

BAB II

TINJAUAN PUSTAKA

II.1. Sistem Informasi Geografis

GIS atau sistem informasi berbasis pemetaan dan geografi adalah sebuah alat bantu manajemen berupa informasi berbantuan komputer yang terkait dengan sistem pemetaan dan analisis terhadap segala sesuatu, serta peristiwa-peristiwa yang terjadi di muka bumi. Teknologi GIS mengintegrasikan operasi pengolahan data berbasis *database* yang biasa digunakan, seperti pengambilan data berdasarkan kebutuhan serta analisis statistic dengan menggunakan visualisasi yang khas serta berbagai keuntungan yang mampu ditawarkan melalui analisis geografis melalui gambar-gambar tertentu.

Konsep GIS telah diperkenalkan di Indonesia sejak pertengahan tahun 1980-an, dan kini telah dimanfaatkan di berbagai bidang baik negeri maupun swasta. Kemampuan dasar dari GIS adalah mengintegrasikan berbagai operasi basis data seperti *query*, menganalisisnya, dan menyimpan serta menampilkannya dalam bentuk pemetaan berdasarkan letak geografisnya. Inilah yang membedakan GIS dengan sistem informasi lain. Komponen GIS terdiri atas *hardware*, *software*, data, dan *user*. Dengan adanya GIS diharapkan tersedia informasi yang cepat, benar dan akurat tentang keadaan di lingkungannya (Hersa Farida Qoriani ; 2012 : 2).

II.1.1. *Data Spasial*

Sebagian besar data yang akan ditangani dalam SIG merupakan data spasial, data yang berorientasi geografis. Data ini memiliki sistem koordinat tertentu sebagai dasar referensinya dan mempunyai dua bagian penting yang berbeda dari data lain, yaitu informasi lokasi (spasial) dan informasi deskriptif (atribut) yang dijelaskan berikut ini :

1. Informasi lokasi (spasial), berkaitan dengan suatu koordinat baik koordinat geografi (lintang dan bujur) dan koordinat XYZ, termasuk diantaranya informasi datum dan proyeksi.
2. Informasi deskriptif (atribut) atau informasi nonspasial, suatu lokasi yang memiliki beberapa keterangan yang berkaitan dengannya. Contoh jenis vegetasi, populasi, luasan, kode pos, dan sebagainya.

II.1.2. *Format Data Spasial*

Secara sederhana format dalam bahasa komputer berarti bentuk dan kode penyimpanan data yang berbeda antara file satu dengan lainnya. Dalam SIG, data spasial dapat direpresentasikan dalam dua format, yaitu:

1. Data vektor

Data vektor merupakan bentuk bumi yang direpresentasikan ke dalam kumpulan garis, area (daerah yang dibatasi oleh garis yang berawal dan berakhir pada titik yang sama), titik dan nodes (titik perpotongan antara dua buah garis).

2. Data raster

Data raster (disebut juga dengan sel grid) adalah data yang dihasilkan dari sistem penginderaan jauh. Pada data raster, obyek geografis direpresentasikan sebagai struktur sel grid yang disebut dengan *pixel (picture element)* (Mohd. Ichsan ; 2012 : 51).

II.1.3. Sumber Data Spasial

Salah satu syarat SIG (Sistem Informasi Geografis) adalah data spasial, yang dapat diperoleh dari beberapa sumber antara lain :

1. Peta Analog Peta analog (antara lain peta topografi, peta tanah dan sebagainya) yaitu peta dalam bentuk cetak. Pada umumnya peta analog dibuat dengan teknik kartografi, kemungkinan besar memiliki referensi spasial seperti koordinat, skala, arah mata angin dan sebagainya. Dalam tahapan SIG (Sistem Informasi Geografis) sebagai keperluan sumber data, peta analog dikonversi menjadi peta digital dengan cara format raster diubah menjadi format vektor melalui proses digitasi sehingga dapat menunjukkan koordinat sebenarnya di permukaan bumi.
2. Data Sistem Penginderaan Jauh Data Penginderaan Jauh (antara lain citra satelit, foto udara dan sebagainya), merupakan sumber data yang terpenting bagi SIG (Sistem Informasi Geografis) karena ketersediaanya secara berkala dan mencakup area tertentu.
3. Data Hasil Pengukuran Lapangan Data pengukuran lapangan yang dihasilkan berdasarkan teknik perhitungan tersendiri, pada umumnya data ini merupakan

sumber data atribut contohnya : batas administrasi, batas kepemilikan lahan, batas persil, batas hak perusahaan hutan dan lain-lain.

4. Data GPS (*Global Positioning System*) Teknologi GPS memberikan terobosan penting dalam menyediakan data bagi SIG (Sistem Informasi Geografis). Keakuratan pengukuran GPS semakin tinggi dengan berkembangnya teknologi satelit navigasi. Pengolahan data yang bersumber dari GPS biasanya dilakukan dalam format vector (G. Manjela Eko Hartoyo ; 2010 : 4)

II.1.4. Komponen Sistem Informasi Geografis

Menurut Bramantiyo Marjuki (2014 : 2) Sebagai salah satu jenis sistem informasi, SIG (Sistem Informasi Geografis) mempunyai sub sistem atau komponen yang bekerja secara bersama untuk menghasilkan fungsionalitas SIG (Sistem Informasi Geografis). Komponen SIG (Sistem Informasi Geografis) terdiri dari *hardware*, *software*, data dan metode.

1. *Hardware* atau perangkat keras merupakan media tempat pelaksanaan proses SIG (Sistem Informasi Geografis). *Hardware* yang diperlukan dalam sebuah SIG (Sistem Informasi Geografis) meliputi perangkat keras untuk masukan data, penyimpanan data, pengolahan dan analisa data dan pembuatan keluaran.
2. *Software* atau perangkat lunak merupakan alat pelaksana pekerjaan SIG (Sistem Informasi Geografis). *Software* standar SIG (Sistem Informasi Geografis) harus mempunyai kapabilitas data input, penyimpanan, manajemen data, transformasi dan konversi data, analisa dan pembuatan keluaran.

3. Data atau representasi dari sebuah obyek/fenomena adalah bahan yang dianalisa di dalam SIG (Sistem Informasi Geografis). SIG (Sistem Informasi Geografis) memerlukan sebuah jenis data yang spesifik agar dapat memberikan keluaran seperti fungsionalitasnya. Data yang digunakan dalam SIG (Sistem Informasi Geografis) adalah data geospasial atau data yang berefrensi geografis (mempunyai informasi lokasi).
4. Metode adalah cara bagaimana data diolah untuk menjadi sebuah informasi. Metode meliputi aspek pemasukan data ke dalam sistem, bagaimana data dikelola dan disimpan, bagaimana data dianalisis, dan bagaimana informasi ditampilkan.

II.1.5. Arsitektur Sistem Informasi Geografis

Untuk dapat melakukan komunikasi dengan komponen yang berbeda-beda di lingkungan web maka dibutuhkan sebuah *web server*. Karena standart dari geo data berbeda beda dan sangat spesifik maka pengembangan arsitektur system mengikuti arsitektur "*Client Server*".

Applikasi berada disisi client yang berkomunikasi dengan Server sebagai penyedia data melalui web Protokol seperti HTTP (*Hyper Text Transfer Protocol*). Applikasi seperti ini bisa dikembangkan dengan *web browser* (Mozilla Firefox, Opera, Internet Explorer, dll). Untuk menampilkan dan berinteraksi dengan data GIS, sebuah browser membutuhkan *Pug-In* atau *Java Applet* atau bahkan keduanya. Web Server bertanggung jawab terhadap proses permintaan dari client dan mengirimkan tanggapan terhadap respon tersebut. Dalam arsitektur

web, sebuah web server juga mengatur komunikasi dengan server side GIS Komponen. Server side GIS Komponen bertanggung jawab terhadap koneksi kepada database spasial seperti menterjemahkan *query* kedalam SQL dan membuat representasi yang diteruskan ke server. Dalam kenyataannya Side Server GIS Komponen berupa software libraries yang menawarkan layanan khusus untuk analisis spasial pada data. Selain komponen hal lain yang juga sangat penting adalah aspek fungsional yang terletak di sisi client atau di server (Denny Charter ; 2011 : 1 – 2).

II.2. Quantum GIS

Quantum GIS (QGIS) adalah *cross-platform* perangkat lunak bebas (open source) desktop pada sistem informasi geografis (SIG). Aplikasi ini dapat menyediakan data, melihat, mengedit, dan kemampuan analisis. Quantum GIS berjalan pada sistem operasi yang berbeda termasuk *Mac OS X* , *Linux* , *UNIX* , dan *Microsoft Windows* . Dalam perizinan, QGIS sebagai perangkat lunak bebas aplikasi di bawah GPL (*General Public License*), dapat secara bebas dimodifikasi untuk melakukan tugas yang berbeda atau lebih khusus (Adam Suseno ; 2012 ; 15).

II.3. Pengertian PHP

PHP adalah kependekan dari *Hyper Text Preprocessor* (suatu akronim rekursif) yang dibangun oleh RasmusLerdorf pada tahun 1994. Dahulu, pada awal pengembangannya PHP disebut sebagai kependekan dari *Personal Home Page*.

PHP merupakan produk *OpenSource* sehingga anda dapat mengakses *sourcecode*, menggunakan dan mengubahnya tanpa harus membayar sepeser pun. Gratis (Antonius Nugraha Widhi Pratama ; 2010 : 9).

II.4. Pengertian Database

Secara sederhana *database* (basis data/pangkalan data) dapat diungkapkan sebagai suatu pengorganisasian data dengan bantuan komputer yang memungkinkan data dapat diakses dengan mudah dan cepat. Pengertian akses dapat mencakup pemerolehan data maupun manipulasi data seperti menambah serta menghapus data. Dengan memanfaatkan komputer, data dapat disimpan dalam media pengingat yang disebut *harddisk*. Dengan menggunakan media ini, keperluan kertas untuk menyimpan data dapat dikurangi. Selain itu, data menjadi lebih cepat untuk diakses terutama jika dikemas dalam bentuk *database*. (Agustinus Mujilan ; 2012 : 23).

II.5. Pengertian MySQL

MySQL adalah suatu sistem manajemen basis data relasional (RDBMS-*Relational Database Management System*) yang mampu bekerja dengan cepat, kokoh, dan mudah digunakan. Contoh RDBMS lain adalah *Oracle*, *Sybase*. Basis data memungkinkan anda untuk menyimpan, menelusuri, menurutkan dan mengambil data secara efisien. *Server MySQL* yang akan membantu melakukan fungsionalitas tersebut. Bahasa yang digunakan oleh MySQL tentu saja adalah *SQL-standar* bahasa basis data relasional di seluruh dunia saat ini.

MySQL dikembangkan, dipasarkan dan disokong oleh sebuah perusahaan Swedia bernama MySQL AB. RDBMS ini berada di bawah bendera GNU GPL sehingga termasuk produk *Open Source* dan sekaligus memiliki lisensi komersial. Apabila menggunakan MySQL sebagai basis data dalam suatu situs Web. Anda tidak perlu membayar, akan tetapi jika ingin membuat produk RDBMS baru dengan basis MySQL dan kemudian menguálnua, anda wajib bertemu mudah dengan lisensi komersial (Antonius Nugraha Widhi Pratama ; 2010 : 10).



Gambar II.1. Tampilan MySQL
(Sumber : Antonius Nugraha Widhi Pratama ; 2010 : 10)

II.6. Teknik Normalisasi

Normalisasi adalah teknik perancangan yang banyak digunakan sebagai pemandu dalam merancang basis data relasional. Pada dasarnya, normalisasi adalah proses dua langkah yang meletakkan data dalam bentuk tabulasi dengan

menghilangkan kelompok berulang lalu menghilangkan data yang terduplikasi dari tabel rasional.

Teori normalisasi didasarkan pada konsep bentuk normal. Sebuah tabel relasional dikatakan berada pada bentuk normal tertentu jika tabel memenuhi himpunan batasan tertentu. Ada lima bentuk normal yang telah ditemukan

II.6.1. Bentuk-bentuk Normalisasi

1. Bentuk tidak normal

Bentuk ini merupakan kumpulan data yang akan direkam, tidak ada keharusan mengikuti format tertentu, dapat saja tidak lengkap dan terduplikasi. Data dikumpulkan apa adanya sesuai keadaanya.

2. Bentuk normal tahap pertama (1st Normal Form)

Definisi :

Sebuah table disebut 1NF jika :

- Tidak ada baris yang duplikat dalam tabel tersebut.
- Masing-masing *cell* bernilai tunggal

Catatan: Permintaan yang menyatakan tidak ada baris yang duplikat dalam sebuah tabel berarti tabel tersebut memiliki sebuah kunci, meskipun kunci tersebut dibuat dari kombinasi lebih dari satu kolom atau bahkan kunci tersebut merupakan kombinasi dari semua kolom.

3. Bentuk normal tahap kedua (2nd normal form)

Bentuk normal kedua (2NF) terpenuhi jika pada sebuah tabel semua atribut yang tidak termasuk dalam *primary key* memiliki ketergantungan fungsional pada *primary key* secara utuh.

4. Bentuk normal tahap ketiga (3rd normal form)

Sebuah tabel dikatakan memenuhi bentuk normal ketiga (3NF), jika untuk setiap ketergantungan fungsional dengan notasi $X \rightarrow A$, dimana A mewakili semua atribut tunggal di dalam tabel yang tidak ada di dalam X, maka :

- X haruslah *superkey* pada tabel tersebut.
- Atau A merupakan bagian dari *primary key* pada tabel tersebut.

5. Bentuk Normal Tahap Keempat dan Kelima

Penerapan aturan normalisasi sampai bentuk normal ketiga sudah memadai untuk menghasilkan tabel berkualitas baik. Namun demikian, terdapat pula bentuk normal keempat (4NF) dan kelima (5NF). Bentuk Normal keempat berkaitan dengan sifat ketergantungan banyak nilai (*multivalued dependency*) pada suatu tabel yang merupakan pengembangan dari ketergantungan fungsional. Adapun bentuk normal tahap kelima merupakan nama lain dari *Project Join Normal Form* (PJNF).

6. Boyce Code Normal Form (BCNF)

-Memenuhi 1st NF

- Relasi harus bergantung fungsi pada atribut superkey (Janner Simarmata ; 2010 : 76).

II.7. UML (*Unified Modeling Language*)

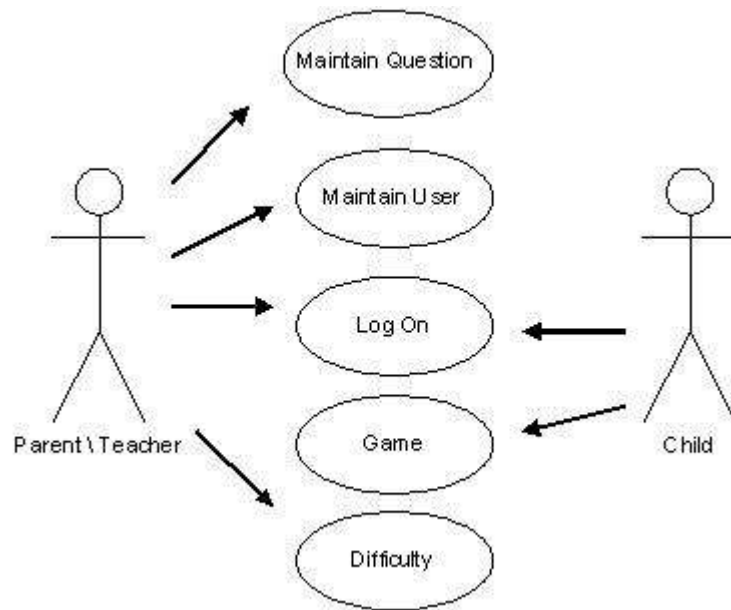
Menurut Sri Dharwiyanti (2013) *Unified Modelling Language* (UML) adalah sebuah "bahasa" yg telah menjadi standar dalam industri untuk visualisasi, merancang dan mendokumentasikan sistem piranti lunak. UML menawarkan sebuah standar untuk merancang model sebuah sistem

1. *Use Case Diagram*

Use case diagram menggambarkan fungsionalitas yang diharapkan dari sebuah sistem. Yang ditekankan adalah “apa” yang diperbuat sistem, dan bukan “bagaimana”. Sebuah *use case* merepresentasikan sebuah interaksi antara aktor dengan sistem. *Use case* merupakan sebuah pekerjaan tertentu, misalnya login ke sistem, meng-*create* sebuah daftar belanja, dan sebagainya. Seorang/sebuah aktor adalah sebuah entitas manusia atau mesin yang berinteraksi dengan system untuk melakukan pekerjaan-pekerjaan tertentu. *Use case diagram* dapat sangat membantu bila kita sedang menyusun *requirement* sebuah sistem, mengkomunikasikan rancangan dengan klien, dan merancang *test case* untuk semua *feature* yang ada pada sistem.

Sebuah *use case* dapat meng-*include* fungsionalitas *use case* lain sebagai bagian dari proses dalam dirinya. Secara umum diasumsikan bahwa *use case* yang di-*include* akan dipanggil setiap kali *use case* yang meng-*include* dieksekusi secara normal. Sebuah *use case* dapat di-*include* oleh lebih dari satu *use case* lain, sehingga duplikasi fungsionalitas dapat dihindari dengan cara menarik keluar fungsionalitas yang *common*.

Sebuah *use case* juga dapat meng-*extend use case* lain dengan *behaviour*-nya sendiri. Sementara hubungan generalisasi antar *use case* menunjukkan bahwa *use case* yang satu merupakan spesialisasi dari yang lain.

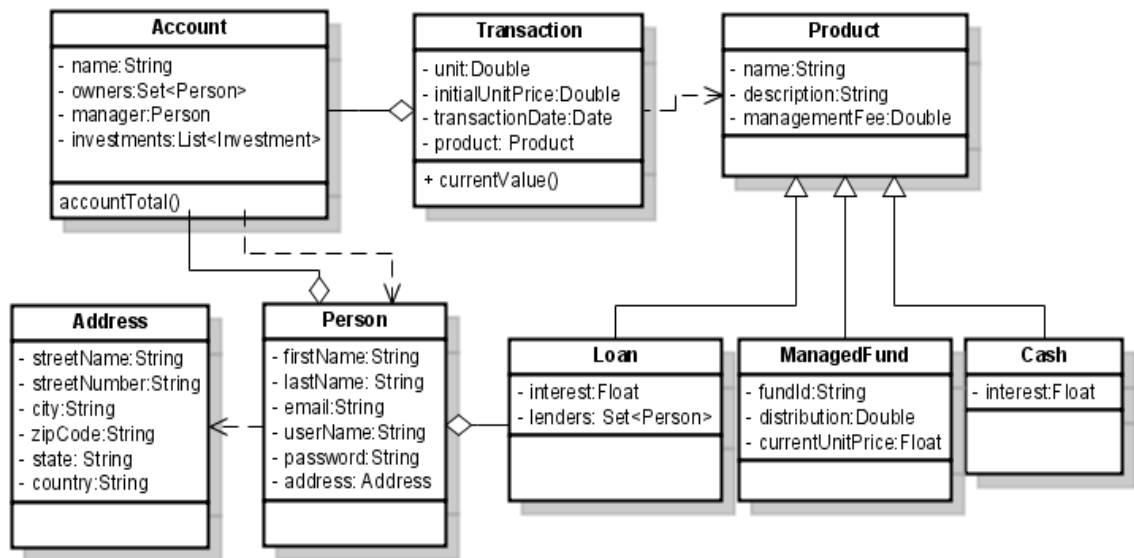


Gambar II.2. Usecase Diagram
(Sumber : Sri Dharwiyanti ; 2013 : 5)

2. Class Diagram

Class adalah sebuah spesifikasi yang jika diinstansiasi akan menghasilkan sebuah objek dan merupakan inti dari pengembangan dan desain berorientasi objek. *Class* menggambarkan keadaan (atribut/properti) suatu sistem, sekaligus menawarkan layanan untuk memanipulasi keadaan tersebut (metoda/fungsi).

Class diagram menggambarkan struktur dan deskripsi *class*, *package* dan objek beserta hubungan satu sama lain seperti *containment*, pewarisan, asosiasi, dan lain-lain.

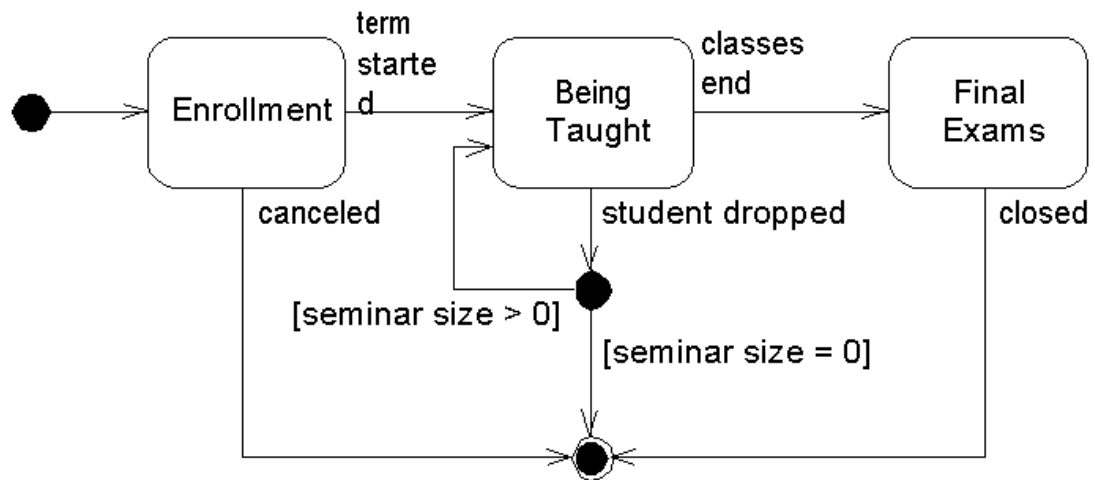


Gambar II.3. Class Diagram
 (Sumber : Sri Dharwiyanti ; 2013 : 6)

3. Statechart Diagram

Statechart diagram menggambarkan transisi dan perubahan keadaan (dari satu *state* ke *state* lainnya) suatu objek pada sistem sebagai akibat dari *stimuli* yang diterima. Pada umumnya *statechart diagram* menggambarkan *class* tertentu (satu *class* dapat memiliki lebih dari satu *statechart diagram*).

Dalam UML, *state* digambarkan berbentuk segiempat dengan sudut membulat dan memiliki nama sesuai kondisinya saat itu. Transisi antar *state* umumnya memiliki kondisi *guard* yang merupakan syarat terjadinya transisi yang bersangkutan, dituliskan dalam kurung siku. *Action* yang dilakukan sebagai akibat dari *event* tertentu dituliskan dengan diawali garis miring. Titik awal dan akhir digambarkan berbentuk lingkaran berwarna penuh dan berwarna setengah.

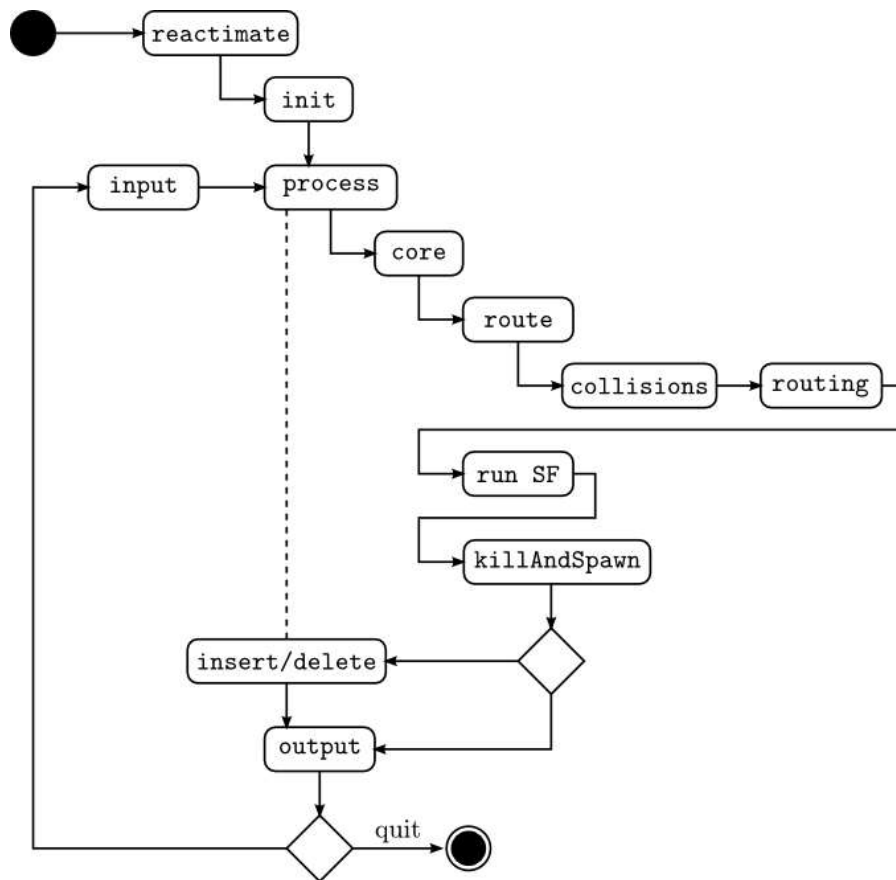


Gambar II.4. Statechart Diagram
 (Sumber : Sri Dharwiyanti ; 2013 : 7)

4. Activity Diagram

Activity diagrams menggambarkan berbagai alir aktivitas dalam sistem yang sedang dirancang, bagaimana masing-masing alir berawal, *decision* yang mungkin terjadi, dan bagaimana mereka berakhir. *Activity diagram* juga dapat menggambarkan proses paralel yang mungkin terjadi pada beberapa eksekusi. *Activity diagram* merupakan *state diagram* khusus, di mana sebagian besar *state* adalah *action* dan sebagian besar transisi di-*trigger* oleh selesainya *state* sebelumnya (*internal processing*). Oleh karena itu *activity diagram* tidak menggambarkan behaviour internal sebuah sistem (dan interaksi antar subsistem) secara eksak, tetapi lebih menggambarkan proses-proses dan jalur-jalur aktivitas dari level atas secara umum.

Sebuah aktivitas dapat direalisasikan oleh satu *use case* atau lebih. Aktivitas menggambarkan proses yang berjalan, sementara *use case* menggambarkan bagaimana aktor menggunakan sistem untuk melakukan aktivitas



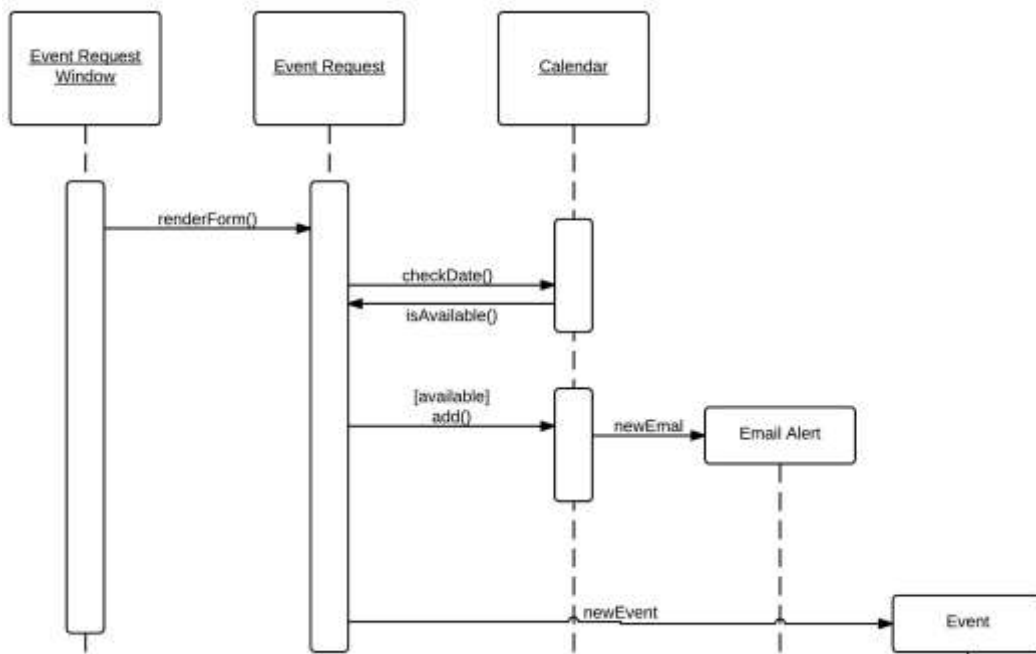
Gambar II.5. Activity Diagram
 (Sumber : Sri Dharwiyanti ; 2013 : 8)

5. Sequence Diagram

Sequence diagram menggambarkan interaksi antar objek di dalam dan di sekitar sistem (termasuk pengguna, *display*, dan sebagainya) berupa *message* yang digambarkan terhadap waktu. *Sequence diagram* terdiri atas dimensi vertikal (waktu) dan dimensi horizontal (objek-objek yang terkait). *Sequence diagram* biasa digunakan untuk menggambarkan skenario atau rangkaian langkah-langkah yang dilakukan sebagai respons dari sebuah *event* untuk menghasilkan *output* tertentu. Diawali dari apa yang men-*trigger* aktivitas tersebut, proses dan

perubahan apa saja yang terjadi secara internal dan *output* apa yang dihasilkan.

Masing-masing objek, termasuk aktor, memiliki *lifeline* vertikal. *Message* digambarkan sebagai garis berpanah dari satu objek ke objek lainnya. Pada fase desain berikutnya, *message* akan dipetakan menjadi operasi/metoda dari *class*. *Activation bar* menunjukkan lamanya eksekusi sebuah proses, biasanya diawali dengan diterimanya sebuah *message*.

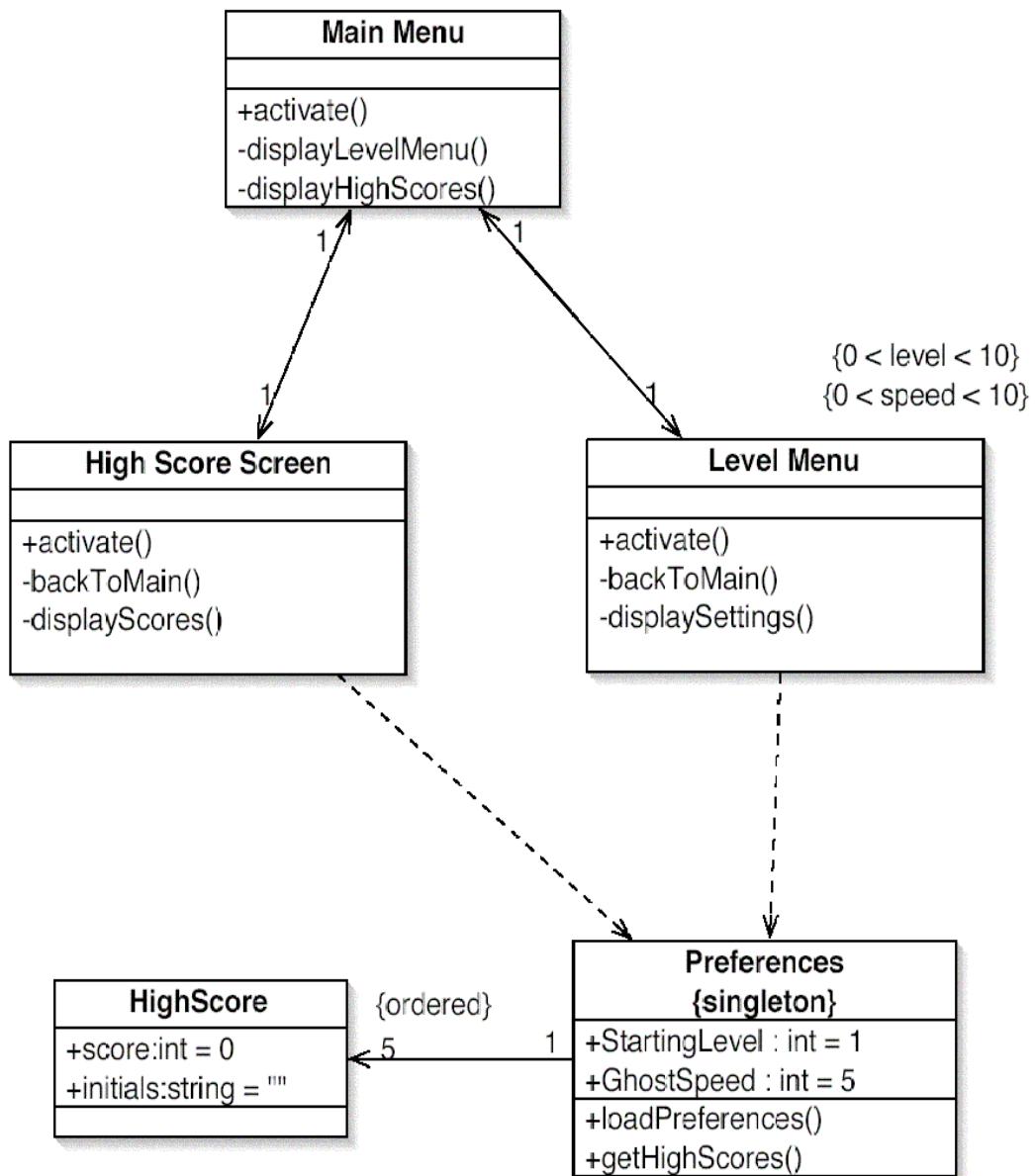


Gambar II.6. Sequence Diagram
(Sumber : Sri Dharwiyanti ; 2013 : 9)

6. Collaboration Diagram

Collaboration diagram juga menggambarkan interaksi antar objek seperti *sequence diagram*, tetapi lebih menekankan pada peran masing-masing objek dan bukan pada waktu penyampaian *message*.

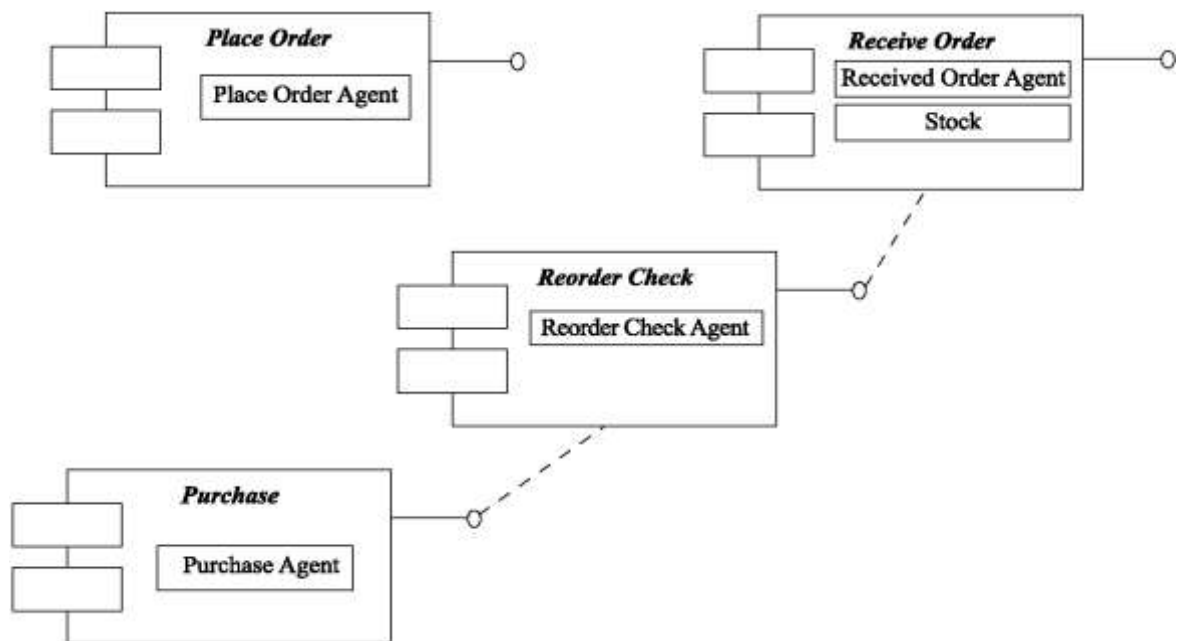
Setiap *message* memiliki *sequence number*, di mana *message* dari level tertinggi memiliki nomor 1. Messages dari level yang sama memiliki prefiks yang sama.



Gambar II.7. Collaboration Diagram
(Sumber : Sri Dharwiyanti ; 2013 : 9)

7. Component Diagram

Component diagram menggambarkan struktur dan hubungan antar komponen piranti lunak, termasuk ketergantungan (*dependency*) di antaranya. Komponen piranti lunak adalah modul berisi *code*, baik berisi *source code* maupun *binary code*, baik *library* maupun *executable*, baik yang muncul pada *compile time*, *link time*, maupun *run time*. Umumnya komponen terbentuk dari beberapa *class* dan/atau *package*, tapi dapat juga dari komponen-komponen yang lebih kecil. Komponen dapat juga berupa *interface*, yaitu kumpulan layanan yang disediakan sebuah komponen untuk komponen lain.

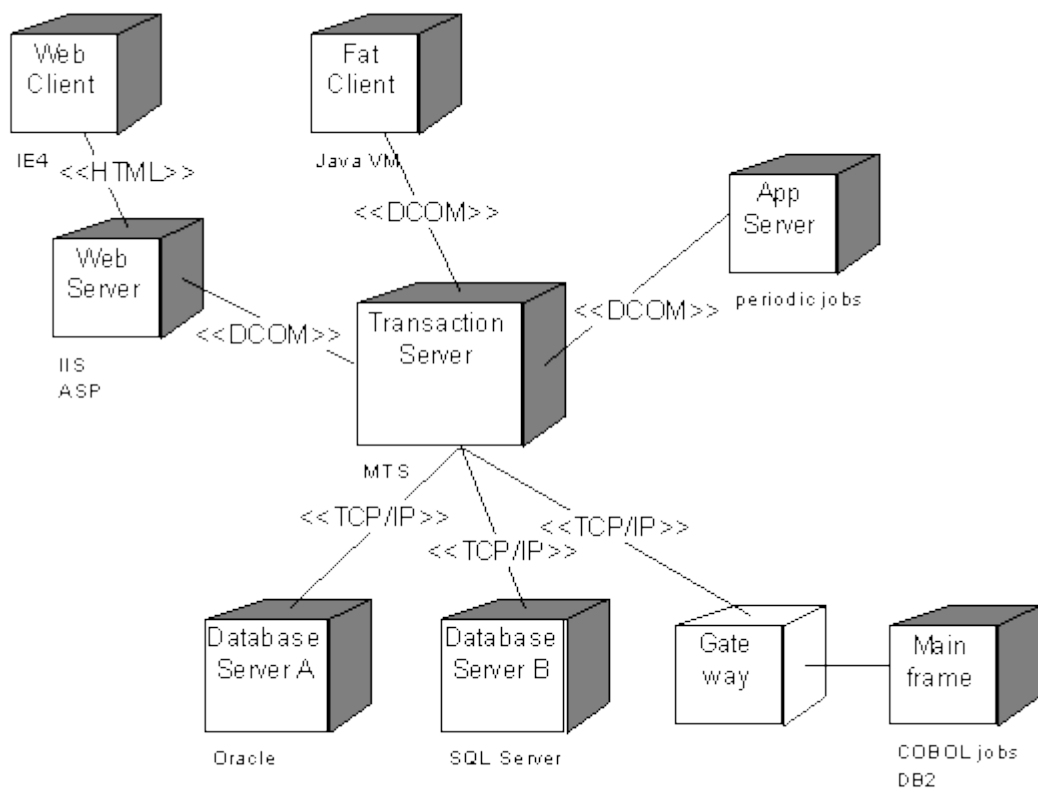


Gambar II.8. Component Diagram
(Sumber : Sri Dharwiyanti ; 2013 : 10)

8. Deployment Diagram

Deployment/physical diagram menggambarkan detail bagaimana komponen di-deploy dalam infrastruktur sistem, di mana komponen akan terletak (pada

mesin, server atau piranti keras apa), bagaimana kemampuan jaringan pada lokasi tersebut, spesifikasi server, dan hal-hal lain yang bersifat fisik Sebuah *node* adalah server, *workstation*, atau piranti keras lain yang digunakan untuk *deploy* komponen dalam lingkungan sebenarnya.



Gambar II.9. Deployment Diagram
(Sumber : Sri Dharwiyanti ; 2013 : 11)