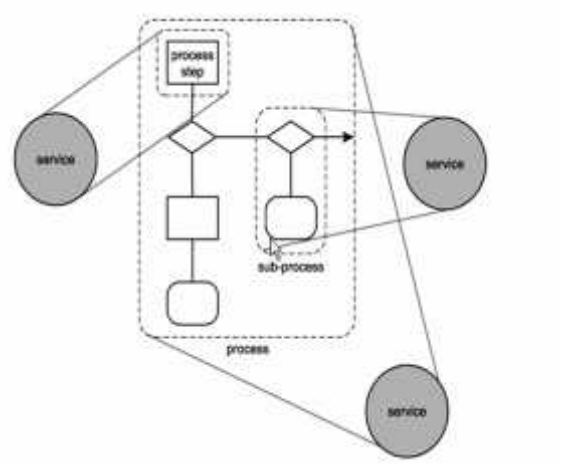
BAB II
LANDASAN TEORI

2.1 *Service Oriented*

Thomas Erl (2005) mengatakan "*Service oriented* merupakan sebuah pendekatan dalam penyelesaian masalah besar dengan membaginya menjadi sekumpulan layanan *(service)* kecil yang menyelesaikan permasalahan secara keseluruhan *(spesifik)"*. Pada dasarnya, *service* adalah suatu enkapsulasi dari unit logis yang dilakukan pada satu atau serangkaian proses pada proses bisnis artinya service merupakan sekumpulan fungsi atau proses yang akan memberi respon jika diminta oleh sebuah client. Dengan penentuan *service* yang didasarkan oleh proses bisnis, maka arsitektur teknologi informasi yang terbentuk dapat lebih mendukung kolaborasi dari segi bisnis dan teknologi informasi. Hal tersebut dapat digambarkan oleh Thomas Erl (2005) seperti pada gambar 2.1.



Gambar 2.1. Enkapsulasi *business process* dengan *service* (Erl,2005)

Seperti dapat dilihat pada gambar 2.1, lingkup dari *service* tidak terbatas, *service* dapat mengenkapsulasi sebuah proses besar atau hanya satu langkah proses kecil. Hal ini dapat disesuaikan tergantung kebutuhan. Misalkan bila dicontohkan dalam kasus belanja online sebuah *service* pengiriman barang yang telah dibayar dapat

didekomposisikan lagi menjadi beberapa langkah yaitu pengepakan barang, pengiriman barang melalui jasa pengiriman barang, pengecekan apakah barang telah dikirim dan sebagainya.

Setelah seluruh permasalahan dapat dibagi dalam beberapa *service*, solusi dari permasalahan tersebut harus bisa diselesaikan dengan memungkinkan seluruh *service* berpatisipasi dalam sebuah orchestra. Untuk itu Thomas Erl (2005) mengatakan bahwa permasalahan yang harus dimiliki oleh *service* yaitu bagaimana *service* berhubungan, bagaimana *service* berkomunikasi, bagaimana *service* didesain dan bagaimana pesan antar *service* didefinisikan.

Pembagian berdasarkan *service* ini sesungguhnya bukan sesuatu yang baru, karena telah banyak diterapkan. Namun hal baru dari pendekatan *service-oriented* ini terkait dengan sifat-sifat yang dimilikinya sesuai dengan pernyataan Erl (2005) yaitu :

1. *Service reusability*, layanan dapat digunakan ulang, sehingga layanan harus didesain agar dapat mendukung penggunaan kembali.
2. *Service contract*, yaitu setiap service memiliki kesepakatan mengenai cara untuk komunikasi.
3. *Service loose coupling*, layanan tidak terhubung erat, maksudnya, layanan didesain agar dapat bekerja tanpa bergantung pada service lain untuk berjalan. Ketergantungan diminimalisir sehingga hanya butuh mekanisme komunikasi satu sama lain.
4. *Service abstraction*, layanan membungkus logika. Bagian yang tampak dari luar hanyalah kontrak layanan. Logika serta implementasi di balik itu tidak kelihatan dan tidak penting bagi peminta layanan.
5. *Service autonomy*, layanan bersifat mandiri. Pengembangan layanan dapat dilakukan terpisah. Layanan memiliki kendali penuh terhadap logika yang dimilikinya.
6. *Service statelessness*, yaitu service tidak memiliki status tertentu terkait dengan aktivitas yang dilakukannya.
7. *Service discoverability*, layanan harus dapat ditemukan. Deskripsi layanan harus dapat dicari dan dimengerti oleh peminta layanan.

2.2 *Service Oriented Architecture* (SOA)

Menurut Mike liu (2010) SOA bukan merupakan suatu teknologi yang spesifik ataupun sebuah bahasa pemrograman yang spesifik, SOA hanya sebuah blueprint/sebuah pendekatan desain sistem. SOA adalah sebuah model arsitektur yang bertujuan untuk menambah kecepatan dan produktifitas dari sistem perusahaan dengan menggunakan komponen atau layanan yang sudah ada.

2.2.1. Karakteristik SOA

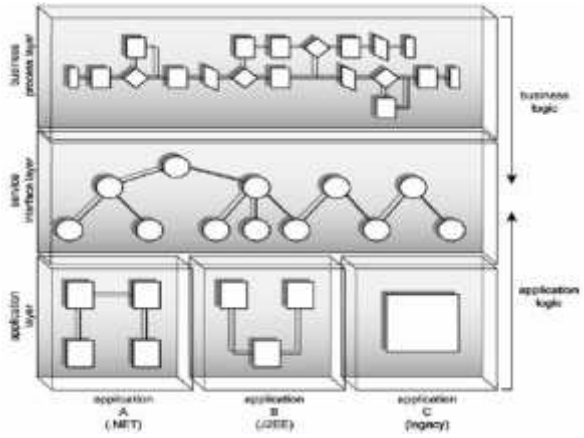
Seperti yang telah didefinisikan sebelumnya, SOA adalah suatu cara pandang aplikasi dengan menggunakan komponen atau layanan yang sudah ada. Dengan kata lain menurut subari (2008), suatu aplikasi dibangun secara modular. Sebenarnya pendekatan modular ini bukanlah sesuatu yang baru. Teknik-teknik pemrograman masa kini seperti object oriented programming, telah mengedepankan pendekatan modular dalam pembangunan aplikasi, namun yang membuat SOA berbeda adalah komponen atau *service* tersebut dibangun dan berinteraksi satu sama lain secara bebas dan lepas *(loose couple)*. Dengan bersifat *loose couple*, sebuah *service* dapat dipanggil oleh program/service lainnya tanpa program pemanggil tersebut perlu memperhatikan dimana lokasi *service* yang dipanggil berada dan platform/teknologi apa yang digunakan oleh service tersebut. Loose coupling sangat penting bagi SOA karena dengan demikian pemanggilan sebuah service oleh service lainnya dapat dilakukan pada saat *run-time.*

Karakteristik lainnya adalah *service* dalam SOA disusun atas 2 hal yaitu *service interface* dan *service implementation*. *Service interface* menyatakan bagaimana *service* tersebut dipanggil seperti parameter input/output dan lokasi ia berada. Misalkan, *service interface* untuk mengecek apakah suatu barang yang dikirim melalui jasa pengiriman telah sampai atau tidak, menyatakan berbagai cara untuk memanfaatkan informasi tentang barang tersebut (dari id barang atau dari pengirim dan sebagainya) dan struktur data barang yang dikembalikan. *Service implementation* adalah bagaimana logic dari *service check* pengiriman barang tersebut dijalankan, *Service implementation* terkait dengan teknologi pemrograman yang

digunakan. Karakter SOA yang terakhir adalah *service* tersebut harus *business oriented* artinya setiap *service* harus melakukan suatu aktivitas bisnis tertentu contohnya seperti *customer lookup, fund transfer, check inventory* (Subari, 2008).

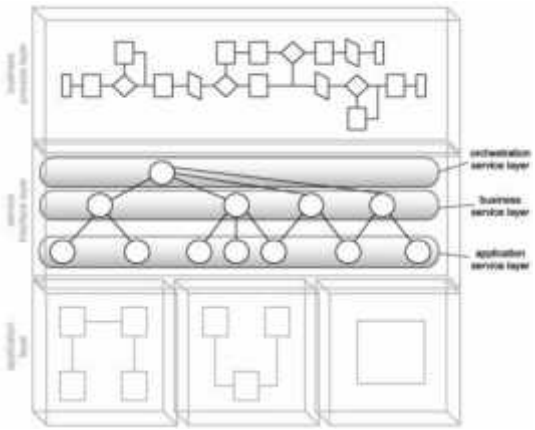
2.2.2 *Layering* Pada SOA

Perangkat lunak yang tidak menggunakan SOA dibagi menjadi dua *layer* utama, yaitu *application layer* dimana aplikasi dijalankan dan *business process layer* yang mendeskripsikan bagaimana proses bisnis dalam perusahaan berjalan (Erl, 2005). Dalam implementasi SOA, konsep *service oriented* yang dipegangnya diimplementasikan dalam sebuah *layer* di antara *business layer* dan *application layer* yang mana keduanya merupakan bagian dari *enterprise logic*. *Layer* tersebut dinamakan *service interface layer* yang fungsinya untuk mengenkapsulasi logic yang ada di *business logic* sehingga dengan pendekatan ini, aplikasi bisa lebih di modularisasi dan menggunakan berbagai macam teknologi seperti dapat dilihat pada gambar 2.2 dimana teknologi .NET pada aplikasi A, J2EE pada aplikasi B, dan aplikasi C akhirnya akan dienkapsulasi oleh *service interface layer* (Erl, 2005).



Gambar 2.2 implementasi layer pada enterprise (Erl,2005)

Service interface ini juga terbagi menjadi 3 layer atau lapisan abstraksi yaitu *application service layer, business service layer, dan orchestration service layer.*



Gambar 2.3 Abstraksi dari *service interface layer* (Erl,2005)

2.2.2.1 *Application Service Layer*

Berdasarkan pendapat Erl (2005), Pada *Application service layer* disediakan sekumpulan *service* yang spesifik untuk mengenkapsulasikan teknologi tertentu yang terdapat didalam *application logic*. Misalkan saja sebuah proses bisnis mengandung aktivitas notifikasi yang mengharuskan pengiriman email kepada pihak yang dinotifikasikan, maka *application service layer* menyediakan layanan untuk mengirim email

2.2.2.2 *Business service layer*

Business service layer merepresentasikan *business logic* dari aplikasi.Layer ini bisa diibaratkan sebagai *controller* dari *application service layer*. Pada *business service layer* inilah fungsi-fungsi bisnis yang berupa aktivitas-aktivitas yang dilakukan untuk menjalankan proses bisnis disediakan Erl (2005).

2.2.2.3 *Orchestration Service Layer*

Menurut Erl (2005) "*Orchestration Service Layer* merupakan sebuah layer yang menyediakan abstraksi dengan level tertinggi dari aplikasi. Pada layer ini semua proses bisnis yang ada di

dalam sistem didefinisikan dan dijalankan dengan menggunakan fungsi-fungsi yang terdapat pada *business service* layer".

2.2.3 Keuntungan SOA

Menurut Subari (2008), Keuntungan yang diberikan oleh SOA tidak hanya terbatas pada penghematan biaya dan tenaga dari upaya pembangunan aplikasi, namun pada akhirnya adalah terwujudnya suatu organisasi yang mampu dengan cepat mengadaptasi proses-proses bisnis didalamnya agar mampu menjawab tuntutan pasar terkini. Ada beberapa pendekatan yang bisa diambil yaitu :

1. Dengan SOA dapat dibuat pemrograman yang sudah ada menjadi *service oriented*. Hal ini bisa dilakukan terhadap program-program yang sudah dibangun secara modular sehingga tinggal menambahkan teknik-teknik web service didalamnya. Jadi dibuatkan *service interface* nya tanpa merubah logika jalannya program. Dengan demikian pengetesannya cukup pada interfacenya saja, sedangkan fungsi/logika didalamnya tidak perlu dites lagi karena tidak berubah.
2. Aplikasi-aplikasi yang sudah ada, terutama aplikasi yang dibeli secara paket seperti ERP, CRM dan lain-lain, dapat "dibungkus" agar dapat dipanggil melalui interface web service. Dari aplikasi-aplikasi tersebut, kita dapat membuat beberapa *service* mulai dari level *service* terendah seperti customer lookup sampai level tertinggi seperti create invoice. Pendekatan ini sangat dimungkinkan mengingat beberapa vendor SOA menyediakan adapter-adapter untuk beberapa paket aplikasi yang terkenal seperti SAP, Siebel dan lain-lain sehingga business logic didalamnya dapat dipublikasikan sebagai *service.*

Dengan membangun suatu koleksi *service*, penghematan dapat dilakukan, Selain itu waktu pengerjaannya juga semakin cepat sehingga memungkinkan perusahaan untuk memberikan layanan bisnis yang makin

responsive terhadap tuntutan pasarnya.Secara garis besar keuntungan SOA yaitu :

1. Kecepatan
2. *Real time* responsive
3. Penghematan
4. Waktu pengembangan lebih singkat
5. *Platform* independent
6. Pengembangan secara *Increment* dan pemanfaatan software yang ada
7. Semakin banyaknya pihak penyedia *Web Service* bukan hanya *Microsoft-Based* tapi juga dari PHP, Oracle, BEA dll.

2.2.4 Cara kerja SOA

Inovasi membutuhkan perubahan dan SOA memudahkannya.SOA bekerja seperti charger untuk semua fungsi, atau dengan kata lain SOA membangun interface yang bisa diakses oleh berbagai macam software. Selama ini, sebuah software dibangun dengan cara mengikat data dan alat pemrosesnya dalam satu rangkaian. Tentu saja, semakin banyak software yang dibutuhkan akan membuat perusahaan mengeluarkan uang dan tenaga lebih banyak lagi. Demikian pula semakin banyaknya lalu lintas data antar software tersebut yang secara otomatis akan meningkatkan ongkos perusahaan. Untuk menggambarkan bagaimana SOA bekerja dalam sebuah perusahaan atau institusi bisnis, dapat mengambil contoh transaksi pembelian barang melalui internet yang dilakukan seorang netter atau pelanggan. Dalam sistem TI pengecer yang menggunakan sebuah SOA, pembelian barang secara online itu memicu serangkaian transaksi lainnya. Misalnya, kartu kredit pelanggan diverifikasi, bagian pengiriman barang diberi tahu, gudang diminta untuk menyesuaikan persediaan barang, dan catatan-catatan pembukuan diperbaharui. Transaksi-transaksi tersebut berupa input informasi yang dikirim melalui sistem atau software-software yang berlainan, yang kadang tidak sesuai dan tidak bisa berhubungan satu sama lain. Namun, teknologi SOA telah memungkinkan infrastruktur yang

mendukung transaksi tersebut untuk dibaurkan dan dikombinasikan secara integral (Afe, 2012).

2.3 SOA dan *Web Services*

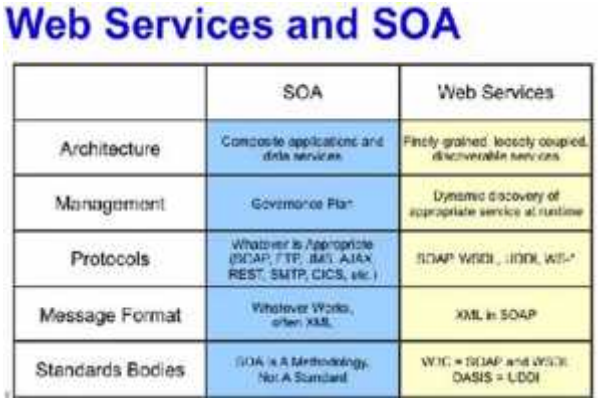
Untuk mengimplementasikan sebuah *service*, terdapat beberapa permasalahan yang harus dimiliki oleh *service* yaitu bagaimana *service* berhubungan, bagaimana *service* berkomunikasi, bagaimana *service* didesain dan bagaimana pesan antar *service* didefinisikan (Erl, 2005). *Web service* merupakan teknologi yang dapat menemukan kebutuhan ini, teknologi ini juga telah jamak digunakan untuk membangun aplikasi SOA. Selain pengaruh dari teknologi web service, ekstensi dari teknologi ini juga memberikan pengaruh terhadap perkembangan SOA. Akibatnya SOA semakin identik dengan *Web Service.* Atas dasar ini Thomas Erl (2005) menyebutnya sebagai *Contemporary* SOA. *Contemporary* SOA merupakan SOA yang menggunakan *Web Service* dan XML dalam implementasinya. Bagaimana *Web Service* sebagai sebuah teknologi dapat memfasilitasi beberapa permasalahan tersebut, berikut penjelasannya :

1. Sebuah *service* dalam SOA adalah sebuah aplikasi web service. Aplikasi ini merepresentasikan sebuah *business logic* atau *automation logic* dari sebuah proses sistem besar yang mencakupinya. Tuntutan dari sistem ini adalah dia harus berdiri sendiri dan bisa berkomunikasi satu sama lain, maka dari itu implementasi *service* dalam SOA adalah *web service*.
2. Hubungan satu *service* dengan yang lainnya di definisikan dengan *web service description language (WSDL)*. Dalam sudut pandang SOA yang menggunakan *web service* sebagai *service*, teknologi WSDL ini menjadi jembatan untuk menghubungkan sebuah *service* ke *service* lainnya.
3. *Service* berkomunikasi satu sama lain menggunakan SOAP *messanging* karna fleksibel dan dapat diperluas.

Web service sangat berbeda dengan *website*, perbedaan yang paling terlihat adalah *website* dibuat untuk memiliki tampilan atau *user interface* yang bagus sedangkan *web service* tidak memiliki tampilan karena *web service* tidak dibuat untuk berinteraksi langsung dengan user melainkan hanya menyediakan *service* atau layanan yang kemudian dapat dipanggil atau digunakan oleh aplikasi lain. Oleh karena itu yangakan menjadi *interface* adalah aplikasi yang memanggilnya bukan *web service* itu sendiri.

Menurut definisi yang dikeluarkan *World Wide Web Consortium* (W3C), *Web Service* adalah suatu sistem perangkat lunak yang dirancang untuk mendukung interaksi mesin ke mesin pada suatu jaringan.*Web Service* menyediakan standar komunikasi di antara berbagai aplikasi *software* yang berbeda – beda. *Web Service* dapat berjalan di berbagai *platform* dan *framework* (Siswoutomo,2004). Untuk memanggil dan memanfaatkan sebuah *web service* dapat digunakan bantuan HTTP (*Hypertext Transfer Protocol*) bisa juga dipanggil dengan menggunakan protocol lain yaitu SMTP (*Simple Transfer Protocol*), namun yang paling umum digunakan adalah protocol HTTP.

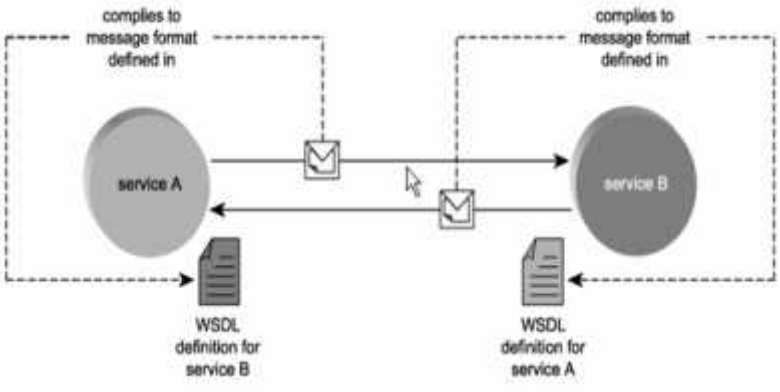
Web service merupakan kumpulan dari fungsi atau method (*service*) yang terdapat pada sebuah server yang dapat dipanggil oleh klien dari jarak jauh. Untuk memanggil method-method tersebut kita bebas menggunakan aplikasi yang dibuat dengan bahasa pemrograman apasaja yang dijalankan diatas platform apa saja. *Web service* dibutuhkan pada saat sekarang ini karena perangkat keras,sistem operasi, aplikasi hingga bahasa pemrograman semakin beraneka ragam jenisnya, keadaan itu dapat menimbulkan masalah dalamproses pertukaran data antar perangkat yang menggunakan aplikasi dan platform yang berbeda. Oleh karena itu digunakan *web service* yang memungkinkan perangkat berbeda itu dapat saling bertukar informasi dengan mudah. Gambaran Web Service dan SOA pada Gambar 2.4 berikut.



Gambar 2.4 Web Service dan SOA (http://www.infoq.com/articles/ieee-software-engineeering-services-cloud-computing)

2.4 WSDL (*Web Service Description Language*)

Sebelum mengakses sebuah *web service*, harus diketahui method-method apa saja yang disediakan oleh *web service* tersebut. Untuk mengetahuinya diperlukan sebuah dokumen yang bernama WSDL. WSDL adalah sebuah dokumen dalam format XML yang isinya menjelaskan informasi detail sebuah *web service*. Didalam WSDL dijelaskan method apa saja yang tersedia dalam web service, parameter apa saja yang diperlukan untuk memanggil sebuah method, dan apa hasilatau type data yang dikembalikan oleh method yang dipanggil tersebut.



Gambar 2.5 Peran WSDL dalam hubungan antar *service* (Erl,2005)

Sebuah dokumen WSDL menggunakan elemen – elemen berikut dalam pendefinisian *network service.*

a. *Types,* sebuah kontainer bagi definisi tipe – tipe data.
b. *Messages,* suatu abstraksi, definisi tipe data yang dikomunikaskan.
c. *Operation,* suatu abstraksi kumpulan operasi – operasi yang didukung oleh satu atau lebih *endpoint.*
d. *Binding*, suatu protokol kongkrit dan spesifikasi format data untuk *port type tertentu.*
e. *Port,* sebuah *endpoint* tunggal yang didefinisikan sebagai suatu kombinasi dari *binding* dan alamat *network.*
f. *Service*, sebuah koleksi dari *endpoint – endpoint* yang berelasi.

2.5 SOAP (*Simple Object Access Protocol*)

Siswoutomo (2004) menyimpulkan bahwa SOAP adalah protokol untuk pertukaran informasi dengan desentralisasi dan terdistribusi.SOAP dibangun dengan menggunakan protokol komunikasi HTTP. Karena HTTP didukung oleh semua *browser* dan *server*, maka SOAP dapat berkomunikasi dengan berbagai aplikasi meskipun terdapat perbedaan sistem operasi, teknologi, dan bahasa pemrogramannya.

Peran SOAP di dalam teknologi *web service* adalah sebagai protokol pemaketan untuk pesan – pesan (*messages*) yang digunakan secara bersama oleh aplikasi – aplikasi penggunanya. Spesifikasi yang digunakan tidak lebih seperti sebuah amplop biasa berbasis XML untuk informasi yang ditransfer, serta sekumpulan aturan bagi translasi aplikasi dan tipe – tipe data *platform* yang spesifik menjadi bentuk XML. Desain bentuk SOAP membuatnya cocok untuk berbagai pertukaran pesan pada aplikasi (Snell, 2001). Sebuah pesan SOAP adalah sebuah dokumen XML yang berisi elemen – elemen berikut:

a. *Envelope element* yang mengidentifikasikan dokumen XML sebagai sebuah pesan SOAP.
b. Elemen *header* yang berisi informasi *header*. Elemen ini bersifat opsional.
c. Elemen *body* yang berisi panggilan dan merespon informasi.

d. *Fault element* yang berisi pesan kesalahan yang terjadi pada waktu proses. Elemen ini opsional.

2.6 XML (*eXtensible Markup Language*)

XML kependekan dari *eXtensible Markup Language*, merupakan sebuah standar W3C-*endorsed* untuk *Markup language* yang dikembangkan mulai tahun 1996 dan baru mendapatkan pengakuan dari W3C pada bulan februari 1998. *Markup language* itu sendiri merupakan suatu bahasa pemrograman untuk menandai suatu dokumen yang disebut dengan tag agar dokumen mudah dibaca, dipahami serta menarik (Kusmayadi, 2008).

Seperti halnya HTML, XML juga menggunakan elemen yang ditandai dengan tag pembuka (diawali dengan ‘<’ dan diakhiri dengan ‘>’), tag penutup(diawali dengan ‘</ ‘diakhiri ‘>’) dan atribut elemen(parameter yang dinyatakan dalam tag pembuka misal <form name=”isidata”>). Hanya bedanya, HTML medefinisikan dari awal tag dan atribut yang dipakai didalamnya, sedangkan pada XML kita bisa menggunakan tag dan atribut sesuai kehendak kita (Junaedi, 2003). Untuk lebih jelasnya berikut contohnya:

<pesan>

<dari>MIS Manager</dari>

<buat>HRD Manager</buat>

<buat>Bagian rekrut</buat>

<buat>Computer Suport team</buat>

<subyek>Permohonan Tenaga kerja baru</subyek>

<isi>Mohon diberikan tenaga kerja baru untuk mengisi lowongan di Departemen MIS</isi>

</pesan>

Jadi, XML adalah bahasa yang digunakan oleh Web service dan aplikasi, sedangkan “tata bahasa” nya adalah SOAP sehingga keduanya dapat saling mengerti dan memahami saat sedang “berbicara” (Lucky, 2008)

2.7 UML (*Unified Modelling Language*)

UML adalah sebuah bahasa untuk menentukan, visualisasi kontruksi, mendokumentasikan *artifacts* (model, deskripsi, atau *software*) dari sistem *software*, untuk memodelkan bisnis, dan sistem *non-software* lainnya.UML merupakan suatu kumpulan teknik terbaik yang telah terbukti sukses dalam memodelkan sistem yang besar dan kompleks.Berikut adalah diagram-diagram UML yang ada:

a. *Usecase Diagram*

Menurut Joseph Schmuler (2004) sebuah use case adalah sebuah deskripsi dari sistem dari sudut pandang user. Untuk seorang pengembang sistem,use case adalah sebuah alat yang penting.

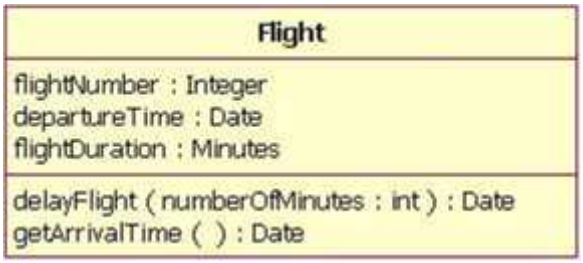
Diagram usecase biasanya digunakan untuk mengumpulkan kebutuhan-kebutuhan dari sistem yang ingin dibuat dari sudut pandang user kemudian pengembang akan melakukan analisa terhadap kebutuhan dari klien tersebut dengan menentukan actor-aktor apa saja yang terlibat didalam sistem atau yang akan menggunakan sistem dan juga menentuka proses apa saja yang ada didalam sistem.Diagram use case itu terdiri dari aktor dan use case

b. *Class diagram*

Merupakan digram yang selalu ada di pemodelan sistem berorientasi objek.Class diagram menunjukkan hubungan antara kelas dalam sistem yang sedang dibangun dan bagaimana mereka saling berkolaborasi untuk mencapai suatu tujuan. Dalam class diagram setiap kelas itu diwakilkan dengan sebuah kotak yang berisi 3 bagian yaitu :

1. Bagian atas berisi nama kelas
2. Bagian tengah berisi atribut-atribut dari kelas

3. Bagian bawah berisi metode atau operasi yang kelas tersebut dapat lakukan



Gambar 2.6 Notasi kelas di dalam class diagram (IBM)

c. *Sequence Diagram*

Sequence diagram menggambarkan sekelompok objek dan interaksi antar objek tersebut, termasuk pesan yang dikirim dari satu objek ke objek lain terhadap waktu.

Sequence diagram adalah suatu diagram yang menggambarkan interaksi objek dan mengindikasikan komunikasi diantara objek-objek tersebut. Diagramini menunjukkan serangkaian pesan yang dipertukarkan oleh objek-objek yang melakukan suatu tugas atau aksi tertentu.

d. *Activity Diagram*

Menurut Schmuler (2004) sebuah activity diagram didesain untuk memperlihatkanapa yang terjadi didalam sebuah operasi atau sebuah proses.

Diagram-diagram yang ada di dalam activity diagram :

1. Initial state
2. Final State
3. Activity
4. Percabangan

2.8 UDDI(*Universal Description, Discovery and Integration*)

UDDI merupakan suatu directory serice yang digunakan untuk meregistrasikan dan mencari web service. Dengan menggunakan UDDI, web

service yang dibuat dapat dicari dan ditemukan oleh orang lain berdasarkan kata kunci dan kategori tertentu (Lucky, 2008).

2.9 Kuesioner

Kuesioner merupakan instrumen pengumpulan data atau informasi yang dioperasionalisasikan ke dalam bentuk *item* atau pertanyaan. Penyusunan kuesioner dilakukan dengan harapan dapat mengetahui variabel- variabel apa saja yang menurut responden merupakan hal yang penting.

Tujuan penyusunan kuesioner adalah untuk memperbaiki bagian-bagian yang dianggap kurang tepat untuk diterapkan dalam pengambilan data terhadap responden. Menurut Suharsini (2010) klasifikasi kuesioner terbagi atas 2 jenis, yaitu:

1. Kuesioner langsung dan tidak langsung

   Suatu kuesioner dikatakan langsung apabila kuesioner tersebut dikirim langsung kepada orang yang dimintai pendapat. Sebaliknya, apabila kuesioner dikirimkan kepada seseorang yang dimintai pendapat mengenai keadaan orang lain, maka disebut kuesioner tidak langsung.

2. Kuesioner terbuka dan tertutup

   Kuesioner tertutup merupakan kuesioner yang menghendaki jawaban pendek, atau jawabannya diberikan dengan membubuhkan tanda tertentu. Daftar pertanyaan disusun dengan disertai alternatif jawaban, responden diminta untuk memilih salah satu jawaban atau lebih dari altenatif yang disediakan. Sedangkan kuesioner terbuka merupakan kuesioner yang berupa item-item pertanyaan yang tidak disertai alternative jawaban, melainkan mengharapkan responden untuk mengisi dan memberi komentar atau pendapat.

2.9.1 Skala Likert

Skala Likert menurut Djaali (2008) ialah skala yang dapat dipergunakan untuk mengukur sikap, pendapat, dan persepsi seseorang atau sekelompok orang tentang suatu gejala atau fenomena pendidikan. Skala Likert adalah suatu skala psikometrik yang umum digunakan dalam

kuesioner, dan merupakan skala yang paling banyak digunakan dalam riset berupa survei. Nama skala ini diambil dari nama Rensis Likert, pendidik dan ahli psikoloog Amerika Serikat. Sewaktu menanggapi pertanyaan dalam skala *Likert*, responden menentukan tingkat persetujuan mereka terhadap suatu pernyataan dengan memilih salah satu dari pilihan yang tersedia. Biasanya disediakan 5 pilihan skala, kadang digunakan juga skala dengan tujuh atau Sembilan tingkat.

2.9.2 Pengolahan Data

Untuk menghitung presentase masing-masing variabel penelitian dari kuesioner dapat menggunakan perhitungan rata-rata persentase dengan rumus sebagai berikut :

$$Y = \frac{P * 100}{Q * R}$$

Keterangan:

P = Banyaknya jawaban responden tiap Variabel

Q = Jumlah responden

R = Banyak Soal tiap jenis pengujian

Y = Nilai Persentase

2.10 Penelitian Sebelumnya

Pada penelitian ini akan diambil beberapa penelitian sejenis yang pernah dilakukan sebelumnya dengan studi kasus yang berbeda sebagai salah satu bahan referensi. Diantara penelitian itu adalah :

1. Penelitian yang dilakukan oleh Deviana Hartati (2011), Mahasiswi Politeknik Negeri Sriwija yang berjudul "Penerapan *XML Web service* Pada Sistem Distribusi Barang". Dalam penelitian ini *Web Service* diimplementasikan pada sistem pengelolaan distribusi barang di sebuah apotek yang memiliki beberapa cabang dengan studi kasus PT. Apotik

Plus Palembang. Menurut Deviana (2011) Dari penelitian ini diambil beberapa kesimpulan bahwa sistem informasi dengan dukungan teknologi *web service* dapat memberikan informasi transaksi data pada proses pelayanan pemesanan maupun pengiriman barang secara lengkap karena XML skema dan tipe data dapat dilihat pada arus transformasi data antara client dan server. *Web service* dapat digunakan untuk mengintegrasikan dua DBMS yang berbeda yaitu Microsoft SQL Server dan Microsoft Access untuk digunakan pada satu aplikasi yang sama.

2. Penelitian yang dilakukan oleh Hendro Widhiarto (2010), Mahasiswa UGM yang berjudul implementasi teknologi web service pada aplikasi mobile Profile Pariwisata Boyolali. Dalam penelitian ini *Web Service* diimplementasikan pada Aplikasi berbasis *Mobile* untuk informasi *Profile* Pariwisata Boyolali. Database yang digunakan adalah MySQL dan Mobile Database. Kesimpulan yang didapat adalah *Web Service* dapat digunakan untuk mengintegrasikan dua database yang berbeda tersebut untuk digunakan pada satu aplikasi yang sama yaitu aplikasi *Mobile Profile* pariwisata Boyolali.