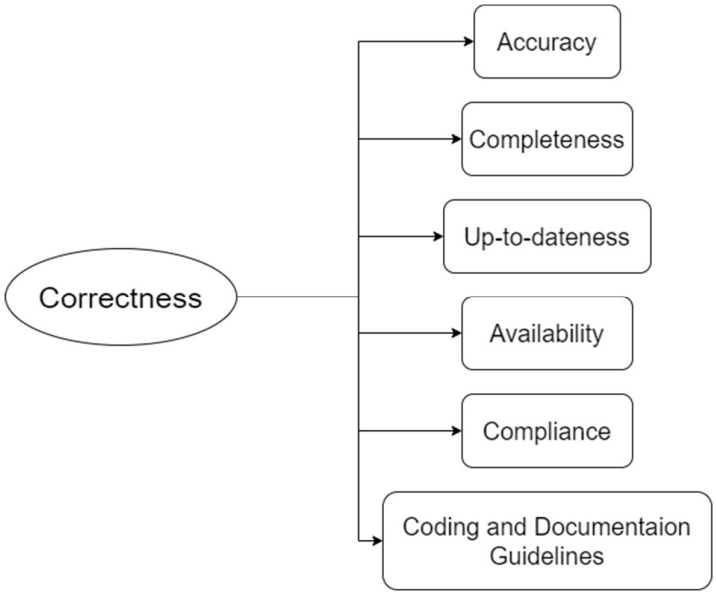
BAB 2
LANDASAN TEORI

2.1 Software Quality Assurance

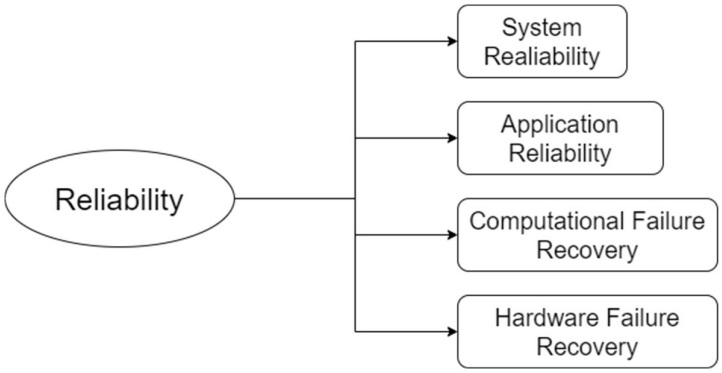
Software Quality Assurance merupakan sebuah perantara bagi para developer untuk bisa menilai kualitas sebuah Software yang telah dibangun. Software Quality terdefinisi sebagai sebuah tolak ukur para developer dalam membuat Software yang baik dan benar. Dalam artian baik dan benar, tentu saja Software tersebut dibuat dengan memperhatikan segala macam faktor kualitas. Faktor-faktor kualitas tersebut dapat didefinisikan oleh banyak Model Kualitas Software, salah satunya adalah Model McCall. McCall sendiri membagi kualitas software menjadi 3 bagian kategori utama, yaitu "Product Revision", "Product Transtion" dan "Product Operation". Terdapat beberapa faktor yang dikategorikan berdasarkan 3 faktor utama, seperti pada gambar di bawah ini.

Gambar 1 Kategori dan Faktor Model McCall

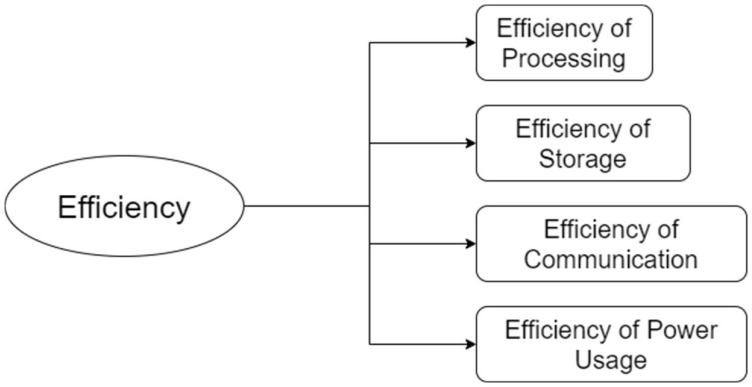
Dalam masing-masing Faktor juga terdapat beberapa sub-faktor yang dapat memfokuskan Software Quality menjadi bagian yang lebih detail, seperti yang di tampilkan pada table di bawah ini :



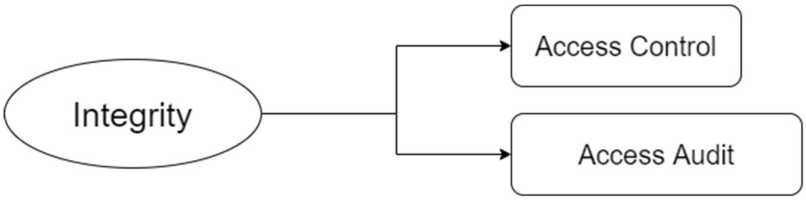
Gambar 2 Correctness



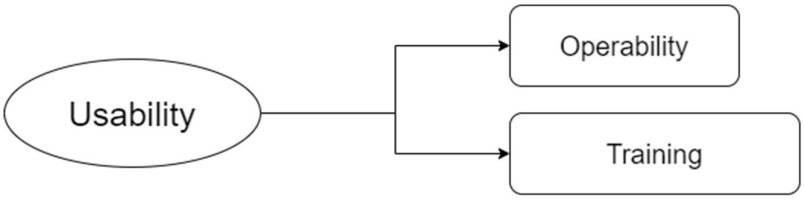
Gambar 3 Reliability



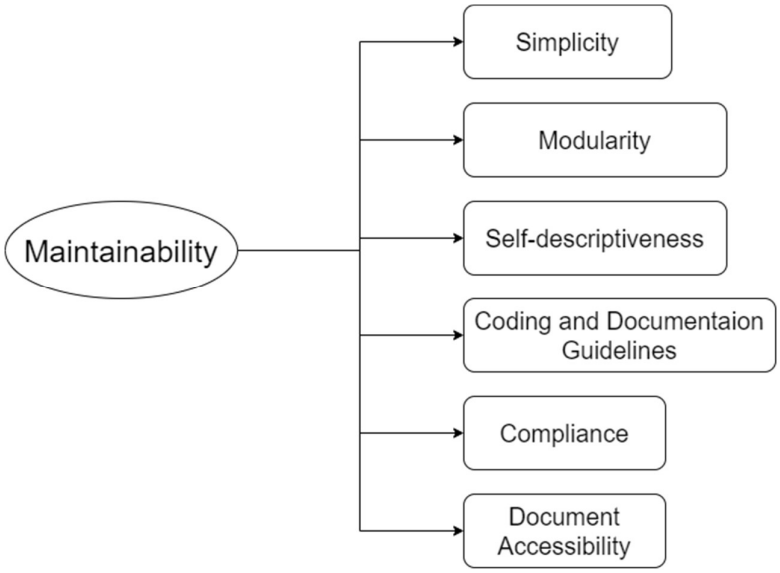
Gambar 4 Efficieny



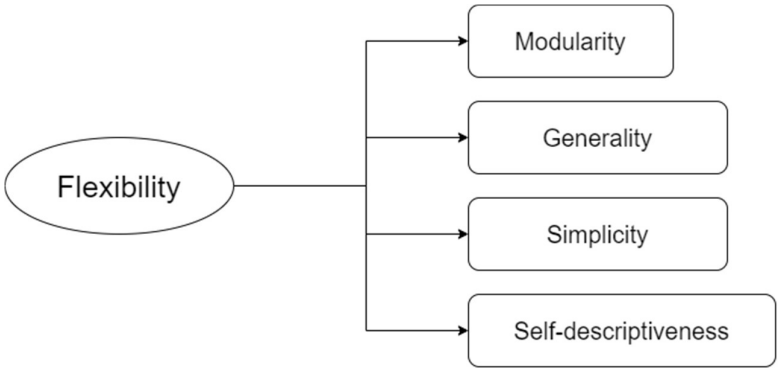
Gambar 5 Integrity



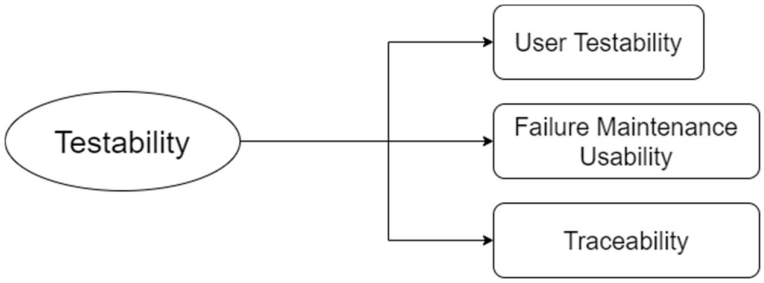
Gambar 6 Usability



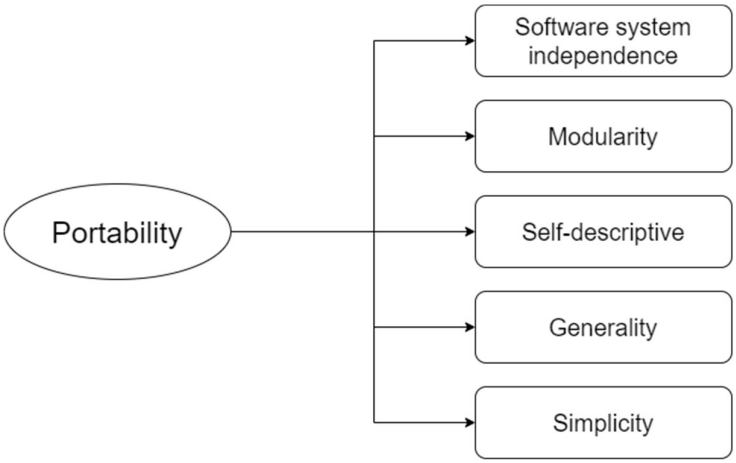
Gambar 7 Maintainabilty



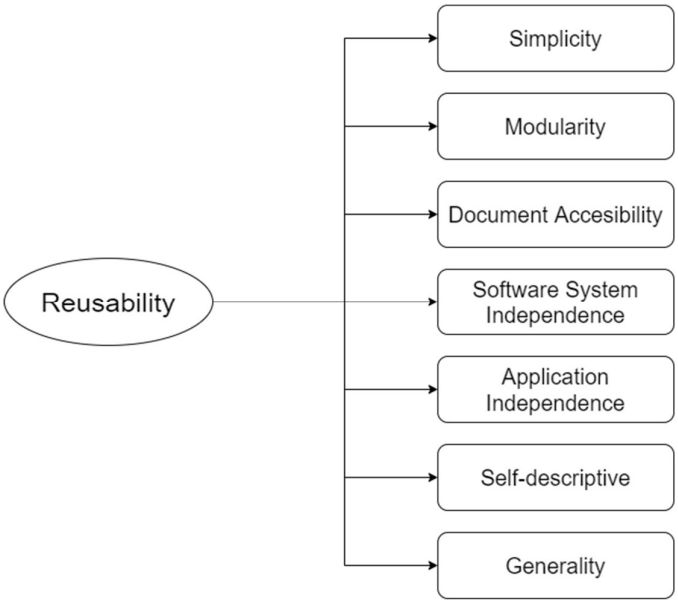
Gambar 8 Flexibility



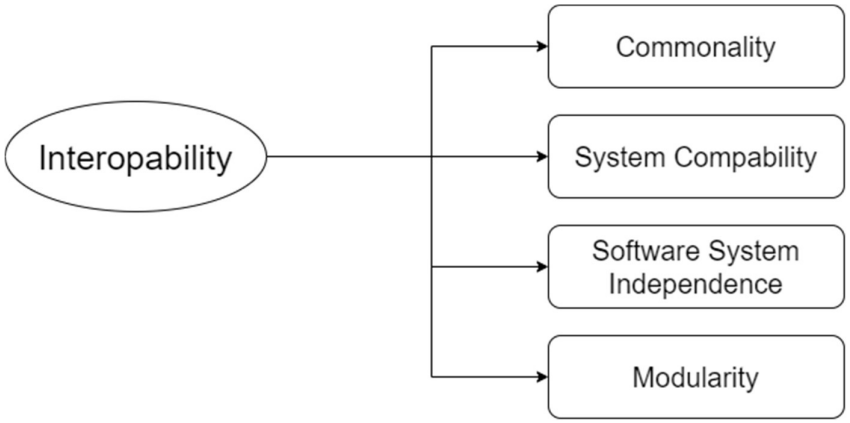
Gambar 9 Testability



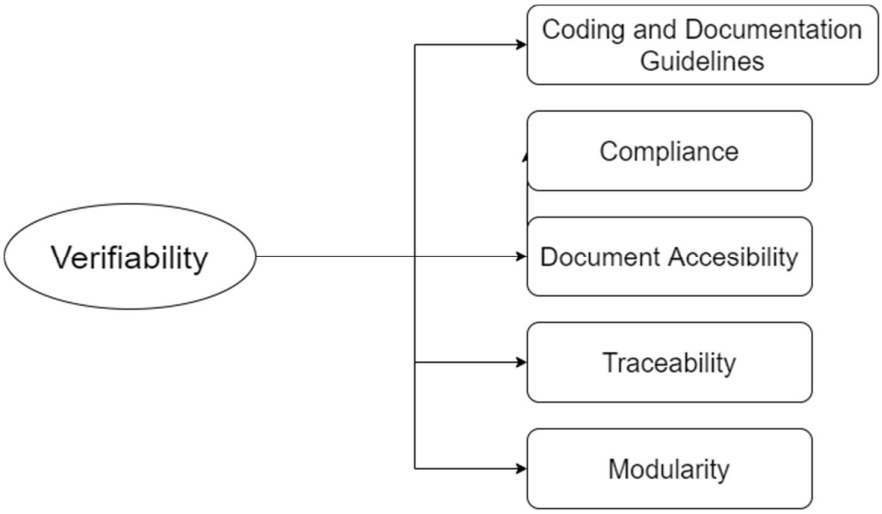
Gambar 10 Portability



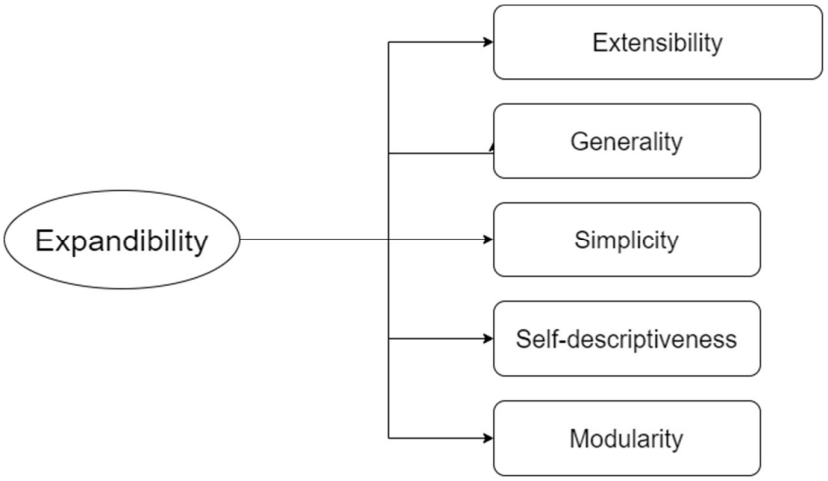
Gambar 11 Reusability



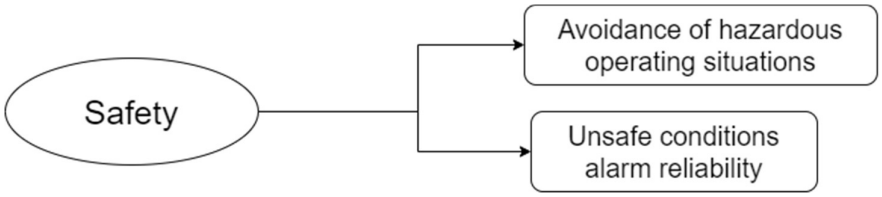
Gambar 12 Interopability



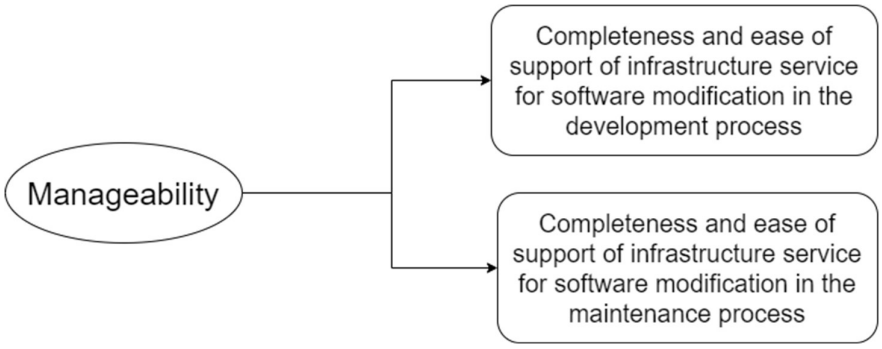
Gambar 13 Veriability



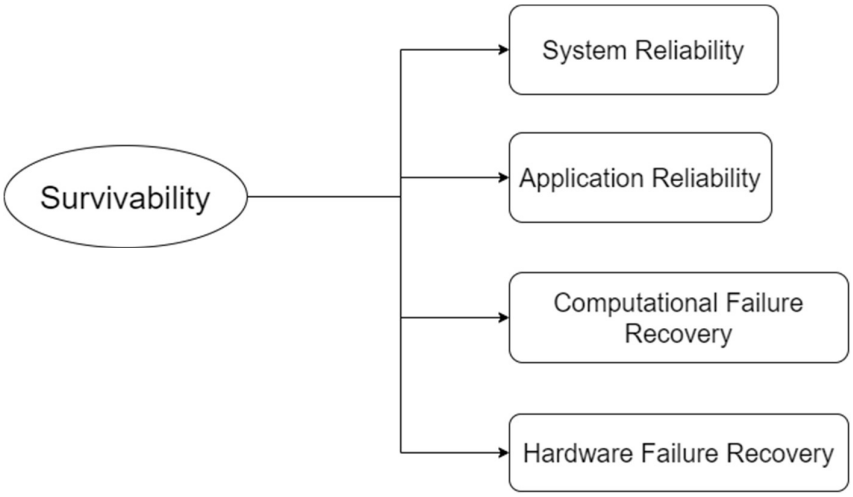
Gambar 14 Expandibility



Gambar 15 Safety



Gambar 16 Manageability

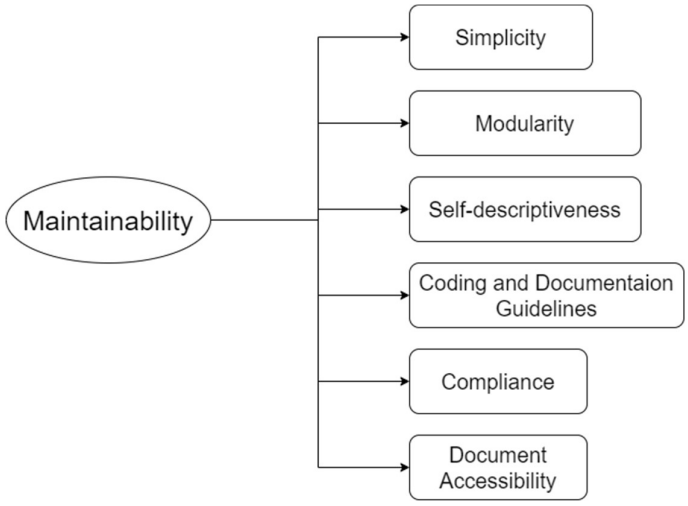


Gambar 17 Survivability

2.2 Software Maintainability

Software Maintainability merupakan salah satu attribute penting kualitas software, dimana attribute software ini dikaitkan erat dengan SDLC. Software Maintainability tentu saja dikaitkan dengan pengembangan dengan improvisasi performance atau mungkin adanya adaptasi software dengan lingkungan baru. Salat satu sub-faktor yang diangkat adalah simplicity, dengan adanya improvisasi dan adaptasi, tentu para developer harus menjaga konsistensi simplicity karena akan sangat mempengaruhi kualitas software yang sebelumnya sudah dibangun.

Seperti yang dijelaskan oleh gambar berikut :



Gambar 18 Maintainability

2.3 Maintainability Indeks

Maintainability Indeks merupakan sebuah software metric yang dapat mengukur seberapa mudahnya source code untuk dimaintain[5]. Maintainability Indeks terstruktur oleh Cyclomatic Complexity dan Halstead Volume. Terdapat 3 formula dasar yang sering digunakan pada Maintainability Indeks, yaitu :

the original formula:

$$MI = 171 - 5.2 \ln V - 0.23 G - 16.2 \ln L$$

Formula yang digunakan oleh SEI:

$$MI = 171 - 5.2 \log_2 V - 0.23 G - 16.2 \log_2 L + 50 \sin(\sqrt{2.4C})$$

Formula yang digunakan oleh Visual Studio:

$$MI = \max [0,100 * (171 - 5.2 \ln V - 0.23 G - 16.2 \ln L) / 171]$$

Dimana :

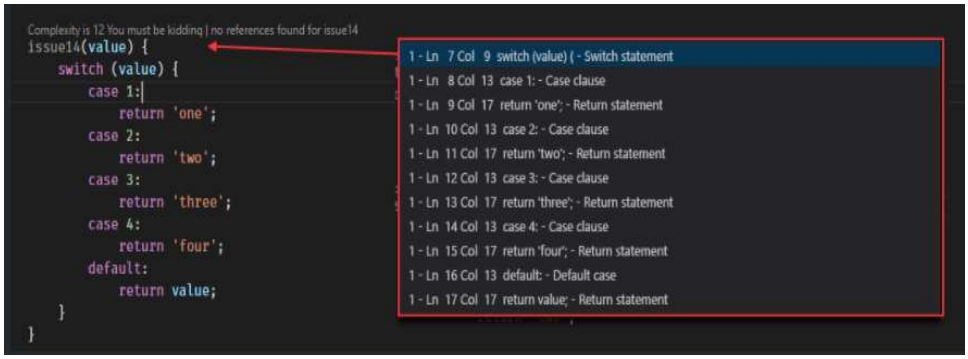
- V adalah Halstead Volume
- G adalah total Cyclomatic Complexity
- L adalah nomor dari Source Lines of Code (SLOC)

- C adalah nilai persen dari comment lines

2.4 CodeMetrics

CodeMetrics adalah sebuah plugin/extensi pada IDE Visual Studio Code yang berfungsi sebagai analisis yang akan menilai kompleksitas sebuah Code Javascript.

Serta dengan berikut sebagai contoh gambar hasil analisis :



Gambar 19 Contoh Analisa CodeMetrics

2.5 Cyclomatic Complexity

Cylcomatic Complexity bertanggung jawab pada jumlah “decisions” dari sebuah blok kode. Cyclomatic Complexity merupakan sebuah metrics pada perhitungan complexity program[7]. Cyclomatic Complexity menggunakan formula sebagai berikut :

$$M = E - N + 2P$$

Note :

M = Cyclomatic Complexity

E = Jumlah edge pada graf kendali

N = Jumlah node(jalur) pada graf kendali

P = Jumlah komponen yang terhubung

2.6 Halstead Metrics

Halstead Metrics merupakan salah satu metrics yang mengukur kompleksitas dengan cara menghitung jumlah setiap operator dan operan yang terdapat pada sebuah Code. Halstead Metrics pada umumnya cukup sulit untuk dikalkulasikan, karena tidak ada kepastian tentang penjelasan mengenai subjek pasti operand dan operator di berbagai bahasa pemrograman, seperti Java, Python, C# dan lain- lain[6]. Terdapat beberapa metric dalam Halstead metrics, yaitu :

- **Program length**, total munculnya operator serta operand dengan estimasi program length yaitu

$$N^* = n1 \log_2 n1 + n2 \log_2 n2$$

- **Halstead Vocabulay**, total jumlah dari operator dan operand yang unik dengan $n = n1 + n2$

- **Program Volume**, merupakan referensi dari ukuran program (dalam ukuran bits) dengan $V = Size * (\log_2 vocabulary) = N^* \log_2 (n)$

- **Potential Minimum Volume**, merupakan kemungkinan nilai terkecil dari volume yang bisa dihasilkan oleh Code, dengan

$$V^* = (2 + n2^*) * \log_2(2 + n2^*)$$

- **Program Level**, tingkatan yang ditentukan berdasarkan bahasa pemrograman, semakin tinggi level bahasanya maka akan semakin sedikit usaha untuk mengembangkan program dengan bahasa tersebut,

$$L = V^* / V$$

2.7 Clean Code

Clean Code merupakan sebuah prinsip dalam penulisan sebuah code yang baik, benar, dan rapih[9]. Struktur code yang mudah dipahami, mudah dikembangkan adalah target utama dari prinsip Clean Code ini. Clean code memiliki beberapa kategori yang bisa dijadikan sebagai pedoman untuk

membandingkan antara Bad Code dan Good Code, diantaranya adalah sebagai berikut :

A. Meaningful Names

Dalam merangkai Code-code, tentu perlu diketahui struktur code sampai yang terkecil seperti data, maupun objects. Dengan penamaan yang baik, setiap developer dapat mencari apapun dengan mudah.

Bad Code	Good Code
Var data1; Var data2;	Var data_from_api; Var data_from_db;

Non-onPoint name pada Pikobar	onPoint name

export default {
props: {
to: {
type: String,
default: '#'
},
}
 	
BarStatHarianAkumulatifV2/index.vue
presetRanges: {
all () {
return {
label: 'Semua Waktu',
active: true,
dateRange: {
start: new Date('2020-03-01'),
	export default {
	props: {
	to_link: {
	type: String,
	default: '#'
	},
	}
	
	BarStatHarianAkumulatifV2/index.vue
	presetDateRanges: {
	all () {
	return {
	label: 'Semua Waktu',
	active: true,
	dateRange: {
	start: new Date('2020-03-01'),

end: new Date() } } },	end: new Date() } } },

Menggunakan nama yang onpoint pada variable akan lebih memudahkan dalam perubahan yang akan dilakukan.

Confusing Code pada Pikobar	Good Code

MapFaskes/index.vue
if (feature.properties.rujukan === true) {
icon = self.iconRSRujukan
} else if
(feature.properties.tipe_faskes === 1) {
icon = self.iconRSLini
} else {
icon = self.iconPuskesmas
}
	MapFaskes/index.vue
	if (feature.properties.rujukan === true) {
	icon = self.iconRSRujukan
	} else if
	(feature.properties.tipe_faskes ===
	‘Rumah sakit”) {
	icon = self.iconRSLini
	} else {
	icon = self.iconPuskesmas
	}

else if (self.filter.lini2 && feature.properties.tipe_faskes === 1 &&
feature.properties.rujukan === null) {
return feature
}
 	
Variable if condition pada tipe_faskes dapat membuat kebingungan developer baru tanpa adanya komentar.	

B. Functions

Kurangi fungsi yang tidak memiliki tujuan jelas, serta ambiguitas.

Bad Code

```
Function CreatePost = (user, post) => {
If (user.member_status === 2) {
    Postdata(user.id, post)
}
}
```

Good Code

```
Function CreatePost = (user, post) => {
If (user.status == 'active') {
    Postdata(user.id, post)
}
}
```

C. Comments

"Buatlah comments, jika itu diperlukan. Jika sudah jelas, tak perlu membuang waktu", merupakan kalimat yang disebutkan dibuku Robert C. Martin

Bad Code

```
// Variable akan diterima jika api memiliki authentication yang
    valid
```

```
Const data = axios.get(https://Api.com/data)
```

Good Code

```
Const data = await axios. get(https://Api.com/data)
```

Masih ada beberapa kriteria Bad Code dan Good Code yang akan dilakukan pada tahap Pemetaan Bad dan Good Code.

2.8 Javascript

JavaScript merupakan sebuah scripting language yang biasa digunakan untuk pembangunan sebuah dynamic website. JavaScript banyak digunakan karena relasi yang sangat besar dengan HTML dan CSS. JavaScript memiliki banyak fungsi pada basis Website, namun javascript hanya bisa dijalan kan dengan server-side (Node.JS).

2.9 Vue JS

Vue JS merupakan salah satu framework yang dikembangkan dengan bahasa pemrograman javascript. Vue adalah progressive framework yang digunakan untuk membangun user interface dengan sifat yang adoptable, serta mudah dalam proses pengembangan. Vue Js juga memiliki supporting libraries yang dapat lebih memudahkan developer untuk membuat sebuah Single-Page Application dan juga terdapat modern tools yang bisa menambah pengalaman yang menarik.