BAB II

TINJAUAN PUSTAKA

II.1. Penelitian Terdahulu

Telah ada beberapa penelitian yang dilakukan terkait dengan penerapan Metode KNN, diantaranya adalah :

1. Penelitian yang dilakukan sebelumnya oleh Raymundus Nandy Irawan, Wawan Laksito YS., Sri Siswanti (2016) (ISSN : 1693-1173), dengan judul Sistem Pendukung Keputusan Untuk Menentukan Status Prestasi Siswa Menggunakan Metode K- *Nearest Neighbor.* Pada penelitian ini peneliti menemukan permasalahan yaitu penentuan status prestasi siswa yang dianggap kurang efektif sehingga digunakan sistem pendukung keputusan dengan metode K- *Nearest Neighbor* sebagai metode yang sesuai dalam pengklasifikasian masalah. Dari hasil penelitian ini peneliti mendapatkan hasil dengan metode ini berhasil didapatkan hasil pengurutan siswa berdasarkan nilai terbesar hingga terkecil.
2. Penelitian selanjutnya adalah penelitian dari Ratih Kumalasari Niswatin (2015) (e-ISSN: 2477-8079), yang berjudul sistem pendukung keputusan penempatan jurusan mahasiswa baru menggunakan metode *k-nearest neighbor.* Pada penelitian Sistem pendukung keputusan penempatan jurusan mahasiswa baru pada Universitas Nusantara PGRI Kediri pada penelitian ini akan menggunakan metode klasifikasi *k-nearest neighbor*. Metode *k-nearest neighbor* dipilih karena sesuai dengan data hasil yang diharapkan, dengan

menggunakan algoritma *k-nearest neighbor* maka proses klasifikasi penempatan jurusan mahasiswa baru bisa diperoleh berdasarkan data jurusan mahasiswa angkatan sebelumnya. Hasil penelitian ini menunjukkan bahwa metode *k-nearest neigbor* merupakan metode yang cukup baik dan sesuai digunakan untuk menyelesaikan permasalahan klasifikasi. Syarat utama penggunaan metode *k-nearest neigbor* untuk menyelesaikan permasalahan klasifikasi adalah tersedianya data training yang baik dan akurat, karena pada metode *k-nearest neigbor* hasil klasifikasi diperoleh dengan mengitung kedekatan antara permasalahan baru (*data testing*) dengan permasalahan lama (*data training*) berdasarkan pada kecocokan bobot / nilai dari fitur – fitur yang telah ditentukan.

3. Penelitian sebelumnya yang dilakukan oleh Asahar Johar T, Delfi Yanosma, Kurnia Anggriani (2016) (ISSN 2355-5920), dengan judul Implementasi Metode *K-Nearest Neighbor* (KNN) Dan *Simple Additive Weighting* (SAW) Dalam Pengambilan Keputusan Seleksi Penerimaan Anggota Paskibraka (Studi Kasus : Dinas Pemuda dan Olahraga Provinsi Bengkulu) dimana pada Penelitian ini penulis ingin membangun sebuah sistem pendukung keputusan seleksi penerimaan anggota Paskibraka. Aplikasi yang dibangun menggunakan metode *K-Nearest Neighbor* (KNN) dan *Simple Addittive Weighting* (SAW). Metode *K-Nearest Neighbor* digunakan untuk melakukan klasifikasi peserta yang akan diterima. Metode *Simple Addittive Weighting* digunakan untuk melakukan perangkingan. Aplikasi ini dibangun dengan menggunakan bahasa pemrograman PHP. Metode pengembangan sistem

yang digunakan untuk membangun aplikasi ini adalah model *waterfall* dan *Unified Modelling Language* (UML) sebagai perancangan sistem. Hasil dari aplikasi ini yaitu berupa rekomendasi nama peserta yang lolos dan tidak lolos seleksi berdasarkan hasil perangkingan nilai masing-masing peserta.

4. Penelitian sebelumnya yang dilakukan oleh Diva Kurnianingtyas, Brillian Aristyo Rahardian, Dyan Putri Mahardika, Amalia Kartika A., Dwi Angraeni K. (2017), dengan judul Sistem Pendukung Keputusan Diagnosis Penyakit Sapi Potong Menggunakan *K- Nearest Neighbour (K- NN)* dimana pada penelitian ini penulis mencari tau mengenai Membangun sistem didasarkan dari data pasien yang disimpan ke dalam database. Sistem pendukung keputusan membantu memperkuat dalam mengambil keputusan, tetapi bisa juga tidak mempengaruhi keputusan yang diambil oleh user akan membantu mempercepat penganalisisan jenis penyakit pada hewan ternak dengan memberikan informasi terkait jenis penyakit yang menjangkit hewan ternak dan solusi penanganannya. Penggunaan metode K-NN dalam melakukan prediksi pada sistem pendukung keputusan dengan mencari jarak terpendek antar data. Jarak tersebut akan dievaluasi berdasarkan nilai K pada data latihnya. Berdasarkan pengujiannya yang dilakukan secara manual, sistem menghasilkan data yang baik.
5. Penelitian sebelumnya yang dilakukan oleh Hendri Risman, Didik Nugroho, Yustina Retno WU (2015) (ISSN : 2338-4018), dengan judul Penerapan Metode K-Nearest Neighbor Pada Aplikasi Penentu Penerima Beasiswa Mahasiswa Di Stmik Sinar Nusantara Surakarta dimana pada penelitian ini

penulis ingin membangun sebuah sistem dimana sistem tersebut nantinya dapat berfungsi sebagai aplikasi yang dapat membantu dalam membuat sebuah keputusan alternatif dalam menentukan calon penerima beasiswa dengan menggunakan metode K-Nearest Neighbor. Metode pengumpulan data dengan wawancara dan observasi, metode analisis sistem dengan diagram Context, HIPO, DAD, desain database dengan ERD, implementasi program dengan PHP dan database mysql. Kriteria dalam penentuan calon penerima beasiswa ini yaitu Semester, Indeks Prestasi Kumulatif (IPK), penghasilan orang tua, dan Tanggungan Orang Tua. Dari pengujian yang dilakukan terhadap 22 data sampel yang di jadikan acuan dalam perhitungan k-Nearest Neighbor dalam menghasilkan keputusan diperoleh nilai kekuratan sebesar 90,90% yang mana dalam algoritma k-Nearest Neighbor nilai tersebut termasuk besar karena algoritma k-Nearest Neighbor tidak menggunakan parameter untuk dijadikan acuan melainkan data sampel yang nilainya bervariasi.

II.2. Landasan Teori

II.2.1. Sistem Pendukung Keputusan / *Decision Support System* (DSS)

Sistem Pendukung Keputusan merupakan sistem informasi interaktif yang menyediakan informasi, pemodelan, dan pemanipulasian data. Sistem itu di gunakan untuk membantu pengambilan keputusan dalam situasi yang semiterstruktur dan situasi yang tidak terstruktur, dimana seorang pun tahu secara pasti bagaimana keputusan seharusnya dibuat.

Sistem pendukung keputusan biasanya dibangun untuk mendukung solusi atas suatu masalah atau untuk mengevaluasi suatu peluang. Sistem pendukung keputusan yang seperti itu disebut aplikasi sistem pendukung keputusan. Aplikasi sistem pendukung keputusan digunakan dalam pengambilan keputusan.

Sistem pendukung keputusan tidak dimaksudkan untuk mengotomatisasikan pengambilan keputusan, tetapi memeberikan perangkat interaktif yang memungkinkan pengambil keputusan untuk melakukan berbagai analisis menggunakan model-model yang tersedia. (Kusrini : 2017)

II.2.2. Tujuan *Decision Support System* (DSS)

Tujuan dari sistem pendukung keputusan adalah sebagai berikut :

1. Membantu manajer dalam pengambilan keputusan atas masalah yang terstruktur.
2. Memberikan dukungan atas pertimbangan manajer dan bukannya dimaksudkan untuk menggantikan fungsi manajer.
3. Meningkatkan efektivitas keputusan yang diambil manajer lebih daripada perbaikan efisiensinya.
4. Kecepatan komputasi. Komputer memungkinkan para pengambil keputusan untuk melakukan banyak komputasi secara cepat dengan biaya yang rendah.
5. Peningkatan produktivitas.
6. Dukungan kualitas. Komputer bisa meningkatkan kualitas keputusan yang dibuat.
7. Berdaya saing.

8. Mengatasi keterbatasan kognitif dalam pemrosesan dan penyimpanan. (Kusrini, 2017)

II.2.3. Karakteristik *Decision Support System* (DSS)

Berdasarkan hasil kutipan Kusrini dalam buku karangan Turban yang berjudul *Decision Support System and Intelligent Systems*, karakteristik dari sistem pendukung keputusan adalah sebagai berikut :

1. Sistem pendukung keputusan memberikan dukungan bagi pengambil keputusan pada situasi semi terstruktur dan tak terstruktur dengan memadukan pertimbangan manusia dan informasi terkomputerisasi.
2. Dukungan untuk semua level manajerial, dari eksekutif puncak sampai manajer lini.
3. Dukungan untuk individu dan kelompok.
4. Dukungan untuk keputusan independen dan sekuensial.
5. Dukungan di semua fase proses pengambilan keputusan, yaitu *intelligence, design, choice*, dan *implementation*.
6. Dukungan di berbagai proses dan gaya yang berbeda-beda.
7. Adaptivitas sepanjang waktu.
8. Mudah untuk digunakan *user*.
9. Peningkatan efektivitas dari pengambilan keputusan daripada efisiensi.
10. Kontrol penuh oleh pengambil terhadap semua langkah proses pengambilan keputusan.

11. Pengguna akhir bisa mengembangkan dan memodifikasi sendiri sistem sederhana.
12. Biasanya,model-model digunakan untukmenganalisis situasi pengambilan keputusan.
13. Akses disediakan untuk berbagai sumber daya, format, dan tipe, mulai dari sistem informasi sampai sistem berorientasi objek.
14. Dapat digunakan sebagai *standalone* oleh seorang pengambil keputusan pada satu lokasi atau didistribusikan di suatu organisasi secara keseluruhan dan di beberapa organasasi sepanjang rantai persediaan. (Kusrini ; 2017)

II.2.4. Komponen-komponen *Decision Support System* (DSS)

Komponen *Decision Support System* dapat berupa :

a. *Data Management,*

   Termasuk *database*, yang mengandung data yang relevan untuk pelbagai situasi dan diatur oleh *software* yang disebut *Database Management Systems* (DBMS).

b. *Model Management,*

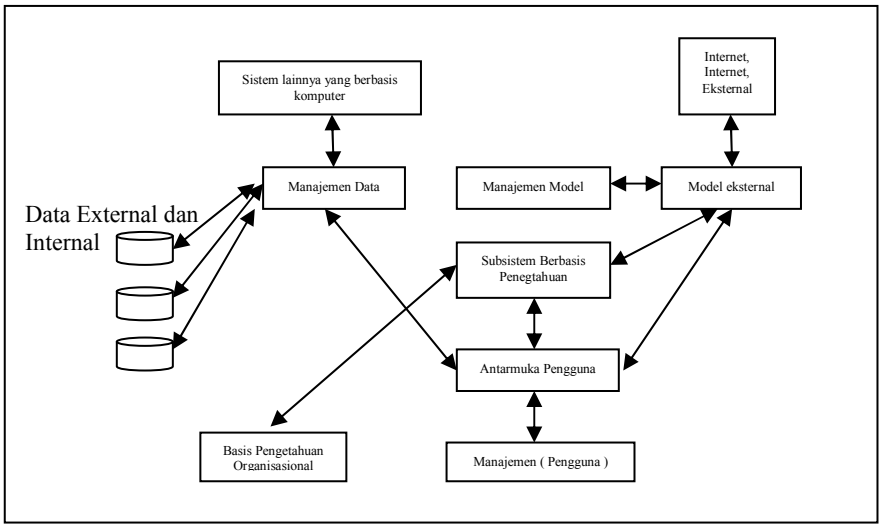
   Melibatkan model *finansial, statistikal, management science,* atau pelbagai model kuantitatif lainnya, sehingga dapat memberikan kesistem suatu kemampuan analitis, dan manajemen *software* yang diperlukan.

c. *Communication (Dialog Subsystem),*

   User dapat berkomunikasi dan memberikan perintah pada *Decision Support System* melalui subsistem ini. Ini berarti menyediakan antarmuka.

d. *Knowledge Management,*

Subsistem optional ini dapat mendukung subsistem lain atau bertindak sebagai komponen yang berdiri sendiri. Gambar II.1 menunjukkan model konseptual *Decision Support System*. (Adil Setiawan dan Surya Darma; 2015).



Gambar II.1. Model Konseptual *Decision Support System* (DSS)
(Sumber : Adil Setiawan dan Surya Darma ; Februari 2015)

II.3. Metode KNN

K-Nearest Neighbor merupakan salah satu metode untuk mengambil keputusan menggunakan pembelajaran terawasi dimana hasil dari data masukan yang baru diklasifikasi berdasarkan terdekat dalam data nilai. Algoritma *K-Nearest Neighbor* (KNN) adalah sebuah metode untuk melakukan klasifikasi terhadap objek yang berdasarkan dari data pembelajaran yang jaraknya paling dekat dengan objek tersebut. KNN merupakan algoritma *supervised learning* dimana hasil dari *query instance* yang baru diklasifikan berdasarkan mayoritas

dari kategori pada algoritma KNN. Dimana kelas yang paling banyak muncul yang nantinya akan menjadi kelas hasil dari klasifikasi. Kedekatan didefinisikan dalam jarak metrik, seperti jarak *Euclidean*. Jarak *Euclidean* dapat dicari dengan menggunakan persamaan 1 berikut ini:

$$d_i = \sqrt{\sum_{i=1}^{p} (x_{2i} - x_{1i})^2} \quad \text{.....(1)}$$

Keterangan :

D : jarak kedekatan

X_1: data *training* / sampel data

X_2: data *testing* / data uji

p: jumlah atribut individu antara 1 s.d. n

f : fungsi *similitary* atribut *i*antara kasus *X* dan kasus *Y*

i : Atribut individu antara 1 sampai dengan n

Langkah-langkah untuk menghitung metode *K-Nearest Neighbor* antara lain :

1. Menentukan parameter *K* (jumlah tetangga paling dekat).
2. Menghitung kuadrat jarak *Euclid* (*queri instance*) masing-masing objek terhadap data sampel yang diberikan menggunakan persamaan 1.
3. Kemudian mengurutkan objek-objek tersebut ke dalam kelompok yang mempunyai jarak *Euclid* terkecil.
4. Mengumpulkan kategori *Y* (Klasifikasi *Nearest Neighbor*), dengan menggunakan kategori *Nearest Neighbor* yang paling mayoritas maka dapat diprediksi nilai *query instance* yang telah dihitung. (*Sumber : Jurnal Pseudocode, Volume III Nomor 2, September 2016, ISSN 2355-5920*).

II.5. *Unified Modelling Language (UML)*

Pada perkembangan teknologi perangkat lunak, diperlukan adanya bahasa yang digunakan untuk memodelkan perangkat lunak yang akan dibuat dan perlu adanya standarisasi agar orang diberbagai negara dapat mengerti pemodelan perangkat lunak. Seperti yang kita ketahui bahwa menyatukan banyak kepala untuk menceritakan sebuah ide dengan tujuan untuk memahami hal yang sama tidaklah mudah, oleh karena itu diperlukan sebuah bahasa pemodelan perangkat lunak yang dapat dimengerti oleh banyak orang.

Banyak orang yang telah membuat bahasa pemodelan pembangunan perangkat lunak sesuai dengan teknologi pemrograman yang berkembang pada saat itu, misalnya yang sempat berkembang dan digunakan oleh banyak pihak adalah *Data Flow Diagram* (DFD) untuk memodelkan perangkat lunak yang menggunakan pemrograman prosedural atau struktural, kemudian juga ada *State Transition Diagram* (STD) yang digunakan untuk memodelkan sistem *real time* (waktu nyata).

Pada perkembangan teknik pemrograman berorientasi objek, muncullah sebuah standarisasi bahasa pemodelan untuk pembangunan perangkat lunak yang dibangun dengan menggunakan teknik pemrograman berorientasi objek, yaitu *Unified Modelling Language* (UML). UML muncul karena adanya kebutuhan pemodelan visual untuk menspesifikasi, menggambarkan, membangun dan dokumentasi dari sistem perangkat lunak.

"UML merupakan bahasa visual untuk pemodelan dan komunikasi mengenai sebuah sistem dengan menggunakan diagram dan teks-teks pendukung"

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. Seperti yang kita ketahui bahwa banyak hal di dunia sistem informasi yang tidak dapat dibakukan, semua tergantung kebutuhan, lingkungan dan konteksnya. Begitu juga dengan perkembangan penggunaan UML bergantung pada *level* abstraksi penggunaannya. Jadi belum tentu pandangan yang berbeda dalam penggunaan UML adalah suatu yang salah, tapi perlu ditelaah dimanakah UML digunakan dan hal apa yang ingin divisualkan. (Shalahuddin, M. dan Rosa A.S, 2018).

II.6. *Use Case Diagram*

Use case atau diagram *use case* merupakan pemodelan untuk melakukan (*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.

Syarat penamaan pada *use case* adalah nama didefinisikan sesimpel mungkin dan dapat dipahami. Ada dua hal utama pada *use case* yaitu pendefinisian apa yang disebut aktor dan *use case.*

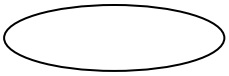
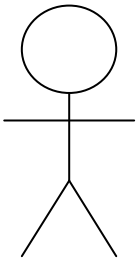
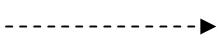
1. Aktor merupakan orang, proses, atau sistem lain yang berinteraksi dengan sistem informasi yang akan dibuat diluar sistem informasi yang akan dibuat

itu sendiri, jadi walaupun simbol dari aktor adalah gambar orang, tapi aktor belum tentu merupakan orang.

2. *Use case* merupakan fungsionalitas yang disediakan sistem sebagai unit-unit yang saling bertukar pesan antar unit atau aktor.

Berikut adalah simbol-simbol yang ada pada diagram *use case* :

Tabel II.1 Simbol-Simbol *Use Case Diagram*

Simbol	Nama	Keterangan
	*Use Case*	Fungsionalitas yang disediakan sistem sebagai unit-unit yang saling bertukar pesan antar unit atau aktor; biasanya dinyatakan dengan menggunakan kata kerja di awal frase nama *use case*.
	*Actor*	Orang, proses, atau sistem lain yang berorientasi 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 frase nama aktor.
	Association	Komunikasi antar aktor dan *use case* yang berpartisipasi pada *use case* atau *use case* memiliki interaksi dengan aktor.
	*Extend*	Relasi *use case* tambahan ke sebuah *use case* dimana *use case* yang ditambahkan dapat berdiri sendiri walau tanpa *use case* tambahan itu; mirip dengan prinsip *inheritance* pada pemrograman berorientasi objek; biasanya *use case* tambahan memiliki

		nama depan yang sama dengan *use case* yang ditambahkan.
	Generalization	Hubungan generalisasi dan spesialisasi (umum-khusus) antara dua buah *use case* dimana fungsi yang satu adalah fungsi yang lebih umum dari lainnya.
<<include>> 	*Include*	Relasi *use case* tambahan ke sebuah *use case* dimana *use case* yang ditambahkan memerlukan *use case* ini untuk menjalankan fungsinya atau sebagai syarat dijalankan *use case* ini.

(Sumber : Shalahuddin, M. dan Rosa A.S, 2018, Rekayasa Perangkat Lunak)

II.7. *Activity Diagram*

Diagram aktivitas atau *activity diagram* menggambarkan *work flow* (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 *actor,* jadi aktivitas yang dapat dilakukan oleh sistem.

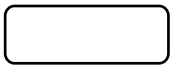
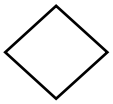
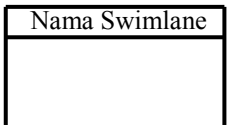
Diagram aktivitas juga banyak digunakan untuk mendefinisikan hal-hal berikut :

1. Rancangan proses bisnis dimana setiap urutan aktivitas yang digambarkan merupakan proses bisnis sistem yang didefinisikan.
2. Urutan atau pengelompokan tampilan dari sistem/ *user interface* dimana setiap aktivitas dianggap memiliki sebuah rancangan antarmuka tampilan.

3. Rancangan pengujian dimana setiap aktivitas dianggap memerlukan sebuah pengujian yang perlu didefinisikan kasus ujinya.
4. Rancangan menu yang ditampilkan pada perangkat lunak.

Berikut adalah simbol-simbol yang ada pada diagram aktivitas :

Tabel II.2 Simbol *Activity Diagram*

Simbol	Nama	Keterangan
	Status Awal	Status awal aktivitas sistem, sebuah diagram aktivitas memiliki sebuah status awal.
	Aktivitas	Aktivitas yang dilakukan sistem, aktivitas biasanya diawali dengan kata kerja.
	Percabangan/ *Decision*	Asosiasi percabangan dimana jika ada pilihan aktivitas lebih dari satu.
	Penggabungan/ *Join*	Asosiasi penggabungan dimana lebih dari satu aktivitas digabungkan menjadi satu.
	Status Akhir	Status akhir yang dilakukan sistem, sebuah diagram aktivitas memiliki sebuah status akhir.
	Swimlane	Memisahkan organisasi bisnis yang bertanggung jawab terhadap aktivitas yang terjadi.

(Sumber : Shalahuddin, M. dan Rosa A.S, 2018, Rekayasa Perangkat Lunak)

II.8. *Class Diagram*

Class diagram atau diagram kelas Merupakan hubungan antar kelas dan penjelasan detail tiap-tiap kelas di dalam model desain dari suatu sistem, juga memperlihatkan aturan-aturan dan tanggung jawab entitas yang menentukan perilaku sistem. *Class Diagram* juga menunjukkan atribut-atribut dan operasi-operasi dari sebuah kelas dan *constraint* yang berhubungan dengan objek yang dikoneksikan. *Class Diagram* secara khas meliputi : Kelas (*Class*), Relasi *Assosiations*, *Generalitation* dan *Aggregation*, attribut (*Attributes*), operasi (*operation/method*) dan *visibility*, tingkat akses objek eksternal kepada suatu operasi atau attribut. Hubungan antar kelas mempunyai keterangan yang disebut dengan *Multiplicity* atau *Cardinality. (Sumber :Jurnal Khatulistiwa Informatika, Vol. IV, No. 2 Desember 2016)*

Tabel II.3. *Multiplicity Class Diagram*

Multiplicity	**Penjelasan**
1	Satu dan hanya satu
0..*	Boleh tidak ada atau 1 atau lebih
1..*	1 atau lebih
0..1	Boleh tidak ada, maksimal 1
n..n	Batasan antara. Contoh 2..4 mempunyai arti minimal 2 maksimal 4

(Sumber :Jurnal Khatulistiwa Informatika, Vol. IV, No. 2 Desember 2016)

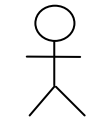
II.9. *Sequence Diagram*

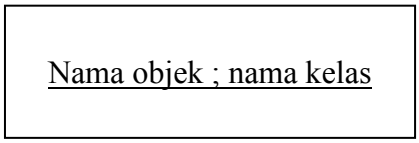
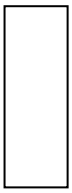
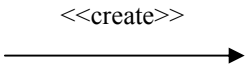
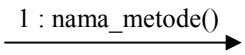
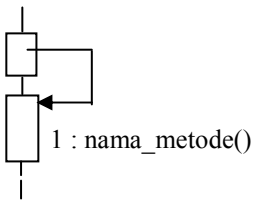
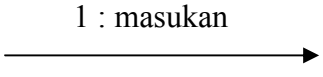
Diagram sekuen menggambarkan kelakuan objek pada *use case* dengan mendeskripsikan waktu hidup objek dan *message* yang dikirmkan dan diterima antar objek. Oleh karean itu untuk menggambar diagram sekuen maka harus diketahui objek-objek yang terlibat dalam sebuah *use case* beserta metode-metode yang dimilik kelas yang diinstansiasi menjadi objek itu. Membuat diagram sekuen juga dibutuhkan untuk melihat scenario yang ada pada *use case*.

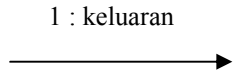
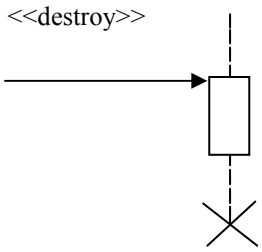
Banyaknya diagram sekuen yang harus digambar adalah minimal sebanyak pendefenisian *use case* yang memiliki proses sendiri atau yang penting semua *use case* yang telah didefenisikan interaksi jalannya pesan sudah dicakup pada diagram sekuen sehingga semakin banyak *use case* yang didefenisikan maka diagram sekuen yang harus dibuat juga semakin banyak.

Berikut adalah simbol-simbol yang ada pada diagram sekuen:

Tabel II.3. Simbol *Sequence Diagram*

Simbol	Deskripsi
 Nama aktor Atau Nama aktor Tanpa waktu aktif	Orang, proses, atau sistem lain yanng 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 frase nama actor
Garis hidup / *lifeline*	Menyatakan kehidupan suatu objek

Objek  	Menyatakan objek yang berinteraksi pesan
Waktu aktif 	Menyatakan objek dalam keadaan aktif dan berinteraksi, semua yang terhubung dengan waktu aktif ini adalah sebuah tahapan yang dilakukan didalamnya.
Pesan tipe create  	Menyatakan suatu objek membuat objek yang lain, arah panah mengarah pada objek yang dibuat.
Pesan tipe call  	Menyatakan suatu objek memanggil operasi/metode yang ada pada objek lain atau dirinya sendiri,  Arah panah mengarah pada objek yang memiliki operasi/metode, karena ini memanggil operasi/metode maka operasi/metode yanng dipanggil harus ada pada diagram kelas sesuai dengan kelas objek yang berinteraksi.
Pesan tipe send  	Menyatakan bahwa suatu objek mengirimkan data/masukan/informasi ke objek lainnya, arah panah mengarah pada objek yang dikirimi.

Pesan tipe return 	Menyatakan bahwa suatu objek yang telah menjalankan suatu operasi atau metode menghasilkan suatu kembalian ke objek tertentu, arah panah mengarah pada objek yang menerima kembalian
Pesan tipe destroy 	Menyatakan suatu objek mengakhiri hidup objek yang lain, arah panah mengarah pada objek yang diakhri, sebaiknya jika ada create maka ada destroy.

(Sumber : Shalahuddin, M. dan Rosa A.S, 2018:165-167, Rekayasa Perangkat Lunak)

II.10. *Visual Basic 2010*

Visual Studio 2010 pada dasarnya adalah sebuah bahasa pemrograman komputer. Dimana pengertian dari bahasa pemrograman itu adalah perintah-perintah atau instruksi yang dimengerti oleh komputer untuk melakukan tugas-tugas tertentu. Visual Studio 2010 (yang sering juga disebut dengan VB .Net 2010) selain disebut dengan bahasa pemrograman, juga sering disebut sebagai sarana (tool) untuk menghasilkan program-progam aplikasi berbasiskan windows.

Visual basic adalah sebuah bahasa pemrograman yang berpusat pada *object* (*Object Oriented Programming*) digunakan dalam pembuatan aplikasi *Windows* yang berbasis *Graphical User Interface*, hal ini menjadikan *Visual Basic* menjadi bahasa pemrograman yang wajib diketahui dan dikuasai oleh setiap programmer. Beberapa karakteristik obyek tidak dapat dilakukan oleh *Visual*

Basic misalnya seperti *Inheritance* tidak bisa module dan *Polymorphism* secara terbatas bisa dilakukan dengan deklarasi *class module* yang mempunyai *Interface* tertentu.

Beberapa kemampuan atau manfaat dari Visual Studio 2010 diantaranya seperti :

a. Untuk membuat program aplikasi berbasiskan windows.
b. Untuk membuat objek-objek pembantu program seperti, misalnya : kontrol *ActiveX, file Help*, aplikasi Internet dan sebagainya.
c. Menguji program (debugging) dan menghasilkan program berakhiran EXE yang bersifat executable atau dapat langsung dijalankan.

Visual Studio 2010 adalah bahasa yang cukup mudah untuk dipelajari. Bagi *programmer* pemula yang baru ingin belajar program, lingkungan Visual Studio dapat membantu membuat program dalam sekejap mata. Sedang bagi programer tingkat lanjut, kemampuan yang besar dapat digunakan untuk membuat program-program yang kompleks, misalnya lingkungan *net-working* atau *client server*.. Bahasa Visual Studio cukup sederhana dan menggunakan kata-kata bahasa Inggris yang umum digunakan. Kita tidak perlu lagi menghafalkan sintaks-sintaks maupun format-format bahasa yang bermacam-macam, di dalam Visual Basic semuanya sudah disediakan dalam pilihan-pilihan yang tinggal diambil sesuai dengan kebutuhan. Selain itu, sarana pengembangannya yang bersifat visual memudahkan kita untuk mengembangkan aplikasi berbasiskan Windows, bersifat mouse-driven (digerakkan dengan mouse) dan berdaya guna tinggi. (Ninuk Wiliani dan Syadid Zambi, 2017 ; ISSN : 2252-7354).