BAB II

KAJIAN TEORI

A. Kajian Teori

1. Pengembangan Sistem Informasi *Internship Program Tracer* Berbasis *Website* Dengan *MEAN Stack*

Sistem Informasi *Internship Program Tracer* Berbasis *Website* Dengan *MEAN Stack* merupakan sebuah sistem informasi berbasis *website* yang dikembangkan menggunakan metode *Agile Unified Process* dan menggunakan *tools* atau teknologi dari *MEAN Stack* (*MongoDB, Express, Angular*, dan *NodeJS*) dengan arsitektur *web RESTful API*, yang menerapkan metode *Information Retrieval* (Temu Kembali Informasi) dengan tujuan untuk menghasilkan informasi riwayat industri/perusahaan/bengkel pelaksanaan Praktik Industri mahasiswa Jurusan Pendidikan Teknik Elektronika dan Informatika Fakultas Teknik Universitas Negeri Yogyakarta. Berikut uraian tentang teori-teori yang terkait dengan variabel penelitian kali ini :

a. Sistem Informasi *Internship Program Tracer* Berbasis *Website* Dengan *MEAN Stack*

1) Sistem Informasi

Pengertian sistem adalah sekumpulan dari elemen atau komponen yang saling berinteraksi antara satu dengan lainnya yang bertujuan untuk mencapai sebuah tujuan tertentu. Komponen dari sistem terdiri dari komponen masukan atau biasa disebut dengan istilah *input*, proses atau disebut dengan istilah *process*, keluaran

(*output*), serta timbal balik (*feedback*). Data merupakan sekumpulan fakta, seperti nomor pegawai, total jam kerja pegawai dalam seminggu, atau nomor pesanan. Sedangkan informasi adalah sekumpulan data yang diorganisasikan sedemikian rupa sehingga dapat menghasilkan nilai tambah. Dari kedua paparan definisi di atas, dapat diambil kesimpulan pengertian dari Sistem Informasi adalah relasi dari sekumpulan komponen untuk menyimpan, memanipulasi mengumpulkan, dan menyebarkan data atau informasi, serta memberikan timbal balik (*feedback*) untuk menggapai tujuan tertentu. (Stair & Reynolds, 2010).

Sistem Informasi bertujuan untuk mendapatkan informasi yang tepat kepada individu yang tepat pada waktu, jumlah, dan format yang tepat. Karena Sistem Informasi dimaksudkan untuk menyediakan informasi yang berguna (Rainer & Cegielski, 2011).

2) Praktik Industri (*Internship Program*)

Internship Program atau Praktik Industri menjadi kewajiban bagi mahasiswa Fakultas Teknik Universitas Negeri Yogyakarta sebagai salah satu mata kuliah atau program kurikuler yang harus ditempuh dengan bobot 3 (tiga) sks. Praktik Industri dilaksanakan dalam jangka waktu minimal selama 2 (dua) bulan.

Praktik Industri memiliki bertujuan agar dapat menambahkan wawasan, IPTEK, dan pengalaman bagi mahasiswa pada serangkaian kegiatan praktek langsung yang dilaksanakan di lapangan dalam hal ini industri/perusahaan/bengkel yang ditempati. Selain itu, mahasiswa yang melaksanakan kegiatan Praktik Industri

akan memperoleh pelajaran terkait kewirausahaan yang memiliki hubungan dengan tempat industri/perusahaan/bengkel yang ditempati selama kegiatan Praktik Industri berlangsung.

Selain menjadi kelengkapan pembelajaran untuk pemenuhan kurikulum, mata kuliah Praktik Industri memiliki beberapa peran strategis bagi Fakultas Teknik Universitas Negeri Yogyakarta. Peran strategis tersebut adalah : sebagai kontrol kualitas mahasiswa, apakah mahasiswa Fakultas Teknik Universitas Negeri Yogyakarta telah memenuhi kaidah keterkaitan dan kesesuaian (*link and match*) bidang keahliannya dengan tuntutan dunia industri. Peran selanjutnya adalah sebagai fungsi kehumasan (*public relation*) bagi FT UNY.

Praktik Industri bekerja sama dengan industri/bengkel/perusahaan yang memenuhi syarat yang telah ditetapkan oleh Fakultas dan relevan dengan program studi yang ditempuh oleh mahasiswa Fakultas Teknik Universitas Negeri Yogyakarta. Oleh karena itu dalam proses mencari, memilih dan menempatkan mahasiswa untuk melaksanakan Praktik Industri harus diperhatikan dan diorganisasikan sesuai prosedur melalui tahapan perencanaan, koordinasi, pelaksanaan, pengendalian, serta melakukan evaluasi secara berkala dan cermat agar dapat mencapai tujuan praktik industri secara efektif serta efisien (Tim Praktik Industri, 2016).

3) *Information Retrieval* (*IR*)

Pada pengembangan Sistem Informasi *Tracer* kali ini menerapkan metode *information retrieval* (Temu Kembali Informasi) agar mahasiswa Jurusan Pendidikan Teknik Elektronika dan Informatika Universitas Negeri Yogyakarta yang belum melaksanakan kegiatan Praktik Industri mendapatkan akses untuk menelusuri data-data riwayat industri/perusahaan/bengkel Praktik Industri yang telah dilakukan mahasiswa angkatan sebelumnya yang tersimpan dalam *database*.

Information retrieval (IR) atau "Temu Kembali Informasi" adalah pencari materi (biasanya berbentuk dokumen) dari sumber yang tidak terstruktur (biasanya teks) yang berfungsi untuk memenuhi kebutuhan informasi, dari dalam koleksi besar yang tersimpan dalam *database* (Manning, 2008). *Information retrieval* juga dapat diartikan ilmu yang mempelajari prosedur dan metode untuk menemukan kembali informasi (dokumen) yang tersimpan dari sumber yang tidak tersetruktur (teks).

Information retrieval memiliki prinsip kerja apabila terdapat sebuah kumpulan dokumen dan seorang pengguna (*user*) yang memformulasikan sebuah pertanyaan (*request* atau *query*). Jawaban pertanyaan tersebut adalah mengambil sekumpulan dokumen yang relevan dan membuang dokumen yang tidak relevan (Gerard, 1989).

4) *Website*

World Wide Web (*WWW*) atau yang biasa diketahui dengan istilah *web* pertama kali ditemukan oleh *Sir Timothy John "Tim" Berners-Lee*, yang merupakan seseorang yang berkebangsaan Inggris pada tahun 1980-an. *Web* pada awalnya dibuat bertujuan untuk memberikan kemudahan tukar menukar atau memperbarui informasi kepada sesama peneliti di tempat dia bekerja di *European Laboratory for Particle Physics* (atau dikenal dengan *CERN*) di kota *Geneva* dekat perbatasan Swiss dan Prancis. Seiring perkembangan teknologi, penggunaan *web* semakin banyak digunakan untuk pembuatan *website* hingga *web application* (Berners-Lee, 1989).

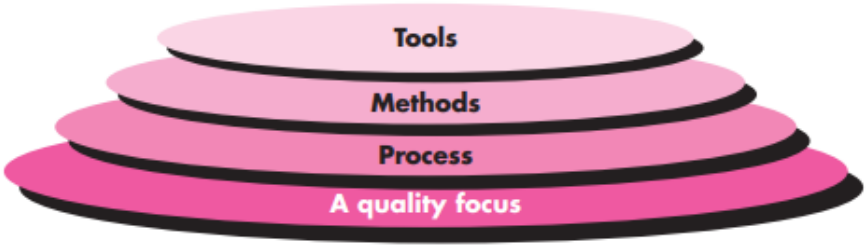
b. Pengembangan Sistem Informasi berbasis *Website*

1) Pengembangan Perangkat Lunak

Untuk mengembangkan sebuah sistem yang memiliki kompleksitas secara sistematis dan terintegrasi, maka dibutuhkan metode pengembangan sistem agar dapat membantu menuntun pengembang sistem (*developer*) untuk menghasilkan suatu sistem/perangkat lunak yang memiliki standar tertentu (Oetomo, 2002).

Pengembangan sistem/*software*/perangkat lunak (*software engineering*) merupakan sebuah lapisan teknologi yang meliputi kumpulan alat, metode, proses, dan fokus kualitas (Pressman, 2010). Secara sederhana dapat dikatakan bahwa pengembangan perangkat lunak (*software enginnering*) merupakan rangkaian dari proses yang sistematis menggunakan metode dan alat (teknologi) tertentu untuk

menghasilkan sebuah sistem/*software*/aplikasi yang memiliki kualitas. Gambar 1 menunjukkan lapisan pengembangan perangkat lunak.



Gambar 1. Lapisan *Software Engineering*

Lapisan alat (*tools layer*) dalam pengembangan sistem/*software*/perangkat lunak mendukung lapisan proses dan metode dalam pengembangan perangkat lunak. Setiap pendekatan pengembangan (termasuk pengembangan perangkat lunak) harus bersandar pada komitmen organisasi kepada kualitas (*software quality*).

Lapisan metode (*methods layer*) pengembangan perangkat lunak menyediakan teknik "bagaimana untuk" membangun sebuah perangkat lunak. Lapisan ini mencakup beragam tugas termasuk *communication, requirements analysis, design modeling, program construction, testing,* dan *support*.

Sedangkan *layer process* merupakan kumpulan aktivitas, aksi, dan tugas yang dilakukan ketika produk sedang akan dibuat. Sebuah aktivitas berupaya mencapai tujuan yang luas, dan diterapkan terlepas dari domain aplikasi. Pada lapisan ini mencakup beragam tugas seperti berikut:

a) *Communication*

Sebelum pekerjaan teknikal lainnya dilakukan, sangat penting untuk melakukan komunikasi and kolaborasi dengan klien, hal ini bertujuan agar klien memahami tujuan dari proyek dan mengumpulkan keperluan (*requirements*) yang membantu mendefinisikan fungsi dan fitur perangkat lunak.

b) *Planning*

Sebuah proyek perangkat lunak pasti memiliki proses-proses yang sulit, namun proses tersebut akan mudah jika terdapat perencanaan dalam pengembangan perangkat lunak. Perencanaan membantu pengembang mudah mendeskripsikan langkah teknikal dalam pengembangan perangkat lunak.

c) *Modeling*

Merupakan proses untuk mengimplementasikan hasil *planning* ke dalam bentuk desain produk, baik itu desain antarmuka maupun desain komponen *back-end*. Tahapan ini akan membuat lebih terarah ketika melakukan tahapan *construction.*

d) *Construction*

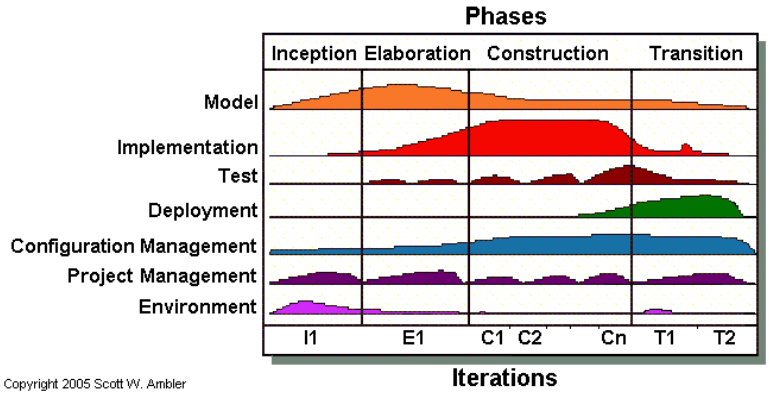
Tahapan ini berfungsi untuk membangun/menghasilkan/memproduksi kode program yang dapat dieskekusi ke dalam bentuk perangkat lunak. Biasanya pada tahap ini juga dilakukan pengujian fungsi dan fitur perangkat dalam skala kecil.

e) *Deployment*

Deployment merupakan tahapan terakhir dari lapisan *process*. Tahapan ini menandakan bahwa produk / perangkat lunak siap didistribusikan dan digunakan kepada pengguna.

2) *Agile Unified Process* (AUP)

Agile Unified Process merupakan salah satu metode pengembangan perangkat lunak. *Agile Unified Process* merupakan versi sederhana dari *Rational Unified Process* (RUP). Metode ini menjelaskan kemudahan untuk memahami pendekatan untuk mengembangkan aplikasi perangkat lunak bisnis menggunakan teknik dan konsep lincah/cepat (Ambler, 2014).



Gambar 2. Siklus Pengembangan *Agile Unified Process* (AUP)

Pada dasarnya *Agile Unified Process* (AUP) memiliki tahapan-tahapan pengembangan yaitu:

a) *Model*

b) *Implementation*

c) *Test*

d) *Deployment*

e) *Configuration Management*

f) *Project Management*

g) *Environment*

Dimana pada setiap tahapan tersebut dilakukan pada 4 fase yang berbeda, yaitu *Inception, Elaboration, Construction*¸dan *Transition*.

c. Perangkat Pengembangan Perangkat Lunak

1) UML (*Unified Modelling Language*)

Grady Booch, Ivar Jacobson, dan James Rumbaugh pada tahun 1995 menggagas sebuah gagasan untuk menciptakan sebuah standar diagram bernama *Unified Modelling Language* (UML) (Alan, Wixom, & Tegarden, 2015). Pada tahun 1997, *Unified Modelling Language* (UML) dirilis dengan versi 1.0. Dengan seiringnya waktu, UML mengalami peningkatan serta mengalami revisi secara berkala. UML telah mencapai versi 2.0 pada tahun 2012 hingga saat penulisan penelitian ini, UML telah mencapai versi 2.5.

UML menyajikan model dalam wujud diagram, yang merepresentasikan bagian dari fakta yang telah dideskripsikan dalam bentuk model (Seidl, Scholz, Huemer, & Kappel, 2015). Pada saat ini, terdapat 15 macam diagram UML yang

dibagi menjadi 2 (dua) kategori yaitu yang pertama adalah diagram terstuktur atau *structure diagrams* yang menyajikan gambaran data dan juga relasi statis dalam sistem informasi dan kategori yang kedua yaitu *behaviour diagrams* yang menyajikan model kebutuhan sistem (Alan, Wixom, & Tegarden, 2015).Diagram terstruktur atau *Structure diagrams* terdiri dari model : *class*, *component*, *composite*, *object*, *package*, *deployment*, *structure*, dan *profile diagrams*. Sedangkan untuk *behaviour diagrams* meliputi *activity*, *sequence*, *timing*, *interaction overview*, *communication*, *protocol state machine*, *behavior state machine*, serta *use-case diagrams*.

Dari 15 macam model UML, pada tugas akhir ini ini peneliti menggunakan 4 (empat) model diagram yaitu :

a) *Use Case Diagram*

Use Case adalah kumpulan daftar dari aksi yang biasanya mendefinisikan interaksi antara peran pengguna dengan sistem untuk meraih sebuah tujuan. *Use Case* menyajikan interaksi antara pengguna dan dalam teknik yang detail. Pada umumnya, *business model* terdapat entitas (orang maupun perusahaan) eksternal untuk target organisasi, disebut pelaku bisnis (*business actors*) dan entitas internal disebut *business workers*. *Business Use Case* merupakan proses yang dilakukan oleh para *business actors* dan *business workers* yang memungkinkan mereka untuk mencapai beberapa tujuan bisnis perusahaan.

Use Case Diagram menyajikan seperangkat *use case* dan aktor (sejenis kelas khusus) serta relasinya. Menerapkan *use case diagram* untuk mengilustrasikan tampilan *use case* statis dari suatu sistem. Menggunakan *use case diagram* sangat penting dalam mengatur dan memodelkan sistem.

b) *Class Diagram*

Class Diagram menampilkan satu set *class*, *interface* (antarmuka), dan kolaborasi atau relasi. *Class Diagram* ini adalah diagram yang paling umum ditemukan dalam proses pembuatan model sistem berorientasi objek (*Object Oriented*). *Class diagram* membahas pandangan desain statis dari suatu sistem. *Class diagram* yang mencakup kelas aktif membahas pandangan proses statis dari suatu sistem.

c) *Activity Diagram*

Adalah jenis khusus dari diagram *statechart* yang menampilkan aliran dari antar aktivitas dalam suatu sistem. *Activity diagram* membahas pandangan dinamis dari suatu sistem. *Activity diagram* memiliki peranan penting dalam pembuatan model fungsi sistem serta menekankan aliran kontrol di antara objek.

d) *Sequence Diagram*

Merupakan diagram interaksi yang lebih menekankan waktu saat melakukan pemesanan pesan. *Sequence diagram* menampilkan satu set objek dan juga pesan yang dikirim dan diterima oleh tiap-tiap objek. Objek-objek biasanya diberi nama atau contoh anonim kelas, tetapi mungkin juga mewakili contoh hal-hal lain, seperti

kolaborasi, komponen, dan simpul. *Sequence diagram* secara sederhana digunakan untuk menggambarkan pandangan dinamis dari suatu sistem.

2) *MEAN Stack*

MEAN Stack merupakan kepanjangan dari *MongoDB, Express JS, Angular*, dan *NodeJS* atau juga biasa disebut *Full-Stack Development*. *Full-Stack Development* dimulai dari *database* dan *web server* pada sisi belakang atau biasa disebut *back-end*, yang terdiri dari logika aplikasi, dan kontrol pada sisi tengah (*middle*), serta bagian antarmuka (*interface*) pada sisi depan atau biasa disebut (*front-end*). *MEAN Stack* terdiri dari empat teknologi *programming* yang digunakan, yaitu :

a) *MongoDB*

Basis data atau biasa juga disebut *database* merupakan sebuah sistem komputerisasi (perhitungan) untuk memelihara informasi dan membuat informasi tersedia ketika dibutuhkan (Irmawati & Indrihapsari, 2014).

MongoDB adalah *database* yang berorientasi pada dokumen (*object-oriented*), bukan relasional. Alasan utama untuk menghindari dari model relasional adalah membuat *scalling* lebih mudah, namun juga tetap memiliki keuntungan lain. *MongoDB* merupakan salah satu media penyimpan basis data (*database*) yang *powerfull*, fleksibel, dan terukur. Ia mampu untuk melakukan pengukuran (*scalling*) dengan berbagai macam fitur yang dimiliki oleh *relational database*, seperti

secondary indexes, *range queries*, dan *sorting* (pengurutan) (Kristina & Dirolf, 2010).

Ide dasar untuk mengganti model atau konsep baris (*row*) ke model yang lebih fleksibel, yaitu "*document*". Dengan menggunakan dokumen dan *array* yang disematkan, konsep *document-oriented* memungkinkan untuk merepresentasikan hubungan hirarki yang kompleks dengan *single-record*.

MongoDB juga mendukung *schema-free*: sebuah *key* dari dokumen tidak ditentukan atau diperbaiki dengan berbagai cara. Tanpa menggunakan skema untuk mengubah, migrasi data secara *massive*, biasanya tidak diperlukan.

b) *ExpressJS*

ExpressJS adalah salah satu *Framework* Aplikasi *Web NodeJS* yang ringkas dan fleksible, yang menyediakan serangkaian fitur canggih untuk aplikasi *web* dan *mobile* (ExpressJS Contributors, 2017).

c) *Angular*

Angular merupakan salah satu *Framework JavaScript* yang dikembangkan oleh *Google* di bawah lisensi *MIT-Style*, yang dapat digunakan untuk mengembangkan sebuah *Reactive Single Page Application* (*SPAs*). Membangun aplikasi dengan *Angular*, memungkinkan *developer* dapat menggunakan kembali kode yang telah dibuat untuk membuat aplikasi yang berjalan pada *platform* lain dan pada target *deployment* yang berbeda seperti aplikasi *web*, *mobile web*, *native mobile*, hingga *native desktop* dengan menggunakan kode yang sama. Hal ini

dikarenakan *Angular* merupakan *Cross-Platform*, sehingga dapat digunakan pada *desktop* maupun *mobile* (*multiplatform*) (Angular Developers, 2018).

d) *NodeJS*

NodeJS merupakan *platform server-side* yang dibangun pada *Google Chrome* menggunakan *JavaScript Engine* (V8). *NodeJS* dikembangkan oleh Ryan Dahl pada tahun 2009 dan versi terbarunya adalah v0.10.36. *NodeJS* merupakan platform yang dibangun di *runtime JavaScript Google Chrome* untuk memudahkan pembuatan *network application* secara cepat dan skalabel. *NodeJS* menggunakan *event-driven*, model *non-blocking I/O* yang membuatnya ringan dan efisien, sempurna untuk aplikasi data-intensif secara *real-time*. *NodeJS* merupakan *Open Source*, *Cross-Platform Runtime Environment* untuk mengembangkan *server-side* dan *networking application*. Seperti yang telah disebutkan di atas, *NodeJS* ditulis dalam *JavaScript*, dan dapat dijalankan pada *Runtime NodeJS* pada *OS X*, *Microsoft Windows*, dan *Linux*. *NodeJS* juga menyediakan berbagai *library* dan modul *JavaScript* yang menyederhanakan pengembangan aplikasi *web* (Brown, 2014).

3) *REST API Web Service*

REST merupakan kepanjangan dari *REpresentational State Transfer*. *REST* sendiri merupakan sebuah standar arsitektur *web* yang digunakan dalam pengembangan layanan berbasis *web*. Pada umumnya, *REST* menggunakan *HTTP* (*Hypertext Transfer Protocol*) sebagai *protocol* untuk komunikasi data. *REST* pertama kali diperkenalkan oleh Roy Fielding pada tahun 2000.

REST server menyediakan sumber daya/data (*resources*) dan *REST client* mengakses serta menampilkan data tersebut untuk penggunaan selanjutnya. Setiap sumber daya/data (*resource*) diidentifikasi oleh URIs (*Universal Resource Identifiers*) atau *global ID. Resource* tersebut direpresentasikan dalam bentuk format data *XML, JSON*, maupun teks (Fielding, 2000).

Berikut ini adalah *HTTP Methods* yang umumnya digunakan dalam arsitektur *RESTful API* :

GET - Digunakan yang berfungsi untuk mengakses sebuah resource
POST - Digunakan yang digunakan untuk mengubah sebuah resource atau membuat sebuah resource baru
PUT - Digunakan untuk membuat sebuah resource baru
DELETE - Digunakan untuk menghapus resource
OPTIONS - Digunakan untuk mendapatkan dukungan operasi pada *resource*.

2. Pengujian Perangkat Lunak dan Standar ISO 25010:2011

a. Pengujian Perangkat Lunak

Pengujian perangkat lunak atau juga bisa disebut *software testing* merupakan teknik untuk menguji perangkat lunak atau sistem dengan mengacu pada standar tertentu. Pengujian perangkat lunak menjadi bagian dari salah satu tahapan pada berbagai macam metode pengembangan perangkat lunak seperti *waterfall*, RUP (*Relational Unified Process*), AUP (*Agile Unified Process*), dan lain sebagainya. Pengujian perangkat lunak sangat diperlukan untuk memastikan produk *software*

atau perangkat lunak yang telah dikembangkan benar-benar memiliki kesiapan untuk digunakan kepada pengguna.

Terdapat berbagai macam model pengujian perangkat lunak, salah satunya adalah pengujian dengan Standar ISO/IEC 25010:2011. Standar ISO/IEC 25010:2011 adalah sebuah standar pengujian perangkat lunak internasional yang berlaku saat ini. Standar ini menentukan 8 (delapan) karakteristik atau aspek yaitu *Functional Suitability*, *Usability*, *Performance Efficiency*, *Reliability*, *Compatibility*, *Security*, *Maintainability* dan juga *Portability* yang dijabarkan menjadi serangkaian sub-karakteristik (Wagner, 2013).

b. Standar ISO/IEC 25010:2011

Pada tahun 2011, ISO/IEC 9126 merupakan acuan atau standar yang dipergunakan untuk menguji kualitas perangkat lunak. Namun di 2011 ISO/IEC 25010 hadir untuk memperbarui versi sebelumnya yaitu ISO/IEC 9126, yang merupakan acuan standar untuk tingkat kualitas *software/*perangkat lunak/sistem informasi (ISO, 2011). ISO/IEC 25010:2011 mempunyai 8 (delapan) karakteristik kualitas perangkat lunak pada sebagai berikut:

1) *Functional Suitability*

Functional Suitability merepresentasikan sudah sejauh mana fungsi-fungsi pada perangkat lunak atau sistem telah memenuhi kebutuhan pengguna yang direncanakan dan diimplementasikan ketika dipergunakan pada suatu keadaan

tertentu. Pada karakteristik *Functional Suitability* sendiri dijabarkan menjadi 3 (tiga) subkarakteristik yaitu :

- *Functional Completeness* adalah karakteristik yang mengatur fungsi-fungsi tersebut mencakup seluruh *task* yang telah ditentukan.
- *Functional Correctness* adalah karakteristik perangkat lunak atau sistem yang dikembangkan mampu memberikan hasil yang presisi.
- *Functional Appropriateness* adalah tingkat dimana fungsi dapat memberikan fasilitas untuk mencapai tugas dan tujuan tertentu.

2) *Performance Efficiency*

Performance Efficiency merepresentasikan performa relatif terhadap kuantitas sumber daya atau *resources* yang dipergunakan pada suatu keadaan. *Performance Efficiency* sendiri dijabarkan dalam 3 (tiga) subkarakteristik atau tingkatan sebagai berikut :

- *Time Behaviour*, adalah tingkat dimana respon dan waktu proses serta tingkat *output* suatu perangkat lunak atau sistem ketika fungsinya berjalan dapat memenuhi persyaratan.
- *Resource Utilization* (Pemanfaat Sumber Daya), yaitu tingkat kuantitas serta dan sumber daya (*resources*) yang dipergunakan oleh suatu perangkat lunak atau sistem ketika menjalankan fungsinya dapat memenuhi persyaratan.
- *Capacity*, adalah tingkatan dimana batasan maksimum produk atau parameter sistem dapat menyesuaikan persyaratan.

3) *Compatibility*

Yaitu tingkatan perangkat lunak, sistem atau komponen mampu saling berbagi informasi terhadap perangkat lunak, sistem maupun komponen lainnya, dan atau menjalankan fungsinya yang dibutuhkan, sedangkan sistem tetap dapat saling berbagi informasi pada *hardware* maupun *software* di suatu lingkungan yang sama.

- *Co-existence*, tingkatan perangkat lunak mampu menjalankan kebutuhan fungsi dengan efisien, sementara sistem sedang berbagi *resources* atau sumberdaya terhadap produk lainnya, tanpa memberikan dampak yang memberikan kerugian pada produk lainnya.
- *Interoperability*, tingkatan dimana dua atau lebih suatu sistem, perangkat lunak atau komponen dapat berbagi informasi satu sama lain dan dapat memakai kembali informasi yang telah ditukar sebelumnya.

4) *Usability*

Yaitu tingkatan dimana perangkat lunak atau sistem memungkinkan untuk digunakan oleh suati pengguna/user tertentu untuk menggapai sebuah tujuan tertentu dengan efektivitas, efisiensi, serta kepuasan dalam konteks penggunaan tertentu. Karakteristik ini dijabarkan menjadi beberapa subkarakteristik sebagai berikut :

- *Accessibility Recognizability*, tingkat pengguna mengetahui apakah sistem yang telah dikembangkan sesuai dengan kebutuhan pengguna.
- *Learnability*, tingkat suatu perangkat lunak atau sistem memungkinkan untuk dioperasikan oleh pengguna tertentu untuk mencapai tujuan dari pembelajaran.

- *Operability*, adalah tingkatan suatu produk atau sistem mempunyai atribut yang membuat mudah dioperasikan.
- *User Error Protection*, yaitu tingkat dimana sistem mampu memberikan perlindungan kepada pengguna dari kesalahan atau *error*.
- *User Interface Aesthetics*, adalah tingkat dimana *user interface* atau antarmuka pengguna mampu memberikan interaksi yang memberikan kesenangan dan memberi kepuasan bagi pengguna.
- *Accessibility*, yaitu tingkat suatu perangkat lunak atau sistem dapat dioperasikan oleh pengguna dalam jangkauan karakteristik dan kemampuan terluas untuk dicapai pada tujuan dan dalam konteks penggunaan tertentu.

5) *Reliability*

Yaitu tingkatan dimana suatu perangkat lunak/sistem atau komponen mampu melakukan suatu fungsi tertentu dalam situasi tertentu serta untuk jangka waktu tertentu. Karakteristik ini terdiri dari 3 (tiga) sub-karakteristik berikut :

- *Maturity*, yaitu tingkat dimana perangkat lunak atau sistem dapat memenuhi kebutuhan akan reliabilitas di bawah operasi normal.
- *Availability*, yaitu tingkat dimana produk atau sistem dapat beroperasi dan diakses ketika diperlukan untuk digunakan.
- *Recoverability*, yaitu tingkat dimana produk atau sistem ketika interupsi atau kegagalan terjadi, perangkat lunak atau sistem mampu mengembalikan data yang terkena dampak dan membangun kembali seperti semula.

6) *Security*

Yaitu tingkatan perangkat lunak atau sistem yang mampu memberikan perlindungan informasi dan data karakteristik ini terdiri dari sub karakteristik berikut :

- *Confidentiality*, tingkatan perangkat lunak atau sistem mampu memberikan kepastian bahwa data hanya dapat diakses kepada pengguna yang memiliki wewenang untuk menggunakannya.
- *Integrity*, tingkatan sistem, perangkat lunak atau komponen mampu memberikan pencegahan terhadap akses illegal untuk mengakses suatu data rahasia.
- *Non-repudiation*, tingkat tindakan dapat dibuktikan telah terjadi, sehingga tidak ada penolakan pada peristiwa atau tindakan.
- *Accountability*, tingkatan tindakan dari suatu entitas dapat dilacak secara unik pada suatu entitas tersebut.
- *Authenticity*, tingkat identitas subjek atau sumber daya dapat dibuktikan menjadi salah satu yang dapat diklaim.

7) *Maintainability*

Karakteristik ini mewakili tingkatan efektifitas dan efisiensi suatu sistem atau perangkat lunak agar dapat dilakukan modifikasi dalam rangka untuk memperbaikinya, serta menyesuaikannya apabila terjadi perubahan lingkungan, dan dalam kebutuhan. Karakteristik ini terdiri dari beberapa sub-karakteristik berikut:

- *Modularity* yaitu tingkatan dimana sistem atau perangkat lunak terdiri dari sekumpulan komponen yang terpisah sehingga apabila terjadi perubahan pada suatu komponen, akan meminimalisir dampak pada komponen lain.
- *Reusability* yaitu tingkat penggunaan kembali suatu aset pada satu sistem, atau penggunaan untuk membangun properti lainnya.
- *Analysability* yaitu tingkatan keefektifan serta efisiensi yang memberikan kemungkinan untuk mengukur dampak pada perubahan satu atau lebih bagian perangkat lunak atau sistem, mencari penyebab error pada suatu produk, serta mengidentifikasi suatu bagian yang akan diubah.
- *Modifiability* yaitu tingkat dimana suatu perangkat lunak atau sistem dapat dilakukan modifikasi dengan efektif serta efisien tanpa memberikan suatu cacat maupun mengurangi kualitas produk yang dikembangkan.
- *Testability* yaitu tingkat efektivitas dan efisiensi pada kriteria pengujian dapat dibangun untuk perangkat lunak, serta pengujian dapat dilakukan untuk menentukan apakah kebutuhan atau kriteria tersebut telah terpenuhi.

8) *Portability*

Tingkatan efektivitas serta efisiensi dimana suatu sistem atau perangkat lunak dapat ditransfer dari suatu perangkat keras, perangkat lunak, atau lingkungan operasional yang berbeda.

- *Adaptability* yaitu tingkat dimana suatu perangkat lunak atau sistem mampu beradaptasi secara efektif dan efisien untuk perangkat keras, perangkat lunak,

atau lingkungan operasional serta penggunaan lain yang berbeda atau sedang dikembangkan.

- *Installability* yaitu tingkatan efektivitas dan efisiensi dimana perangkat lunak atau keberhasilan sistem dapat dilakukan pemasangan dan atau pelepasan pada sebuah lingkungan tertentu.
- *Replaceability* yaitu tingkat di mana suatu perangkat lunak mampu menggantikan perangkat lunak lain yang telah ditentukan untuk tujuan yang sama dalam lingkungan yang sama.

B. Hasil Penelitian yang Relevan

1. Hasil penelitian yang berjudul "Aplikasi Sistem Rekomendasi Pemilihan Tempat Praktik Industri Yang Tepat Bagi Mahasiswa Pendidikan Teknik Elektronika Berbasis Android" yang ditulis oleh Rahardyan Bisma Setya Putra dalam penelitian yang dilakukan pada tahun 2017. Penelitian tersebut menghasilkan sebuah Sistem Informasi berbasis *mobile* yang menggunakan *platform android* yang menyediakan data tempat praktik industri, serta dapat memberikan rekomendasi yang mengacu pada minat mahasiswa. Dan menjamin sistem yang dikembangkan agar dapat memberikan rekomendasi yang tepat, yang memiliki kesesuaian dengan minat mahasiswa dengan cara melakukan pengujian terhadap hasil rekomendais aplikasi terhadap minat dan konsentrasi mata kuliah yang akan diambil oleh mahasiswa dengan menggunakan pengujian tingkat akurasi informasi yang dihasilkan. Aplikasi rekomendasi tempat industri memberikan hasil rekomendasi dengan metode

rekomendasi yang berbasis konten dengan pembobotan *term frequency – inverse document frequency* yang berjalan pada *platform Android.*

2. Hasil penelitian yang berjudul "Analisis Perancangan Sistem Informasi *Tracer Study* Berbasis *Web* Dengan Menggunakan *Codeignier*" yang dilakukan oleh Anisa Nur Hidayati dalam penelitian yang dilakukan pada tahun 2017. Tujuan pada penelitian ini adalah untuk memastikan kualitas dari sistem informasi tracer study berbasis *web* yang dikembangkan dengan bahasa pemrograman *PHP* yang menggunakan sebuah *framework* yang bernama *CodeIgniter* yang mengacu pada standar ISO 9126, sehingga diperoleh hasil tingkat kelayakan dari produk yang dikembangkan pada penelitian ini. Produk yang dihasilkan pada penelitian ini adalah sebuah sistem informasi *tracer study* yang telah diuji kualitasnya. Penelitian ini menerapkan metode *Research and Development* yang berpedoman dari pendapat Borg and Gall. Sedangkan pengujiannya mengacu pada ISO 9126 pada aspek *functional suitability, reliability, portability,* dan *usability*. Instrumen penelitian pada *functional suitability* peneliti menggunaka pengujian dengan metode *black box testing*, aspek *reliability* diuji dengan *software WebServer Stress Tool*. Pada aspek *portability* diuji dengan *software powermapper* dan *browserstack*, dan untuk menguji *usability* menggunakan kuisioner dari J.R Lewis. Hasil pengujian *functional* untuk sub karakteristik *accuracy and suitability* memperoleh hasil dengan kelayakan 100%. Sedangkan untuk aspek *security* dinyatakan aman dari beberapa macam serangan seperti *blacklisting, injected SPAM, defacement, ,*

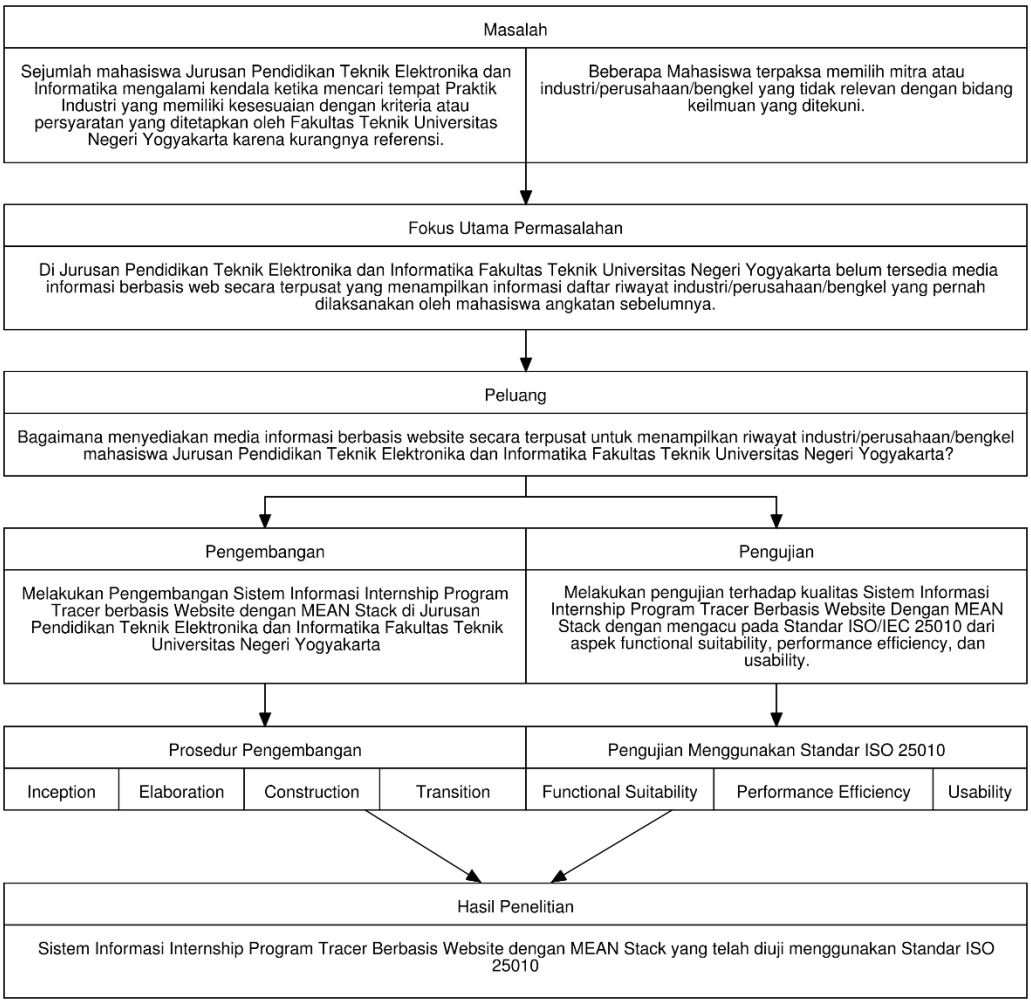
website, *malware,* dan *SQLInjection*. Pengujian *reliability* sistem dinyatakan layak karena produk dapat berjalan baik dengan 10 stimultan user dengan rata-rata waktu 2 detik serta keberhasilan akses 95,08%. Pengujian *portability* memperoleh hasil kelayakan 100%, dan pada *usability* memperoleh hasil kelayakan 70% dengan hasil reliabilitas 0,929.

3. Hasil penelitian yang berjudul "Analisis Kualitas Sistem Informasi Kegiatan Sekolah Berbasis Mobile *Web* di SMK Negeri 2 Yogyakarta" yang dilakukan oleh Taufiq Abdul Ghaffur & Nurkhamid dalam penelitannya di tahun 2017. Penelitian ini memiliki tujuan untuk menjamin tingkat kualitas Sistem Informasi Kegiatan Sekolah berbasis *mobile web,* untuk menghindari terjadinya *error,* serta dapat menyesuaikan fitur dan fugsionalitasnya sesuai yang direncanakan dengan dilakukan pengujian yang mengacu pada standar ISO 25010. Pengujian pada penelitian ini menggunakan 4 (empat) karakteristik yang terdapat pada standar ISO 25010 yaitu *functional suitability*, *compatibility*, *usability*, dan *performance efficiency*. Instrumen penelitian yang digunakan yaitu *black-box testing*, *USE Questionnaire*, *Yslow tools* dan *Pingdom Website Speed Test*. Penelitian ini menghasilkan Sistem Informasi Kegiatan Sekolah yang memenuhi standar ISO 25010 pada 4 (empat) karakteristik di atas. pada aspek *functional suitability* mendapatkan nilai 100%, pada aspek *compatibility* memperoleh nilai 100%, dan pada *usability* memperoleh nilai 82% dengan *alpha Cronbach* sebesar 0,981, dan aspek

performance efficiency diperoleh nilai 94,2 dengan rata-rata kecepatan akses sebesar 0,9305 detik setiap halaman (baik).

C. Kerangka Pikir

Penelitian tugas akhir ini dimulai dari permasalahan yang ditemukan ketika masa-masa pemilihan tempat Praktik Industri berlangsung. Sehingga dibutuhkan sebuah alternatif sebagai solusi untuk masalah tersebut. Adapun salah satu solusi pada permasalahan tersebut adalah dengan mengembangkan Sistem Informasi *Internship Program Tracer* berbasis *website* dengan *MEAN Stack* yang menampilkan data riwayat industri/perusahaan/bengkel Praktik Industri mahasiswa yang telah menyelesaikan pelaksanaan Praktik Industri. Setelah sistem dibuat, dilakukan pengujian terhadap sistem tersebut yang telah dikembangkan oleh peneliti. Adapun kerangka pikir dalam penelitian ini adalah sebagai berikut :



Gambar 3. Kerangka Pikir Penelitian

D. Pertanyaan Penelitian

Terdapat 5 (lima) pertanyaan penelitian dari Sistem Informasi *Internship Program Tracer* berbasis *website* dengan *MEAN Stack* sebagai berikut:

1. Bagaimanakah mengembangkan Sistem Informasi *Internship Program Tracer* berbasis *website* dengan *MEAN Stack* sesuai dengan model pengembangan *Agile Unified Process* (AUP) ?

2. Apakah Sistem Informasi *Internship Program Tracer* berbasis *website* dengan *MEAN Stack* memenuhi aspek *functional suitability* ?
3. Apakah Sistem Informasi *Internship Program Tracer* berbasis *website* dengan *MEAN Stack* memenuhi aspek *usability* ?
4. Apakah Sistem Informasi *Internship Program Tracer* berbasis *website* dengan *MEAN Stack* memenuhi aspek *perfomance efficiency* ?