BAB II
LANDASAN TEORI

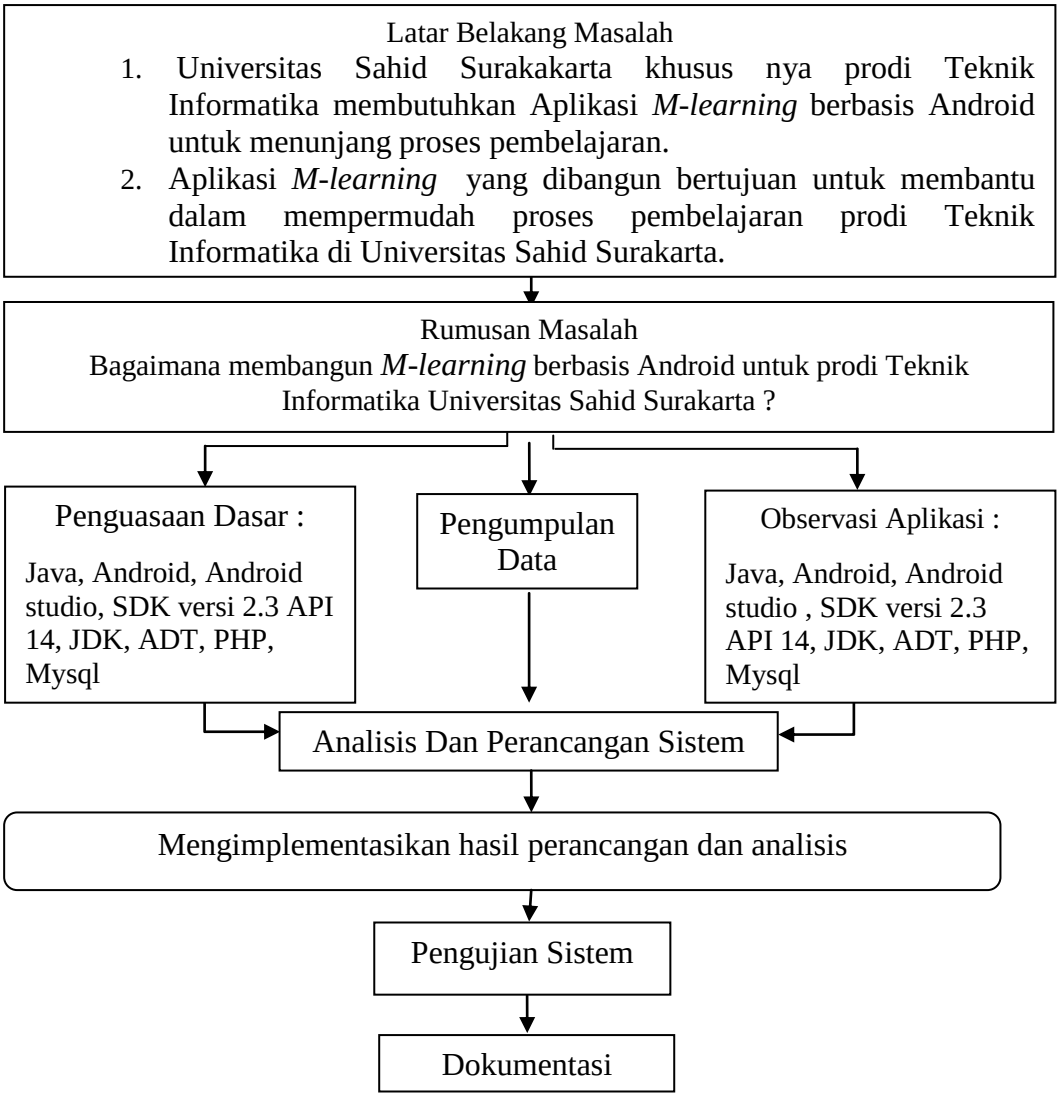
2.1. Tinjauan Pustaka

Astra, dkk (2012), menjelaskan bahwa dari penelitian yang berjudul aplikasi *mobile learning* fisika dengan menggunakan *adobe flash* sebagai media pembelajaran pendukung dapat disimpulkan bahwa aplikasi *mobile learning* fisika dengan menggunakan *adobe flash* pada materi pelajaran perpindahan kalor khususnya dapat dijadikan sebagai media pembelajaran pendukung untuk siswa SMA. Media pembelajaran ini memiliki kelebihan, yaitu siswa dapat mengakses materi pelajaran dari mana saja tanpa dibatasi oleh ruang dan tempat serta memiliki fleksibilitas, karena tidak terkait dengan waktu.

Yuniati (2011), melalui penelitian yang berjudul pengembangan media pembelajaran *mobile learning* efek *Doppler* sebagai alat bantu dalam pembelajaran fisika yang menyenangkan dapat disimpulkan bahwa *mobile learning* efek *Doppler* dikembangkan dengan format multimedia yang menyajikan teks, gambar, audio, dan animasi. Media pembelajaran tersebut berfungsi sebgai alat bantu dalam pembelajaran fisika yang menyenangkan sehingga mempermudah belajar siswa dimanapun dan kapanpun, siswa dapat mengakses bahan-bahan belajar setiap saat dan berulang-ulang, dengan demikian siswa dapat lebih memantapkan penguasaannya terhadap materi pembelajaran.

Rizal, dkk (2013), melalui penelitiannya yang berjudul perancangan dan pembuatan *mobile learning* interaktif berbasis android dengan metode personal *extreme* programming dapat disimpulkan bahwa aplikasi *mobile learning* diciptakan untuk mengintegrasikan sumber daya pada aplikasi *client* dengan server *e-learning* memiliki fitur untuk menampilkan informasi materi tiap kuliah. Informasi materi unduh *file*, informasi penugasan, dan informasi forum serta menambah diskusi forum. Penerapan metode PXP memberi kemudahan yang juga berperan sebagai penggunaan perangkat lunak dalam mengestimasi dan memperkirakan segala prioritas.

2.2. Kerangka Pemikiran



Gambar 2.1 Diagram Kerangka Pemikiran

Kerangka pemikiran menjelaskan alur proses pembuatan laporan Tugas Akhir (Gambar 2.1)

1. Latar Belakang Masalah

Universitas Sahid Surakarta khususnya prodi Teknik Informatika membutuhkan Aplikasi *M-learning* berbasis Android untuk menunjang proses pembelajaran.

Aplikasi *M-learning* yang dibangun bertujuan untuk membantu dalam mempermudah proses pembelajaran prodi Teknik Informatika di Universitas Sahid Surakarta.

2. Rumusan Masalah

Bagaimana membangun *M-learning* berbasis Android untuk prodi Teknik Informatika Unversitas Sahid Surakarta ?

3. Pengumpulan Data Tertulis dan Tidak Tertulis

Mengumpulkan semua data yang diperlukan dalam penelitian, baik melalui *interview*, observasi maupun dokumentasi.

4. Penguasaan Dasar

Melakukan beberapa percobaan membuat aplikasi sederhana dengan tujuan agar dapat lebih menguasai pemrograman android menggunakan Android studio yang merupakan *tool* lengkap yang terdiri dari SDK Android dan API sebagai alat untuk desain tampilan, *coding*, emulasi aplikasi android bersifat *free.*

5. Observasi Aplikasi

Mencari beberapa aplikasi atau tinjauan pustaka yang berkaitan dengan *M-learning* dan pemrograman Android, karya ilmiah, buku yang dapat dijadikan referensi dalam membangun *M-learning* berbasis Android untuk prodi teknik informatika Universitas Sahid Surakarta.

6. Analisis dan Perancangan Sistem Berbasis Objek

Menganalisa dan merancang aplikasi yang akan dibangun, yang meliputi gambaran aplikasi, desain aplikasi, dan isi dari aplikasi yang akan dibangun.

7. Implementasi

Membangun Aplikasi *M-learning* dengan basis Android sesuai dengan data-data yang didapatkan dari prodi Teknik Informatika Universitas Sahid surakarta.

8. Pengujian Sistem

Pada tahap ini dimana sistem yang telah siap digunakan kemudian dilakukan pengujian untuk mengetahui jika ternyata masih ada kesalahan atau kekurangan pada sistem yang telah dibuat.

9. Dokumentasi

Pada tahap akhir dimana sistem telah siap digunakan di prodi Teknik Informatika Universitas Sahid Surakarta dan membuat dokumentasi dari keseluruhan penelitian tugas akhir ini

2.3. Landasan Teori

2.3.1. Pengertian Sistem

Sistem adalah suatu jaringan kerja dari prosedur-prosedur yang saling berhubungan, berkumpul bersama-sama untuk melakukan suatu kegiatan atau untuk menyelesaikan suatu sasaran yang tertentu (Hartono, 2005).

2.3.2 Pengertian informasi

Informasi dapat diartikan sebagai data yang diolah menjadi bentuk yang lebih berguna dan lebih berarti bagi yang menerimanya (Hartono, 2005).

2.3.3 Sistem informasi

Sistem Informasi adalah kumpulan dari sub-sub sistem baik fisik maupun non fisik yang berhubungan satu sama lain dan bekerjasama secara harmonis untuk mencapai satu tujuan yaitu mengolah data menjadi informasi yang berarti dan berguna (Susanto, 2009).

2.3.4 *M-learning*

M-learning yaitu sebuah metode yang menggunakan teknologi divais bergerak tanpa kabel untuk sistem pembelajaran yang mampu melakukan pengaksesan terhadap lingkungan komputer yang berbasis desktop.

M-learning merupakan sebuah metode yang terintegrasi pada perangkat bergerak dan pemrosesan teknologi tanpa kabel dengan sebuah sistem pembelajaran untuk memperbaiki keefektifan belajar menggunakan metode pembelajaran tradisional. *Mobile learning* memanfaatkan kemajuan dari teknologi baik *wireless network* maupun *mobile communication.* Dengan keberadaan teknologi tanpa kabel (*wireless*) dan teknologi divais bergerak telah memacu pengembangan dari sistem pembelajaran yang dilakukan secara tradisional *(conventional learning)* (Robson, dkk, 2003).

2.3.5 Android

Android adalah sebuah sistem operasi *mobile* yang berbasiskan pada versi modifikasi dari Linux. Pertama kali sistem operasi ini dikembangkan oleh perusahaan Android Inc. Nama perusahaan inilah yang pada akhirnya digunakan sebagai nama proyek sistem operasi *mobile* tersebut, yaitu sistem operasi Android (Komputer, 2013).

Android adalah sistem operasi berbasis Linux bagi telepon seluler seperti telepon pintar dan komputer tablet. Android juga menyediakan platform terbuka bagi para pengembang untuk menciptakan aplikasi mereka sendiri yang akan digunakan untuk berbagai macam piranti gerak. Awalnya, Google Inc. membeli Android Inc., pendatang baru yang membuat piranti lunak untuk ponsel. kemudian dalam pengembangan Android, dibentuklah *Open Handset Alliance*, konsorsium dari 34 perusahaan piranti keras, piranti lunak, dan telekomunikasi, termasuk Google, HTC, Intel, Motorola, Qualcomm, T-Mobile, dan Nvidia (Kasman, 2013).

2.3.6 Android studio

Android studio merupakan lingkungan pengembangan Android baru berdasarkan intelliJ IDEA. Mirip dengan eclipse dengan ADT *plugin*, Android studio menyediakan alat pengembang terintegrasi untuk pengembangan dan *debugging* (Enterprice, 2015).

2.3.7 SDK

Android SDK adalah *tools* API *(Aplication Programming Interface)* yang diperlukan untuk memulai mengembangkan aplikasi pada *platform* Android menggunakan bahasa pemrograman java. Android SDK menyediakan *tools* dan API untuk pengembangan *platform* aplikasi Android dengan menggunakan bahasa pemrograman java (Mulyadi, 2010).

2.3.8 ADT (*Android Development Tools*)

ADT (*Android Development Tools*) adalah *plugin* yang didesain untuk IDE Eclipse yang memberikan kita kemudahan dalam mengembangkan aplikasi Android dengan menggunakan *IDE Eclipse*. Dengan menggunakan ADT untuk Eclipse akan memudahkan kita dalam membuat aplikasi *project* Android, menggunakan GUI, dan menambahkan komponen-komponen yang lainnya, begitu juga kita dapat melakukan *running* aplikasi menggunakan Android SDK melalui Eclipse. Dengan ADT juga kita dapat melakukan pembuatan *package* Android (.apk) yang digunakan untuk distribusi Android yang kita rancang (Safaat, 2012).

2.3.9 JDK

JDK adalah *Sun Microsystem* produk ditujukan untuk pengembangan java. Sejak diperkenalkan java, SDK java yang paling banyak digunakan. Pada tanggal 17 November 2006, Sun mengumumkan bahwa akan dirilis di bawah GNU *General Public License* (GPL), sehingga membuat perangkat lunak bebas (Satyaputra, dkk, 2002).

2.3.10 PHP

PHP (*Hypertext Preprocessor*) adalah bahasa *server-side scripting* yang menyatu dengan HTML untuk membuat halaman *web* yang dinamis. Karena PHP merupakan *server-side scripting* maka sintaks dan perintah-perintah PHP akan dieksekusi di *server* kemudian hasilnya dikirimkan ke *browser* dalam format HTML. Dengan demikian kode program yang ditulis dalam PHP tidak akan terlihat oleh *user* sehingga keamanan *web* lebih terjamin. PHP dirancang untuk membentuk suatu tampilan berdasarkan permintaan terkini, seperti menampilkan isi basis data ke halaman *web* (Arief, 2011).

2.3.11 MySQL

SQL merupakan kepanjangan dari *Structured Query Language*, yang merupakan bahasa terstruktur yang khusus digunakan untuk mengolah database. SQL pertama kali didefinisikan oleh ANSI pada tahun 1986. Sedangkan MySQL adalah sebuah *system managemen* database yang bersifat *open source* yang dibuat dan dikembangkan oleh MySQL AB yang berada di Swedia.

MySQL dapat digunakan untuk membuat dan mengelola *database* beserta isinya. MySQL dapat dimanfaatkan untuk menambah, mengubah, dan menghapus data yang berada dalam *database*, selain itu merupakan *system managemen database* yang akan diletakkan pada beberapa tabel yang terpisah sehingga manipulasi data akan menjadi jauh lebih cepat (Nugroho, 2008).

2.3.12 Xampp

Menurut Yogi Wicaksono (2008:7), Xampp adalah sebuah *software* yang berfungsi untuk menjalankan *website* berbasis PHP dan menggunakan pengolahan data MySQL di komputer lokal. Menurut Alexius (2010:3), Xampp merupakan sebuah aplikasi PHP, MySQL, dan Apache nyang diperlukan untuk melakukan

instalasi CMS, bahkan di dalam Xampp terdapat pula FFTP server dan *Email Server*.

Menurut Ramadhan (2006:4), Xampp merupakan sebuah *tool* yang menyediakan beberapa paket perangkat lunak ke dalam satu buah paket.

2.3.13 JSON

JSON (*JavaScript Object Notation*) adalah format pertukaran data yang ringan, mudah dibaca dan ditulis oleh manusia, serta mudah diterjemahkan dan dibuat (*generate*) oleh komputer. Format ini dibuat berdasarkan bagian dari Bahasa Pemprograman JavaScript, Standar ECMA-262 Edisi ke-3 – Desember 1999. JSON merupakan format teks yang tidak bergantung pada bahasa pemprograman apapun karena menggunakan gaya bahasa yang umum digunakan oleh programmer keluarga C termasuk C, C++, C#, Java, JavaScript, Perl, Python dan lainnya. Oleh karena sifat-sifat tersebut, menjadikan JSON ideal sebagai bahasa pertukaran-data (Kasman, 2013).

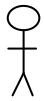
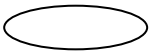
2.3.14 Metode Perancangan Sistem Berorientasi Objek

Metode pemrograman berorientasi objek mencoba melihat permasalahan lewat pengamatan dunia nyata dimana setiap objek adalah entitas tunggal yang memiliki kombinasi struktur data dan fungsi tertentu (Nugroho, 2002).

2.3.14.1 Use Case Diagram

Use case diagram mengambarkan sejumlah *external actors* dan hubungannya ke *use case* yang diberikan oleh sistem. *Use case* adalah deskripsi fungsi yang disediakan oleh sistem dalam bentuk teks sebagai dokumentasi dari *use case symbol* namun dapat juga dilakukan dalam *activity diagrams*. *Use case* digambarkan hanya dilihat dari luar oleh *actor* (keadaan lingkungan sistem yang dilihat *user*) dan bukan bagaimana fungsi yang ada dalam sistem (Kusumo 2004). Simbol-simbol yang digunakan pada *use case diagram* disajikan pada Tabel 2.1.

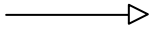
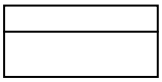
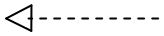
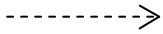
Tabel 2.1. Simbol *Use Case Diagram*

NO	GAMBAR	NAMA	KETERANGAN
1.		*Aktor*	Idealization orang eksternal, proses, ata u hal yang berinteraksi dengan sistem,su bsistem atau kelas.
2.		*Use case*	Sebuah *use case* menggambar-kan interaksi dengan*actor* sebagai uruta n pesan antarasistem dan aktor satu atau lebih.
3.		*System Boundary*	Menspesifikasikan paket yang menampilkan sistem secara terbatas.
4.		*Generalization*	Hubungan antara *use case* umum dan *use case* yang lebih spesifik yang mewarisi dan menambahkan fitur.
5.		*Comunication Association*	Jalur komunikasi antara *actor* dan *use case* yang berpartisipasi didalam.
6.		*Extend*	Penyisipan perilaku tambahan kedalam basis *use case* yang tidak tahu tentang hal itu.
7.		*Include*	Penyisipan perilaku tambahan kedalam basis *use case* yang secara eksplisit menggambarkan penyisipan.

2.3.14.2 Class Diagram

Class diagram menggambarkan struktur statis *class* di dalam sistem. *Class* merepresentasikan sesuatu yang ditangani oleh sistem. *Class* dapat berhubungan dengan yang lainmelalui berbagai cara: *associated* (terhubung satu sama lain), *dependent* (satu *class* tergantung/menggunakan *class* yang lain), *specialed* (satu *class* merupakan spesialisasi dari *class* lainya), atau *package* (grup bersama sebagai satu unit). Sebuah sistem biasanya mempunyai beberapa *class diagram* (Kusumo 2004). Simbol-simbol yang digunakan pada *class diagram* disajikan pada Tabel 2.2.

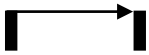
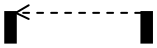
Tabel 2.2. Simbol *Class Diagram*

NO	GAMBAR	NAMA	KETERANGAN
1.		*Generalization*	Hubungan dimana objek anak (*descendent*) berbagai perilaku dan struktur data dari objek yang ada diatas objek induk (*ancestor*).
2.		*Nary Association*	Upaya untuk menghindari asosiasi dengan lebih dari dua objek.
3.		*Class*	Himpunan dari objek–objek yang terbagi atribut serta operasi yang sama.
4.		*Collaboration*	Deskripsi dari urutan aksi-aksi yang ditampilkan sistem yang menghasilkan suatu hasil yang terukur bagi suatu aktor.
5.		*Realization*	Operasi yang benar-benar dilakukan oleh suatu objek.
6.		*Dependency*	Hubungan yang terjadi pada suatu elemen mandiri (*independent*) akan mempengaruhi elemen yang bergantung pada elemen yan tidak mandiri.
7.		*Association*	Untuk menghubungkan objek satu dengan objek yang lainnya.

2.3.14.3 Sequence Diagram

Sequence diagram menggambarkan kolaborasi dinamis antara sejumlah objek. Kegunaan untuk menunjukan rangkaian pesan yang dikirim Antara objek juga interaksi antara objek, sesuatu yang terjadi pada titik tertentu dalam eksekusi sistem (Kusumo 2004). Simbol-simbol yang digunakan pada *sequence diagram* disajikan pada Tabel 2.3.

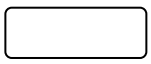
Tabel 2.3. Simbol *Sequence Diagram*

NO	GAMBAR	NAMA	KETERANGAN
1.		*LifeLine*	Objek *entity*, antarmuka yang saling berinteraksi.
2.		*Message*	*Message* ditampilkan sebagai anak panah dari lifeline dari satu objek ke objek yang lain.
3.		*Return*	Spesifikasi dari komunikasi antar objek yang memuat informasi-informasi tentang aktifitas yang terjadi.

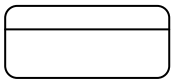
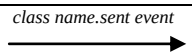
2.3.14.4 Statechart Diagram

Statechart diagram menggambarkan semua *state* (kondisi) yang dimiliki oleh suatu *object* dari suatu *class* dan keadaan yang menyebabkan *state* berubah. Kejadian dapat berupa *object* lain yang mengirim pesan. *State class* tidak digambarkan untuk semua *class*, hanya yang mempunyai sejumlah *state* yang terdefini dengan baik dan kondisi *class* berubah oleh *state* yang berbeda (Kusumo 2004). Simbol-simbol yang digunakan pada *statechart diagram* disajikan pada Tabel 2.4.

Tabel 2.4. Simbol *Statechart Diagram*

NO	GAMBAR	NAMA	KETERANGAN
1.		*State*	Meliputi seluruh pesan dari *object* yang dapat mengirim dan menerima.
2.		*Start state*	Masing-masing diagram harus mempunyai satu dan hanya satu *start state*

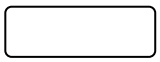
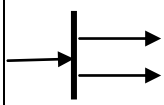
Lanjutan Tabel 2.4. Simbol *Statechart diagram*

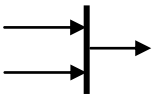
NO	GAMBAR	NAMA	KETERANGAN
3.		*State Detail*	*Action-action* yang mengiringi seluruh *state transition* ke sebuah *state*.
4.		*Stop State*	Sebuah *object* boleh mempunyai banyak *stop state*
5.		*State Transition*	Sebuah kejadian yang memicu sebuah *state object* dengan cara memperbarui satu atau lebih nilai atributnya.

2.3.14.5 Activity Diagram

Activity diagram menggambarkan rangkaian aliran aktifitas, digunakan untuk mendeskripsikan aktifitas yang dibentuk dalam suatu operasi sehingga dapat juga digunakan untuk aktifitas lainnya seperti *use case* atau interaksi (Kusumo 2004). Simbol-simbol yang digunakan pada *use activity diagram* disajikan pada Tabel 2.5.

Tabel 2.5. Simbol *Activity Diagram*

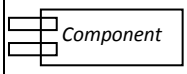
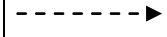
NO	GAMBAR	NAMA	KETERANGAN
1.		*Actifity State*	Aktivitas yang mewakili pelaksanaan dalam pernyataan dalam prosedur atau pelaksanaan kegiatan dalam alur kerja.
2.		*Branch/Merge*	Asosiasi percabangan dimana jika ada pilihan aktivitas lebih dari satu.
3.		*Initial State*	Status awal aktivitas sistem, sebuah diagram aktivitas memiliki sebuah status awal.
4.		*Final State*	Status akhir yang dilakukan sistem.
5.		*Fork*	Percabangan yang menunjukkan aliran pada *activity diagram*

6.		*Join*	Penggabungan yang menjadi arah aliran pada *activity diagram*

2.3.14.6 Component Diagram

Component diagram menggambarkan struktur fisik kode dari komponen. Komponen dapat berupa *sourcecode*, komponent biner, atau *executable component*. Sebuah komponen berisi informasi tentang *logic clas*s atau *class* yang diimplementasikan sehingga membuat pemetaan dari *logical view* ke *component view* (Kusumo 2004). Simbol-simbol yang digunakan pada component *diagram* disajikan pada Tabel 2 .6.

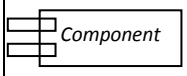
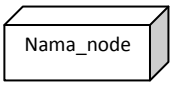
Tabel 2.6. Simbol *Component Diagram*

NO	GAMBAR	NAMA	KETERANGAN
1.		*Component*	Sebuah komponen melambangkan sebuah entitas *software* dalam sebuah sistem. Sebuah komponen dinotasikan sebagai sebuah kotak segiempat dengan dua kotak kecil tambahan yang menempel disebelah kirinya.
2.		*Dependency*	Sebuah *Dependency* digunakan untuk menotasikan relasi antara dua komponen.

2.3.14.7 Deployment Diagram

Deployment Diagram menggambarkan arsitektur fisik dari perangkat keras dan perangkat lunak sistem, menunjukkan hubungan komputer dengan perangkat (*nodes*) satu sama lain dan jenis hubungannya. Di dalam *nodes*, *executeable component* dan *object* yang dialokasikan untuk memperlihatkan unit perangkat lunak yang dieksekusi oleh *node* tertentu dan ketergantungan komponen (Kusumo 2004). Simbol-simbol yang digunakan pada component *diagram* disajikan pada Tabel 2.7.

Tabel 2.7. Simbol *Deployment Diagram*

NO	GAMBAR	NAMA	KETERANGAN
1.		*Component*	Komponen-komponen yang ada diletak-kan didalam *node*.
2.		*Node*	*Node* menggambarkam bagian-bagian *hardware* dalam sebuah sistem.
3.		*Association*	Sebuah *association* digambarkan sebagai sebuah garis yang menghubungkan dua *node* yang mengindikasikan jalur komunikasi antara element-elemen *hardware*.

2.3.14.8 Metode Pengujian Sistem *Black Box Testing*

Terfokus pada apakah unit program memenuhi kebutuhan (*requirement*) yang disebutkan dalam spesifikasi. Pada *black box testing*, cara mengujian hanya dilakukan dengan menjalankan atau mengeksekusi unit atau modul, kemudian diamati apakah hasil dari unit itu sesuai dengan proses bisnis yang diinginkan.

Black-Box testing berusaha untuk menemukan kesalahan dalam kategori berikut:

1. Fungsi yang tidak benar atau fungsi yang hilang.
2. Kesalahan antarmuka.
3. Kesalahan dalam struktur data atau akses *database* eksternal.
4. Kesalahan perilaku *(behavior)* atau kesalahan kinerja.
5. Inisialisasi dan pemutusan kesalahan.

Tes ini dirancang untuk menjawab beberapa pertanyaan-pertanyaan berikut ini:

1. Bagaiman validasi fungsional diuji?
2. Bagaimana perilaku dan kinerja sistem diuji?
3. Apa kelas *input* akan membuat kasus uji yang baik?
4. Apakah sistem sensitive terhadap nilai *input* tertentu?

5. Bagaimana batas-batas kelas data yang terisolasi?
6. Kecepatan dan volume data seperti apa yang dapat ditolerir sistem?
7. Efek apakah yang akan menspesifikasikan kombinasi data dalam sistem operasi?