マルチメディア通信と分散処理　78－11
（１９９６．９．２７）

EDIFACT対応EDIトランスレータの実装と評価

杉山敬三　　小花貞夫

ＫＤＤ研究所

概要

本稿では，汎用データフォーマット変換方式に基づくEDIFACT対応トランスレータの実装概要と評価結果について述べる．ここでは，先に実装したCII対応トランスレータと対比したEDIFACT対応トランスレータ実装の課題として，データセグメントのネスト/反復処理，データエレメントの順序確認・省略処理，並びにシンタックスルールにおける要素の対応付け処理があることを示し，その解決方法を述べる．また，その解決方法に従ったEDIFACT対応トランスレータをWindowsパソコン上に実装し，先に開発したCII対応トランスレータとの比較評価を行う．EDIFACTはCIIと比べてデータセグメントや複合データエレメント等シンタックスルールの階層が深いため辞書ファイルへのアクセスが多く，修飾データエレメントも存在するため，プログラムサイズや変換時間が大きくなる．

Implementation and Evaluation of EDI(Electronic Data Interchange) Translator for EDIFACT Syntax Rule

Keizo SUGIYAMA　　Sadao OBANA

KDD R&D Laboratories

Abstract

This paper describes an implementation and an evaluation of an EDI translator for EDIFACT syntax rule. Its design is based on a translation method which accommodates multi data formats and usage. We first identify the subjects for implementation and discuss the solutions for these subjects by comparing with the CII translator we have already developed. There are three subjects: appearance order investigation and omission processing for data elements, nest and repeat processing for data segments and mapping processing between elements included in syntax rules. Next we describe the implementation of the EDIFACT translator on Windows PC based on the above solutions and evaluate through the comparison with the CII translator. The program size and the translation time of the EDIFACT translator is greater than those of the CII translator because of the frequent access to the dictionary file and the existence of qualifier data element.

1.はじめに

インターネット等のデータ通信ネットワークの発達に伴い，個人と企業あるいは企業間で商取引きに関する情報を電子的に交換する EC(エレクトロニックコマース)や EDI(電子データ交換)[1]が注目を集めている．

EDI の標準には国際・国内・業界標準等の多数の標準が存在し，しかも各々扱うデータフォーマットが異なるため，データフォーマットの変換を行うトランスレータが必要となる．また，トランスレータは，企業内で扱うローカルなデータフォーマットと企業間で扱う標準のデータフォーマットの変換，標準のデータフォーマット間の変換並びに端末におけるプログラム内部のデータ構造と標準のデータフォーマットとの変換という３つの利用形態で必要となる．

そこで筆者等は，扱うデータフォーマットや利用形態が異なるトランスレータのプログラムを効率的に開発するための汎用データフォーマット変換方式を提案し[2]，これに基づき国内の標準である CII(産業情報化推進センター)[3]のデータフォーマットとローカルデータフォーマットである固定長データフォーマットを変換するトランスレータ(CII トランスレータ)を既に開発している[4]．

今回，CALS(Commerce At Light Speed)[5]や公衆網に関する網管理情報を顧客から電子的にアクセス可能とする CNM(Customer Network Management)[6]等でその使用を想定している EDI の国際標準である EDIFACT (Electronic Data Interchange for Administration, Commerce and Transport)[7]のデータフォーマットと固定長のデータフォーマットを変換する EDI トランスレータ(EDIFACT トランスレータ)を上記の汎用データフォーマット変換方式に基づき開発した[8]．

本稿では，この EDIFACT トランスレータの実装と評価について述べる．まず CII 標準と EDIFACT 標準の比較を通して EDIFACT トランスレータ実装の課題と解決方法について述べる．次に，その方法に従った EDIFACT トランスレータの実装概要について述べ，最後に評価と考察を行う．

2. EDI 標準の概要及び CII 標準と EDIFACT 標準の差異

2.1.EDI 標準の概要

EDI の標準は一般に，①標準メッセージ，②シンタックスルール，③データエレメントディレクトリから構成される[1]．①は，注文書等の特定の帳票に含めることのできるデータ項目のリストである．②は，標準メッセージから必要なデータ項目を抽出してシステム間で交換するデータフォーマットを組み立てるための文法規則である．③は，全ての標準メッセージに含まれるデータ項目の一覧表である．

EDI のデータフォーマットは図1のような階層構造を示し，各階層にはヘッダやトレーラを付与する．EDIFACT の場合，UNH や UNT 等のヘッダやトレーラはサービスセグメントと呼ばれる．また，EDIFACT における使用可能な文字セット(レベル A/B)や，データエレメントのネスト/反復の表現方法(明示的/暗黙的表現)等，シンタックスルールには一般にオプションが存在する．このシンタックスルールに従って，種々の帳票に対応する標準メッセージを定義する．EDIFACT では，ORDERS(発注)や INVOIC(インボイス)等の 100 以上の標準メッセージ(UNSM)が存在する．

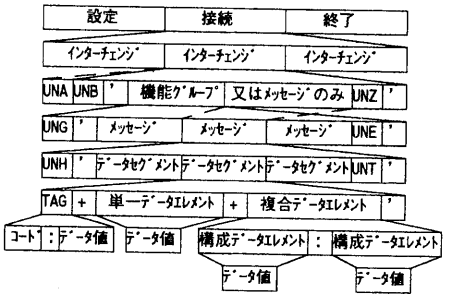

図1 EDIFACT データフォーマットの階層構造

2.2.CII 標準と EDIFACT 標準の差異

表1に CII 標準[3](以下 CII と呼ぶ)と EDIFACT 標準[7](以下 EDIFACT と呼ぶ)のデータフォーマットの主な差異を挙げる．

表1 CII 標準と EDIFACT 標準の主な差異

	CII	EDIFACT
機能グループ	必ず存在する	存在しなくてもよい
メッセージの構成	データエレメントの組み合わせ	データセグメントの組み合わせ
エレメントの構成	単一データエレメント(TFD と呼ぶ)のみ	単一/複合データエレメントが存在する
要素の区切り	TFD の長さフィールドから検出する	セパレタ(+,',:)の文字を使用する
要素の順序	エレメントの順序は基本的に任意	セグメントやエレメントの順序は固定
要素のネスト/反復	エレメント単位：明示 エレメントのセット単位：明示	エレメント単位：暗黙 セグメント単位：明示/暗黙
エレメントの省略	必要なエレメントのみで構成する	省略した部分にセパレタを挿入する
セグメント/エレメントの修飾	エレメント自体が特定の用途を表す(ex. 納期)	修飾子により用途が区分される場合あり(ex. 納入[修飾子]+日時)

注:エレメント及びセグメントは各々データエレメント及びデータセグメントを指し，要素は両方を含む

2.2.1.メッセージ

CII のデータフォーマットには必ず同種のメッセージの集合である機能グループを含むが，EDIFACTでは機能グループを含まず直接メッセージから構成できる．CII のメッセージはデータエレメント(TFD: Transfer Form Data)から構成されるが，EDIFACT では NAD (Name And Address)等のデータセグメントから構成され，関連する複数のデータセグメントはグループ化(セグメントグループ)できる．

また，ネスト/反復に関しては，CII では表形式のデータを扱うマルチ明細があり，対象となる TFD の範囲を注釈等により指定する．マルチ明細では，最初と最後にマルチ明細ヘッダ・トレーラを，行の区切りに改行文字を付与する．EDIFACT では"NAD:2"が NAD データセグメントの 2 回反復を示すように，各データセグメントがネストや反復可能であり，その回数を明示的に指定できる．EDIFACT のメッセージ中のデータセグメントの出現順序や最大出現回数並びにセグメントグループは，メッセージダイアグラムで規定する．

2.2.2.データエレメント

CII の TFD は「タグ+長さ+データ値」の 3 つのフィールドで構成され，マルチ明細を除けばその出現順序は任意である．一方，EDIFACT ではデータセグメント中の出現順序でデータエレメントを特定するためタグは存在しない．データ値は'+'や':'等のセパレタで区切られ，データ値が存在しない場合セパレタのみとなる．

また，CII のデータエレメントには TFD しか存在しないが，EDIFACT には CITY NAME 等単独で意味をなす単一データエレメントと，File name と Item description を組合わせて FILE IDENTIFICATION を表すように，複数の構成データエレメントを組合わせて一つの意味をなす複合データエレメントが存在する．

さらに，CII では個々の TFD が納期等の特定の用途を表すが，EDIFACT では修飾データエレメントにより用途を示す．例えば，DATE/TIME/PERIOD 複合データエレメント中の修飾データエレメントである Date/time/period qualifier 構成データエレメントの値を 2(Delivery date/time, requested)とすることで納期を表す．

3. 汎用データフォーマット変換方式と CII トランスレータの実装概要

3.1.汎用データフォーマット変換方式

汎用データフォーマット変換方式[2]では，データフォーマット変換においてメッセージを特定のシンタックスルールに従って符号化したビット列のデータである"シンタックス"とメッセージの持つ情報の内容である"セマンティクス"の 2 つの側面を分離して扱い，セマンティクスを表すメッセージのプログラム内部表現として，EDI の各種標準データフォーマットに共通に適用可能である汎用的なデータ構造(C 言語の構造体)を定義した．

この構造体は，図 1に示した EDIFACT のデータフォーマットの階層構造における各要素の構造体を階層毎にリスト構造として保持しており，これと EDI の特定のデータフォーマット間で，対応するシンタックスルールに基づき符号化/復号を行う(エンコーダ/デコーダ)．また，その構造体上で，シンタックスルールにおける要素間での対応付けと，型や長さ等のデータエレメントにおける属性の変換からなるセマンティクスの対応付けを行う(ゲートウェイ)．これらエンコーダ/デコーダとゲートウェイの組み合わせを変えることで，3 つの利用形態やデータフォーマットが異なるトランスレータを効率的に実現できる．

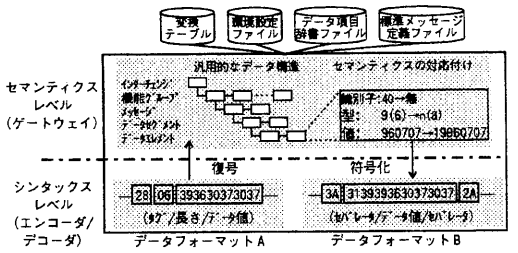

図2 汎用データフォーマット変換方式

本方式では，標準メッセージ中のデータエレメントの一覧を示す標準メッセージ定義ファイル，データエレメントディレクトリに相当するデータ項目辞書ファイル，動作モード等を決定する環境設定ファイル並びに 2 つのデータフォーマット間でのデータエレメントの対応関係を示す変換テーブルを使用する．

3.2.CII トランスレータの実装概要[4]

上記の汎用データフォーマット変換方式に基づいて，CII トランスレータを先に実装した．ここでは，Windows パソコン上で CII のデータフォーマットと固定長のデータフォーマットのエンコーダ/デコーダを DLL(動的リンクライブラリ)として実現し，CII-固定長ゲートウェイのプログラムから呼び出すようにした．3.1.で述べた変換テーブル等のファイルは，Windows の INI フ

ファイルとして実現した．この変換テーブルには，CII の TFD のタグの値と，それに対応する固定長のデータ項目の順序番号を記述した．

4.EDIFACT トランスレータ実装の課題と解決方法

2.2.で述べた CII と EDIFACT の差異に基づいて，EDIFACT トランスレータ実装の主な課題と解決方法を検討する．

4.1.エンコーダ/デコーダ実装関連

シンタックスルールの階層構造に従って処理を行う点は CII エンコーダ/デコーダと同様であるが，以下の点で異なる．

4.1.1.データセグメントのネスト/反復処理

●課題

データセグメントのネストや反復の回数を精査し，明示的に表現できなければならない．

●解決方法

標準メッセージ定義ファイルの内容を，図 3に示すメッセージダイアグラムに対応する情報で構成する．トランスレータの起動時にこれを読み込んで，メッセージダイアグラムをリスト構造としてメモリ上に展開する．

符号化/復号時には，一つのデータセグメントを処理する毎にネストや反復の回数を計算してデータセグメントに対応する構造体のメンバに設定する．併せてメッセージダイアグラムのポインタを左から右へ，また垂直構造が存在する場合上から下へ進めることで，出現順序や最大繰返し数との比較及び必須データセグメントの存在の確認等を行う．ネスト/反復が明示的に指定された場合，符号化時には構造体に設定された値を書き出す処理を設ける．

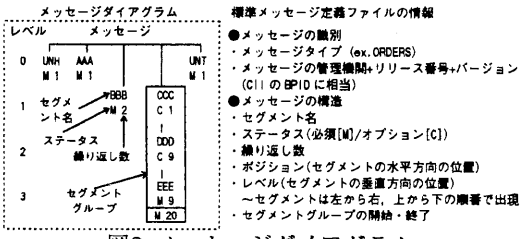

図3 メッセージダイアグラム

4.1.2.データエレメントの順序確認・省略処理

●課題

データエレメントの区切りをセパレタにより検出して順序を精査したり，省略部分にセパレタを挿入しなければならない．

●解決方法

符号化時は，データエレメントに対応する構造体の長さを示すメンバを参照し，その長さ分データフォーマットファイルに書き出してセパレタを付与する．長さが 0 の場合は省略と判断しセパレタを挿入する．復号時はセパレタを検出するまでデータフォーマットファイルから読み出し，構造体のメンバに値と長さを設定する．省略を検出した場合，省略されたデータエレメントに対応する構造体の長さを0に設定することで省略を表す．

また，データ項目辞書ファイルの内容を，データエレメントの種類や順序等の表 2に示す情報から構成する．トランスレータの起動時にこれを読み込み，符号化/復号時には表 2の項目との間で属性や順序の比較処理を行い，異なる場合はエラーを返す．

表2 データ項目辞書ファイルの内容

データ項目	情報	値の例
データセグメント	セグメント名*	ADR
	単一/複合データエレメント情報+	
	-識別番号	C090
	-ステータス(必須[M]/オプション[C])	C
複合データエレメント	複合データエレメント識別番号*	C090
	構成データエレメント情報+	
	-識別番号	3477
	-ステータス(必須[M]/オプション[C])	M
単一/構成データエレメント	単一/構成データエレメント識別番号*	3477
	属性(型[a=英字, n=数字]/長さ[..=最大])	an..3
	コード値の有無	coded
コード	単一/構成データエレメント識別番号*	3477
	コード値+	4

注：*はキー項目を，+は複数回出現することを表す

4.2.ゲートウェイ実装関連

4.2.1.要素間での対応付け処理

●課題

データセグメントや複合データエレメントの階層を含めたり，修飾データエレメントにより対応付けを変更できる必要がある．

●解決方法

EDIFACT側をキーとして変換テーブルを記述し，データセグメントや複合データエレメントの情報も明示的に記述する．これらの情報と単一/構成データエレメントの識別番号を組み合せて固定長のデータ項目と対応付ける．また，修飾データエレメントが存在する場合，修飾データエレメントの値毎に修飾されるデータセグメント/データエレメントを記述する．

ORDERS メッセージに対する変換テーブルの記述例を図 4に示す．ここでは，各行に一つのデータセグメント，データエレメント，セグメントグループの情報を記述している．例えば No.0014 から No.0019 迄の行は"GR1"というセグメントグループであり，反復が最大 2 回でここは 1 回目を示している．"C506"(No.0016)の複合データエレメントには"1153"(No.0017) と "1154" (No.0018)という構成データエレメントが含ま

れ，"1153"は修飾データエレメントで"OP(Original Purchase Order)"という値で修飾しており，"1154"は固定長の"0004"番目のデータ項目と対応している．

```
[ORDERS]                ;メッセージタイプ
LINECNT=286             ;行数
...
; EDIFACTの符号化例「RFF:1+OP:960627094814'」
No.0014=GR1,001/002,|   ;セグメントグループ開始
No.0015=RFF             ;セグメントタグ
No.0016=C506            ;複合データエレメント
No.0017=,1153,"OP"      ;修飾データエレメント
No.0018=,1154,0004      ;構成データエレメント
No.0019=GR1,001/002,|   ;セグメントグループ終了
...
```

図4 変換テーブルの記述例

5.EDIFACT トランスレータの実装

5.1.基本方針

5.1.1.モジュール構成

汎用データフォーマット変換方式に基づき，EDIFACT エンコーダ/デコーダ及び EDIFACT-固定長ゲートウェイは新規に開発し，固定長エンコーダ/デコーダは開発済み[4]のものを流用する．

5.1.2.サポートする属性

EDIFACT は a 属性(英字)と n 属性(数字)を，固定長は X 属性(8bit 文字列)，9 属性(固定小数点正数)，N 属性(浮動小数点数)，Y 属性(日付)をサポートする．

5.1.3.プログラミング環境

Windows95/NT 上で動作する 32bit アプリケーションとして，C 言語(Visual C++)によりプログラミングを行う．

5.2.EDIFACT エンコーダ/デコーダ

4.1.1.のメッセージダイアグラムと構造体の比較処理では，再帰呼び出しによりメッセージダイアグラム中のデータセグメントのポインタを進めるようにした．また，4.1.2.のセパレタは文字セット(レベル A/B)により異なる上，セパレタを変更する UNA サービスセグメントも存在するため，セパレタは変数として定義し，UNA サービスセグメントの未使用時にはレベルに応じたセパレタをプログラム中で設定するようにした．

エラー処理では，標準メッセージの一つであり受信したメッセージに対する受諾/拒否を表す CONTRL メッセージに含まれるエラーコード(例:セグメント反復数が不正)に加え，メモリ割当てエラー等の本実装固有のエラーコードを戻り値として返すようにした．

5.3.EDIFACT-固定長ゲートウェイ

EDIFACT-固定長ゲートウェイでは，セマンティクスの対応付けのために，まず変換テーブルを読み込んで4.2.1.で述べた要素間での対応付け処理を行った後，属性の変換を行う．属性の変換では，EDIFACT の a/n 属性と固定長の X/9/N/Y 属性との間で任意の組み合わせによる変換を可能とした．EDIFACT から固定長に変換する場合，X 属性は左詰め，9 及び N 属性は右詰めにする等，セマンティクスを保持しながら長さや型等の変換を行う．また，固定長から EDIFACT への変換時に図5に示すような EDIFACT のデータフォーマットに修飾データエレメントの値を補う処理を持たせた．

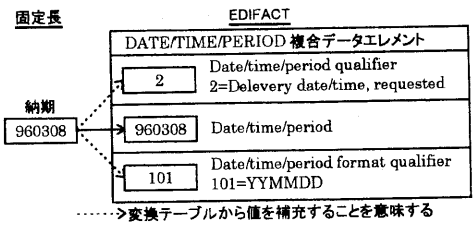

図5 修飾データエレメント値の補充処理

5.4.各種ファイル

標準メッセージ定義ファイル(図 3)やデータ項目辞書ファイル(表 2)は，電子的に入手可能な EDIFACT の標準メッセージ(UNSM)やデータセグメントディレクトリ(EDSD, UNCL 等)の定義ファイルを直接読み込むことで実現した．また，文字セットのレベル等のサービスセグメントの情報や，ネスト/反復の明示的/暗黙的表現等のシンタックスルールのオプション情報を含む環境設定ファイルや4.2.1.で述べた変換テーブルは，Windows の INI ファイルとして実現した．これらのファイルは，検索処理の高速化のため，トランスレータ起動時にメモリマップトファイルとして読み込む．

6.評価と考察

6.1.トランスレータ実装の課題について

6.1.1.データセグメントのネスト/反復処理

CII にもマルチ明細という概念があるが，基本的に表形式であり，TFD の出現位置は計算式により導出できた．しかしながら，EDIFACT のデータセグメントはメッセージダイアグラムにより木構造として定義されるため，対象となるデータセグメントの出現位置は計算式では導出できない．そのため，メッセージダイアグラムとの比較処理を再帰呼び出しにより実現する等の工夫が必要となる．

6.1.2.データエレメントの順序確認・省略処理

CII トランスレータにおいても各データエレメントの属性と辞書との比較処理は必要であるが，

タグにより TFD が一意に決まるため辞書の検索は比較的容易であった．しかしながら，EDIFACT はデータセグメントや複合データエレメントなど階層構造が深い上，データエレメントの決定のために常に辞書ファイルを検索しながら順序確認や省略の処理を行う必要がある．このため，個々のデータエレメントあたりの処理の負荷が大きくなっている．

6.1.3.要素間での対応付け処理

EDIFACTでは修飾データエレメントの値毎に対応付けを記述する必要があるため，CII と比べ同じ規模のメッセージでも変換テーブルのサイズは大きくなった．これを回避するには，変換テーブルに変数や条件式を設け，変数値によって対応付けを変える等の記述を可能とすることが考えられる．

また，今回は変換テーブルをテキストエディタで作成するようにしたが，エンドユーザが個々のデータセグメントのネストや反復を把握しながら含まれるデータエレメントの対応関係を記述するのは容易ではない．このため，メッセージダイアグラムの構造や関連するデータセグメントの内容を適宜表示しながら対応関係を入力できるよう簡便化を図る必要がある．

6.2.プログラムサイズについて

開発した C 言語のソースプログラムのサイズを CII トランスレータと実際に比較したところ，ゲートウェイは約 70K バイトで大差ないが，エンコーダ/デコーダが約 230K バイトで 40K バイト程度大きくなった．これは上記で述べたように，主に階層構造の深さとデータセグメントやデータエレメントの順序確認処理によるものであった．また，今回流用した固定長エンコーダ/デコーダは約 40K バイトであった．

6.3.変換速度について

CPU が i486DX4(100MHz)のパソコン上で，約 1K バイトの固定長ファイル(1 メッセージ，50 データエレメント)を復号してゲートウェイでセマンティクスの対応付けを行い，約 1K バイトの EDIFACT 及び CII ファイルへ符号化する迄の時間を測定した．この値は，各々約 1.5 秒と約 0.6 秒であった．

この違いは，6.1.で述べたように EDIFACT ではエンコーダ/デコーダにおいて辞書ファイルの検索処理が必要になったり，ゲートウェイにおいて修飾データエレメント値により対応付けが異なることによると考えられる．今回の実装では辞書ファイルをメモリマップトファイルとして扱ったため，上記の条件では同一メッセージの 2 回目以降の変換が 0.5 秒(2.0→1.5)ほど短縮されることが確認できた．これにより必要なメモリは大きくなるが，現在規定されている全ての標準メッセージやデータセグメントを読み込んだとしても 4MB 程度であり，近年のハードウェア環境においてはそれほど問題にならない．

7.おわりに

本稿では，先に提案した汎用データフォーマット変換方式に基づいた EDIFACT トランスレータの実装概要と評価結果について述べた．ここでは，先に実装した CII トランスレータと対比した EDIFACT トランスレータ実装の課題として，データセグメントのネスト/反復処理，データエレメントの順序確認・省略処理並びに要素間での対応付け処理を挙げ，その解決方法を示した．データセグメントのネスト/反復処理では，ネスト/反復の回数をデータセグメントの構造体に設定し，メッセージダイアグラムとの比較を行う．データエレメントの順序確認・省略処理では，セパレタにより検出したデータエレメントに対し，データ項目辞書ファイルとの比較を行う．要素間での対応付け処理では，データセグメント等の情報を変換テーブルに含め，修飾データエレメントの値毎に対応付けを記述する．また，これらの解決方法に従った EDIFACT トランスレータを Windows パソコン上に実装し，CII トランスレータとの比較評価を行った．EDIFACT は CII と比べてデータセグメントや複合データエレメント等シンタックスルールの階層が深いため辞書ファイルへのアクセスが多く，修飾データエレメントも存在するため，プログラムサイズや変換時間が大きくなる．最後に日頃ご指導頂く KDD 研究所村上所長に感謝します．

参考文献

[1]:通商産業省編「最新EDI事情」,工業調査会
[2]杉山,小花,鈴木「EDI用汎用トランスレータの実装」情処学会DPSワークショップ, 1994年
[3]:(財)日本情報処理開発協会 産業情報化推進センター,「CIIシンタックスルール1.11及び1.51」, 1994年
[4]:杉山,小花,鈴木「Windows対応パソコンEDIトランスレータの実装」,第50回情処全大1T-1
[5]:CALS推進協議会編「日本版CALS」オーム社
[6]:ITU-T Draft Rec. X.163, "Definition of Management Information for Customer Network Management Service for Public Data Networks to be used with the CNMe Interface", 1994
[7]:ISO9735,"Electronic Data Interchange for Administration, Commerce and Transport(EDIFACT) - Application Level Syntax Rules", 1988
[8]:杉山,小花「EDIFACT対応EDIトランスレータの実装」,第52回情処全大7Bb-5