BAB II
LANDASAN TEORI

2.1. Tinjauan Pustaka

Beberapa penelitian yang berkaitan dengan penerapan sistem pendukung bantuan oprasional pendidikan dalam menentukan kelayakan berdasarkan jurnal penelitian terlihat pada Tabel 2.1:

Tabel 2.1 Tinjauan Pustaka

No Literatur	Penulis	Metode	Judul	Hasil	Penjelasan Sistem	Perbedaan
Literatur 01	(Wahyuni and Santosa, 2019)	*Forward Chaining*	Implementasi Metode Forward Chaining Untuk Mendeteksi Penyakit Pada Tanaman Kelengkeng	1. System pendeteksian penyakit tanaman kelengkeng ini mudah digunakan karena berbasis web. 2. Memberikan pengetahuan bagi masyarakat yang sedang membudidayakan tanaman kelengkeng agar mudah mendeteksi penyakit tanaman kelengke	Sistem yang dibangun berbasis web yang dikelola oleh admin	Menggunakan metode pengembangan sistem *extreme programming* dan pengujian sistem ISO 25010

No Literatur	Penulis	Metode	Judul	Hasil	Penjelasan Sistem	Perbedaan
				ng dan mampu mencari solusinya. 3. Ke depannya system ini diharapkan mampu memberikan gambaran penyakit tanaman kelengkeng, sekaligus memberikan solusi guna mencegah penyakit tersebut.		
Literatur 02	(Ahmad and Iskandar, 2020)	*Forward Chaining*	Metode *Forward Chaining* untuk Deteksi Penyakit Pada Tanaman Kentang	Penggunaan sistem pakar akan sangat membantu dalam menyampaikan informasi kepada orang lain sesuai dengan bidang permasalahan yang	Pada sistem yang dibangun terdapat halaman diagnosa penyakit pada tanaman kentang akan disajikan pertanyaan-pertanyaan kepada user (para	Membahas penyakit pada tanaman kelengkeng, metode pengembangan sistem *extreme programming* dan pengujian sistem ISO 25010 serta sistem dapat diakses oleh masyarakat untuk

No Literatur	Penulis	Metode	Judul	Hasil	Penjelasan Sistem	Perbedaan
				ada. Sistem pakar akan menjadi asisten yang tepat bagi pakar dalam memberikan informasi dan solusi untuk penanganan suatu masalah tertentu. Forward chaining yang merupakan salah satu metode di dalam sistem pakar menjadi metode yang tepat untuk digunakan dengan menerapkan aturan penalaran ke depan, dimana fakta nya merupakan gejala-gejala yang ada yang	petani). User akan memberikan jawaban "Ya" dan "Tidak". Jika yang dijawab oleh user adalah "Ya" dari pertanyaan atau fakta yang ditanyakan oleh aplikasi (sistem pakar) maka aplikasi (sistem pakar) akan memberikan kesimpulan dari aturan kaidah produksi yang telah ditentukan. Jika yang dijawab oleh user adalah "Tidak" dari pertanyaan atau fakta yang ditanyakan oleh	melakukan prediksi penyakit tanaman kelengkeng

No Literatur	Penulis	Metode	Judul	Hasil	Penjelasan Sistem	Perbedaan
				dikombinasikan dengan kaidah produksi menggunakan premis dan konsekuen dalam bentuk implikasi (IF-THEN) yang akan mendapatkan kesimpulan yang tepat	sistem maka sistem akan melanjutkan dengan memberikan pertanyaan atau fakta yang lain yang sesuai dengan aturan kaidah produksi yang telah ditentukan sehingga mendapatkan kesimpulan.	
Literatur 03	(Honggowibowo, 2019)	*Forward* Dan *Backward Chaining*	Sistem Pakar Diagnosa Penyakit Tanaman Padi Berbasis Web Dengan *Forward* Dan *Backward Chaining*	Sistem pakar diagnosa penyakit tanaman padi berbasis web yang telah dikembangkan mempunyai keunggulan dalam kemudahan akses dan kemudahan pemakaian. Dengan fitur yang	Pada penelitian ini dirancang sistem pakar berbasis web menggunakan basis aturan (rule based reasoning) dengan metode inferensi forward chaining dan backward chaining	Membahas penyakit pada tanaman kelengkeng, metode pengembangan sistem *extreme programming* dan pengujian sistem ISO 25010

No Literatur	Penulis	Metode	Judul	Hasil	Penjelasan Sistem	Perbedaan
				berbasis web yang dimiliki, sistem pakar untuk diagnosa penyakit tanaman padi yang telah dibangun dapat digunakan sebagai alat bantu untuk diagnosa penyakit tanaman padi dan dapat diakses oleh petani dimanapun juga untuk mengatasi persoalan keterbatasan jumlah pakar pertanian dalam membantu petani mendiagnosa penyakit tanaman padi.	yang dimaksudkan untuk membantu petani dalam mendiagnosa penyakit tanaman padi. Sistem	
Literatur 04	(Novianti, Gunawan and Sukerti, 2020)	*Forward Chaining*	Implementasi Forward Chaining Untuk Mendiagnosi	1. Sistem pakar diagnosa penyakit tanaman kopi	sistem pakar untuk mendiagnosis penyakit	Membahas penyakit pada tanaman kelengkeng, metode

No Literatur	Penulis	Metode	Judul	Hasil	Penjelasan Sistem	Perbedaan
			s Penyakit Tanaman Kopi	Arabika dapat menghasilkan diagnosa jenis penyakit tanaman kopi Arabika yang sesuai berdasarkan aturan-aturan yang telah dibuat dengan menerapkan metode forward chaining. 2. Sistem pakar yang telah dibuat dapat digunakan untuk mendiagnosa jenis penyakit tanaman kopi Arabika berdasarkan gejala yang terlihat dan menghasilkan solusi tindakan berdasarkan hasil	tanaman kopi dibangun berbasis web dan basis data yang digunakan adalah MySQL dan PhpMyAdmin. Sistem yang dihasilkan dapat secara langsung digunakan untuk mendiagnosis penyakit pada tanaman kopi. Pengguna memilih gejala yang terjadi pada tanaman kopi nya dan mengisi checkbox pilih gejala lalu memproses diagnosis	pengembangan sistem *extreme programming* dan pengujian sistem ISO 25010 serta sistem dapat diakses oleh masyarakat untuk melakukan prediksi penyakit tanaman kelengkeng

No Literatur	Penulis	Metode	Judul	Hasil	Penjelasan Sistem	Perbedaan
				diagnosa penyakitnya. 3. Pengujian sistem menggunakan metode pengujian black box testing dan didapatkan hasil sesuai untuk semua fungsi yang diharapkan.		
Literatur 05	(Noviani, Prambudi and Mulyadi, 2020)	*Backward Chaining*	Sistem Pakar Diagnosis Penyakit Pada Tanaman Pepaya Menggunakan Metode *Backward Chaining* Berbasis Web	sistem pakar dengan menggunakan metode Backward Chaining dalam penelitian ini dapat membantu petani dalam melakukan diagnosa awal terhadap suatu penyakit tanaman pepaya dan memberikan solusi	Pembuatan Sistem Pakar ini mampu membuktikan kebenaran suatu penyakit papaya dan memberi solusi dalam menangani penyakit pepaya tersebut. Pembuatan aplikasi sistem pakar diagnosa penyakit	Membahas penyakit pada tanaman kelengkeng, metode pengembangan sistem *extreme programming* dan pengujian sistem ISO 25010 serta sistem dapat diakses oleh masyarakat untuk melakukan prediksi penyakit tanaman kelengkeng

No Literatur	Penulis	Metode	Judul	Hasil	Penjelasan Sistem	Perbedaan
				untuk mengatasi penyakit pada tanaman pepaya tersebut. Hasil	pada tanaman pepaya berbasis web mengguna kan bahasa pemrogra man PHP(Hype rtext Pre-processor) dan Database MY Structured Query Language (MYSQL). Kata	

2.2. Tanaman Kelengkeng

Tanaman Kelengkeng atau *Dimocarpuslongan* merupakan salah satu tanaman buah yang banyak dikembangkan oleh masyarakat. Kelengkeng memiliki ciri-ciri buahnya kecil, bergerombol, mempunyai rasa manis yang memikat, serta mempunyai daging yang tebal. Buah kelengkeng memiliki cita rasa yang khas. Aroma nya harum, rasa manis berair merupakan buah eksklusif karena harganya relatif mahal sehingga konsumen buah kelengkeng adalah masyarakat menengah ke atas. Buah kelengkeng sangat menyehatkan dengan kandungan nutrisi nya. Sebab itulah masyarakat mulai membudidayakan tanaman kelengkeng guna memenuhi kebutuhan buah kelengkeng yang sangat kurang suplay nya (Wahyuni and Santosa, 2019).

2.3. Metode *Forward Chaining*

Forward chaining: Pencocokan fakta atau pernyataan dimulai dari bagian sebelah kiri dulu (IF dulu). Dengan kata lain penalaran dimulai dari fakta terlebih dahulu untuk menguji kebenaran hipotesis. Kadang disebut: data-driven karena inference engine menggunakan informasi yang ditentukan oleh user untuk memindahkan ke seluruh jaringan dari logika „AND‟ dan „OR‟ sampai sebuah terminal ditentukan sebagai objek. Bila inference engine tidak dapat menentukan objek maka akan meminta informasi lain. Aturan (*Rule*) di mana menentukan objek, membentuk path (lintasan) yang mengarah ke objek. Oleh karena itu, hanya satu cara untuk mencapai satu objek adalah memenuhi semua aturan (Permana, Saputra and Cabai, 2019).

Langkah – langkah dalam membuat sistem pakar dengan menggunakan metode *forward chaining* yaitu Menurut (Riskadewi dan Hendrik, 2005) :

a. Pendefinisian masalah dimulai dengan pemilihan domain masalah dan akuisi pengetahuan.
b. Pendefinisian data input untuk memulai inferensi karena diperlukan oleh sistem forward chaining.
c. Pendefinisian struktur pengendalian data untuk membantu mengendalikan pengaktifan suatu aturan.
d. Penulisan kode awal dalam domain pengetahuan.
e. Pengujian sistem agar dapat mengetahui sejauh sistem berjalan mana
f. Perancangan antarmuka dengan basis pengetahuan
g. Pengembangan system h. Evaluasi system

2.4. *Backward chaining*

Backward chaining adalah pencocokan fakta atau pernyataan yang dimulai dari bagian sebelah kanan, dengan kata lain penalaran dimulai dari hipotesis tersebut harus dicari fakta-fakta yang ada di dalam basis pengetahuan. Teknik pencarian yang dimulai dari fakta yang diketahui, kemudian mencocokkan fakta-fakta tersebut dengan IF dari rule IF-THEN.

Backward chaining menggunakan pendekatan goal-driven, dimulai dari ekspektasi apa yang diinginkan terjadi (hipotesis), kemudian mencari bukti yang mendukung (atau kontradiktif) dari ekspektasi tersebut (Honggowibowo, 2019).

Backward Chaining adalah suatu alasan berkebalikan dengan hipotesis, dimana hipotesis dihasilkan setelah mengumpulkan fakta-fakta yang sudah ada secara lengkap lalu diambil kesimpulan (conclusion) atau hipotesis nya sedangkan Backward Chaining akan memperkirakan potensial kesimpulan (conclusion) yang mungkin terjadi atau terbukti, karena adanya fakta yang mendukung hipotesis tersebut (Noviani, Prambudi and Mulyadi, 2020).

2.5. Pengumpulan Data

Metode pengumpulan data yang bisa dilakukan untuk mendapatkan data yang akurat (Sugiyono, 2010) sebagai berikut :

1. Wawancara

Wawancara adalah proses memperoleh keterangan untuk tujuan penelitian dengan cara tanya jawab, sambil bertatap muka antara penanya atau pewawancara dengan penjawab atau responden dengan menggunakan alat yang dinamakan *interview guide* (panduan wawancara). Wawancara dapat dilakukan dengan tatap muka maupun melalui komunikasi jaringan telepon.

2. Pengamatan

Pengumpulan data dengan pengamatan langsung adalah cara pengambilan data dengan menggunakan mata tanpa ada pertolongan alat standar lain untuk keperluan tersebut.

3. Dokumentasi

Dokumentasi merupakan catatan peristiwa yang sudah berlalu. Dokumen bisa berbentuk tulisan, gambar, atau karya-karya monumental dari seorang. Dokumen yang berbentuk tulisan misalnya catatan harian, sejarah kehidupan (*life histories*), ceritera, biografi, peraturan, kebijakan. Dokumen yang berbentuk gambar misalnya foto, gambar hidup, sketsa dan lain-lain.

4. Tinjauan Pustaka

Studi kepustakaan adalah metode pengumpulan data dengan mengadakan studi penelaahan terhadap buku-buku, literatur-literatur, catatan-catatan, dan laporan-laporan yang ada hubungannya dengan masalah yang dipecahkan.

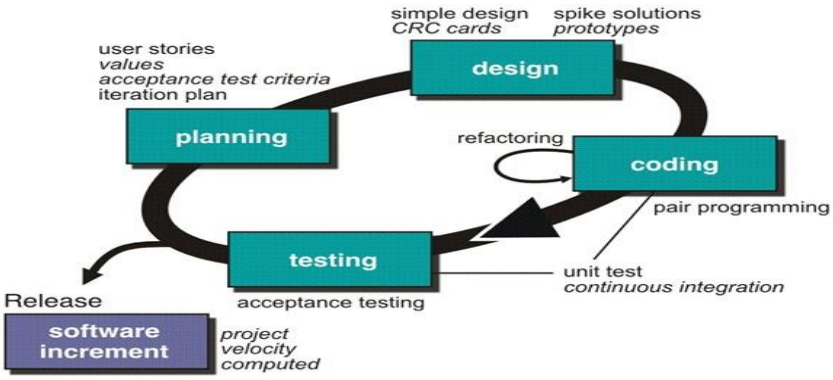
2.6. Metode Pengembangan Sistem

XP adalah metodologi pengembangan perangkat lunak yang ditujukan untuk meningkatkan kualitas perangkat lunak dan tanggap terhadap perubahan kebutuhan pelanggan. Jenis pengembangan perangkat lunak semacam ini dimaksudkan untuk meningkatkan produktivitas dan memperkenalkan pos pemeriksaan dimana persyaratan pelanggan baru dapat diadopsi (Pressman, 2015).

Tahapan-tahapan dari *Extreme Programming* terdiri dari *planning* seperti memahami kriteria pengguna dan perencanaan pengembangan, *designing* seperti perancangan *prototype* dan tampilan, *coding* termasuk pengintegrasian dan yang terakhir adalah testing. Unsur-unsur lain dari *Extreme Programming* meliputi

paired programming pada tahapan *coding*, unit testing pada semua kode, penghindaran pemrograman fitur kecuali benar-benar diperlukan, struktur manajemen yang datar, kode yang sederhana dan jelas, dan seringnya terjadi komunikasi antara programmer dan pelanggan ketika terjadi perubahan kebutuhan pelanggan seiring berlalunya waktu berlalu.

Metode ini membawa unsur-unsur yang menguntungkan dari praktek rekayasa perangkat lunak tradisional ke tingkat "ekstrem", sehingga metode ini dinamai *Extreme Programming*. Unsur-unsur yang menjadi karakteristik metodologi adalah kesederhanaan, komunikasi, umpan balik, dan keberanian. Gambar tahapan XP dapat dilihat pada gambar 2.1:



Gambar 2. 1 Tahapan Extreme Programming

Dibawah ini adalah penjelasan tahapan *Extreme Programming* yaitu :

1. *Planning* (Perencanaan)

Kegiatan Perencanaan disebut juga *planning game* biasanya dimulai dengan mendengarkan suatu kegiatan yang bertujuan mengumpulkan kebutuhan-kebutuhan untuk memahami konteks bisnis dan perlunya keluaran-keluaran (*output*), fungsi utama dan fungsionalitas.

Pada perencanaan terdapat *user stories values* yaitu story dengan value tertinggi akan dipindahkan dari jadwal dan diimplementasikan pertama, *acceptance test criteria interaction plan* melakukan perhitungan kecepatan *project* selama *development*, kustomer dapat menambah story, merubah value, membagi story atau menghapusnya.

2. *Design* (Perancangan)

Perancangan yang menarik, dan sederhana selalu memberikan hasil yang lebih disukai daripada gambaran-gambaran yang lebih kompleks. Perancangan XP memberikan panduan implementasi untuk suatu cerita ketika ditulis, tidak kurang, tidak lebih.

Terdapat *simple design CRC Cards* untuk mengenali dan mengatur *object oriented class* sesuai dengan *software increment* dan *spike solutions prototypes* melakukan spesifikasi solusi dari *object oriented class.*

3. *Coding* (Pengkodean)

Pengkodean ini dilanjutkan setelah cerita yang telah dikembangkan dan rancangan yang telah dilakukan oleh tim perangkat lunak. Pengkodean ini tidak langsung mengarah ke kode-kode program. Tim akan mengembangkan serangkaian unit pengujian lalu beralih ke pengkodean.

Pada tahapan *pair programming* melakukan kerja sama untuk membuat code dari satu story dan *refactoring* adalah proses restrukturisasi kode program computer yang ada tanpa mengubah perilaku eksternal nya.

4. *Testing* (Pengujian)

Unit pengujian yang harus dibuat dan kemudian dijalankan menggunakan kerangka kerja yang memungkinkan mereka untuk di otomatisasi sehingga dapat

dijalankan dengan mudah dan berulang kali.

Pada tahapan pengujian yaitu *unit test continuous integration* yaitu tahapan pengujian code yang diintegrasikan dengan kerja lainnya dengan pengujian yang dilakukan oleh customer dan fokus pada keseluruhan dan fungsional sistem dan *acceptance testing* yaitu pengujian yang dilakukan *customer stories* yang akan diimplementasikan sebagai bagian dari *software release.*

Selanjutnya terdapat tahapan *software increment project velocity computed* yaitu tahapan yang telah diimplementasikan dari *software release* yang nantinya akan diterapkan dalam suatu sistem.

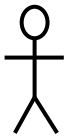
2.7. Bahasa Pemodelan Pengembangan Sistem (*UML*)

Bahasa Pemodelan Pengembangan Sistem (*Unified Modeling Language*) adalah salah satu standar bahasa yang banyak digunakan di dunia industri untuk mendefinisikan *requirement* (kebutuhan), membuat analisis & desain, serta menggambarkan arsitektur dalam pemrograman berorientasi objek (Rosa & Shalahuddin, 2018).

2.7.1. *Use Case Diagram*

Use case diagram atau diagram *use case* merupakan pemodelan untuk kelakuan *(behavior)* sistem informasi yang akan dibuat (Rosa & Shalahuddin, 2018). *Use case* mendeskripsikan sebuah interaksi antara satu atau lebih aktor dengan sistem informasi yang akan dibuat menjelaskan simbol-simbol yang ada pada diagram *use case* dapat dilihat pada gambar 2.6 di bawah ini:

Tabel 2. 2 Simbol Diagram *Use Case*

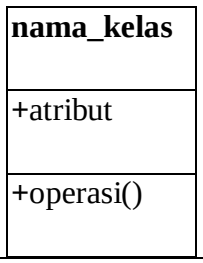
Simbol	Deskripsi
Use Case	Fungsionalitas yang disediakan sistem sebagai unit-unit yang saling bertukar pesan antar unit atau aktor, biasanya dinyatakan dengan menggunakan kata kerja di awal frase nama *use case*
Aktor/*actor* 	Orang, proses atau sistem lain yang berinteraksi dengan sistem informasi yang 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*
Asosiasi/*association*	Komunikasi antara aktor dan *use case* yang berpatisipasi pada *use case* atau *use case* memiliki interaksi dengan *actor*
Ekstensi/*extend*	Relasi *use case* tambahan ke sebuah *use case* dimana *use case* yang ditambahkan
<<*extend*>>	dapat berdiri sendiri walau tanpa *use case* tambahan itu mirip dengan prinsip *inheritance* pada pemrograman berorientasi objek biasanya *use case* tambahan memiliki nama depan
Generalisasi/***generalization***	Hubungan generalisasi dan spesialisasi (umum-khusus) antara dua buah *use case* dimana fungsi yang satu adalah fungsi yang lebih umum dari lainnya.
Menggunakan/*Include/uses*	Relasi *use case* tambahan ke sebuah *use case* dimana *use case* yang ditambahkan

<<*include*>>	memerlukan *use case* ini untuk menjalankan fungsinya atau sebagai syarat dijalankan *use case* ini

2.7.2. *Class Diagram*

Diagram kelas atau *class diagram* menggambarkan struktur sistem dari segi pendefinisian kelas-kelas yang akan dibuat untuk membangun sistem. Kelas memiliki apa yang disebut atribut dan metode atau operasi (Rosa and Shalahudin, 2018). menjelaskan simbol-simbol yang ada pada diagram kelas pada tabel *class diagram* 2.3.

Tabel 2. 3 Simbol *Class Diagram*

Simbol	**Deskripsi**
Kelas 	Kelas pada struktur sistem
Antarmuka/*Interface* **nama_*interface***	Sama dengan konsep *interface* dalam pemrograman berorientasi objek
Asosiasi/*asociation*	Relasi antar kelas dengan makna umum, asosiasi biasanya juga disertai dengan *multiplicity*
Asosiasi berarah/*directed association*	Relasi antar kelas dengan makna kelas yang satu digunakan oleh kelas yang lain, asosiasi biasanya disertai dengan *multiplicity*

Generalisasi	Relasi antar kelas dengan makna generalisasi-spesialisasi(umum khusus)
Kebergantungan/*dependecy*	Relasi antar kelas dengan makna kebergantungan antar kelas
Agregasi/*agregation*	Relasi antar kelas dengan makna semua bagian (*whole-part*)

2.7.3.*Activity Diagram*

Activity diagram atau Diagram aktivitas menggambarkan *workflow* (aliran kerja) atau aktivitas dari sebuah sistem atau proses bisnis (Rosa & Shalahuddin, 2018), menjelaskan Simbol-simbol yang ada pada *activity diagram*dapat dilihat pada tabel 2.4 di bawah ini :

Tabel 2. 4 Simbol *Activity Diagram*

Simbol	Deskripsi
Status awal	Status awal aktivitas sistem,sebuah diagram aktivitas memiliki sebuah status awal.
Aktivitas 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
Tabel	Suatu file komputer dari mana data bisa

	dibaca atau direkam selama kejadian bisnis
Dokumen	Menunjukan dokumen sumber atau laporan
Status akhir	Status akhir yang dilakukan sistem, sebuah diagram aktivitas memiliki sebuah status akhir.
Swimlane nama swimlane	Memisahkan organisasi bisnis yang bertanggung jawab terhadap aktivitas yang terjadi

2.8. Alat Implementasi

2.8.1. *Xampp*

Menurut MADCOMS (2016) *Xampp* adalah sebuah paket kumpulan software yang terdiri dari *Apache*, *MySQL*, *PhpMyAdmin*, PHP, *Perl*, *Filezilla*, dan lain.

Xampp berfungsi untuk memudahkan instalasi lingkungan PHP, di mana biasanya lingkungan pengembangan web memerlukan PHP, *Apache*, *MySQL dan PhpMyAdmin* (MADCOM, 2016).

2.8.2 *Dreamweaver*

Adobe Dreamweaver adalah :aplikasi desain dan pengembangan web yang menyediakan editor WYSIWYG visual (bahasa sehari-hari yang disebut sebagai

Design view) dan kode editor dengan fitur standar seperti *syntax highlighting*, *code completion*, dan *code collapsing* serta fitur lebih canggih seperti *real-time syntax checking* dan *code introspection* untuk menghasilkan petunjuk kode untuk membantu pengguna dalam menulis kode (Destiningrum and Adrian, 2017).

2.8.3 MYSQL

Menurut MADCOM (2016) MySQL adalah sistem manajemen *Database* SQL yang bersifat *Open Source* dan paling popular saat ini. Sistem *Database* MySQL mendukung beberapa fitur seperti multithreaded, multiuser dan SQL Database managemen system (DBMS).

2.9. PHP

Menurut Arif (2011), PHP (*Personal Home Page*) adalah bahasa pemrograman yang digunakan secara luas untuk penanganan pembuatan dan pengembangan sebuah web dan bisa digunakan pada HTML. PHP merupakan singkatan dari "PHP: *Hypertext Preprocessor*", dan merupakan bahasa yang disertakan dalam dokumen HTML sekaligus bekerja di sisi *server* (*server-side* HTML-*embedded scripting*). Artinya sintaks dan perintah yang diberikan akan sepenuhnya dijalankan di *server* tetapi disertakan pada halaman HTML biasa, sehingga script-nya tak tampak di sisi *client* (Destiningrum and Adrian, 2017)..

PHP dirancang untuk dapat bekerja sama dengan database server dan dibuat sedimikian rupa sehingga pembuatan dokumen HTML yang dapat mengakses database menjadi begitu mudah. Tujuan dari bahasa scripting ini adalah untuk membuat aplikasi dimana aplikasi tersebut yang dibangun oleh PHP pada umumnya akan memberikan hasil pada web *browser*, tetapi prosesnya secara keseluruhan dijalankan di server.

2.10. *CodeIgneter*

Menurut MADCOM (2016) *CodeIgniter* adalah sebuah *framework* PHP yang dapat membantu mempercepat *developer* dalam pengembangan aplikasi *web* berbasis PHP dibanding jika menulis semua kode program dari awal. Adapun beberapa keuntungan menggunakan *CodeIgniter,* diantaranya:

1. Gratis

   CodeIgniter berlisensi dibawah *Apache*/BSD *opensorce.*

2. Ditulis Menggunakan PHP 4

   Meskipun *CodeIgniter* dapat berjalan di PHP 5, namun sampai saat ini kode program *CodeIgniter* masih dibuat dengan menggunakan PHP 4.

3. Berukuran Kecil

   Ukuran *CodeIgniter* yang kecil merupakan keunggulan tersendiri. Dibanding dengan *framework* lain yang berukuran besar.

4. Menggunakan Konsep MVC

   CodeIgniter menggunakan konsep MVC yang memungkinkan pemisahan *layer application-logic* dan *presentation.*

5. URL yang Sederhana

   Secara *default*, URL yang dihasilkan *CodeIgniter* sangat bersih dan *Serach Engine Friendly* (SEF).

6. Memiliki Paket *Library* yang Lengkap

   CodeIgniter mempunyai *library* yang lengkap untuk mengerjakan operasi-operasi yang umum dibutuhkan oleh sebuah aplikasi berbasis web, misalnya mengakses *database*, mengirim email, memvalidasi form, menangani *session* dan sebagainya.

7. *Extensible*

   Sistem dapat dikembangkan dengan mudah menggunakan *plugin* dan *helper*, atau dengan menggunakan *hooks.*

8. Tidak Memerlukan *Template Engine*

   Meskipun *CodeIgniter* dilengkapi dengan *template* parser sederhana yang dapat digunakan, tetapi hal ini tidak mengharuskan kita untuk menggunakannya.

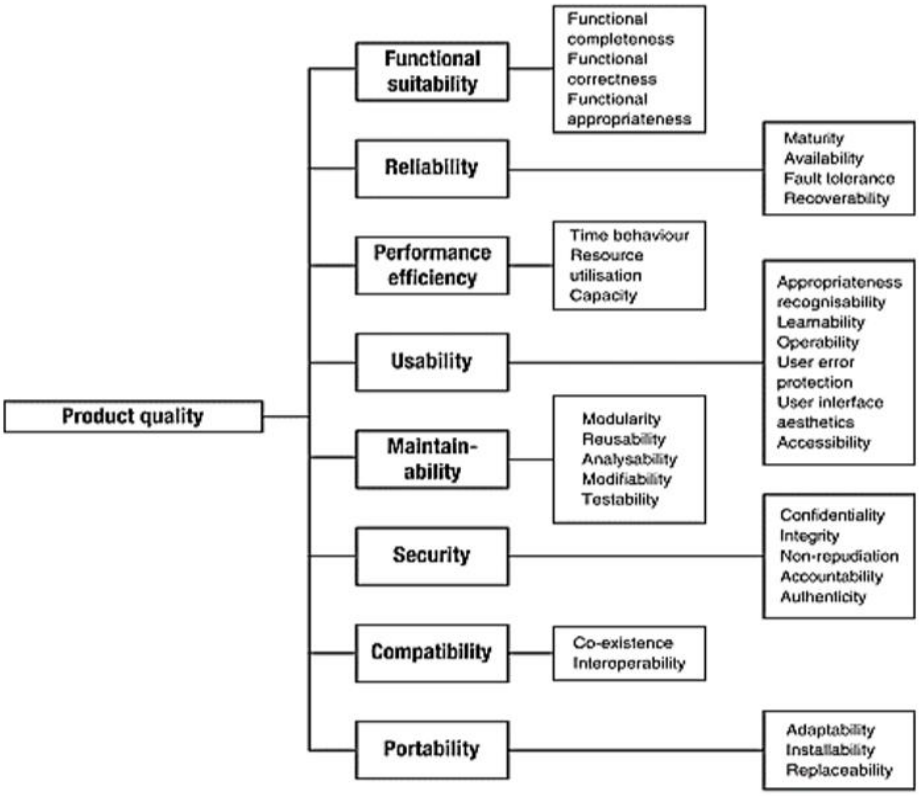
9. Dokumentasi Lengkap dan Jelas

   Dari sekian banyak *framework*, *CodeIgniter* adalah satu-satunya *framework* dengan dokumentasi yang lengkap dan jelas.

2.11. Pengujian Sistem ISO 25010

ISO/IEC 25010 merupakan model kualitas sistem dan perangkat lunak yang menggantikan ISO/IEC 9126 tentang *software engineering* (Iqbal, 2016). Product quality ini juga digunakan untuk tiga model kualitas yang berbeda untuk produk perangkat lunak antara lain:

1. Kualitas dalam model penggunaan
2. Model kualitas produk
3. Data model kualitas



Gambar 2. 2 Model kualitas produk ISO/IEC 25010

Adapun dimensi yang pertama terdapat beberapa faktor elemen diantaranya :

1) *Functionality* (Fungsionalitas). Kemampuan perangkat lunak Merupakan tingkatan dimana perangkat lunak dapat menyediakan fungsionalitas yang dibutuhkan ketika perangkat lunak digunakan pada kondisi spesifik tertentu dalam hal ini perangkat lunak dapat memenuhi kelayakan dari sebuah fungsi untuk melakukan pekerjaan yang spesifik bagi pengguna dan dapat memberikan hasil yang tepat dan ketelitian terhadap tingkat kebutuhan pengguna.
2) *Reliability* Merupakan tingkatan dimana perangkat lunak dapat bertahan pada tingkatan tertentu ketika digunakan oleh pengguna pada kondisi yang spesifik dalam hal ini perangkat lunak dapat beroperasi dan siap ketika dibutuhkan untuk digunakan dan juga dapat bertahan pada tingkat kemampuan tertentu

terhadap kegagalan, kesalahan serta perangkat lunak kembali pada tingkat tertentu dalam mengembalikan pengembalian data yang disebabkan kegagalan atau kesalahan pada perangkat lunak.

3) *Performance efficiency* Merupakan tingkatan dimana perangkat lunak dapat memberikan kinerja terhadap sejumlah sumber daya yang digunakan pada kondisi tertentu dalam hal ini *performance efficiency* dapat memberikan reaksi dan waktu yang dibutuhkan ketika melakukan aksi dari sebuah fungsi dan perangkat lunak dapat menggunakan sejumlah sumber daya ketika melakukan aksi dari sebuah fungsi.
4) *Usability* Perangkat lunak dapat dimengerti, dipelajari, digunakan dan menarik pengguna ketika digunakan dalam hal ini perangkat lunak mudah dipelajari oleh pengguna, perangkat lunak dapat digunakan dan dioperasikan oleh pengguna.
5) *Security* Merupakan perlindungan terhadap perangkat lunak dari berbagai ancaman atau ke ganjalan dalam hal ini perangkat lunak memiliki perlindungan terhadap data atau informasi dari pengguna dan merupakan dari kelengkapan, ketepatan dari sejumlah *asset* yang telah dijaga sehingga aksi atau tindakan yang dilakukan telah terbukti dan hal tersebut tidak dapat ditolak.
6) *Compability* Faktor ini merupakan kemampuan dari dua atau lebih komponen perangkat lunak dapat melakukan pertukaran informasi dan melakukan fungsi yang dibutuhkan ketika digunakan pada *hardware* atau lingkungan perangkat lunak yang sama.

7) *Maintainability* Merupakan tingkat dimana sebuah perangkat lunak dapat dimodifikasi. Dalam hal ini modifikasi adalah perbaikan, perubahan atau penyesuaian perangkat lunak untuk dapat berubah pada lingkungan, kebutuhan dan fungsionalitas yang spesifik. Selain itu perangkat lunak dapat dianalisis untuk mengetahui apa yang menyebabkan kegagalan pada perangkat lunak untuk mengidentifikasi bagian yang dapat dimodifikasi.

8) *Transferability* Merupakan kemudahan dimana sistem atau komponen dapat berpindah dari lingkungan satu ke lingkungan yang lain dalam hal ini perangkat lunak dapat beradaptasi dengan cepat pada spesifikasi lingkungan yang berbeda tanpa menerapkan aksi atau cara lain dari pada memberikan tujuan tertentu terhadap perangkat lunak yang telah ada