# BAB 2. TINJAUAN PUSTAKA

## 2.1 Penelitian Terdahulu

Penelitian terdahulu ini menjadi salah satu acuan penulis dalam melakukan penelitian sehingga penulis dapat memperkaya teori yang digunakan dalam mengkaji penelitian yang dilakukan. Dari penelitian terdahulu, penulis tidak menemukan penelitian dengan judul yang sama seperti judul penelitian penulis. Namun penulis mengangkat beberapa penelitian sebagai referensi dalam memperkaya bahan kajian pada penelitian penulis. Berikut merupakan penelitian terdahulu berupa beberapa jurnal terkait dengan penelitian yang dilakukan penulis.

**Tabel 2 - 1 Penelitian Terdahulu**

| No | Judul Penelitian | Nama Peneliti | Penjelasan Penelitian | Pembeda |
|---|---|---|---|---|
| 1. | Aplikasi Sistem Informasi Geografis Pelayanan Kesehatan Kota Depok Berbasis Web Menggunakan Quantum GIS. [2] | Endah Dharmaputeri | Penelitian yang dilakukan mengenai sebuah aplikasi sistem informasi yang berbasis 'WebGIS Pelayanan Kesehatan Kota Depok' yang mampu menyajikan peta digital yang didalamnya terdapat informasi mengenai titik-titik lokasi sarana kesehatan dan Dinas Kesehatan di Kota Depok serta informasi yang terkait didalamnya. | Penelitian yang sekarang akan dilaksanakan ruang lingkupnya lebih luas se- Provinsi Jawa Barat. |
| 2. | Implementasi QGIS | Mella | Penelitian yang | Penelitian yang |

| | | | | |
|---|---|---|---|---|
| | Dalam Hal Pemetaan Informasi Lokasi Rumah Sakit Di Kota Bandung (Studi Kasus Dinas Kesehatan Kota Bandung) [3] | Rahmawati | dilakukan mengenai penyebaran lokasi rumah sakit, fasilitas dan jam praktek dokter yang berada di Kota Bandung. | sekarang akan dilaksanakan ruang lingkupnya se Provinsi Jawa Barat dan data yang cakupannya lebih luas seperti fasilitas-fasilitas kesehatan yang berada di Provinsi Jawa Barat. |
| 3. | Sistem Informasi Geografis Pemetaan Klinik Bersalin Di Kabupaten Bantul [4] | Rinawati Puji Astuti | Pada penelitian ini output nya berupa program SIG berbasis web, dengan menggunakan Map Server | Software yang digunakan pada penelitian yang akan dilaksanakan menggunakan Qgis, dan ruang lingkupnya pun bukan saja membahas mengenai pemetaan klinik bersalin |
| 4. | Rancang Bangun Sistem Pemetaan Tempat Pelayanan Kesehatan Di Kabupaten Klaten [5] | Rahmatullah Priyo Kusuma | Pada penelitian ini membangun sistem informasi geografis berbasis web yang menampilkan pelayanan kesehatan Kabupaten Klaten menggunakan metode *SDLC (System Development Life Cycle)* | Pada penelitian yang akan dilaksanakan menggunakan metode *RUP (Rational Unified Process)* |

| | | | | |
|---|---|---|---|---|
| 5. | Implementasi Sistem Informasi Geografis Menggunakan Google Maps API dalam pemetaan asal mahasiswa. [6] | Fauzan Masykur | Pemetaan mahasiswa di Fakultas Teknik Universitas Muhammadiyah Ponorogo. Pemetaan mahasiswa tersebut dilakukan dengan memanfaat peta yang sudah disediakan oleh google yakni google Maps API. Google Maps adalah layanan *free* yang diberikan oleh google. | Penelitian saat ini menggunakan Software Quantum gis dan basis data PostgreSQL, dan menggukan website *one page* untuk menjadi wadah atau *interface* bagi pengguna yang akan mengakses. |

## 2.2 Kesehatan

Menurut Undang-Undang Republik Indonesia Nomor 36 Tahun 2009 [7]. Kesehatan adalah keadaan sehat, baik secara fisik, mental, spritual maupun sosial yang memungkinkan setiap orang untuk hidup produktif secara sosial dan ekonomis. Sumber daya di bidang kesehatan adalah segala bentuk dana, tenaga, perbekalan kesehatan, sediaan farmasi dan alat kesehatan serta fasilitas pelayanan kesehatan dan teknologi yang dimanfaatkan untuk menyelenggarakan upaya kesehatan yang dilakukan oleh Pemerintah, pemerintah daerah, dan/atau masyarakat.

## 2.3 Dinas Kesehatan Provinsi Jawa Barat

Dinas Kesehatan sebagai salah satu Satuan Kerja Perangkat Daerah Pemerintah Provinsi Jawa Barat berkepentingan untuk memberikan kontribusi yang bermakna dalam mewujudkan Visi dan Misi Pemerintah Provinsi Jawa Barat dengan mempertimbangkan kesesuaian dan keterkaitan dengan Visi dan Misi Kementerian Kesehatan serta Visi Pembangunan Provinsi Jawa Barat, maka telah disusun Visi Dinas Kesehatan Provinsi Jawa Barat yaitu : ”Masyarakat yang

Mandiri Untuk Hidup Sehat" Untuk mewujudkan pencapaian visi yang telah ditetapkan dengan memperhatikan kondisi dan permasalahan yang ada, tantangan ke depan, serta memperhitungkan peluang yang dimiliki, maka ditetapkan 4 (empat) misi Pembangunan Kesehatan di Jawa Barat sebagai berikut:

1) Membangun kemandirian masyarakat untuk hidup sehat
2) Menjamin pelayanan kesehatan yang prima
3) Mendukung sumber daya pembangunan kesehatan
4) Regulator pembangunan kesehatan di Jawa Barat Mobilitas Penduduk

Mobilitas Penduduk dibagi menjadi dua macam yaitu: mobilitas penduduk vertikal atau perubahan status dan mobilitas penduduk horizontal atau mobilitas penduduk geografis [8].

## 2.4 Peta

Peta merupakan gambaran wilayah geografis, bagian permukaan bumi yang disajikan dalam berbagai cara yang berbeda, mulai dari peta konvensional yang tercetak hingga peta digital yang tampil di layar komputer. Peta dapat digambarkan dengan berbagai gaya, masing-masing menunjukkan permukaan yang berbeda untuk subjek yang sama untuk memvisualisasikan dunia dengan mudah, informatif dan fungsional.

Peta berbasis komputer (digital) lebih serba guna dan dinamis karena bisa menunjukkan banyak tampilan yang berbeda dengan subjek yang sama. Peta ini juga memungkinkan perubahan skala, animasi gabungan, gambar, suara, dan bisa terhubung ke sumber informasi tambahan melalui internet. Peta digital dapat diperbaharui ke peta tematik baru dan bisa menambahkan detail informasi geografi lainnya [9].

## 2.5 Peta Tematik

Dalam pemetaan tematik [10], data divisualisasikan berdasarkan kategori tertentu dengan konsep spasial. Konsep spasial berasal dari beberapa data spasial. Data spasial terdiri dari data *raster* dan data *vector*, dimana data *raster* yaitu data yang berasal dari foto-foto seperti foto satelit maupun foto yang dihasilkan dari kamera telepon seluler, sedangkan data *vector* merupakan data peta berdasarkan garis, titik, atau *polygon*. Contoh dari peta tematik yaitu visualisasi jumlah

penduduk di suatu daerah. Berdasarkan variabel grafis, peta diklasifikasikan dalam berbagai jenis, yaitu:

a. *Chorochromatic maps*, menampilkan perbandingan data kualitatif dengan menggunakan warna sebagai pembeda.
b. *Choropleth maps*, menampilkan perbandingan kuantitas dengan menggunakan nilai atau warna sebagai pembeda.
c. *Proportional symbol maps*, menampilkan kuantitas dengan besaran ukuran peta.
d. *Diagram maps*, menggunakan diagram untuk menampilkan suatu titik atau area.
e. *Flow maps*, menampilkan alur, atau arah dari pergerakan.
f. *Dot maps*, merepresentasikan distribusi fenomena dengan titik untuk menunjukan persamaan kuantitas

## 2.6 Pengertian Sistem Informasi

Sistem informasi adalah sekumpulan komponen yang membentuk suatu sistem yang memiliki koneksi antara satu komponen dan komponen lain yang bertujuan untuk menghasilkan informasi dalam bidang tertentu. Sistem informasi memerlukan klasifikasi aliran suatu informasi karena keragaman kebutuhan informasi pengguna informasi. Kriteria untuk sistem informasi fleksibel, efektif dan efisien [11].

## 2.7 Sistem Informasi Geografis

Istilah sistem informasi geografis merupakan gabungan dari tiga unsur pokok yaitu sistem, informasi dan geografis. Dengan demikian pengertian terhadap tiga unsur tersebut akan sangat membantu dalam memahami sistem informasi geografis. Dengan demikian maka sistem informasi geografis merupakan sistem informasi dengan tambahan unsur geografis.

Istilah sistem informasi geografis mengandung pengertian informasi mengenai tempat-tempat yang terletak di permukaan bumi, pengetahuan mengenai posisi dimana suatu obejek terletak di permukaan bumi dan informasi mengenai keterangan-keterangan yang terdapat di permukaan bumi yang posisinya diberikan atau diketahui. Oleh karena itu sistem informasi geografis adalah suatu kesatuan yang terdiri dari berbagai sumberdaya fisik dan logika yang berkenaan dengan

objek-objek yang terdapat di permukaan bumi, sehingga sistem informasi geografis merupakan perangkat lunak yang dapat digunakan untuk pemasukan, penyimpanan, manipulasi dan menampilkan informasi geografis beserta dengan keterangan-keterangannya.

Istilah *geography* digunakan karena SIG dibangun berdasarkan pada geografi atau spasial. Objek ini mengarah pada spesifikasi lokasi dalam suatu *space*. *Geographic Information System* (GIS) merupakan sistem komputer yang berbasis pada sistem informasi yang digunakan untuk memberikan bentuk digital dan analisis terhadap permukaan geografi bumi [12].

Geografi adalah informasi mengenal permukaan bumi dan semua obyek yang berada diatasnya, sedangkan sistem informasi geografis(SIG) atau dalam bahasa inggris disebut *Geographic Information System* (GIS) adalah sistem informasi khusus yang mengelola data yang memiliki informasi spasial (bereferensi keruangan). Sistem informasi geografis adalah bentuk sistem informasi yang menyajikan informasi dalam bentuk grafis denganmenggunakan peta sebagai antarmuka. SIG tersusun atas konsep beberapa lapisan (*layer*) dan relasi.

## 2.8 Quantum GIS

Quantum GIS atau yang lebih dikenal dengan QGIS adalah sistem informasi geografis *open source* yang berlisensi GNU *General Public License*. QGIS adalah proyek resmi dari organisasi *Open Source* Geospatial Foundation (OSGeo). QGIS dapat berjalan di berbagai sistem operasi seperti Linux, Windows, Mac OS dan Android, serta mendukung berbagai format dan fungsionalitas vektor, raster dan basis data [13].

QGIS merupakan alat yang dapat digunakan untuk mengelola data spasial dan data yang bereferensi geografi seperti *input*, manajemen, proses serta *output*. QGIS juga memiliki kemampuan untuk bekerjasama dengan paket aplikasi komersial terkait. QGIS menyediakan semua fungsionalitas dan fitur-fitur yang dibutuhkan oleh pengguna sistem informasi geografis pada umumnya. Menggunakan *plugin* dan fitur inti dimungkinkan untuk visualisasi pemetaan untuk kemudian diubah dan dicetak sebagai sebuah peta yang langkap. QGIS terhubung dengan OSGeo4W sebagai perangkat lunak geospasial. OSGeo4W

membutuhkan *server* dalam menampilkan peta, yang mana terdapat QGIS *Server* didalam perangkat OSGeo4W.

## 2.9 QGIS *Server*

QGIS *Server* merupakan perangkat *open source*, selain itu QGIS *Server* juga dapat mengimplementasikan pemetaan tematik. QGIS *Server* merupakan aplikasi CGI (*Common Gateway Interface*, yang merupakan penghubung program aplikasi kedalam halaman *web*) yang ditulis dalam C++ yang bekerja dengan *web server* (seperti Apache dan Lighttpd).

QGIS *Server* menggunakan QGIS Desktop sebagai *back end* untuk melakukan proses pembuatan peta sekaligus pembuatan logika SIG. Karena QGIS Desktop dan QGIS *Server* menggunakan visualisasi *libraries* yang sama, maka peta yang dipublikasikan di *web* melalui QGIS *Server* terlihat sama seperti peta yang telah di buat di GIS Desktop [14].

## 2.10 WEB GIS

WEB GIS adalah jenis sistem informasi terdistribusi, yang terdiri dari setidaknya *server* dan *client*, di mana *server* yang dimaksud adalah GIS *server* dan *client* yang dimaksud adalah *browser web*, aplikasi *desktop*, atau aplikasi *mobile*. Dalam bentuk yang paling sederhana, WEB GIS dapat didefinisikan sebagai SIG yang menggunakan *server web* untuk berkomunikasi antara *server* dan *client* [19].

Dengan memanfaatkan internet untuk mengakses informasi melalui *web* tanpa memperhatikan seberapa jauh jarak *server* dan *client* dari satu sama lain, WEB GIS memperkenalkan keunggulan berbeda dibandingkan GIS *desktop* tradisional, termasuk yang berikut:

1. Jangkauan global
2. Jumlah pengguna yang banyak
3. Kemampuan lintas *platform* yang lebih baik
4. Biaya rendah
5. Mudah digunakan
6. Pembaruan terpadu
7. Beragam aplikasi

## 2.11 PostgreSQL

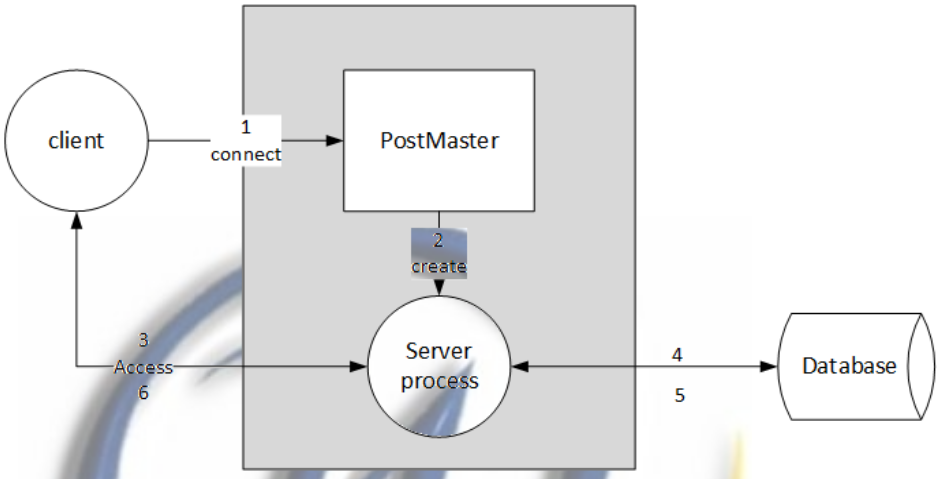
PostgreSQL atau sering disebut Postgres merupakan salah satu database besar yang menawarkan skalabilitas, keluwesan, dan kinerja yang tinggi. PostgreSQL termasuk dalam kategori free open source software (FOSS). Penggunaannya begitu meluas di berbagai platform dan didukung oleh banyak bahasa pemrograman.

PostgreSQL adalah salah satu sistem perangkat lunak aplikasi berbasis data (DBMS) yang bersifat objek-relasional (ORDBMS) dan masih memiliki fitur-fitur khas DBMS tradisional, tetapi dengan sejumlah perbaikan unjuk-kerja dan fungsional sebagaimana juga bisa ditemukan pada sistem-sistem DBMS generasi pada saat ini. Selain itu, postgreSQL juga merupakan sistem perangkat lunak yang bersifat *free* dan *open source*. Sebagian dikembangkan oleh anggota-anggota tim atau kelompok komunitas pengembang yang sebagian besar tersebar di dunia dan tidak dibayar secara khusus atau bersifat *volunteer*. Para anggota tim pengembang ini saling berkomunikasi dan berkoordinasi melalui internet dan menyebut pekerjaan mulia tersebut sebagai proyek komunitas yang sama sekali tidak dikendalikan oleh perusahaan tertentu.

PostgreSQL merupakan sistem perangkat lunak aplikasi DBMS *open source* turunan sistem sejenis yang berasal dari Universitas California yang berada di Berkeley, Amerika Serikat. Perangkat lunak DBMS ini memiliki beberapa keuntungan atau kelebihan dengan mendukung sebagian besar pengimplementasian konsep-konsep SQL standar, berikut beberapa fitur umum DBMS modern seperti berikut:

1 Mengimplementasikan konsep relasional yang berorientasi objek
2 Mengimplementasikan konsep integritas referensial (*reference integrity*)
3 Mendukung standar *query* SQL
4 Mendukung multi bahasa pemrograman prosedur atau *trigger*.
5 Dilengkapi dengan berbagai variasi tipe data
6 Mendukung atau memfasilitasi kebutuhan penyimpanan data objek (tipe biner) yang berukuran cukup besar seperti file gambar, suara dan video
7 Mendukung penggunaan berbagai program aplikasi *client* API

8 Jika memiliki suatu masalah seputar DBMS PostgreSQL, para pengguna dapat segera mengajukan masalah, pertanyaan atau solusi dalam komunitas diskusi yang terdapat pada *website* resmi PostgreSQL. Dalam komunitas tersebut semua pihak baik pengguna maupun pengembang dapat berdiskusi dan memecahkan masalah.




**Gambar 2 - 1 Skema koneksi clent/server DBMS PostgreSQL**

Salah satu kekuatan PostgreSQL adalah arsitektur *client/server*. Dengan arsitektur seperti ini, banyak manfaat yang akan mengalir baik pada pengembang aplikasi maupun pada para penggunanya. Dengan arsitektur *client/server* maka paket DBMS menugaskan PostMaster agar terus berjaga atau menunggu *request* (permohonan koneksi ke basis data) yang masuk dan berasal dari program aplikasi *client*. Jika terdapat *request* yang datang maka PostMaster akan segera membuat sebuah proses *server* baru yang secara khusus akan melayani detail *request* dari *client* yang bersangkutan untuk mengakses data yang terdapat di dalam tabel-tabel basis data yang diminta [16].

## 2.12 PostGIS

PostGIS adalah suatu program, *tool*, *add-on*, *spatial database extender*, *spatial database engine*, atau *extension* yang dapat menambah dukungan dalam pendefinisian dan pengelolaan unsur-unsur spasial bagi DBMS objek relasional PostgreSQL. Secara praktis, PostGIS berperan sebagai penyedia layanan spasial bagi DBMS ini, memungkinkan PostgreSQL untuk digunakan sebagai *backend*

basis data spasial. Singkatnya, PostGIS juga menambahkan tipe-tipe *query* SQL, operator dan fungsi-fungsi analisis yang kemudian menyebabkan DBMS PostgreSQL menjadi bersifat *Spatially-enabled*.

Hingga saat ini PostGIS masih dikembangkan oleh institusi bernama *Refractions Research*. Berkaitan dengan hal ini, *Refractions Research* masih melanjutkan proses pengembangan PostGIS sedemikian rupa hingga dapat menyediakan beberapa fitur-fitur seperti *user inteface tool*, dukungan topologi dasar, transformasi koordinat, validasi data dan fasilitas pemrograman API. Pengembangan proyek berikutnya akan dilanjutkan pada penyediaan dukungan topologi secara penuh, dukungan terhadap data raster, jaringan, permukaan tiga dimensi, kurva dan unsur lainnya.

Sebagai perangkat lunak spatial database extender yang bersifat *open source* dan *free*, PostGIS memiliki beberapa fitur yang menjadi unggulan. Fitur-fitur tersebut antara lain yaitu:

1 Mengelola tipe unsur spasial dasar geometri seperti garis, titik dan area
2 Mengelola tipe unsur spasial lanjutan seperti *multipoints, multilinestrings* dan *multipolygons*
3 Menyediakan operator spasial untuk menentukan pengukuran seperti jarak, luas, panjang dan keliling
4 Menyediakan operator spasial untuk menentukan operasi spasial seperti *union/overlay, difference* dan *buffer*
5 Mendukung pengelolaan tipe data raster [16]

## 2.13 Metodologi Penelitian C.R Kothari

Proses penelitian terdiri dari serangkaian tindakan atau langkah-langkah yang diperlukan untuk secara efektif melakukan penelitian dan urutan yand diinginkan dari langkah-langkah tersebut. Untuk menghindari masalah yang timbul dari proses penelitian, maka diperlukan prosedur-prosedur yang sebelumnya diperhitungkan.

Menurut C. R. Kothari [17], langkah-langkah detail proses untuk memberikan panduan prosedural yang berguna dalam proses penelitian adalah sebagai berikut:

1. *Formulating the Research Problem*

Ada dua jenis masalah penelitian, yaitu yang berhubungan dengan keadaan alamiah dan yang berhubungan dengan hubungan antar variabel. Pada saat awal peneliti harus memilih masalah yang ingin dipelajari, yaitu harus memutuskan area umum yang menarik atau aspek dari subjek yang ingin ditanyakan. Awalnya masalah dapat dinyatakan dengan cara umum yang luas dan kemudian ambiguitas, jika ada, terkait dengan masalah tersebut diselesaikan. Kemudian, kelayakan solusi tertentu harus dipertimbangkan sebelum melakukan rumusan masalah, dengan demikian, identifikasi masalah merupakan langkah pertama dalam melakukan penelitian ilmiah. Pada dasarnya dua langkah dilibatkan merumuskan masalah penelitian, yaitu, memahami masalah secara menyeluruh, dan menyusun ulang sama menjadi istilah yang bermakna dari sudut pandang analitis.

2. *Extensive Literature Survey*

Pada langkah ini, peneliti harus melakukan survei literatur yang luas terkait dengan masalah. Untuk tujuan ini, diperlukan jurnal yang diterbitkan atau tidak dipublikasikan, bibliografi adalah tempat pertama yang dikunjungi. Jurnal akademik, proses konferensi, laporan, buku, dll., harus di*filter* tergantung pada sifat masalahnya. Dalam proses ini, seharusnya Ingatlah bahwa satu sumber akan mengarah ke yang lain.

3. *Development of Working Hypotheses*

Setelah survei literatur dilakukan, peneliti harus nyatakan dengan jelas hipotesis kerja atau hipotesis. Hipotesis kerja adalah asumsi tentatif dibuat untuk menarik dan menguji konsekuensi logis atau empirisnya. Dengan demikian cara masuk hipotesis penelitian mana yang dikembangkan sangat penting karena mereka memberikan titik fokus untuk penelitian. Mereka juga mempengaruhi cara di mana tes harus dilakukan dalam analisis data dan secara tidak langsung kualitas data yang diperlukan untuk analisis. Dalam sebagian besar jenis penelitian, pengembangan hipotesis kerja memainkan peran penting. Hipotesis harus sangat spesifik dan terbatas pada bagian penelitian di tangan karena harus diuji. Peran hipotesis adalah untuk membimbing peneliti dengan membatasi bidang penelitian dan membuatnya tetap di jalur yang benar. Pemikirannya menajam dan memusatkan perhatian pada aspek-aspek masalah yang lebih penting. Ini juga

menunjukkan jenis data yang dibutuhkan dan jenis metode analisis data yang akan digunakan.

4. *Preparing the Research Design*

Masalah penelitian telah dirumuskan dalam bentuk yang jelas syaratnya, peneliti akan diminta untuk menyiapkan desain penelitian, yaitu, ia harus menyatakan struktur konseptual di mana penelitian akan dilakukan. Persiapan desain semacam itu memfasilitasi penelitian agar seefisien mungkin menghasilkan informasi yang maksimal. Dengan kata lain, fungsi desain penelitian adalah untuk menyediakan pengumpulan bukti yang relevan dengan pengeluaran minimal usaha, waktu dan uang. Tetapi bagaimana semua ini dapat dicapai tergantung pada penelitian tujuan.

5. *Determining Sample Design*

Semua item yang dipertimbangkan dalam bidang pertanyaan apa pun merupakan ‘semesta’ atau ‘populasi’. Penghitungan lengkap semua item dalam 'populasi' dikenal sebagai penyelidikan sensus. Dapat dianggap bahwa dalam penelitian seperti itu ketika semua item tercakup elemen peluang dibiarkan dan akurasi tertinggi diperoleh. Tetapi dalam praktiknya ini mungkin tidak benar. Bahkan elemen sekecil apa pun dalam penyelidikan semacam itu akan menjadi lebih besar dan lebih besar sebagai jumlah pengamatan meningkat. Selain itu, tidak ada cara untuk memeriksa elemen atau luasnya kecuali melalui *resurvey* atau penggunaan sampel cek. Selain itu, jenis penyelidikan ini melibatkan banyak waktu, uang dan energi.

6. *Collecting the Data*

Dalam berurusan dengan masalah kehidupan nyata, sering ditemukan bahwa data yang ada tidak memadai, dan karenanya, perlu untuk mengumpulkan data yang sesuai. Ada beberapa cara pengumpulan data yang sesuai yang sangat berbeda dalam konteks biaya uang, waktu dan sumber daya lain yang tersedia bagi peneliti. Data primer dapat dikumpulkan baik melalui eksperimen atau melalui survei. Jika peneliti melakukan eksperimen, ia mengamati beberapa pengukuran kuantitatif, atau data, dengan bantuan yang dia periksa kebenaran yang terkandung dalam hipotesisnya.

7. *Execution of the Project*

Eksekusi proyek adalah langkah yang sangat penting dalam proses penelitian. Jika pelaksanaan proyek berlangsung pada jalur yang benar, data yang akan dikumpulkan akan memadai dan dapat diandalkan. Peneliti harus melihat bahwa proyek dijalankan secara sistematis tepat dan sesuai waktu. Jika survei akan dilakukan dengan menggunakan kuesioner terstruktur, data dapat siap diproses. Dalam situasi seperti itu, pertanyaan serta jawaban yang mungkin mungkin berkode. Jika data harus dikumpulkan melalui pewawancara, pengaturan harus dibuat dengan benar seleksi dan pelatihan pewawancara. Pelatihan dapat diberikan dengan bantuan instruksi manual yang menjelaskan dengan jelas pekerjaan pewawancara pada setiap langkah. Pemeriksaan lapangan sesekali harus dibuat untuk memastikan bahwa pewawancara melakukan pekerjaan yang ditugaskan dengan tulus dan efisien.

8. *Analysis of Data*

Setelah data dikumpulkan, peneliti beralih ke tugas menganalisis mereka. Analisis data memerlukan sejumlah operasi yang terkait erat seperti pendirian kategori, penerapan kategori ini untuk data mentah melalui pengkodean, tabulasi dan kemudian menggambar kesimpulan statistik. Data yang berat harus diringkas menjadi beberapa yang dapat dikelola kelompok dan tabel untuk analisis lebih lanjut. Dengan demikian, peneliti harus mengklasifikasikan data mentah menjadi beberapa kategori terarah dan bermanfaat. Operasi pengkodean biasanya dilakukan pada tahap ini melalui mana kategori data ditransformasikan menjadi simbol yang dapat ditabulasi dan dihitung. Editing adalah prosedur yang meningkatkan kualitas data untuk pengkodean. Dengan pengkodean panggung siap untuk tabulasi.

9. *Hypotheses-Testing*

Setelah menganalisis data sebagaimana dinyatakan di atas, peneliti berada dalam posisi untuk uji hipotesisnya, jika ada, yang dia rumuskan sebelumnya. Apakah fakta mendukung hipotesis atau kebalikannya? Ini adalah pertanyaan biasa yang harus dijawab saat menguji hipotesis. Berbagai tes, seperti uji Chi square, uji-t, uji-F, telah dikembangkan oleh ahli statistik untuk tes tujuan seperiti ini. Hipotesis dapat diuji melalui penggunaan satu atau lebih dari tes tersebut,

tergantung sifat dan objek penyelidikan penelitian. Pengujian hipotesis akan menghasilkan menerima hipotesis atau menolaknya. Jika peneliti tidak memiliki hipotesis untuk memulai, generalisasi didirikan pada dasar data dapat dinyatakan sebagai hipotesis untuk diuji oleh penelitian selanjutnya di masa mendatang.

10. *Generalisations and Interpretation*

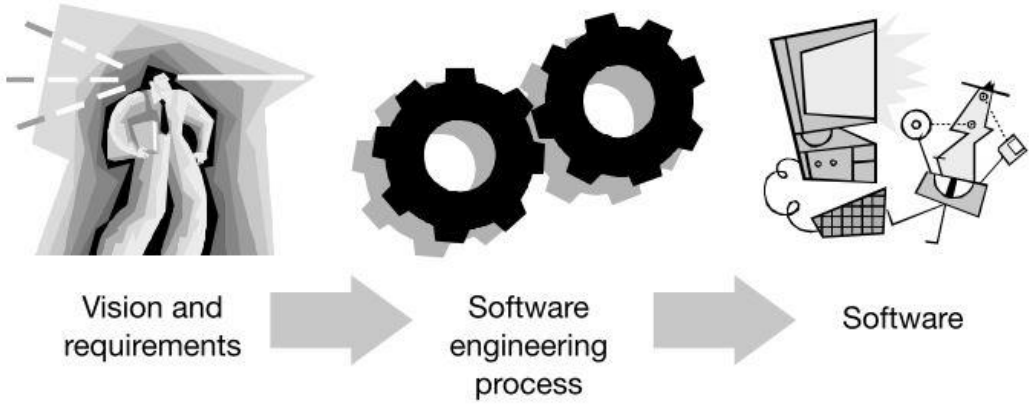
Jika suatu hipotesis diuji dan ditegakkan beberapa kali, itu mungkin bagi peneliti untuk sampai pada generalisasi, yaitu, untuk membangun teori. Faktanya, nilai sebenarnya dari penelitian terletak pada kemampuannya untuk sampai pada generalisasi tertentu. Jika peneliti tidak punya hipotesis awalnya, dia mungkin berusaha menjelaskan temuannya berdasarkan beberapa teori. Yang diketahui sebagai interpretasi. Proses penafsiran mungkin cukup sering memicu pertanyaan baru yang masuk dapat menyebabkan penelitian lebih lanjut.

11. *Preparation of the Report or Presentation of the Result*

Peneliti harus menyiapkan laporan apa yang telah dilakukan olehnya, apakah berbentuk laporan atau persentasi hasil penelitian.

## 2.14 *Unified Process*

*Software Development Process* (SDP), atau yang dikenal juga sebagai *Software Engineering Process* (SEP), menjelaskan tentang siapa, apa, kapan dan bagaimana cara mengembangkan perangkat lunak. Sebagaimana tentang penjelasan gambar berikut ini, SEP adalah proses dimana kita mengubah kebutuhan pengguna menjadi perangkat lunak.




**Gambar 2 - 2 *Unified Process[18]***

*Unified Software Development Process* (USDP) adalah standar industri SEP dari pengembang *Unified Modeling Language* (UML) dan biasa disebut sebagai proses terpadu atau *Unified Process* (UP) [19].

### 2.14.1 Sejarah *Rational Unified Process*

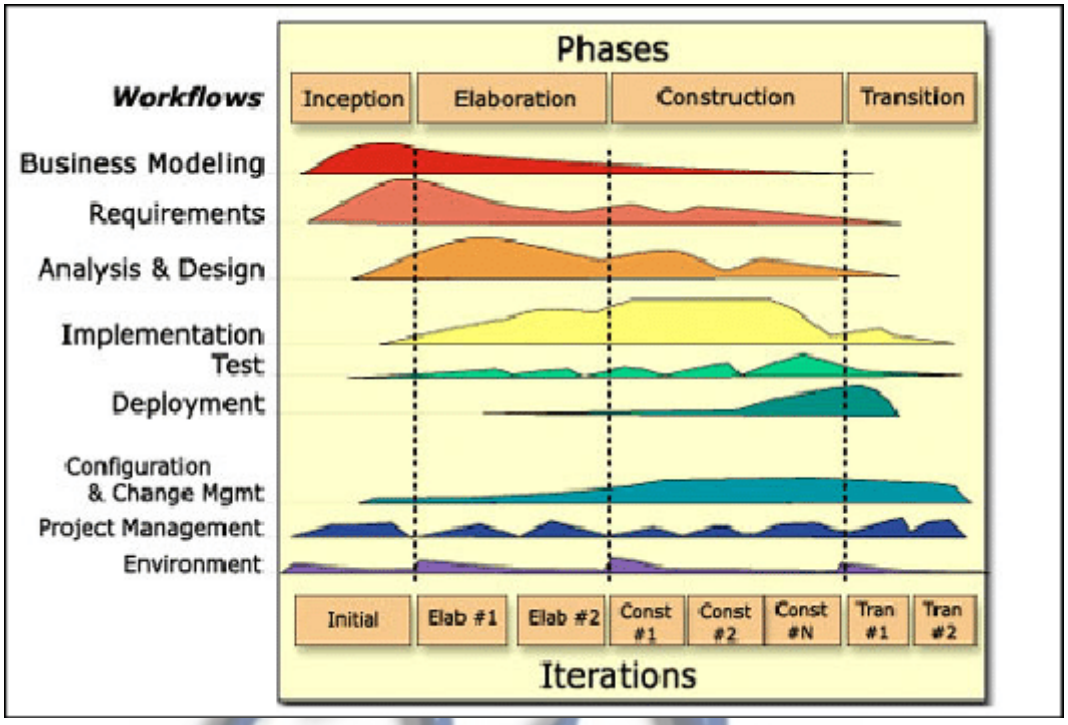
RUP, singkatan dari *Rational Unified Process*, adalah suatu kerangka kerja proses pengembangan perangkat lunak_iteratif yang dibuat oleh *Rational Software*, suatu divisi dari IBM sejak 2003. RUP bukanlah suatu proses tunggal dengan aturan yang konkrit, melainkan suatu kerangka proses yang dapat diadaptasi dan dimaksudkan untuk disesuaikan oleh organisasi pengembang dan tim proyek perangkat lunak yang akan memilih elemen proses sesuai dengan kebutuhan mereka.

RUP merupakan produk proses perangkat lunak yang awalnya dikembangkan oleh *Rational Software*. *Rational Software* diakuisisi oleh IBM pada februari 2003. Produk ini memuat basis-pengetahuan yang bertautan dengan artefak sederhana disertai deskripsi detail dari beragam aktivitas. RUP dimasukkan dalam produk IBM Rational Method Composer (RM C) yang memungkinkan untuk kustomisasi proses [20].

### 2.14.2 *Rational Unified Process*

*Rational Unified Process* (RUP) adalah proses rekayasa perangkat lunak yang terdefinisi dengan baik dan terstruktur dengan baik. RUP dengan jelas mendefinisikan mengenai siapa yang bertanggung jawab atas apa, bagaimana hal-hal dilakukan, dan kapan melakukannya. RUP juga menyediakan struktur yang terdefinisi dengan baik untuk siklus hidup proyek RUP, dengan jelas mengartikulasikan tonggak penting dan poin keputusan. Dalam proses rekayata perangkat lunak, RUP menggunakan artefak sebagai parameter sehingga RUP menggunakan pendekatan *Object Oriented* karena dalam *Object Oriented* kegiatan beroprasi atau bergantung pada objek aktif [21].

*Rational Unified Process* mengandung 2 dimensi yaitu aspek dinamis (yang digambarkan secara horizontal) dan aspek statis (yang digambarkan secara vertikal).




**Gambar 2 - 3 *Rational Unified Process***

Aspek dinamis merepresentasikan waktu dan menunjukan dari proses, yaitu siklus, tahap, iterasi dan milestone. RUP terdiri dari 4 fase, yaitu

1. Fase *Inception*

*Inception* adalah fase pertama dari siklus hidup RUP. *Inception* memiliki seperangkat tujuan yang jelas dan disimpulkan oleh *Lifecycle Objctive Milestone*. Gunakan tujuan-tujuan ini untuk membantu memutuskan kegiatan mana yang harus dilakukan dan penentuan perangkat lunak yang akan dihasilkan.

2. Fase *Elaboration*

*Elaboration* merupakan fase kedua dalam RUP. *Elaboration* merupakan tahap untuk menentukan aktifitas (*Use Case*) dari perangkat lunak serta perancangan arsitekturnya.

3. Fase *Construction*

*Construction* yaitu membangun produk perangkat lunak secara lengkap yang siap diserahkan kepada pemakai.

4. Fase *transition*

*Transition* yaitu menyerahkan perangkat lunak kepada pemakai, mengujinya di tempat pemakai, dan memperbaiki masalah-masalah yang muncul saat dan setelah pengujian.

Aspek statis merepresentasikan aktivitas, pelaksana kerja dan aliran kerja. Pada RUP terdapat 2 jenis aliran kerja yaitu aliran kerja utama dan aliran kerja pendukung. Untuk detail aliran kerja dijelaskan sebagai berikut:

1. Aliran Kerja Utama
   a. Pemodelan Bisnis (*Business Modeling*) mendeskripsikan struktur dan proses-proses bisnis organisasi
   b. Kebutuhan (*Requirements*) mendefinisikan kebutuhan perangkat lunak dengan menggunakan metode *Use Case*.
   c. Analisis dan perancangan (*Analysis and Design*) mendeskripsikan berbagai arsitektur perangkat lunak dari berbagai sudut pandang.
   d. Implementasi (*implementation*) menulis kode-kode program, dan mengintegrasikan unit-unit program.
   e. Pengujian (*Testing*) mendeskripsikan kasus uji, prosedur, dan alat ukur pengujian.
   f. *Deployment* menangani konfigurasi sistem yang akan diserahkan

2. Aliran Kerja Pendukung
   a. Manajemen konfigurasi dan perubahan (*configuration and change management*) mengendalikan perubahan dan memelihara artefak-artefak proyek.
   b. Manajemen proyek (*project management*) mendeskripsikan berbagai strategi pekerjaan dengan proses yang berulang.
   c. Lingkungan (*environment*) menangani infrastruktur yang dibutuhkan untuk mengembangkan sistem.

## 2.15 *Unified Modeling Language* (UML)

*Unified modeling language* (UML) adalah keluarga notasi grafis yang didukung oleh meta-model tunggal yang membantu pendeskripsian dan desain sistem perangkat lunak khususnya sistem yang dibangun menggunakan pemrograman berorientasi objek.

Selain itu UML adalah bahasa pemodelan yang menggunakan konsep orientasi *object*. UML dibuat oleh Grady Booch, James Rumbaugh, dan Ivar

Jacobson di bawah bendera *Rational Software Corps*. UML menyediakan notasi-notasi yang membantu memodelkan sistem dari berbagai prespetktif. UML tidak hanya digunakan dalam pemodelan perangkat lunak, namun hampir dalam semua bidang yang membutuhkan pemodelan [22].

### 2.15.1 *Use Case* Diagram

*Use case* diagram merupakan pemodelan untuk kelakuan (*behavior*) sistem yang akan dibuat. *Use case* mendeskripsikan sebuah interaksi antara satu atau lebih actor dengan sistem informasi yang akan dibuat. Secara kasar, *use case* digunakan untuk mengetahui fungsi apa saja yang ada di dalam sebuah sistem informasi dan siapa saja yang berhak menggunakan fungsi-fungsi itu.

Syarat penamaan pada *use case* adalah nama didefenisikan sesederhana mungkin dan dapat dipahami. Ada dua hal utama pada *use case* yaitu pendefenisian apa yang disebut *actor* dan *use case*.

1. Aktor merupakan orang, proses, atau sistem lain yang berinteraksi dengan sistem informasi yang akan dibuat di luar sistem informasi yang akan dibuat itu sendiri, jadi walaupun symbol dari actor adalah gambar orang, tapi actor belum tentu merupakan orang.

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

Berikut adalah symbol-simbol yang ada pada diagram *use case*.

**Tabel 2 - 2 *Use Case* Diagram**

| Simbol | Deskripsi |
|---|---|
| *Use case*<br>  | 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*<br>  <br>**nama actor** | orang, proses, atau sistem lain yang berinteraksi dengan sistem informasi yang akan dibuat di luar sistem informasi yang akan dibuat itu sendiri, jadi walaupun |

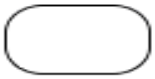
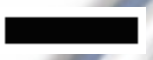
| | |
|---|---|
| | symbol dari actor adalah gambar orang, tapi actor belum tentu merupakan orang; biasanya menggunakan kata benda di awal frase nama *actor* |
| Asosiasi<br>——————— | komunikasi antara aktor dan *use case* yang berpartisipasi pada *use case* atau *use case* memiliki interaksi dengan actor |
| Ekstensi/ *extend*<br><<*extend*>><br>----------> | relasi *use case* tambahan ke sebuah *use case* dimana *use case* yang ditambahkan dapat berdiri sendiri walau tanpa *use case* tambahan itu; biasanya *use case* tambahan memiliki nama depan yang sama dengan *use case* yang ditambahkan |
| Generalisasi | hubungan generalisasi dan spesialisasi (umum – khusus) antara dua buah *use case* dimana fungsi yang satu adalah fungsi yang lebih umum dari lainnya. |
| Menggunakan/ *include*<br><<*include*>><br>----------> | relasi *use case* tambahan ke sebuah *use case* dimana *use case* yang ditambahkan memerlukan *use case* ini untuk menjalankan fungsinya atau sebagai syarat dijalankan *use case* ini |

### 2.15.2 *Activity* Diagram

Menggambarkan rangkaian aliran dari aktivitas, digunakan untuk mendeskripsikan aktivitas yang dibentuk dalam suatu operasi sehingga dapat juga digunakan untuk aktifitas lainnya. Diagram ini sangat mirip dengan *flowchart* karena memodelkan *workflow* dari satu aktivitas ke aktivitas lainnya atau dari aktivitas ke status. Pembuatan *activity diagram* pada awal pemodelan proses dapat membantu memahami keseluruhan proses. *Activity diagram* juga digunakan untuk menggambarkan interaksi antara beberapa *use case*. *Activity Diagram* melengkapi

*Use Case* dengan menampilkan representasi grafis dari aliran interaksi dalam skenario tertentu.

**Tabel 2 - 3 *Activity* Diagram**

| No | Gambar | Nama | Keterangan |
|---|---|---|---|
| **1** |  | *Activity* | Memperlihatkan bagaimana masing-masing kelas antarmuka saling berinteraksi satu sama lain |
| **2** |  | *Action* | State dari sistem yang mencerminkan eksekusi dari suatu aksi |
| **3** |  | *Initial Node* | Bagaimana objek dibentuk atau diawali. |
| **4** |  | *Activity Final Node* | Bagaimana objek dibentuk dan diakhiri |
| **5** |  | *Fork or Join Node* | Satu aliran yang pada tahap tertentu berubah menjadi beberapa aliran |

### 2.15.3 *Class* Diagram

*Class diagram* menggambarkan struktur sistem dari segi pendefenisian kelas-kelas yang akan dibuat untuk membangun sistem. Kelas memiliki apa yang disebut atribut dan metode atau operasi. Atribut merupakan variable-varibel yang dimiliki oleh suatu kelas sedangkan operasi atau metode adalah fungsi-fungsi yang dimiliki oleh suatu kelas.

Kelas-kelas yang ada pada struktur sistem harus dapat melakukan fungsi-fungsi sesuai dengan kebutuhan sistem. Susunan struktur kelas yang baik pada diagram kelas sebaiknya memiliki jenis-jenis kelas berikut :

1.Kelas main

2.Kelas yang menangani tampilan sistem

3.Kelas yang diambil dari pendefenisian *use case*

4.Kelas yang diambil dari pendefenisian data

Jenis-jenis kelas di atas juga dapat digabungkan satu sama lain sesuai dengan pertimbangan yang dianggap baik asalkan fungsi-fungsi yang sebaiknya ada pada struktur kelas tetap ada. Susunan kelas juga dapat ditambahan kelas

utilitas seperti koneksi ke basis data, membaca *file* teks, dan lain sebagainya sesuai kebutuhan.

Berikut adalah simbol-simbol yang ada pada *class diagram.*

**Tabel 2 - 4 *Class Diagram***

| Simbol | Deskripsi |
|---|---|
| Kelas<br>**nama_kelas**<br>+atribut<br>+operasi() | kelas pada struktur sistem |
| asosiasi / *association* | relasi antar kelas dengan makna umum, asosiasi biasanya juga disertai dengan *multiplicity* |
| antarmuka / *interface*<br>**nama_interface** | sama dengan konsep *interface* dalam pemrograman berorientasi objek |
| asosiasi berarah / *directed association* | relasi antar kelas dengan makna kelas yang satu digunakan oleh kelas yang lain, asosiasi biasanya juga disertai dengan *multiplicity* |
| generalisasi | relasi antar kelas dengan makna generalisasi – spesialisasi (umum khusus) |
| kebergantungan/ *dependency* | relasi antar kelas dengan makna kebergantungan antar kelas |
| agregasi / aggregation | relasi antar kelas dengan makna semua-bagian *(whole-part)* |

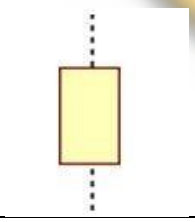
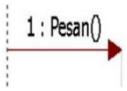
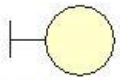
### 2.15.4 *Sequence* Diagram

*Sequence diagram* menggambarkan kelakuan objek pada *use case* dengan mendeskripsikan waktu hidup objek dan *message* yang dikirimkan dan diterima antar objek. Oleh karena itu untuk menggambar *sequence diagram* maka harus diketahui objek-objek yang terlibat dalam sebuah *use case* beserta metode – metode yang dimiliki kelas yang diinstansi menjadi objek itu.

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

Berikut adalah simbol-simbol yang ada pada *sequence diagram*.

**Tabel 2 - 5 *Sequence Diagram***

| No. | Gambar | Nama | Keterangan |
|---|---|---|---|
| 1 |  | *Actor* | Menspesifikasikan himpunan peran yang pengguna mainkan ketika berinteraksi dengan *use case*. |
| 2 |  | *lifeline* | Mengindikasikan keberadaan sebuah object dalam basis waktu. Notasi untuk *Lifeline* adalah garis putus-putus vertikal yang ditarik dari sebuah *object*. |
| 3 |  | *Activation* | Dinotasikan sebagai sebuah kotak segi empat yang digambar pada sebuah lifeline. *Activation* mengindikasikan sebuah *object* yang akan melakukan sebuah aksi. |
| 4 |  | *Massage* | Digambarkan dengan anak panah horizontal antara *activation*. *Message* mengindikasikan komunikasi antara *object-object* |
| 5 |  | *Boundary* | Menemukan dan mengidentifikasi kelas-kelas pembatas dapat dilakukan dengan menguji *diagram use case*. Minimal harus ada satu kelas pembatas untuk setiapa interaksi antara *actor* - *use case*. Kelas pembatas adalah apa saja yang memungkinkan actor |

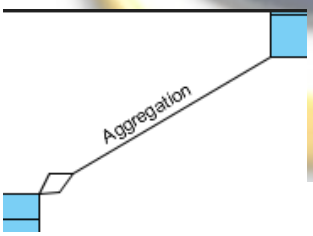
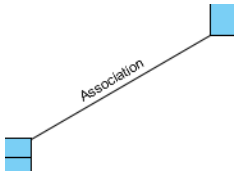
| | | | |
|---|---|---|---|
| | | | berinteraksi dengan *system* |
| 6 |  | *Entity* | Menentukan aliran kejadian (*flow of event*), dan aliran kejadian menentukan obyek-obyek, kelas-kelas, dan attribut-attribut dalam kelas. |
| 7 |  | *Control* | Kelas kontrol bertanggung jawab untuk mengkoordinasikan kegiatan-kegiatan terhadap kelas lainnya. |

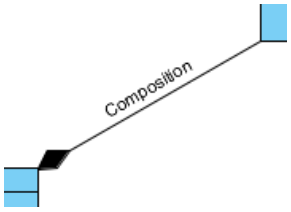
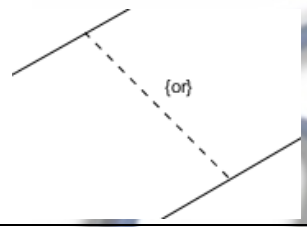
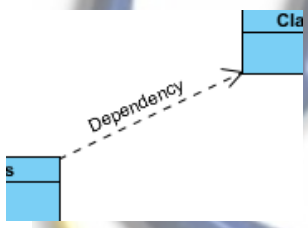
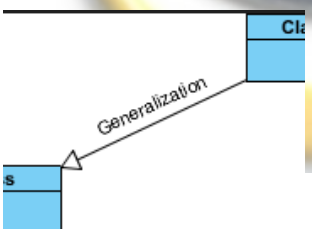
### 2.15.5 *Deployment* Diagram

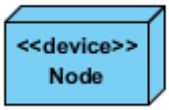
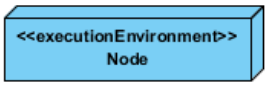
*Deployment Diagram* mengindikasikan bagaimana fungsionalitas perangkat lunak dan subsistemnya dialokasikan dalam lingkungan yang akan mendukung perangkat lunak.

*Deployment diagram* menunjukan lingkungan perangkat lunak, akan tetapi tidak secara eksplisit menjelaskan mengenai rincian atau spesifikasi konfigurasinya. Sebagai contoh, dalam gambar diatas digambarkan mengenai "*Personal Computer*" yang tidak diidentifikasi lebih lanjut mengenai spesifikasi *Personal Computer* tersebut seperti berbasis *windows*, atau *linux* [22].

**Tabel 2 - 6 *Deployment* Diagram**

| No | Gambar | Nama | Keterangan |
|---|---|---|---|
| 1 |  | *Aggregation* | Semacam asosiasi yang memiliki salah satu ujungnya ditandai sebagai agregasi, artinya memiliki agregasi bersama. |
| 2 |  | *Association* | Sebuah asosiasi menentukan hubungan semantik |

| | | | |
|---|---|---|---|
| 3 |  | *Artifact* | Spesifikasi informasi fisik yang digunakan atau diproduksi oleh proses pengembangan perangkat lunak, atau dengan penyebaran dan pengoperasian sistem. |
| 4 |  | *Composition* | Asosiasi dapat mewakili gabungan gabungan (yaitu, keseluruhan / hubungan bagian). |
| 5 |  | *Constraint* | Suatu kondisi atau pembatasan yang diungkapkan dalam teks bahasa alami atau bahasa yang mudah dibaca mesin untuk tujuan mendeklarasikan beberapa semantik elemen. |
| 6 |  | *Dependency* | Ketergantungan adalah hubungan yang menandakan bahwa satu atau satu set elemen model memerlukan elemen model lain untuk spesifikasi atau implementasinya |
| 7 |  | *Generalization* | Generalisasi adalah hubungan taksonomi antara pengklasifikasi yang lebih umum dan klasifikasi yang lebih spesifik. |
| 8 |  | *Component* | Bagian modular dari sistem yang merangkum isinya dan manifestasinya dapat diganti di lingkungannya. |

| | | | |
|---|---|---|---|
| 9 |  | *Realization* | Realisasi adalah hubungan abstraksi khusus antara dua elemen model, satu yang mewakili spesifikasi (pemasok) dan yang lainnya mewakili implementasi yang terakhir (klien). |
| 10 |  | *Note* | *Note* memberi kemampuan untuk melampirkan berbagai ucapan ke elemen. |
| 11 |  | *Deployement Specification* | Menentukan sekumpulan properti yang menentukan parameter eksekusi dari artefak komponen yang digunakan pada node |
| 12 |  | *Device Node* | Sumber komputasi fisik dengan kemampuan pemrosesan dimana artefak dapat digunakan untuk eksekusi |
| 13 |  | *Execution Environtment Node* | Node yang menawarkan lingkungan eksekusi untuk jenis komponen tertentu yang dikerahkan di dalamnya dalam bentuk artefak yang dapat dieksekusi. |

## 2.16 HTML

HTML (*Hyper Text Markup Language*) adalah suatu format data yang digunakan untuk membuat dokumen *hypertext* yang dapat dieksekusi dari suatu *platform* komputer ke *platform* komputer lainnya tanpa perlu melakukan suatu perubahan apapun dengan suatu alat tertentu.

HTML sebenarnya bukan sebuah bahasa pemrograman, karena HTM adalah bahasa *mark up*. HTML digunakan untuk penanda terhadap suatu dokumen teks. Simbol penanda yang digunakan oleh HTML ditandai dengan tanda lebih kecil (<) dan tanda lebih besar (>). Kedua tanda ini disebut tag. Tag yang digunakan sebagai tanda penutup diberi karakter garis miring (</...>) [23].

## 2.17 *JavaScript*

*JavaScript* adalah bahasa yang berbentuk kumpulan skrip yang fungsinya digunakan untuk menambahkan interaksi antara halaman web dengan pengunjung halaman web. *JavaScript* dijalankan pada sisi klien yang akan memberikan kemampuan fitur-fitur tambahan halaman web yang lebih baik dibandingkan fitur-fitur yang terdapat pada HTML.

*JavaScript* adalah bahasa pemrograman yang sederhana karena bahasa ini tidak dapat digunakan untuk membuat aplikasi ataupun *applet*. dengan *JavaScript* kita dapat dengan mudah membuat sebuah halaman web yang interaktif [23].

## 2.18 PHP

PHP (*Perl Hypertext Preprocessor*) merupakaan bahasa berbentuk skrip yang di tempatkan dalam *server* dan di proses di *server*. Selain itu juga PHP merupakan salah satu dari sekian banyak bahasa pemrograman HTML yang dibuat oleh Rasmus Lerdorf diawali dengan membuatnya sebagai *personal project* dan disempurnakan oleh *group sixof developers* dan lahir kembali dengan nama PHP.

Secara khusus, PHP dirancang untuk membentuk web dinamis. Artinya, PHP dapat membentuk suatu tampilan berdasarkan permintaan. PHP memiliki kemampuan yang baik dalam hal perhitungan matematika, dalam hal informasi jaringan *e-mail* dan *regular expretion*. Selain itu PHP juga mampu sebagai antarmuka dengan basis data secara baik, *support* dengan bermacam-macam basis data *server* seperti PostgreSQL, MySQL, ORACLE, Sysbase.

PHP adalah bahasa *scripting* yang dirancang khusus untuk digunakan di web. PHP memiliki fitur untuk membantu pengguna dalam memprogram tugas yang diperlukan untuk mengembangkan aplikasi web dinamis.

Perangkat lunak PHP bekerja dengan *web server*, yang merupakan perangkat lunak yang mengirimkan halaman web agar dapat diakses. saat pengguna mengetikkan URL ke pada *browser*, pengguna mengirim pesan ke *web server* di URL itu, memintanya untuk memberi pengguna file HTML. *Web server* merespon dengan mengirim file yang diminta. *Browser* pengguna membaca file HTML dan menampilkan halaman web. Pengguna juga meminta file dari *web*

*server* saat pengguna menekan tautan di halaman web. Selain itu, *web server* memproses file ketika pengguna menekan tombol pada halaman web yang mengirimkan formulir. Proses ini pada dasarnya sama ketika PHP dipasang. Pengguna meminta *file*, *web server* menjalankan PHP dan mengirimkan HTML kembali ke *browser* pengguna [24].

## 2.19 *Black-Box Testing*

*Black Box Testing* adalah metode pengujian perangkat lunak yang menguji fungsionalitas aplikasi tanpa melihat ke dalam struktur internal atau cara kerja dari aplikasi atau sistem tersebut. Metode pengujian ini dapat diterapkan secara virtual untuk setiap tingkat pengujian perangkat lunak seperti unit, integrasi, sistem, dan penerimaan [25].

Dalam melakukan *Black Box Testing* tidak memerlukan pengetahuan tentang mendalam mengenai sistem atau aplikasi tersebut, akan tetapi mengetahui kasus atau fungsi dasar dari hal yang akan diuji. Masalah utama yang dihadapi dalam melakukan pengujian *Black Box* adalah pemilihan model yang cocok untuk menentukan perilaku aplikasi atau sistem yang akan diuji.

*Functional Testing* adalah program perangkat lunak atau sistem yang diuji diamati sebagai "*Black Box*". Pemilihan kasus uji untuk pengujian fungsional didasarkan pada persyaratan atau spesifikasi desain entitas perangkat lunak yang diuji. Contoh hasil yang diharapkan terkadang disebut *test oracle*, termasuk spesifikasi kebutuhan atau desain, *hand calculated values*, dan hasil simulasi. Pengujian fungsional terutama berfokus pada perilaku eksternal dari entitas perangkat lunak [26].