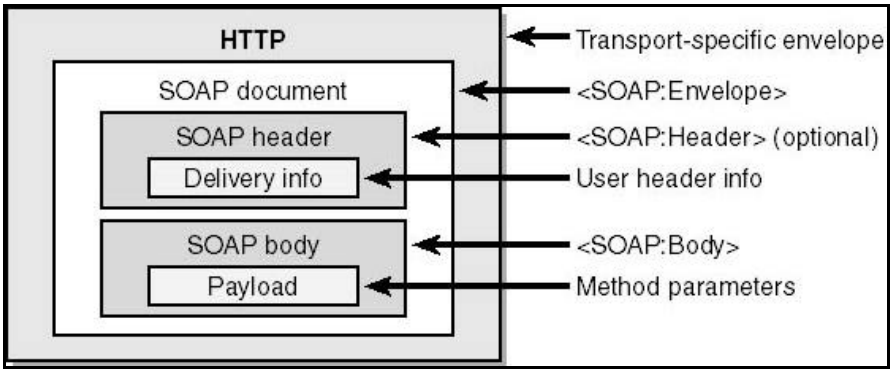
2. TEORI DASAR

2.1. SOAP (*Simple Object Access Protocol*)

SOAP menyediakan suatu mekanisme sederhana untuk pertukaran informasi terstruktur antara *peers* (bagian dari suatu jaringan) dalam suatu kelompok tertentu, mendistribusikannya dalam suatu ruang lingkup dengan menggunakan XML (*eXtensible Markup Language*). SOAP tidak didefinisikan berupa beberapa aplikasi semantik seperti model pemrograman atau implementasi *semantic* tertentu, melainkan didefinisikan sebagai mekanisme sederhana untuk mengekspresikan aplikasi *semantic* dengan menyediakan model pengemasan atau penanganan suatu modul dan mekanisme pembawaan data guna pembacaan data di dalam modul tersebut. Dengan definisi tersebut SOAP mencakup beragam sistem mulai dari sistem pertukaran *message* (pesan) hingga RPC (*Remote Procedure Call).* SOAP berpotensi untuk digunakan dalam banyak macam kombinasi dengan *protocol* lainnya.

2.1.1. Struktur SOAP *Message*

Dokumen SOAP adalah sebuah amplop elektronik untuk diisi dengan informasi yang ingin disampaikan. Isi dari informasi tersebut terdiri dari tanda-tanda yang menjelaskan tentang metode apa yang digunakan untuk mengambil dan mengirimkan data. Pada gambar 2.1 dapat dilihat model SOAP *package* .



Gambar 2.1

SOAP *Package*.

Sebuah dokumen SOAP mempunyai 2 elemen, SOAP *header* dan SOAP *body*. Elemen SOAP *header* mengandung informasi tentang transaksi yang sedang dilakukan. Informasi ini ditentukan oleh *user* dan isinya tegantung pada apa yang ingin diinginkan. Pada gambar 2.2 diberi contoh elemen SOAP *header* yang mengindikasikan sebuah elemen *Transaction* dengan nilai *5*. Sisi penerima dari *message* dapat melakukan sesuatu dengan nilai tersebut untuk kepentingan tertentu. Elemen *Transaction* pada gambar 2.2 didefinisikan sebagai *namespace* yang ditentukan oleh *user* dengan nama *trans.*

```
<SOAP:Header>
   <trans:Transaction
      xmlns:trans="http://schemas.architag.com/transaction.xsd"
      SOAP:mustUnderstand="1">5</trans:Transaction>
</SOAP:Header>
```

Gambar 2.2

Contoh SOAP *Header*

2.1.2. Hubungan SOAP dan XML

Semua SOAP *message* ditulis dengan menggunakan XML (*eXtensible Markup Language*). XML merupakan sintak sederhana yang dapat digunakan untuk membuat bahasa terstruktur atau dikenal dengan *schema* dengan penjelasan data-data di dalamnya. Dalam prakteknya, XML lebih dari sekedar bahasa terstruktur, XML memungkinkan komunikasi terjadi antara beberapa pihak dengan jalan yang tidak mungkin terjadi pada teknologi yang berbeda-beda.

Aplikasi SOAP seharusnya menyertakan *namespace* yang sesuai pada semua elemen dan atribut, sesuai dengan atribut yang didefinisikan dalam SOAP *message* yang dihasilkan. Sebuah aplikasi SOAP harus mampu untuk memproses SOAP *namespace* yang diterimanya, dan harus mengabaikan *namespace* dalam *message* yang tidak sesuai. SOAP *message* tidak boleh memuat deklarasi tipe dokumen dan kumpulan instruksi pemrosesan *message*.

SOAP menentukan dua *namespace*, yaitu :

a. Identitas *namespace* SOAP *envelope* "*http://schemas.xmlsoap.org/soap/envelope/"*
b. Identitas *namespace* SOAP *serialization* "*http://schemas.xmlsoap.org/soap/encoding/*"

2.1.3. SOAP *Encoding*

SOAP *encoding* berbasis pada sistem sederhana yaitu generalisasi fitur-fitur utama pada beberapa sistem bahasa pemrograman, database dan data semi terstruktur. *Namespace* untuk elemen and atribut yang dibahas pada bagian ini adalah "*http://schemas.xmlsoap.org/soap/encoding/*".

2.1.3.1. Aturan Penulisan SOAP dalam XML

XML memberikan kemudahan dan kefleksibelan dalam penulisan data. SOAP menentukan aturan sederhana dalam penyusunannya, untuk itu digunakan terminologi, antara lain:

1. Suatu nilai adalah *string*, nama dari suatu obyek (angka, tanggal, dan lainnya) atau terdiri dari gabungan beberapa nilai primitif. Semua nilai mempunyai tipe spesifik.
2. Nilai sederhana ialah nilai tanpa nama. Misalnya nilai sederhana dari beberapa *string* adalah *string* khusus, *integer*, angka-angka tertentu.
3. Nilai gabungan merupakan kumpulan dari hubungan nilai-nilai lainnya. Misalnya nilai gabungan dari order pembelian, laporan stok, alamat tempat tinggal, dan lainnya.
4. Di dalam nilai gabungan, tiap-tiap nilai penghubung potensial untuk dibedakan berdasarkan nama aturan. Hal ini disebut juga sebagai *accessor.*
5. Suatu *struct* merupakan nilai gabungan dimana nama *accessor* hanya membedakan diantara nilai-nilai di dalamnya, dan tidak ada *accessor* memiliki nama sama.
6. Tipe sederhana merupakan suatu *class* dari nilai sederhana. Contoh tipe sederhana adalah *class* dengan nama *string*, *integer*, dan *class* lainnya.

7. Tipe gabungan merupakan *class* dari nilai gabungan. Tipe gabungan adalah *class* dari order nilai pembelian yang berbagi dengan *accesso*r yang sama (tujuan, total biaya, isi, dan lainnya) meski memiliki nilai yang berbeda-beda.
8. Di dalam tipe gabungan, jika suatu *accessor* memiliki nama yang berbeda dengan tipenya tetapi tidak berbeda dengan yang diharapkan tipe lainnya, maka nama dan tipe keduanya dibutuhkan untuk membuat identitas unik, yang disebut *locally scoped*. Bagaimanapun nama tipe berdasar pada URI , secara langsung atau tidak langsung, nama tersebut dikatakan *universally scoped*.
9. Pemberian informasi dalam *schema* secara relatif suatu grafik dikenali, sangat memungkinkan untuk menerangkan bahwa beberapa nilai dihubungkan dengan *accessor* tunggal. Lainnya, tidak mungkin untuk dilakukan. Jika hanya satu *accessor* dapat digunakan untuk mereferensikan sesuatu, nilainya disebut sebagai *single-reference*. Jika mereferensikan lebih dari satu nilai, secara aktual atau potensial disebut *multi-reference*. Perlu dicatat bahwa mungkin untuk nilai tertentu untuk dinyatakan sebagai hubungan *single-reference* pada satu *schema* dan *multi-reference* terhubung dengan lainnya.
10. Secara sintaks, suatu elemen mungkin berdiri sendiri atau terikat dengan lainnya. Sebuah elemen independen adalah setiap elemen yang muncul pada level teratas dari suatu tingkatan level. Elemen lainnya adalah elemen tambahan.

Meskipun memungkinkan untuk menggunakan *xsi:type* seperti nilai dari suatu gambar adalah bersifat *self-describing* dalam strukturnya sendiri dan tipe dari nilai-nilainya, aturan yang ada membolehkan tipe nilai untuk ditentukan dengan referensi pada suatu *schema*. Mungkin beberapa *schema* dijelaskan dengan cara *"XML Schema Part 1: Structures" [10]* dan *"XML Schema Part 2: Datatypes" [11]* atau mungkin dengan cara penulisan lainnya Beberapa aturan menerapkan menggabungkan tipe selain *struct* dan *array*, beberapa *schema* hanya memuat tipe *struct* dan *array* saja.

Beberapa aturan *schema* yang digunakan :

1. Semua nilai direpresentasikan sebagai isi elemen. Nilai *multi-reference* harus direpresentasikan sebagai isi dari sebuah elemen independen, hal ini tidak berlaku untuk *single-reference* tetapi kadang-kadang mungkin berlaku.

2. Untuk tiap elemen mengandung nilai, tiap nilai harus direpresentasikan setidaknya satu dari kondisi berikut:
   - Elemen yang mengandung atribut *xsi:type*
   - Elemen yang elemen itu sendiri di dalamnya mengandung elemen atribut *SOAP-ENC:arrayType*.
   - Atau nama dari elemen menunjang definisi yang berhubungan dengan tipe, tipe yang berkaitan dengan *schema*.
3. Nilai sederhana direpresentasikan sebagai data karakter tanpa sub elemen. Setiap nilai sederhana harus memiliki satu tipe yang terdapat pada spesifikasi skema *XML*..
4. Nilai gabungan dibuat dengan urutan elemen, tiap *accessor* direpresentasikan dengan elemen tambahan dengan nama yang menunjuk pada nama dari *accessor*.

2.1.4. SOAP *Envelope*

Struktur SOAP *envelope* merupakan gambaran keseluruhan pekerjaan untuk mengekspresikan apa yang terdapat di dalam sebuah *message*, siapa yang seharusnya terlibat, dan kondisi atau persyaratan sebagai pilihan dalam penggunaannya. Sebuah SOAP *message* merupakan dokumen XML yang terdiri dari gabungan SOAP *envelope*, SOAP *header* dan SOAP *body*. Dokumen XML ini menunjukkan suatu SOAP *message* dengan isi seperti tersebut di atas. *Namespace* mengidentifikasikan elemen dan atribut-atribut dengan "*http://schemas.xmlsoap.org/soap/envelope/*". Suatu SOAP *message* memuat hal-hal sebagai berikut :

a. *Envelope* adalah elemen paling atas dari dokumen XML merepresentasikan isi *message*.
b. *Header* merupakan mekanisme generik untuk menambahkan fitur-fitur pada SOAP *message* dalam kegunaan tertentu tanpa persetujuan terlebih dahulu antara pihak-pihak yang saling berhubungan. SOAP menentukan sedikit atribut yang dapat digunakan untuk mengindikasikan kepada pihak mana yang dituju dengan sebuah fitur dan pilihan kondisi atau perintah.

c. *Body* memuat informasi perintah untuk penerima *message.* SOAP mendefinisikan satu elemen yaitu elemen *fault* untuk melaporkan kesalahan yang terjadi.

Global SOAP *encodingStyle* atribut dapat digunakan untuk mengindikasikan serial dari aturan yang digunakan dalam SOAP *message.* Nilai atribut adalah daftar dari satu atau lebih URI[1] yang mengidentifikasikan aturan serial atau aturan-aturan yang dapat digunakan untuk memproses SOAP *message* agar didapatkan indikasi dari hal paling spesifik hingga kurang spesifik. Contoh dari nilai-nilai adalah :

```
"http://schemas.xmlsoap.org/soap/encoding/"
```

Pada umumnya sintaks URI diawali dengan "*http://schemas.xmlsoap.org/soap/encoding/*" mengindikasikan keterkaitan dengan SOAP *encoding rules.* Nilai dari URI kosong (“”) menjelaskan bahwa tidak ada *encoding style* yang digunakan oleh elemen di dalamnya.

SOAP tidak menjelaskan dengan menyertakan seri angka untuk menyatakan model versinya. Sebuah SOAP *message* harus memiliki sebuah elemen *envelope* yang telah terasosiasi dengan “*http://schemas.xmlsoap.org/soap/envelope/*”. Jika sebuah *message* diterima oleh sebuah aplikasi SOAP dimana elemen SOAP *envelope* terasosiasi dengan *namespace* yang lain, aplikasi tersebut harus mampu mengetahui bahwa terjadi suatu kesalahan versi dan mengabaikan *message*. Jika *message* tersebut diterima melalui sebuah protokol *request/response* seperti HTTP, aplikasi tersebut harus merespon dengan sebuah pesan kesalahan SOAP *VersionMismatch* menggunakan *namespace "http://schemas.xmlsoap.org/soap/envelope/*".

2.1.5. SOAP *Header*

SOAP *header* menyediakan mekanisme fleksibel untuk menangani sebuah *message* tanpa pemahaman sebelumnya antara pihak-pihak yang saling berkomunikasi. Beberapa contoh implementasi sebagai *header* adalah pemberitahuan, transaksi, pembayaran dan sejenisnya.

---

[1] *Uniform Resource Identifier* (URI) adalah sekumpulan string untuk mengidentifikasikan abstraksi atau bentuk fisik dari *resource*.

Elemen *header* tercantum sebagai elemen pertama dari elemen XML SOAP *envelope*. Semua elemen dari elemen *header* disebut sebagai *header entries* Beberapa aturan penulisan *header entries* tercantum sebagai berikut :

1. Suatu *header entry* diidentifikasikan dengan nama penuh kualifikasi elemen, terdiri dari *namespace* URI dan nama lokal. Semua elemen anak dari elemen SOAP *header* harus berupa kualifikasi *namespace*.
2. Atribut SOAP *encodingStyle* dapat digunakan untuk mengindikasikan tipe atau model *encoding* dari *header entries*.
3. Atribut SOAP *mustUnderstand* dan atribut SOAP *actor* dapat digunakan untuk mengindikasikan bagaimana cara memproses *entry* dan oleh siapa boleh dilakukan

2.1.5.1. Atribut SOAP *Header*

Atribut SOAP *header* menjelaskan bagaimana penerima SOAP *message* seharusnya memproses *message*. Suatu aplikasi SOAP memproses sebuah SOAP *message* seharusnya hanya menggunakan atribut SOAP *header* sebagai elemen anak dari elemen SOAP *header*. Penerima tersebut harus mengabaikan semua atribut SOAP *header* yang tidak tercantum sebagai elemen anak dari elemen SOAP *header*. Sebuah contoh *header* dengan elemen identitas *"Transaction*", nilai dari "*mustUnderstan*d" adalah *"1*", dan nilai *5*. Menjadi seperti berikut:

```
<SOAP-ENV:Header>
   <t:Transaction
      xmlns:t="some-URI" SOAP-ENV:mustUnderstand="1">
          5
   </t:Transaction>
</SOAP-ENV:Header>
```

2.1.5.2. Atribut SOAP *Actor*

Sebuah SOAP *message* berpindah dari pengirim ke tujuan secara potensial melalui sekumpulan media penghubung SOAP sepanjang jalur *message* tersebut. SOAP *intermediary* adalah aplikasi yang memungkinkan keduanya menerima dan mengirimkan SOAP *message*. Tidak semua bagian dari SOAP *message* dimaksudkan untuk tujuan utama dari SOAP *message*, melainkan dapat

ditujukan untuk satu atau lebih perantara sepanjang jalur yang dilalui oleh *message*. Atribut global SOAP *actor* dapat digunakan untuk mengindikasikan penerima dari elemen *header*. Nilai dari atribut SOAP *actor* adalah suatu URI. URI spesial *"http://schemas.xmlsoap.org/soap/actor/next"* mengindikasikan bahwa elemen *header* ditujukan untuk aplikasi SOAP yang paling awal memprosesnya.

2.1.5.3. Atribut SOAP *mustUnderstand*

Atribut ini dapat digunakan untuk mengindikasikan bagaimana sebuah *header entry* adalah bersifat instruksi atau pilihan bagi penerimanya. Penerima dari suatu *header entry* didefinisikan dengan atribut SOAP *actor*. Nilai atribut *mustUnderstand* adalah "1" dan "0".

Ketiadaan atribut SOAP *mustUnderstand* dilambangkan dengan penulisan atribut ini dengan nilai "0".

Jika elemen header dengan sebuah atribut SOAP *mustUnderstand* dengan nilai "1", penerima dari SOAP *message* harus mematuhi aturan dan melakukan proses sesuai dengan aturan tersebut, atau harus benar-benar memprosesnya. Atribut SOAP ini memungkinkan terjadinya perubahan. Elemen yang ditandai dengan atribut SOAP *mustUnderstand* dengan nilai "1" harus diartikan sebagai cara untuk memodifikasi induk atau anak suatu elemen. Menandai elemen dengan cara tersebut menjamin perubahan yang dilakukan tidak diabaikan oleh mereka yang tidak memahami sepenuhnya.

2.1.6. SOAP *Body*

Elemen SOAP *body* menyediakan mekanisme sederhana untuk pertukaran informasi kepada penerima *message*. Beberapa bentuk penggunaan elemen *body* disertai dengan pemanggilan RPC (*Remote Procedure Calls*) dan laporan kesalahan yang terjadi. Elemen ini dimuat sebagai elemen anak dari elemen SOAP *envelope* XML. Jika terdapat suatu elemen *header* maka elemen *body* harus segera mengikuti elemen *header*, dengan kata lain elemen ini harus menjadi elemen anak pertama pada *envelope*. Semua elemen anak dari elemen *body* disebut *body entries* dan tiap-tiap *body entry* dibentuk sebagai elemen

independen di dalam elemen SOAP *body*. Aturan yang harus diperhatikan dalam pembuatan *body entries* adalah sebagai berikut:

a. *Body entry* diidentifikasikan dengan nama elemen, yang mana terdiri dari *namespace* URI dan nama lokal. Elemen anak selanjutnya dari elemen SOAP *body* bisa saja menggunakan *namespace*.
b. Atribut SOAP *encodingStyle* dapat digunakan untuk menunjukkan bentuk yang digunakan oleh *body entries*.

SOAP memiliki satu *body entry*, dimana *Fault entry* digunakan untuk pelaporan *error* (kesalahan).

Sementara *header* dan *body* dideklarasikan sebagai elemen independen, nyatanya kedua elemen tersebut memiliki hubungan. Hubungan diantara suatu *body entry* dan suatu *header entry* adalah suatu *body entry* mempunyai nilai *header entry* yang sama dan dengan satu atribut SOAP *mustUnderstand* dengan nilai "1".

2.1.7. SOAP *Fault*

Elemen SOAP *fault* digunakan untuk memuat status *error* dalam sebuah SOAP *message*. Jika terjadi kesalahan, elemen SOAP *fault* muncul sebagai *body entry* dan tidak dapat muncul lebih dari satu kali pada elemen *body*. Ada 4 elemen dalam SOAP *fault*, yaitu :

a. *Faultcode*, elemen ini digunakan untuk menyediakan mekanisme tersusun untuk mengetahui kesalahan yang terjadi.
b. *Faultstring*, elemen ini ditujukan untuk memudahkan pengguna untuk mengetahui keterangan atau penjelasan berupa kata-kata atas kesalahan yang terjadi.
c. *Faultactor*, elemen ini memberitahukan penyebab kesalahan yang terjadi di sepanjang jalur yang dilalui oleh *message*.
d. *Detail*, elemen *detail* menyampaikan informasi *error* dari aplikasi yang berhubungan dengan elemen *body*. Elemen ini harus dihadirkan jika isi dari elemen *body* tidak dapat diproses dengan sukses, selain itu elemen *detail* tidak seharusnya digunakan untuk membawa informasi *error* berkaitan dengan *header entries*. Keterangan informasi kesalahan atas *header entries* harus

disampaikan melalui *header entries.* Ketiadaan elemen *detail* pada elemen *fault* mengindikasikan kesalahan tidak berkaitan dengan elemen *body*, sehingga dapat membingungkan apakah elemen *body* telah diproses atau tidak pada kesalahan yang sedang terjadi.

2.1.8. Penggunaan SOAP pada HTTP

Implementasi SOAP pada HTTP memberikan keuntungan yaitu kemudahan untuk menentukan bentuk formal dan fleksibilitas dari SOAP dengan banyak fasilitas dari HTTP seperti HTTP *request/response*. Penyampaian/pengiriman SOAP melalui HTTP tidak berarti menumpuki atau mengganti aturan-aturan yang ada di HTTP tetapi melainkan digunakannya aturan-aturan standar HTTP. SOAP secara dasar mengacu pada model HTTP *request/response message* dengan menyediakan parameter SOAP *request* dalam sebuah HTTP *request* dan parameter SOAP *response* pada HTTP *response*. Sehingga media penghubung SOAP tidak sama dengan media penghubung yang dimiliki HTTP, artinya media penghubung HTTP dialamatkan dengan HTTP *connection header field*, tidak dapat digunakan untuk memproses SOAP *body* yang disampaikan melalui HTTP *request*. Aplikasi HTTP harus menggunakan tipe media "*text/xml*" mengacu pada RFC 2376 ketika menyertakan SOAP *body* dalam HTTP *message*.

2.1.8.1. SOAP HTTP *Request*

Dalam penyampaian SOAP *message* melalui metode HTTP *request*, digunakan SOAP *action* HTTP *request header field* untuk mengindikasikan maksud dari SOAP HTTP *request*. Nilai dari URI menjelaskan maksud dari *request* tersebut. HTTP pada klien harus menggunakan *header field* ketika melakukan suatu SOAP HTTP *Request*. Contoh dari HTTP *header field* yang digunakan adalah sebagai berikut:

```
soapaction    = "SOAPAction" ":" [ <"> URI-reference <"> ]
```

Isi dari SOAPAction *header field* dapat digunakan oleh *server* atau semacam untuk mengetahui SOAP *request message* dalam HTTP. Nilai *header*

field tersebut mempunyai arti bahwa SOAP *message* diatur oleh HTTP *Request*-URI.:

2.1.8.2. SOAP HTTP *Response*

SOAP HTTP mengikuti aturan dari pengkodean HTTP status untuk mengkomunikasikan informasi status dalam HTTP. Contohnya, kode status '2xx' mengindikasikan *request* dari klien disertai komponen SOAP telah sepenuhnya diterima dan dimengerti. Sedangkan kesalahan pemprosessan SOAP terjadi pada saat *request* , *server* SOAP HTTP harus memberitahukan HTTP 500 "*Internal Server Error*" dan menyertakan SOAP *message* dalam *response* yang berisi suatu elemen SOAP *fault* untuk mengindikasikan kesalahan pemrosesan SOAP.

2.1.9. Penggunaan SOAP untuk RPC

Salah satu kegunaan dari SOAP adalah untuk memungkinkan pertukaran RPC (*Remote Procedure Calls*) dengan memanfaatkan ekstensibilitas dan fleksibilitas dari XML. Penggunaan SOAP untuk RPC berdasar pada protokol SOAP. HTTP sebagai protokolnya, suatu panggilan RPC dipetakan pada HTTP *request* dan suatu RPC *response* dipetakan pada HTTP *response*. Untuk melakukan suatu metode, beberapa informasi diperlukan seperti berikut:

a. Nama metode.
b. Pilihan metode *signature*.
c. Parameter yang digunakan dalam metode tersebut.
d. *Header* dari data bila diperlukan.

SOAP berdasar pada protokol yang digunakan untuk menyelenggarakan sebuah mekanisme untuk memuat URI. Contohnya, untuk HTTP permintaan URI mengindikasikan pada *resource* bahwa suatu proses sedang dilakukan. Selanjutnya bila URI tersebut mempunyai nilai kebenaran, SOAP menentukan tidak adanya larangan pada alamat yang dituju. Suatu SOAP *message* menggunakan cara standar untuk menserialisasikan obyek dan struktur data lainnya dalam XML, tersedianya RPC diartikan sebagai klien yang berpikir tentang metode pemanggilan pada obyek lokal adn diwujudkan dengan mentranslasikannya melalui XML dan terkirim melalui kabel (*internet*), diproses

di ujung yang lain, dan *response message* tanpa diketahui memuat nilai balik dan metedo pemanggilannya. Suatu pemanggilan RPC pada *server* bisa diartikan dengan mengaktifkan obyek dan memanggil metode atau properti yang ada di dalamnya.

2.2. EDI (*Electronic Data Interchange*)

Selama lebih dari 20 tahun, EDI telah dikembangkan untuk mengembangkan sistem *paperless trading*, yaitu dengan mengusahakan sedikit mungkin penggunaan dokumen fisik/*hard copy* dalam transaksi bisnis. Penggunaan komputer dan komunikasi merupakan bagian terpenting dari EDI. Konsep dasar dari EDI yaitu mengolah dokumen yang dipertukarkan, seperti daftar order dan surat penagihan, dokumen ditransmisikan secara langsung ke komputer *partner* dari transaksi itu sendiri melalui jaringan telekomunikasi.

Layanan EDI terdiri dari jaringan komunikasi dan layanan *mailbox* utama dengan menggunakan *software* sebagai penerjemah dan alur dari data elektronik atau EDI *form*. Teknologi ini sudah disertai dengan *system* terintegrasi dengan komputer pusat yang diakses oleh jaringan melalui sarana *leased line*, metode pengaksesan telekomunikasi, protokol konversi, proses translasi dokumen EDI, *software* untuk pengiriman *file.* Pada beberapa tahun terakhir ini, keamanan dan aplikasi dengan standar internasional menjadi suatu kebutuhan penting. Translasi dokumen EDI merupakan proses konversi dari bentuk dokumen tertentu menjadi dokumen yang siap diolah oleh komputer *server* EDI. Sejalan dengan perkembangan protokol komunikasi muncul banyak perdebatan tentang standar yang seharusnya digunakan, tetapi perdebatan ini tidak lagi menjadi suatu halangan besar bagi EDI *scheme.*

Standarisasi EDI telah terdaftar pada ISO (*Inetrnational Standards Organization*) dengan nama EDIFACT(*EDI for Administration, Commerce, and Transportation*). ISO memuat segala standar EDI dan mengkodekannya dalam suatu skala yang menyatakan keserasian dengan standar X.400 *messaging*.

2.2.1. Teknologi EDI

Untuk menciptakan suatu sistem fungsi EDI ada beberapa pilihan kategori komponen yang harus disertakan, antara lain:

a. *Hardware*. Sistem komputer diperlukan untuk membuat suatu *system* EDI. Ada 2 bagian terpenting, bagian *server* dan bagian *client*, kedua nya berupa PC dengan kemampuan bagian *server* diutamakan lebih mempunyai kinerja yang lebih daripada bagian *client*. Hal ini dikarenakan bagian *server* lebih kompleks dalam menangani proses komputasi daripada bagian *client*. *Fixed* Modem dan *wireless* modem dan peralatan komunikasi menjadi tulang punggung dalam melakukan pertukaran data.
b. EDI *server*. Komputer yang digunakan sebagai jantung dari jasa EDI, secara umum berbasis seperti kotak surat dengan spesifikasi tambahan untuk keamanan, validasi *user* , manajemen sistem, pengarsipan dan *backup*. *Server* ini dapat diakses oleh satu atau lebih *user* yang terdaftar untuk kepentingan layanan tertentu.
c. *Software*. Sistem EDI butuh adanya *protocol* konversi, perpindahan dokumen, pengiriman *file*. Perpindahan dokumen meliputi sistem penerimaan dan pengiriman dokumen. Perpindahan dokumen dapat terjadi pada PC terpisah yakni dengan adanya perpindahan dokumen dalam bentuk (fisik) *file* secara manual atau secara terintegrasi secara langsung (*online*).

2.2.2. Sejarah EDI

Munculnya EDI bermula pada penemuan-penemuan dari berabad-abad lalu. Untuk beberapa generasi, sinyal api dan asap, sistem *semaphore* digunakan sebagai prinsip dari komunikasi antar jarak yang berbeda. Dengan penemuan oleh Leyden Jar pada tahun 1746, elektrik statis dapat disimpan dan digunakan sebagai bahan eksperimen. Fenomena kelistrikan ini melahirkan kelahiran industri telekomunikasi. Berikutnya Benjamin Franklin menyelidiki cara listrik dapat berpindah melalui kabel. Tahun 1774 Georges Lesage dari Jenewa membuat telegraf menggunakan mesin elektrostatis, bekerja dengan kabel untuk tiap huruf abjad, 40 tahun kemudian Sir Francis Ronalds mendemonstrasikan telegraf dengan satu kabel listrik tunggal. Penemuan-penemuan baru terus bermunculan

untuk melengkapi dan memperbarui penemuan sebelumnya, hingga pada tahun 1875 Alexander Graham Bell menciptakan pesawat telepon dimana tetap menggunakan kabel namun tidak hanya mengirimkan sinyal listrik seperti pada telegraf tetapi sinyal tersebut diubah menjadi suara, sehingga memungkinkan penemuan ini menjadi sarana komunikasi nyata dua arah.

Penggunaan teknik pada pesawat telepon dan telegraf berkembang pesat, dengan prinsip *analog* sebagai teknologi standarnya karena kebutuhan telekomunikasi telah memungkinkan adanya penemuan untuk mengkonversi sinyal *analog* menjadi *digital*. Keberadaan sinyal *digital* diintegrasikan pada teknologi komputer dimana pemrosesan datanya menggunakan sinyal *digital*. Kemudian bermunculan perubahan-perubahan baru dengan adanya teknologi transmisi data *digital* memungkinkan pada akhir abad ke-20 orang mentransmisikan suara, data, dan gambar bergerak melalui kabel serat optik, satelit dan gelombang *microwave* dengan kecepatan transfer data rata-rata mencapai 2 *Mbit/sec* hingga 10 *Mbit/sec*.

2.2.3. Sistem Perdagangan

Sistem perdagangan yang telah dilakukan selama lebih dari ratusan tahun memiliki langkah kerja mendasar, seperti berikut ini:

- Langkah 1. Konsumen menentukan barang yang akan dibeli, menanyakan harga dan masalah pengiriman barang. Hal ini dapat dilakukan selain dengan bertatap muka secara langsung antara penjual dan pembeli, juga melalui percakapan telepon, surat, faksimili. Tenggang waktu dalam menangani permintaan dari konsumen dapat disebabkan oleh metode pengiriman data dan waktu yang diperlukan data untuk diproses dalam suatu sistem. Kesalahan mengartikan permintaan, kesalahpahaman sering terjadi pada tahap ini dikarenakan tenggang waktu dan biaya yang timbul.
- Langkah 2. Berdasar pada langkah ke-1, setelah permintaan dari konsumen telah diterima dan dimasukkan ke dalam sistem, diterjemahkan dan diproses dengan benar, maka komputer akan memproses dengan hasil cetakan *form* untuk segera dikirimkan ke rekan usahanya. Perlu dicatat bahwa seringkali langkah ke-1 dan ke-2 ini berdasar pada negosiasi dari konsumen dan penjual /

produsen untuk mendapatkan aturan yang saling menguntungkan. Transaksi dapat berhenti pada tahap ini atau dilanjutkan pada langkah berikutnya.

- Langkah 3. Setelah menerima pemberitahuan atau balasan dari penjual, konsumen segera membuat order pembelian untuk selanjutnya dikirimkan kepada penjual.
- Langkah 4-7. Setelah order pembelian diterima, penjual memeriksa kebenaran dari order pembelian tersebut, jika benar maka selanjutnya data order pembelian tersebut dimasukkan ke sistem order penjualan internal dari penjual. Artinya data dimasukkan dengan diketik menggunakan mesin ketik atau diinputkan ke dalam komputer dan terdapat tenggang waktu dalam pengerjaannya. Tentunya jika terdapat kesalahan antara persetujuan dan order maka langkah ke-1 hingga ke-4 diulangi lagi.

  Sistem internal kemudian memproses menjadi *form* dan dokumen yang dibutuhkan untuk identifikasi dan menentukan lokasi dari stok barang, dimana barang tersebut dapat diambil, dikemas dan dikirim kepada konsumen. Proses ini merupakan suatu rutin internal. Dalam proses ini mungkin terjadi beberapa item dari order kosumen mengalami kekurangan stok, sehingga langkah ke-4 hingga ke-6 perlu untuk diulang dan diselesaikan.

  Konsumen harus diberitahu perihal keterlambatan pengiriman dan kekurangannya, dan permintaan status dari order untuk dijawab melalui tulisan atau secara lisan. Pencatatan internal harus diperbarui sesuai dengan perubahan pengiriman, jumlah barang, isi order, dan semacamnya.
- Langkah 8. Surat pengiriman dicetak sebagai bagian dari inputan internal order dan harus disertakan pada saat pengiriman barang. Bilamana terjadi kesalahan atau perubahan dengan order konsumen misalnya kesalahan jumlah, spesifikasi atau tanggal maka *form* baru harus dibuat untuk merefleksikan perubahan tersebut. Langkah ini harus dilakukan hingga pengiriman selesai dilakukan sesuai dengan order semula.
- Langkah 9. Penjual menerbitkan surat tagihan, tagihan atas barang-barang yang telah dikirimkan.
- Langkah 10-11. Suatu pernyataan mungkin disertakan pada akhir bulan, misalnya pengingat jadwal pembayaran hingga pembayaran telah diterima.

Konsumen memberikan keterangan atas kecocokan barang yang dipesan dan catatan pengiriman atas bentuk fisik dari barang yang telah diterima kepada penjual.

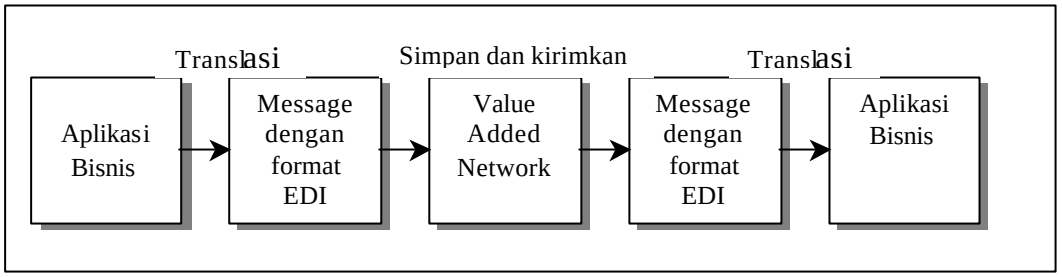
Tampak pada penjelasan langkah-langkah di atas, proses berjalan secara sekuensial, berulang-ulang, boros waktu, dan memungkinkan terjadinya kesalahan-kesalahan. Jika komputer pihak konsumen dapat berkomunikasi secara langsung dengan komputer pihak penjual, maka waktu dan biaya yang diperlukan dapat dihemat, serta beberapa kesalahan-kesalahan pokok dapat dikurangi.

2.2.4. Penerapan EDI

EDI merupakan kombinasi dari beberapa teknologi dan teknologi yang memfasilitasi perdagangan dengan sedikit penggunaan fisik surat menyurat (*paperless*) atau perdagangan elektronik. Dalam usaha untuk mempertukarkan dokumen sebagai kebutuhan dunia perdagangan sangat diperlukan sebuah aplikasi pada satu komputer dapat berkomunikasi dengan aplikasi pada komputer lainnya melalui jaringan elektronik. Sebuah jaringan yang diperlukan untuk terciptanya komunikasi antara dua aplikasi tersebut memerlukan adanya protokol untuk memahami adanya pertukaran dokumen yang sedang terjadi. Dengan sebuah personal komputer dan sebuah sarana komunikasi, memungkinkan untuk mengirimkan informasi secara langsung dari satu komputer ke komputer lainnya.

2.2.4.1. EDI Tradisional

EDI menerjemahkan dokumen-dokumen menjadi bahasa bisnis yang dipahami secara global dan mentransmisikannya antara rekanan-rekanan usaha menggunakan jalur telekomunikasi yang aman seperti terlihat pada gambar 2.3 berikut:



Gambar 2.3
EDI Tradisional

Beberapa standar EDI yang populer adalah *United Nations EDI for Administration, Commerce, and Trade* (*EDIFACT*). Di Amerika Serikat, standar yang paling banyak digunakan adalah ANSI X.12. Pengguna EDI tradisional menggunakan jalur telepon atau VAN, seperti pada IBM dan AT&T untuk mentransferkan data.

Pada tahun 1998 kurang dari 100.000 perusahaan di Amerika Serikat telah mengadopsi EDI dalam kegiatan perdagangan mereka. Kebanyakan dari perusahaan tersebut hanya memiliki rekan bisnis EDI dalam jumlah kecil dikarenakan biaya tinggi dalam pengoperasiannya. Sehingga tidak memiliki keuntungan dari EDI, faktor utama yang menyebabkan antara lain:

- Dibutuhkannya investasi awal dalam jumlah besar.
- Proses perubahan struktur bisnis sulit dilakukan untuk memenuhi kriteria yang dibutuhkan EDI.
- Waktu yang diperlukan untuk memulai penggunaan EDI cukup lama.
- Mahalnya biaya penggunaan VAN
- Biaya operasi EDI tinggi
- Banyaknya standar EDI
- Sistem terlalu kompleks untuk dijalankan

- Dibutuhkan adanya pengubah untuk menerjemahkan transaksi bisnis ke dalam standar EDI

Faktor-faktor tersebut menyebabkan EDI tradisional bergantung pada software formal translasi suatu transaksi , dan VAN tidak cocok untuk solusi jangka panjang bagi beberapa perusahaan karena faktor-faktor tersebut tidak memenuhi kriteria :

- Memudahkan makin banyak perusahaan menggunakan EDI
- Mencakup integrasi EDI secara penuh ke proses bisnis dari rekanan
- Menyederhanakan implementasi EDI
- Menambah kemampuan pertukaran informasi secara *online*

2.2.4.2. *Internet-Based* EDI

Untuk menetapkan jalur yang tepat bagi EDI, *internet* tampaknya alternatif paling memungkinkan untuk menerapkan perdagangan *business to bussines* (*B2B*) dengan kemudahan dijangkau oleh organisasi atau perusahaan , baik skala besar atau kecil. Ada beberapa alasan mengapa perusahaan menggunakan EDI dengan memanfaatkan sarana internet seperti di bawah berikut:

- *Internet* adalah jaringan dengan kemudahan pengaksesan oleh publik, dengan koneksi skala besar (tidak diperlukan arsitektur jaringan komputer khusus), mendukung pertumbuhan beberapa aplikasi bisnis secara cepat.
- Koneksi jaringan *internet* global menawarkan kemudahan untuk menjangkau rekanan bisnis dalam jumlah besar dari sekian banyak alternatif rekanan bisnis yang ada.
- Penggunaan *internet* memangkas biaya komunikasi hingga 50 persen lebih.
- Penggunaan *internet* untuk transaksi pertukaran EDI sangat konsisten dengan perkembangan bisnis pengiriman, jasa elektonik, dan jasa berbasis *web* pada umumnya.
- *Internet-based* EDI dapat melengkapi atau mengganti aplikasi EDI yang telah ada.
- Perlengkapan *internet* seperti *browser* dan *search engine* mudah dimengerti oleh *user* dan telah banyak *user* saat ini mengerti cara mengoperasikannya.

Kehadiran *internet* mendukung penerapan EDI dalam banyak bentuk dan cara antara lain:

- *Electronic mail* (*e-mail*) dapat digunakan sebagai transportasi *message* dari sistem EDI. Di tempat tertuju, *Internet Engineeering Task Force* (*IETF*) menggunakan standar untuk memperoleh *message* dengan menggunakan *Secure Internet Mail Extension* (S/MIME)
- Sebuah perusahaan dapat membuat jaringan ekstra untuk memungkinkan rekanan bisnis untuk memperoleh informasi melalui *web form* dimana *field* yang ada mewakili *field* yang ada pada EDI *message* dan dokumen EDI.
- Perusahaan dapat menspesialisasikan jasa dari *web-based* EDI dalam banyak cara sesuai dengan kondisi dari rekanan yang tergabung di dalamnya.

Perusahaan dengan tradisional EDI menanggapi positif hadirnya *internet-based* EDI. Sebuah badan penelitian *Forester Research, Inc*. mensurvei 1000 perusahaan menunjukkan bahwa mendekati separuhnya merencanakan pengunaan EDI melalui internet pada akhir abad ke-20. Beberapa pengguna EDI berubah pada penggunaan *internet-based* EDI untuk mengeliminasi biaya yang harus disediakan untuk VAN. Pada EDI tradisional, mereka juga harus membayar untuk biaya transportasi, translasi, dan jalur dari EDI *message* hingga sistem secara keseluruhan selesai. Sering dijumpai beberapa perusahaan mengkombinasikan EDI tradisional dengan internet dengan cara order ditransmisikan ke VAN atau penyedia jasa untuk mentranslasikan data menjadi format EDI dan dikirimkan ke komputer utama dengan fasilitas *internet*. *Internet* menyederhanakan mekanisme transportasi *message* dibandingkan dengan penyewaan saluran komunikasi khusus. Kombinasi penggunaan *web*, XML, dan Java membuat EDI lebih interaktif sehingga memudahkan *user* untuk terlibat di dalamnya.