang mengalami peningkatan jumlah nasabah setiap bulannya sehingga menyebabkan meningkatnya pengelolaan jumlah data nasabah. Pada proses Koperasi Bengkawas Jaya perhitungan nilai data nasabah yang akan menerima kredit pinjaman dilakukan secara manual dan sering terjadi kesalahan, sehingga proses perangkingan nilai akan menjadi tidak efisien serta membutuhkan waktu yang cukup lama. Oleh sebab itu diperlukan sebuah Sistem Pendukung Keputusan Pemberian Kredit Pinjaman Menggunakan Metode Simple Additive Weighting (SAW) untuk menyelesaikan permasalahan tersebut. Sistem Pendukung Keputusan Pemberian Kredit Pinjaman pada penelitian ini dibangun dengan menggunakan bahasa pemrograman PHP dengan didukung basis data MySQL. Tujuan dan manfaat dari analisa dan perancangan sistem pendukung keputusan ini yaitu menghasilkan sistem pendukung keputusan dengan basis Web yang dapat membantu kinerja pegawai Koperasi Bengkawas Jaya dalam pemilihan nasabah yang menerima kredit pinjaman.

1.1.4. Tinjauan pada literature 4

(Oktavina, Dedih and Yanitasari, 2019). Sistem Penunjang Keputusan Penilaian Kesehatan Organisasi Koperasi Menggunakan Metode Simple Additive Weighting (SAW). Penilaian kesehatan merupakan proses untuk mengukur tingkat kesehatan koperasi simpan pinjam (KSP) dan usaha simpan pinjam (USP). Penilaian dilakukan terhadap organisasi koperasi pada perusahaan atau instansi. Ada 7 aspek penilaian kesehatan yaitu permodalan, kualitas aktiva produktif, manajemen, efisiensi, likuiditas, kemandirian, dan jatidiri. Dalam proses penilaian tidak jarang terjadi kesalahan dalam perhitungannya. Oleh karena itu dibuatlah sistem penunjang keputusan menggunakan metode Simple Additive Weighting dengan metode pengembangan System Development Life Cycle (SDLC) Waterfall, dengan 5 data koperasi diperoleh hasil nilai tertinggi 0,89 kategori sehat oleh kopkar Rs Bayukarta dan nilai terendah 0,71 kategori cukup sehat oleh KSP Tirta Karawang Sejahtera.

1.1.5. Tinjauan pada literature 5

(Mi'rajul and Dona, 2018). Sistem Pendukung Keputusan Pemilihan Koperasi Berprestasi Menggunakan Metode *Simple Additive Weighting* Pada Kabupaten Rokan Hulu. Koperasi sebagai dasar perekonomian diwujudkan dalam pembangunan perekonomian nasional bertujuan untuk mewujudkan kedaulatan politik dan ekonomi Indonesia. Berdasarkan data Asosiasi Kader Sosio-Ekonomi Strategis (AKSES), saat ini dari 206.000 koperasi di Indonesia, 70 persen di antaranya sudah tidak beroperasi lagi, 23 persen di antaranya mati suri, dan sisanya bertahan dengan berbagai tekanan. Penilaian Koperasi Berprestasi dalam kurun waktu tertentu dilihat penting dengan tujuan untuk memberikan motivasi pada koperasi agar dapat berfungsi sebagai lembaga ekonomi yang mampu meningkatkan pendapatan anggota. Dengan kerumitan kriteria-kriteria dalam penilaian koperasi berprestasi di Kabupaten Rokan Hulu, maka pengambilan keputusan dalam menentukan koperasi berprestasi selain membutuhkan waktu yang lama bisa menimbulkan kesalahan pada proses penilaian atau tidak objektifnya penilaian pada alternatif-alternatif (koperasi). Sistem pendukung keputusan untuk penilaian koperasi berprestasi ini menggunakan metode SAW (Simple Additive Weighting). Dengan metode ini, pemecahan masalah dapat dilakukan penilaian secara lebih tepat karena didasarkan pada nilai kriteria dan bobot preferensi yang sudah ditentukan.

Perbedaan dengan penelitian yang dilakukan adalah penentuan koperasi sehat ini menggunakan aspek yaitu Modal, Kualitas Aktiva Produktif, Manajemen, Efisiensi, Likuiditas, Kemandirian, dan Jati Diri Koperasi dan dilakukan pada PLUT KUMKM Provinsi Lampung, sedangkan persamaannya yaitu menggunakan metode SAW.

1.2. Sistem

Menurut (Romney and Steinbart, 2014), Sistem adalah rangkaian dari dua atau lebih komponen-komponen yang saling berhubungan, yang berinteraksi untuk mencapai suatu tujuan. Sebagian besar sistem terdiri dari subsistem yang lebih kecil yang mendukung sistem yang lebih besar. Menurut (Mulyadi, 2016), Sistem adalah "suatu jaringan prosedur yang dibuat menurut pola yang terpadu untuk melaksanakan kegiatan pokok perusahaan".

Berdasarkan pengertian diatas dapat disimpulkan sistem adalah kumpulan dari komponen-komponen yang saling berkaitan satu dengan yang lain untuk mencapai tujuan dalam melaksanakan suatu kegiatan pokok perusahaan.

1.3. Infromasi

(Krismaji, 2015), Informasi adalah "data yang telah diorganisasi dan telah memiliki kegunaan dan manfaat".

Menurut (Romney and Steinbart, 2014), Informasi adalah data yang telah dikelola dan diproses untuk memberikan arti dan memperbaiki proses pengambilan keputusan. Sebagaimana perannya, pengguna membuat keputusan yang lebih baik sebagai kuantitas dan kualitas dari peningkatan informasi.

Berdasarkan pengertian diatas dapat disimpulkan bahwa pengertian informasi adalah data yang diolah agar bermanfaat dalam pengambilan keputusan bagi penggunanya.

1.4. Sistem Pendukung Keputusan

Sistem pedukung keputusan (*Decision Support System*) merupakan sistem informasi intraktif yang menyediakan informasi, pemodelan, dan pemanipulasian data. Sistem ini digunakan untuk membantu pengambilan keputusan dalam situasi yang semiterstruktur dan situasi yang tidak terstruktur, dimana tak seorang pun tahu secara pasti bagaimana keputusan seharusnya dibuat. DSS biasanya dibangun untuk medukung solusi atas suatu masalah atau untuk mengevaluasi suatu

peluang. DSS seperti itu disebut aplikasi DSS. Aplikasi DSS menggunakan CBIS (*Computer based information system*) yang fleksibel, interaktif, dan dapat diadaptasi, yang dikembangkan untuk mendukung solusi atas masalah manajemen spesifikasi yang tidak terstruktur. Aplikasi DSS menggunakan data, memberikan antarmuka pengguna yang mudah, dan dapat menggabungkan pemikiran pengambilan keputusan (Anggraeni and Irviani, 2017).

Menurut (Virgiawan, 2016), konsep Sistem Pendukung Keputusan (SPK) / Decision Support Sistem (DSS) pertama kali diungkapkan pada awal tahun 1970-an oleh Michael S. Scott Morton dengan istilah Management Decision Sistem. Sistem tersebut adalah suatu sistem yang berbasis komputer yang ditujukan untuk membantu pengambil keputusan dengan me-manfaatkan data dan model tertentu untuk memecahkan berbagai persoalan yang tidak terstruktur.

1.5. Simple Additive Weighting

Prinsip kerja metode SAW adalah mencari penjumlahan terbobot dari rating kinerja pada setiap alternatif di semua atribut. Metode SAW membutuhkan proses normalisasi matriks keputusan (X) ke suatu skala yang dapat diperbandingkan dengan semua rating alternatif yang ada. Sedangkan Fuzzy SAW (F-SAW) merupakan penggabungan metode SAW dengan logika matematika fuzzy (Yasdomi *et al.*, 2017). Teori himpunan Fuzzy digunakan untuk mempresentasikan permasalahan ketidakpastian. Sebuah bilangan fuzzy memiliki himpunan fuzzy yang ditandai dengan pemberian interval dari 0 sampai 1. Perbedaan utama antara Fuzzy SAW dengan SAW adalah pada proses penentuan nilai kriteria. Metode SAW menerapkan nilai tegas sedangkan pada FSAW nilai diubah kedalam bentuk bilangan fuzzy. Sehingga implementasi nilai pada matrix perbandingan, yakni diwakili oleh tiga variabel (a, b, c) yang disebut Triangular Fuzzy Numbers (TFN). Hal ini berarti nilai yang ditemukan bukan satu melainkan tiga, sesuai dengan fungsi keanggotaan segitiga yang meliputi tiga bobot yang berurutan. Secara umum, prosedur F-SAW mengikuti langkah-langkah sebagai berikut (Roszkowska and Kacprzak, 2016).

1.6. Perhitungan Simple Additive Weighting

Metode SAW sering dikenal dengan istilah metode penjumlahan terbobot. Konsep dasar metode SAW (*Simple Additive Weighting*) adalah mencari penjumlahan terbobot dari rating kinerja pada setiap alternatif pada semua atribut (Oktavina *et al.*, 2019). Metode SAW dapat membantu dalam pengambilan keputusan suatu kasus, akan tetapi perhitungan dengan menggunakan metode SAW ini hanya yang menghasilkan nilai terbesar yang akan terpilih sebagai alternatif yang terbaik. Perhitungan akan sesuai dengan metode ini apabila alternatif yang terpilih memenuhi kriteria yang telah ditentukan. Metode SAW ini lebih efisien karena waktu yang

dibutuhkan dalam perhitungan lebih singkat (Putu Adi Wiryawan, I Gede Suardika and Suniantara, 2020). Metode SAW membutuhkan proses normalisasi matriks keputusan (X) ke suatu skala yang dapat diperbandingkan dengan semua rating alternatif yang ada.

Untuk melakukan normalisasi tabel pada tahap analisa, dengan rumus berikut ini

$$r_{ij} = \begin{cases} \dfrac{x_{ij}}{\underset{i}{Max} x_{ij}} & \text{Jika j adalah atribut keuntungan (}\textit{benefit}\text{)} \\ \dfrac{\underset{i}{Min} x_{ij}}{x_{ij}} & \text{Jika j adalah atribut biaya (}\textit{cost}\text{)} \end{cases}$$

Keterangan:

r_{ij} = Rating kinerga ternormalisasi dari alternative Ai (i = ,2,…,m)

Maxi = Nilai maximal dari setiap baris dan kolom

Mini = Nilai minimum dari setiap baris dan kolom

xij = Baris dan kolom dari matriks

dimana r_{ij} adalah rating kinerja ternormalisasi dari alternatif Ai pada atribut Cj; i=1,2,..., m dan j=1,2,...,n. Nilai preferensi untuk setiap alternatif (Vi) diberikan sebagai:

$$V_i \sum_{j=1}^{n} W_{ij} X_{ij}$$

Keterangan:

Vi = Nilai akhir dari alternative

Wi = Bobot yang telah ditentukan

Rij = Normalisasi matriks

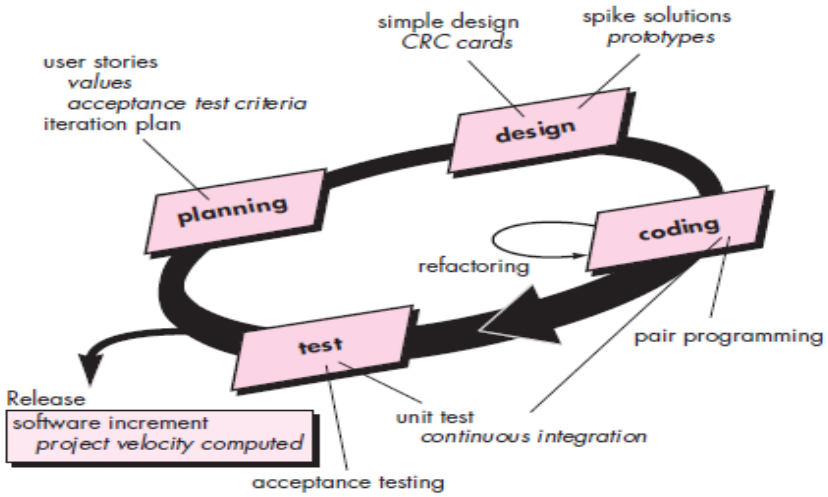
Nilai Vi yang lebih besar mengindikasikan bahwa alternative Ai lebih terpilih.

1.7. Metode Pengembangan Sistem

Agile software engineering merupakan salah satu model proses iteratif yang memberikan suatu alternatif yang layak dari berbagai macam metode konvensional untuk membangun berbagai jenis perangkat lunak dan berbagai macam tipe proyek pengembangan perangkat lunak (Pressman, 2010). *Agile software development* adalah Metode dari beberapa kumpulan prinsip untuk pengembangan *software* di mana persyaratan dan solusi melalui upaya kolaboratif dari antar tim fungsional dan klien sebagai pendukung perencanaan adaptif, perkembangan evolusi, awal pengiriman, dan perbaikan terus-menerus, dan itu mendorong respon yang cepat dan fleksibel untuk dirubah. Prinsip-prinsip ini mendukung definisi dan evolusi dari banyak metode pengembangan perangkat lunak (Bentley and Whitten, 2007).

Agile software development interaksi dan personel lebih penting dari pada proses dan alat, *software* yang berfungsi lebih penting daripada dokumentasi yang lengkap, kolaborasi dengan *client* lebih penting daripada negosiasi kontrak, dan sikap tanggap terhadap perubahan lebih penting daripada mengikuti rencana. Namun demikian, sama seperti model proses yang lain, *agile software development* memiliki kelebihan dan tidak cocok untuk semua jenis proyek, produk, orang dan situasi. *Agile software development* memungkinkan model proses yang toleransi terhadap perubahan kebutuhan sehingga perubahan dapat cepat ditanggapi.

Extreme Programming (XP) adalah metode pengembangan *software* yang cepat, efisien, beresiko rendah, *fleksibel*, terprediksi, *scientific*, dan menyenangkan. *Extreme Programming* (XP) merupakan suatu pendekatan yang paling banyak digunakan untuk pengembangan perangkat lunak cepat (Pressman, 2010). Alasan menggunakan metode XP karena sifat dari aplikasi yang di kembangkan dengan cepat melalui tahapan-tahapan yang ada meliputi : *Planning* (perencanaan), *Design* (perancangan), *Coding* (Pengkodean), dan *Test* (pengujian).



Gambar 2.2. *Extreme Programming* (Pressman, 2010)

Tahapan dalam metode pengembangan sistem *Extreme Programming* yaitu :

1. ***Planning***

   Pada tahap perencanaan ini dimulai dari pengumpulan kebutuhan yang membantu tim teknikal untuk memahami konteks bisnis dari sebuah aplikasi. Selain itu pada tahap ini juga mendefinisikan *output* yang akan dihasilkan, fitur yang dimiliki oleh aplikasi dan fungsi dari aplikasi yang dikembangkan. *Planning* Membentuk *user stories*, menentukan *cost* (Nur Cahya and Suaidah, 2021). Semua *story* segera diimplemetasikan (dalam beberapa minggu) *Story* dengan *value* tertinggi akan dipindahkan dari jadwal dan dimplementasikan pertama. *Story* dengan resiko paling tinggi akan diimplemetasikan lebih dulu. Setelah project pertama di*release* dan di*delivery*, XP team memperhitungkan kecepatan project.

2. ***Design***

   Metode ini menekankan desain aplikasi yang sederhana, untuk mendesain aplikasi dapat menggunakan *Class-Responsibility-Collaborator* (CRC) *cards* yang mengidentifikasi dan mengatur *class* pada *object-oriented*. Design menggunakan CRC card, untuk mengenali dan mengatur *object oriented class* yang sesuai dengan *software increment*.

3. ***Coding***

   Konsep utama dari tahapan pengkodean pada *extreme programming* adalah *pair programming*, melibatkan lebih dari satu orang untuk menyusun kode.

4. ***Test***

   Pada tahapan ini lebih fokus pada pengujian fitur dan fungsionalitas dari aplikasi.

1.8. 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.

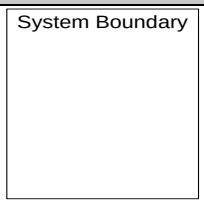
Unified Modeling Language (UML) adalah satu kumpulan konvensi pemodelan yang digunakan untuk menentukan atau menggambarkan sebuah sistem *software* yang terkait dengan objek (Bentley and Whitten, 2007).

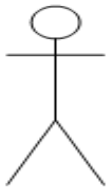
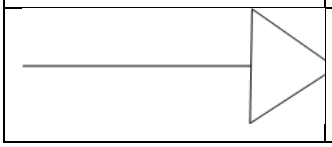
1.8.1. Use Case Diagram

Menurut (Bentley and Whitten, 2007), *use case diagram* adalah diagram yang menggambarkan interaksi antara sistem, eksternal sistem dan *user*. Dengan kata lain, diagram ini menjelaskan sistem tersebut dan bagaimana cara *user* berinteraksi dengan sistem. *Use case diagram* menunjukkan hubungan statis antara *actor* dan *use case* dalam sistem. Mereka menyediakan pandangan awal dari struktur sistem. *Use case* berguna dalam membangun dan mengkomunikasikan pandangan umum sistem. Mereka menyediakan satu titik awal untuk desain, khususnya objek identifikasi dan diagram urutan. Elemen-elemen diagram *use case* adalah *use case*, aktor, kegunaan, dan tanda panah. Aktor diwakili oleh segala sesuatu yang berada diluar sistem, seperti jenis pengguna atau sistem eksternal, yang berinteraksi dengan sistem.

Simbol-simbol yang digunakan dalam *use case diagram* :

Tabel 2.2. Simbol-simbol Use Case Diagram

Simbol	Nama Simbol	Keterangan
	*Boundary* *System*	Digambarkan dengan kotak disekitar *use case*, untuk menggambarkan jangkauan sistem anda *(scope of of your system).* Biasanya digunakan apabila memberikan beberapa alternatif sistem yang dapat dijadikan pilihan.
	*Use-Case*	Gambaran fungsional sistem yang akan di buat, agar pengguna lebih mengerti penggunaan *system*

	*Actor*	*Actor* adalah pengguna sistem. *Actor* tidak terbatas hanya manusia saja, jika sebuah sistem berkomunikasi dengan aplikasi lain dan membutuhkan *input* atau memberikan *output*, maka aplikasi tersebut juga dianggap sebagai *actor*.
	Association	Asosiasi digunakan untuk menghubungkan *actor* dengan *use case*. Asosiasi digambarkan dengan sebuah garis yang menghubungkan antara *actor* dengan *use case.*
	*Extend*	Metode yang hanya berjalan di bawah kondisi tertentu.
	*Include*	Metode yang harus terpenuhi agar sebuah *event* dapat terjadi.
	*Generalization*	Sebuah elemen yang menjadi spesialisasi dari elemen yang lain.

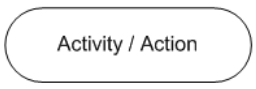
Sumber : (Bentley and Whitten, 2007).

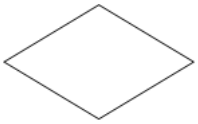
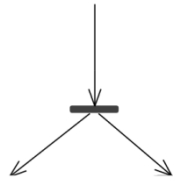
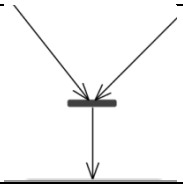
1.8.2. Activity Diagram

(Bentley and Whitten, 2007), *activity diagram* digunakan untuk menggambarkan alur dari proses bisnis atau langkah-langkah *use case* secara berurutan. Diagram ini juga digunakan untuk menggambar *action* (tindakan) yang akan dieksekusikan ketika suatu proses sedang berjalan dan berserta hasil dari proses eksekusi tersebut.

Berikut adalah simbol-simbol yang ada pada diagram aktivitas :

Tabel 2.3. Simbol-simbol Activity Diagram

Simbol	Nama Simbol	Keterangan
	*Initial node*	Awal sebuah proses.
	*Action*	Urutan tindakan membentuk total aktivitas yang ditunjukkan oleh diagram.

	*Flow*	Kebanyakan aliran tidak membutuhkan kata-kata untuk mengidentifikasi mereka kecuali keluar dari keputusan.
	*Decision*	Aliran yang keluar ditandai untuk menunjukkan kondisi.
	*Fork*	bar hitam dengan satu alur masuk dan dua atau lebih alur keluar, aksi di bawah percabangan dapat terjadi dalam urutan apapun atau bahkan secara bersamaan.
	*Join*	bar hitam dengan dua atau lebih alur masuk dan satu alur keluar untuk menyatukan lagi alur aksi yang dipisahkan oleh *fork*.
	*Activity Final*	mewakili akhir proses atau aktivitas.
	*Swimlane*	pembagian *activity diagram* untuk menunjukkan siapa melakukan apa

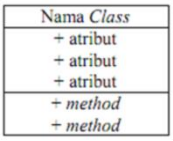
Sumber : (Bentley and Whitten, 2007).

1.8.3. Class Diagram

Class diagram merupakan gambaran grafis dari struktur objek statis dari sebuah sistem yang menunjukkan kelas objek yang tersusun dari hubungan antara kelas- kelas objek yang lain. *Class diagram* digunakan untuk menggambarkan tampilan desain statis sebuah sistem (Bentley and Whitten, 2007).

Tabel 2.4. Simbol Class Diagram

Simbol	Nama Simbol	Keterangan

	*Class*	*Class* terdiri atas 3 bagian yaitu bagian atas, bagian tengah, dan bagian bawah. Bagian atas adalah bagian nama dari *class*. Bagian tengah mendefinisikan *property*/atribut *class*. Bagian akhir mendefinisikan *method-method* dari sebuah *class*.
	*Association*	Sebuah asosiasi merupakan sebuah *relationship* paling umum antara 2 *class* dan dilambangkan oleh sebuah garis yang menghubungkan antara 2 *class*. Garis ini bisa melambangkan tipe-tipe *relationship* dan juga dapat menampilkan hukum-hukum multiplisitas pada sebuah *relationship*.
	*Composition*	Jika sebuah *class* tidak bisa berdiri sendiri dan harus merupakan bagian dari *class* yang lain, maka *class* tersebut memiliki relasi *composition* terhadap *class* tempat dia bergantung tersebut.
	*Dependency*	Kadang kala sebuah *class* menggunakan *class* yang lain. Hal ini disebut *dependency*. Umumnya penggunaan *dependency* digunakan untuk menunjukkan operasi pada suatu *class* yang menggunakan *class* yang lain. Sebuah *dependency* dilambangkan sebagai sebuah panah bertitik-titik.
	*Aggregation*	*Aggregation* mengindikasikan keseluruhan bagian *relationship* dan biasanya disebut sebagai relasi

Sumber : (Bentley and Whitten, 2007).

1.9. Black Box Testing

Blackbox Testing adalah sebuah metode yang dipakai untuk menguji sebuah software tanpa harus memperhatikan detail *software*. Pengujian ini hanya memeriksa nilai keluaran berdasarkan nilai masukan masing-masing. Tidak ada upaya untuk mengetahui kode program apa yang output pakai. Proses *Black Box Testing* dengan cara mencoba program yang telah dibuat dengan mencoba

memasukkan data pada setiap formnya. Pengujian ini diperlukan untuk mengetahui program tersebut berjalan sesuai dengan yang dibutuhkan (Cahya Ningrum *et al.*, 2019).

Menurut (Shadiq, Safei and Loly, 2021), Blackbox Testing adalah metode merupakan pengujian yang berfokus pada spesifikasi fungsional dari perangkat lunak, tester dapat mendefinisikan kumpulan kondisi input dan melakukan pengetesan pada spesifikasi fungsional program. Proses Black Box Testing dengan caramencoba program yang telah dibuat denganmencoba memasukkan data pada setiap formnya. Pengujian dengan metode Blackbox Testing memiliki dua teknik yaitu Equivalence Partitioning dan Boundary Value Analysis

1.10. ISO 25010

Menurut (Hengki, Saputro and Rizan, 2018), Model ini merupakan bagian dari *Software product Quality Requirements and Evaluation* (SQuaRE), dimana model ini berkaitan dengan model kualitas perangkat lunak yang merupakan pengembangan dari model sebelumnya. Pada model ini terdapat beberapa sub-karakteristik tambahan dan beberapa sub-karakteristik yang dipindahkan ke karakteristik lain. Berikut ini merupakan karakteristik atau faktor kualitas internal dan eksternal yang terdapat pada model ISO-25010

Tabel 2.5. ISO 25010

Indikator	Sub – Indikator
Functional Suitability	*1. Functional Completeness* *Functional Correctness* *3. Functional Appropriateness*
Performance Efficiency	*1. Time behaviour* *2. Resource utilization* *3. Capacity*
Compatibility	*1. Co-existence* *2. Interoperability*
Usability	*1. Appropriateness Recognisability* *2. Learnability* *3. Operability* *4. User error protection* *User interface aesthetics6. Accessibility*

Reliability	*1. Maturity* *2. Availability* *3. Fault tolerance* *4. Recoverability*
Security	*1. Confidentiality* *2. Integrity* *3. Non-repudiation* *4. Accountability* *Authenticity*
Maintainability	1. *Modularity* 2. *Reusability* *Analyzability*

Berikut ini merupakan pengertian dari masing-masing faktor dan sub-faktor yang terdapat pada model ISO-25010, antara lain :

1. *Functional Suitability*, seberapa jauh sistem mampu memberikan fungsi yang memenuhi kriteria kebutuhan yang ada (ISO/IEC, 2011).
   a. *Functional Completeness*: Seberapa jauh rangkaian fungsi tersebut mencangkup semua tujuan dan tugas penggunanya.
   b. *Functional Correctness*: Seberapa jauh produk tersebut dapat memberikan/membenarkan hasil yang sesuai dengan tingkatan tertentu sesuai kebutuhan.
   c. *Functional Appropriateness*: Seberapa jauh fungsi dari sistem tersebut memberikan fasilitas dalam penyelesaikan tugas dan tujuan yang sudah ditentukan
2. *Performance Efficiency*, seberapa jauh tingkat kemampuan kapasitas sistem yang relative baik dengan jumlah sumber daya yang digunakan (ISO/IEC, 2011).
   a. *Time Behaviour*: Seberapa jauh respon dan waktu proses dan tingkat hasil dari suatu sistem pada saat fungsi tersebut dijalankan dalam melengkapi persyaratan yang diberikan.
   b. *Resource Utilization*: Seberapa banyak sumber daya yang dipakai pada produk saat fungsi tersebut dijalankan sebagai pemenuh persyaratan.
   c. *Capacity*: Seberapa jauh batas maxsimal atau daya tampung dari produk berdasarkan parameter sistem yang memenuhi persyaratan.

3. *Compatibility*, seberapa jauh sistem tersebut mampu bertukar informasi dalam menjalankan sistem lainnya yang digunakan dan secara bersamaan di berbagai lingkungan perangkat lunak maupun keras yang sama (ISO/IEC, 2011).
    a. *Co-existence*: Seberapa jauh sistem tersebut mampu melakukan fungsi yang lebih efisien dengan cara berbagi lingkungan dan sumber daya antara satu sistem dengan yang lain. Tidak perlu memberikan dampak merugikan untuk sistem tersebut.
    b. *Interoperability*: Seberapa jauh satu, dua atau lebih dan sistem tersebut dapat bertukar data serta memakai informasi yang sudah ditukarkan sebelumnya.
4. *Usability*, yaitu seberapa jauh sistem tersebut mampu digunakan oleh penggunanya dalam menempuh tujuan yang sudah ditentukan dengan efektifitas, efisiensi, dan kepuasan. (ISO/IEC, 2011).
    a. *Appropriateness Recognizability*: Seberapa jauh pemakai mampu mengenali produk tersebut, apakah sudah sesuai dengan keperluan mereka atau kurang sesuai.
    b. *Learnability*: Seberapa jauh produk tersebut agar dapat dipakai oleh pengguna agar dapat mencapai pembelajaran yang sudah ditentukan dalam penggunaan produk. Terutama dibutuhkan pencapaian Kefektivitas, efisiensi, kebebasan dari resiko serta kepuasan dalam penggunaan tertentu.
    c. *Operability*: Seberapa jauh produk tersebutkmemiliki kemampuan yang membuat produk tersebut lebih mudah untuk digunakan.
    d. *User Error Protection*: Seberapa jauh sistem tersebut mampu melindungi pengguna jika terjadi suatu kegagalan.
    e. *User Interface Aesthetics*: Seberapa jauh antarmuka dari pengguna/user mampu untuk menciptakan suatu interaksi yang baik untuk pengguna.
    f. *Accessibility*: Seberapa jauh produk tersebut dapat digunakan oleh berbagai kalangan dengan jangkauan karakteristik dengan mencapai tujuan dalam konteks tertentu.
5. *Reliability*, seberapa jauh sistem dapat menjalankan fungsi yang ditentukan selama batas waktu yang ditentukan (ISO/IEC, 2011).
    a. *Maturity*: Seberapa jauh sistem, produk/kreasi atau unsur yang melengkapi keperluan untuk keunggulan dalam operasi standart.
    b. *Availability*: Seberapa jauh metode, produk/kreasi, atau unsur siap dioperasikan dan bisa diakses ketika dibutuhkan untuk digunakan.

c. *Fault Tolerance*: Seberapa jauh metode, produk/kreasi, atau unsur yang akan beroperasi seperti halnya walaupun ada kegagalan hardware (perangkat keras) atau software (perangkat lunak).
d. *Recoverability*: Seberapa jauh, ketika suatu hal terjadi kelangsungan kendala atau rintangan, suatu sistem atau struktur tersebut dapat mengembalikan data yangKterkena kerusakan langsung dan dapat membentuk kembali ke sistemkyangkdiinginkan.

6. *Security*, seberapa jauh sistem dapat melindungi data dan informasi yang diakses (ISO/IEC, 2011).
   a. *Confidentiality*: Seberapa jauh produk tersebut memastikan jika data tersebut hanya bisa dimasuki sama pihak yang berkuasa saja.
   b. *Integrity*: Seberapa jauh sistem tersebut mencegah terjadinya jalur masuk yang tidak valid, adanya perubahan program ataupun data.
   c. *Non-repudiation*: Seberepa jauh sistem tersebut dapat memebrikan bukti jika suatu tindakan telah terjadi sehingga tidak akan ada penyangkalan pada peristiwa tersebut.
   d. *Authenticity*: Seberapa jauh identitas seseorang dapat dibuktikan sebagai yang diklaim.
   e. *Accountability*: Seberapa jauh tindakan dari suatu identitas dapat dilacak dan ditelusuri.
7. *Maintainability*, seberapa jauh keefektifan dan keefisiensian sistem dapat dirawat (ISO/IEC, 2011).
   a. *Modularity*: Seberapa jauh sistem dapat memperkecil dampak terhadap komponen lain jika terjadi modifikasi pada salah satu komponen.
   b. *Reusability*: Seberapa jauh sistem dapat dipakaisehingga dapat membangun aset lain.
   c. *Analysability*: Seberapa jauh sistem dapat mengkaji dampak perubahan sistem untuk mendiagnosis kekurangan.
   d. *Modifiability*: Seberapa jauh suatu hasil atau metode tersebut bisa berguna secara efektif dan efisien dirubah tanpa ada memperlihatkan kegagalan atau diturunkannya suatu kualitas produk tersebut.
   e. *Testability*: Seberapa jauh nilai level efektivitas dan efisiensi yang sesuai dengan kebutuhan pengujian dan bisa diterapkan oleh suatu sistem, produk atau komponen serta dicek juga dipengujian tersebut apakah nilai itu telah terpenuhi atau belum.

8. *Portability*, seberapa jauh sistem dapat ditransfer atau dipindahkan dari satu perangkat keras atau lunak ke hardware (perangkat keras) atau software (perangkatKlunak)Kyang lain pada lingkungan operasional yang berbeda (ISO/IEC, 2011).
    a. *Adaptability*: Seberapa jauh suatu hasil atau metode tersebut yang efektif maupun efisien dan dilakukan penyesuaian untuk hardware (perangkat keras), software (perangkat lunak) maupun lingkunganKoperasional / penggunaankyang berbeda.
    b. *Installability*: Seberapa jauh perangkat sistem tersebut berhasil diinstal (dipasangkan) dan / atau dihapus (dilepaskan) dalam tindakan tertentu.
    c. *Replaceability*: Seberapa jauh penggunaan sistem tersebut bisa merubah sistem lainnya dan perubahan/penggantian tersebut ditentukan dari tujuan yang sama pada lingkungan yang sama.