

## BAB II

### LANDASAN TEORI

#### 2.1 Studi Kepustakaan

Dari tinjauan kepustakaan diketahui ada beberapa referensi terkait yang telah dilakukan oleh para peneliti terdahulu yaitu sebagai berikut:

Pipi Wijia Astuti (2015), tentang Sistem Pendukung Keputusan Pemilihan Kegiatan Ekstrakurikuler Dengan Metode Smart Pada MAS PAB1 Sampali. Bertujuan untuk mempermudah proses pemilihan kegiatan ekstrakurikuler agar siswa lebih mudah dalam memilih kegiatan ekstrakurikuler yang di minatnya dan memudahkan guru dalam memilih siswa yang memiliki minat dan bakat pada kegiatan tersebut.

Rika Yunitarini(2013),tentang Sistem Pendukung Keputusan Pemilihan Penyiar Radio Terbaik. Bertujuan untuk mempermudah pemilihan penyiar terbaik yang sesuai berdasarkan kriteria yang menggunakan perhitungan dengan metode SMART.

Atiqah (2013), tentang Implementasi Metode SMART pada Sistem Pengambilan Keputusan Pembelian Mobil Keluarga. Bertujuan untuk menemukan solusi *multicriteria* dan *multiobjective* dalam pembelian mobil, adapun hasil dari implementasi tersebut, dapat membantu konsumen dalam memilih mobil sesuai dengan keinginan mereka.

Pada penelitian ini penulis melakukan pengembangan dari referensi penelitian diatas dengan membuat Sistem Pendukung Keputusan Dalam Penilaian Perawat Terbaik di Lingkungan Rumah Sakit Umum Daerah (RSUD ARIFIN

ACHMAD PEKANBARU) yang bertujuan, untuk mempermudah dalam penyekelsian perawat tersebut yang berhak mendapatkan peringkat maupun penilaian sesuai dengan kriteria-kriteria semestinya, serta memudahkan pimpinandalam melakukan penilaian.

## 2.2 Teori

### 2.2.1 Sistem Pendukung Keputusan

Menurut Turban & Aronson (2001), sistem pendukung keputusan sebagai sistem yang digunakan untuk mendukung dan membantu pihak manajemen melakukan pengambilan keputusan pada kondisi semi terstruktur dan tidak terstruktur. Pada dasarnya konsep DSS hanyalah sebatas pada kegiatan membantu para manajer melakukan penilaian serta menggantikan posisi dan peran manajer.

Secara Umum, sistem pendukung keputusan adalah sebuah sistem yang mampu memberikan kemampuan, baik kemampuan pemecahan masalah maupun kemampuan pengkomunikasian untuk masalah semi terstruktur. Secara Khusus, sistem pendukung keputusan adalah sebuah sistem yang mendukung kerja seorang manajer maupun sekelompok manajer dalam memecahkan masalah semi-terstruktur dengan cara memberikan informasi ataupun usulan menuju pada keputusan tertentu.

Menurut Jogiyanto(2005), kerangka dasar pengambilan keputusan Manajerial dalam tipe keputusan dibagi menjadi:

1. Keputusan Terstruktur (*structured decision*) adalah keputusan yang berulang – ulang dan rutin, sehingga dapat diprogram. Keputusan

terstruktur terjadi dan dilakukan terutama pada manajemen tingkat bawah. Contoh dari keputusan tipe ini misalnya adalah keputusan pemesanan barang, keputusan penagihan piutang dan lain sebagainya.

2. Keputusan Tidak Terstruktur (*unstructured decision*) adalah keputusan yang tidak terjadi berulang – ulang dan tidak selalu terjadi. Keputusan ini terjadi di manajemen tingkat atas. Informasi untuk pengambilan keputusan tidak terstruktur tidak mudah untuk didapatkan dan tidak mudah tersedia dan biasanya berasal dari lingkungan luar. Pengalaman manajer merupakan hal yang sangat penting di dalam pengambilan keputusan tidak terstruktur. Keputusan untuk bergabung dengan perusahaan lain adalah contoh keputusan tidak terstruktur yang jarang terjadi.
3. Keputusan Semi Terstruktur (*semi – structured decision*) adalah keputusan yang sebagian dapat diprogram, sebagian berulang-ulang dan rutin dan sebagian tidak struktur. Keputusan tipe ini seringkali bersifat rumit dan membutuhkan perhitungan – perhitungan serta analisis yang terperinci. Contoh dari keputusan tipe ini misalnya adalah keputusan membeli sistem komputer yang lebih canggih. Contoh yang lainnya misalnya adalah keputusan alokasi dana promosi.

Keuntungan Sistem Pendukung Keputusan :

1. Mampu mendukung pencarian solusi dari masalah yang kompleks.
2. Respon cepat pada situasi yang tidak diharapkan dalam kondisi yang berubah-ubah.

3. Mampu untuk menerapkan berbagai strategi yang berbeda pada konfigurasi berbeda secara cepat dan tepat.
4. Pandangan dan pembelajaran baru.
5. Memfasilitasi komunikasi.
6. Meningkatkan kontrol manajemen dan kinerja.
7. Menghemat biaya.
8. Keputusannya lebih tepat.
9. Meningkatkan efektivitas manajerial, menjadikan manajer dapat bekerja lebih singkat dan dengan sedikit usaha.

Keputusan merupakan kegiatan memilih suatu strategi atau tindakan dalam pemecahan masalah tersebut. Tujuan dari keputusan adalah untuk mencari target atau aksi tertentu yang harus dilakukan (Kusrini, 2007). Kriteria atau ciri-ciri dari sebuah keputusan adalah :

1. Banyak pilihan/alternatif
2. Ada kendala atau syarat
3. Mengikuti suatu pola/model tingkah laku, baik yang terstruktur maupun tidak terstruktur.
4. Banyak input/variabel
5. Ada faktor resiko
6. Dibutuhkan kecepatan, ketepatan, dan keakuratan.

### 2.2.2 Perawat

Perawat(bahasa Inggris: *nurse*, berasal dari bahasa Latin: *nutrix* yang berarti merawat atau memelihara) adalah profesi yang difokuskan pada perawatan individu, keluarga, dan masyarakat sehingga mereka dapat mencapai, mempertahankan, atau memulihkan kesehatan yang optimal dan kualitas hidup dari lahir sampai mati.

Saat ini profesi perawat telah mendapatkan perlindungan hukum melalui disahkannya undang undang keperawatan nomor 38 tahun 2014. Dengan adanya undang - undang ini diharapkan perawat dapat bekerja sesuai peran profesinya secara lebih profesional, bertanggungjawab dan lebih optimal.

Fungsi perawat yang utama adalah membantu pasien atau klien dalam kondisi sakit maupun sehat, untuk meningkatkan derajat kesehatan melalui layanan keperawatan. Dalam menjalankan perannya, perawat akan melaksanakan berbagai fungsi yaitu : Fungsi dependen perawat, fungsi independen perawat dan fungsi interdependen perawat.

#### 1. Fungsi Independen Perawat

Fungsi independen ialah fungsi mandiri dan tidak tergantung pada orang lain, dimana perawat dalam menjalankan tugasnya dilakukan secara sendiri dengan keputusan sendiri dalam melakukan tindakan untuk memenuhi kebutuhan dasar manusia.

#### 2. Fungsi Dependen Perawat

Fungsi dependen ialah fungsi perawat dalam melaksanakan kegiatannya atas atau instruksi dari perawat lain.

### 3. Fungsi Interdependen Perawat

Fungsi Interdependen ialah fungsi yang dilakukan dalam kelompok tim yang bersifat saling ketergantungan di antara satu dengan yang lain.

#### 2.2.3 SMART (*Simple Multi Attribute Rating Technique*)

SMART (*Simple Multi Attribute Rating Technique*) merupakan metode dalam pengambilan keputusan multiatribut. Teknik pengambilan keputusan multiatribut ini digunakan untuk mendukung pembuat keputusan dalam memilih beberapa alternatif. Setiap pembuat keputusan harus memiliki sebuah alternatif yang sesuai dengan tujuan yang dirumuskan. Alternatif terdiri dari sekumpulan atribut dan setiap atribut mempunyai nilai - nilai. Nilai ini dirata-rata dengan skala tertentu.

SMART menggunakan linier adaptif model untuk meramal nilai setiap alternatif. SMART lebih banyak digunakan karena kesederhanaannya dalam merespon kebutuhan pembuat keputusan dan caranya menganalisa respon. Analisis yang terbaik adalah transparan sehingga metode ini memberikan pemahaman masalah yang tinggi dan dapat diterima oleh pembuat keputusan. Pembobotan pada SMART menggunakan skala 0 sampai 1, sehingga mempermudah perhitungan dan perbandingan nilai pada masing-masing alternatif. (Edwards, 1977)

Dengan SMART bobot diperoleh dalam dua tahap (Edwards, 1977; von Winterfeldt dan Edwards, 1986):

1. Tentukan pentingnya perubahan atribut dari atribut level terburuk hingga yang tingkat terbaik.
2. Buat perkiraan rasio kepentingan relatif masing-masing atribut relatif terhadap satu peringkat terendah pentingnya.

Persamaan yang digunakan dalam *SMART*, Rumus 2.1:

$$\text{maximize } \sum_{j=1}^k (W_j \cdot u_{ij}, \forall i = 1, \dots, n) \quad (2.1)$$

Keterangan Persamaan (2.1) :

$w_j$  adalah nilai pembobotan kriteria ke- $j$  dari  $k$  kriteria.

$u_{ij}$  adalah nilai utility alternatif  $i$  pada kriteria  $j$ .

Pemilihan keputusan adalah mengidentifikasi. Mana dari  $n$  alternatif yang mempunyai nilai fungsi terbesar.

Nilai fungsi ini juga dapat digunakan untuk meranking alternatif.

Teknik Metode SMART :

Langkah 1: menentukan jumlah kriteria

Langkah 2: sistem secara default memberikan skala 0-100 berdasarkan prioritas yang telah diinputkan kemudian dilakukan normalisasi.

$$\frac{w_j}{\sum w_j} \quad (2.2)$$

Keterangan Persamaan (2.2) :

$W_j$  : bobot suatu kriteria

$\sum w_j$  : total bobot semua kriteria

Langkah 3: memberikan nilai kriteria untuk setiap alternatif.

Langkah 4: hitung nilai utility untuk setiap kriteria masing-masing.

$$u_i(a_i) = 100 \frac{(C_{mac} - C_{out i})}{(C_{max} - C_{min})} \% \quad (2.3)$$

Keterangan Persamaan (2.3) :

$u_i(a_i)$  : nilai utility kriteria ke-1 untuk kriteria ke-i

$C_{max}$  : nilai kriteria maksimal

$C_{min}$  : nilai kriteria minimal

$C_{out i}$  : nilai kriteria ke-i

Langkah 5: hitung nilai akhir masing-masing.

SMART memiliki beberapa kelebihan dibandingkan dengan metode pengambilan keputusan yang lain yaitu :

1. Mungkin melakukan penambahan/pengurangan alternatif pada metode SMART penambahan atau pengurangan alternatif tidak akan mempengaruhi perhitungan pembobotan karena setiap penilaian alternatif tidak saling bergantung.
2. Sederhana, perhitungan pada metode SMART lebih sederhana sehingga tidak diperlukan perhitungan matematis yang rumit dengan pemahaman matematika yang kuat.



3. Transparan proses dalam menganalisa alternatif dan kriteria dalam SMART dapat dilihat oleh user sehingga user dapat memahami bagaimana alternatif tertentu dapat dipilih. Alasan-alasan bagaimana alternatif itu dipilih dapat dilihat dari prosedur – prosedur yang dilakukan dalam SMART mulai dari penentuan kriteria, pembobotan, dan pemberian nilai pada setiap alternatif.
4. Fleksibilitas pembobotan - pembobotan yang dipakai di dalam metode SMART ada 3 jenis yaitu pembobotan secara langsung (*direct weighting*), pembobotan swing (*swing weighting*), pembobotan centroid (*centroid weighting*).

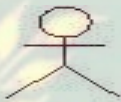


#### 2.2.4 Pengenalan dan Pengertian UML

UML (*Unified Modeling Language*) adalah metode pemodelan secara visual sebagai sarana untuk merancang dan atau membuat software berorientasi objek. Karena UML ini merupakan bahasa visual untuk pemodelan bahasa berorientasi objek, maka semua elemen dan diagram berbasiskan pada paradigma *object oriented*. UML adalah salah satu *tool* / model untuk merancang pengembangan software yang berbasis *object oriented*.






UML sendiri juga memberikan standar penulisan sebuah sistem *blueprint*, yang meliputi konsep bisnis proses, penulisan kelas-kelas dalam bahasa program yang spesifik, skema database, dan komponen-komponen yang diperlukan dalam sistem software. UML sebagai sebuah bahasa yang memberikan *vocabulary* dan tatanan penulisan kata-kata dalam 'MS Word' untuk kegunaan komunikasi.

Sebuah bahasa model adalah sebuah bahasa yang mempunyai *vocabulary* dan konsep tatanan / aturan penulisan serta secara fisik mempresentasikan dari sebuah sistem. Adapun ada beberapa simbol UML terlihat pada tabel 2.1


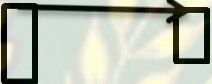
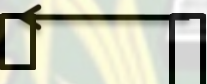
**Tabel 2.1** Simbol UML (*Unified Modelling Language*)

no	Simbol	Keterangan
1		actor menunjukkan user yang akan menggunakan sistem baru
2		use Case, menunjukkan proses yang terjadi pada sistem baru
3		Unidirectional Association, menunjukkan hubungan antara actor dan use case atau antar use case

**Tabel 2.2** Simbol Activity Diagram

No	Gambar	Nama	Keterangan
1		Activity	Memperlihatkan bagaimana masing – masing kelas antar muka 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 di hancurkan.
5		Fork Node	Suatu aliran yang pada tahap tertentu berubag menjadi beberapa aliran

**Tabel 2.3** Simbol Swquence Diagram

No	Gambar	Nama	Keterangan
1		LifeLine	objek entity, antarmuka yang saling berinteraksi.
2		Message	spesifikasi dari komunikasi antar objek yang memuat informasi - informasi tentang aktifitas yang terjadi.
3		Message	spesifikasi dari komunikasi antar objek yang memuat informasi - informasi tentang aktifitas yang terjadi.

UML juga mendefinisikan diagram-diagram sebagai berikut:

1. use case diagram
2. class diagram
3. statechart diagram
4. activity diagram
5. sequence diagram
6. collaboration diagram
7. component diagram
8. deployment diagram

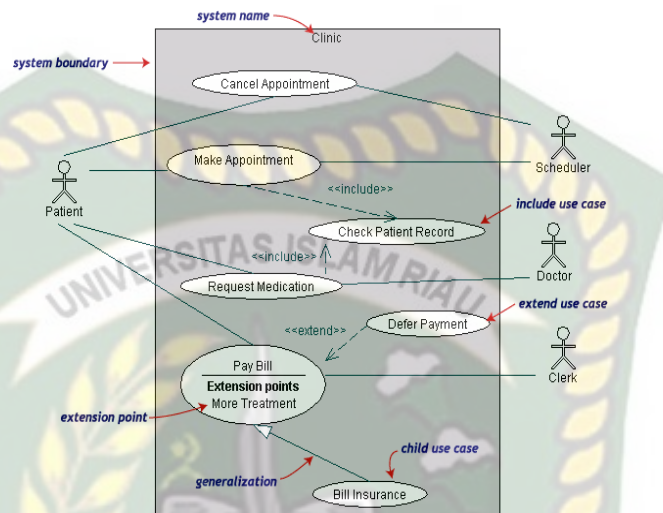
### 2.2.5 Use Case Diagram

*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 sistem 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 *usecase* 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. Contoh Use Case ditampilkan pada Gambar 2.1

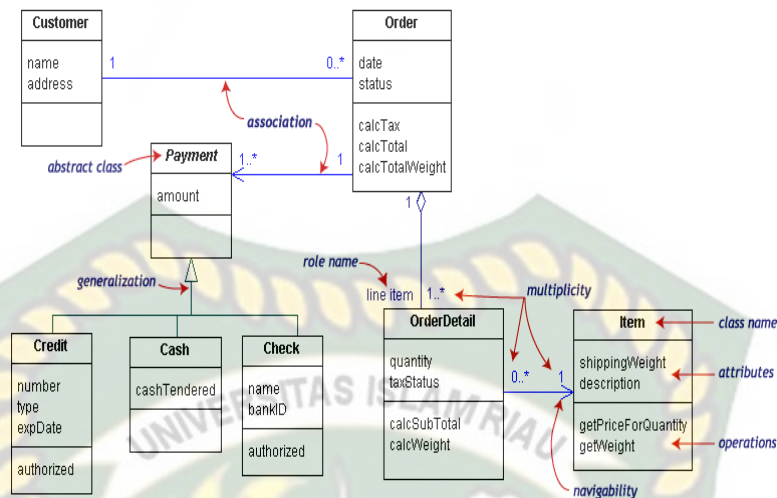


**Gambar 2.1** Use Case Diagram

## 2.2.6 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).

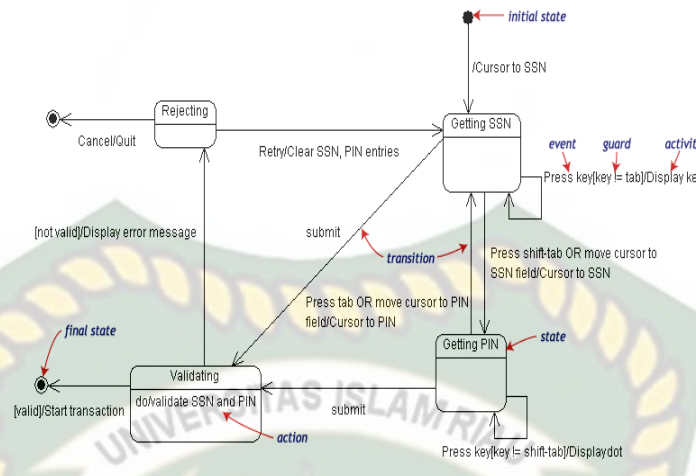
*Class diagram* menggambarkan struktur dan deskripsi *class*, *package* dan objek beserta hubungan satu sama lain seperti *containment*, pewarisan, asosiasi, dan lain-lain. Contoh Class Diagram ditampilkan pada Gambar 2.2



**Gambar 2.2** Class Diagram

### 2.2.7 Statechart Diagram

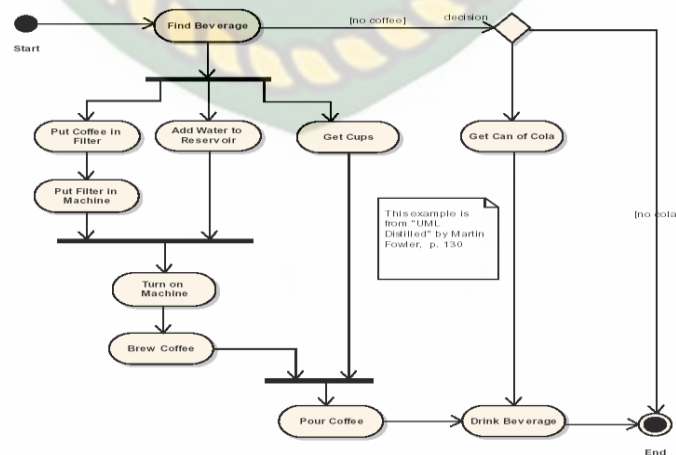
*Statechart diagram* menggambarkan transisi dan perubahan keadaan (dari satu *state* ke *state* lainnya) suatu objek pada sistem sebagai akibat dari *stimuli* yang diterima. Pada umumnya *statechartdiagram* menggambarkan *class* tertentu (satu *class* dapat memiliki lebih dari satu *statechartdiagram*). 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. Contoh gambar Statechart Diagram ditampilkan pada Gambar 2.3



Gambar 2.3 Statechart Diagram

## 2.2.8 Activity Diagram

*Activity diagrams* menggambarkan berbagai alir aktivitas dalam system 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. Contoh *Activity diagrams* ditampilkan pada Gambar 2.4

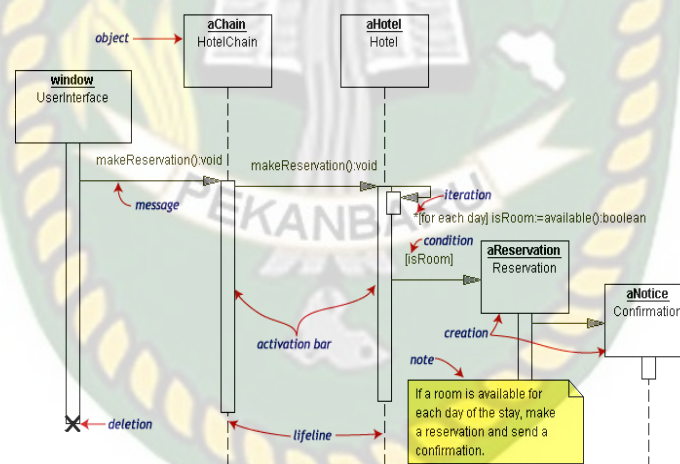


Gambar 2.4 Activity Diagram

### 2.2.9 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. *Sequencediagram* 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.

Contoh Sequence Diagram ditampilkan pada Gambar 2.5



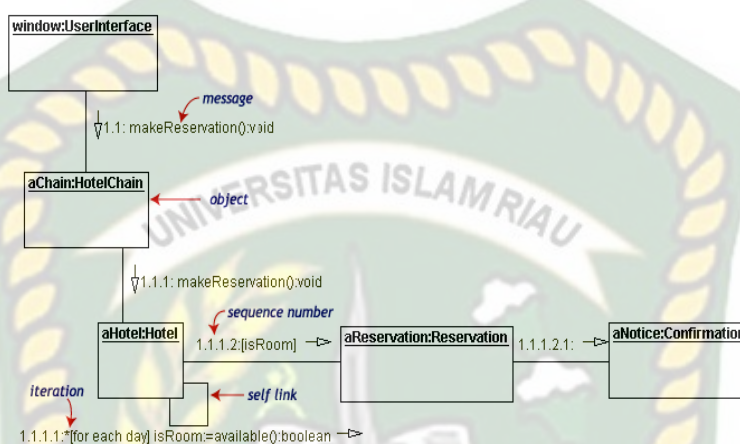
Gambar 2.5 Sequence Diagram

### 2.2.10 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. Contoh Collaboration Diagram ditampilkan pada Gambar 2.6

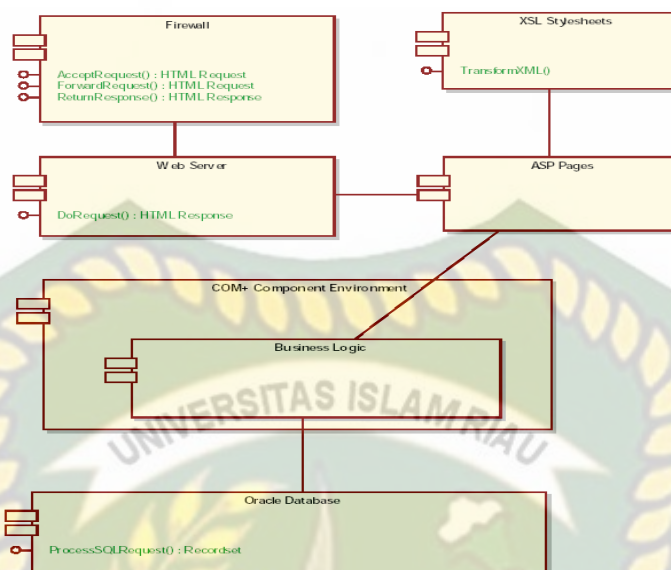


**Gambar 2.6** Collaboration Diagram

### 2.2.11 Komponen 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. Contoh Komponen Diagram ditampilkan pada Gambar 2.7

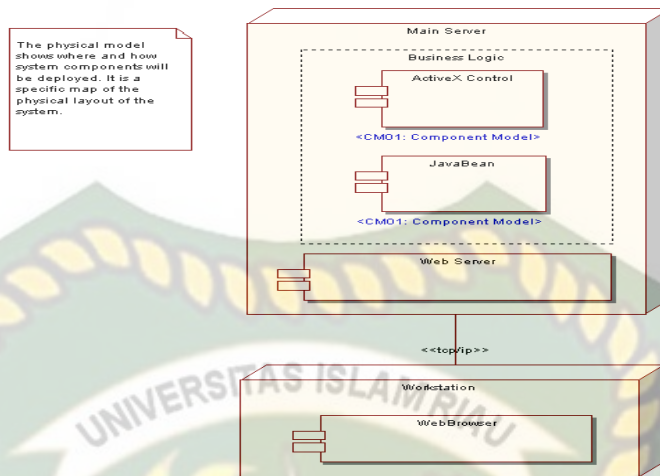


**Gambar 2.7** *Componen Diagram*

### 2.2.12 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 men-*deploy* komponen dalam lingkungan sebenarnya. Hubungan antar *node* (misalnya TCP/IP) dan *requirement* dapat juga didefinisikan dalam diagram ini Contoh Deploement Diagram ditampilkan pada Gambar 2.8



**Gambar 2.8** *Deployment Diagram*

## 2.3 Bahasa Pemrograman Dan Database

### 2.3.1Php Hypertext Processor (PHP)

Php merupakan singkatan dari PHP Hypertext Processor, ia merupakan bahasa yang berbentuk skrip yang ditempatkan dalam server. Hasilnya yang dikirimkan ke klien, tempat pemakai menggunakan browser Abdul Kadir (2008). PHP merupakan singkatan dari "PHP: Hypertext Preprocessor", adalah sebuah bahasa scripting yang terpasang pada HTML. Sebagian besar sintaks mirip dengan bahasa C, Java, asp dan Perl, ditambah beberapa fungsi PHP yang spesifik. Tujuan utama bahasa ini adalah untuk memungkinkan perancang web untuk menulis halaman web dinamik dengan cepat.

### 2.3.2 Mysql

MySQL merupakan salah satu *database server* yang sangat terkenal. Kepopulerannya disebabkan MySQL menggunakan SQL sebagai bahasa dasar untuk mengakses *databasesnya*. Selain itu ia bersifat *open source* pada pelbagai platform (kecuali untuk jenis enterprise, yaitu bersifat komersial). MySQL termasuk jenis RDBMS (*Relational Database Management system*). Itulah sebabnya istilah seperti tabel, garis, baris dan kolom digunakan pada MySQL. Pada MySQL mengandung satu atau sejumlah tabel. Tabel terdiri atas sejumlah baris dan setiap baris mengandung satu atau beberapa kolom Abdul Kadir,(2008). Pengembangan MySQL dilakukan secara bebas, sehingga sama seperti PHP, MySQL dapat diperoleh dan digunakan secara bebas. Meskipun bebas MySQL memiliki kemampuan yang baik sekali dan bisa dibandingkan dengan database lain seperti oracle, PostgreSQL dan lainnya.

MySQL merupakan suatu *engine database* yang *multithread*. Sistem *multithread* mengijinkan MySQL untuk melakukan banyak kegiatan pada waktu yang bersamaan. Thread-thread yang berbeda akan diciptakan untuk menangani setiap koneksi yang masuk, dibantu dengan sebuah *thread* ekstra yang selalu berjalan untuk menangani koneksi ini. Dengan demikian maka banyak client dapat bekerja pada waktu yang sama tanpa mengganggu satu dan lainnya.