情 報 シ ス テ ム 53−10
（１９９５．３．２２）

オブジェクト指向分析における共同作業支援の検討

松澤由香里　山城明宏

(株) 東芝　システム・ソフトウェア生産技術研究所

近年オブジェクト指向分析／設計手法が数多く提唱されているが，実システムにおいて適用するにはまだまだ多くの問題点が存在する．特に，共同開発という着眼点からはどの方法論においても論じられていない．本稿では，オブジェクト指向分析をGUIアプリケーションの共同開発において適用した結果，特に作成枚数が大量になり，共同作業上ネックとなった仕様書である事象トレース図にターゲットを当て，具体的な問題点と，これを解決するために考案した「シナリオ分割図」について紹介し，評価を行う．この「シナリオ分割図」は事象トレース図の部品化の枠組みを与えるものであり，これを利用することで事象トレース図の記述量を減らし，システムの振る舞いの大局的な把握を可能にする．

Support of the Collaborative Development for Object-Oriented Analysis

Yukari Matsuzawa, Akihiro Yamashiro

Systems & Software Engineering Lab., Research & Development Center,
Toshiba Corporation

This paper proposes the Distributed Scenario Diagram (DSD) that facilitates the reuse of Event Trace Diagrams (ETDs). By using DSD, analysts may describe less amount of ETDs and can concentrate on the main issues on system's dynamic behavior.

Although a number of Object-Oriented Analysis (OOA) methods have been proposed, they are still insufficient for practical software development. Particularly, no method is argued from a pint of view of the collaborative development. When we applied one of OOA methods to the collaborative development of the Graphic User Interface application, writing a large quantity of ETDs are laborious and confusing. We explain how we solve these problems using DSD.

1 はじめに

ソフトウェアが大規模化するに従って，いかに効率よい共同開発を行うかが重要になる．しかし，既存の手法では個々の機能の追加・変更の影響がソフトウェア構造のあらゆる箇所に波及し，担当者間に矛盾を生じさせ，その結果保守・拡張が容易でなくなる傾向にあった．このため共同開発を効率よく行うためには，担当者間の独立性の高いソフトウェア構造を構築する必要があった．この要求に応える手法がオブジェクト指向手法である．オブジェクト指向手法の特徴を一言でいうと，従来のソフトウェア構造が"手続き"と"データ"という2要素の集まりから構成されていたのに対し，オブジェクト指向ではソフトウェアの構造を手続きとデータを一体化した"オブジェクト"という1要素の集まりにより構成するということである．そして，オブジェクト内のデータにアクセスできるのはそのオブジェクト内の手続きだけ（情報隠蔽）という独立性がある．このオブジェクトの独立性は共同開発における分業を容易にすると考えられる．このため我々は，オブジェクトの独立性を十分生かすことができるソフトウェア構造を構築するために，分析／設計／実装を通した共同開発におけるオブジェクト指向開発の実用化を目指している．オブジェクト指向手法は現在数多く提唱されているが，特に分析／設計フェーズにはポピュラーでわかり易いOMT法[1]を，また実装にはC++を適用している．

このOMT法は，図1に示すように以下の3つのモデルを基本としている．

・オブジェクトモデル（オブジェクト間の静的関係）
・動的モデル（オブジェクト間の相互作用）
・機能モデル（オブジェクトの機能とデータ変換）

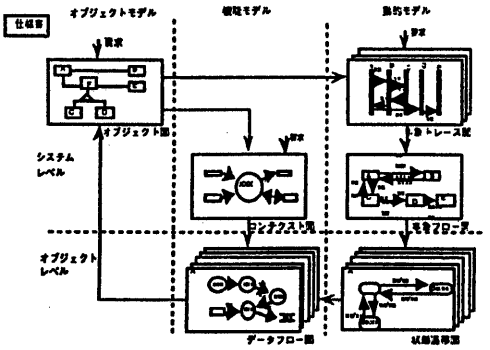

図1 OMT法による分析のモデルと仕様書一覧

OMT法では図1の仕様書の作成経路に表されるように，ソフトウェアの構造を示すオブジェクト図を中心に，オブジェクト毎の振る舞いを記述する状態遷移図と，機能とデータ変換を記述するデータフロー図が最終的に作成される．このように，オブジェクト単位の仕様書が作成されるわけだが，分業ではオブジェクト毎の仕様がいかに独立性の高いものになっているかが鍵となる．従って，オブジェクト毎の仕様化をする以前の，オブジェクト間の関係からオブジェクトがどのような手続きを持つのか，すなわちその役割を検討するための事象トレース図が重要になる．しかし，分析フェーズはオブジェクトの役割を決定するための過程なので，共同開発における事象トレース図作成時にはグループ内のコンセンサスがうまくはかれず，網羅性，一貫性，記述の重複といった側面から問題が発生する．そこで，我々はこれらの問題を解決する手段として，事象トレース図を部品化した「シナリオ分割図」を提案する．このシナリオ分割図により事象トレース図作成を効率良く行えば，オブジェクトの独立性を高くするという過程を共同開発でうまく進めることができ，事象トレース図作成以降の状態遷移図，データフロー図といった仕様書や，後の設計／実装フェーズにおいてもこのオブジェクトを単位として分業化がうまく進むことになる，と我々は考えた．

本稿では，OMT法の事象トレース図について簡単に説明し，共同開発における事象トレース図作成時の問題点を具体的に述べたあと，これを解決するために提案したシナリオ分割図の説明とその評価を行う．

2 OMT法の事象トレース図とその問題点

2.1 事象トレース図

まず，OMT手法における事象トレース図とは何かを説明する．オブジェクト指向手法では"オブジェクト"の持っている手続き（関数）をオブジェクト同士が呼び合うことによってシステムの振る舞いを実現していく．例えば，「顧客から要求されたシステムを作成する」という振る舞いを考えてみる．この場合オブジェクト指向では，図2に表すようにまず初めに"顧客オブジェクト"が"営業マンオブジェクト"に対して「システムを受注して下さい」という"手続き"の依頼（呼び出し）を行なう．すると，"営業マンオブジェクト"は"開発者オブジェクト"に対して「システムを作成して下さい」という"手続き"の依頼を行なう．この結果，実際にシステムを作成することができる．

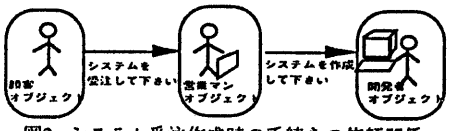

図2 システム受注作成時の手続きの依頼関係

このように，オブジェクト間の一連の手続きの依頼によってシステムの振る舞いの一例が表現できるが，OMT法ではこれを分析時に図3のような事象トレース図を用いて記述する．縦線はオブジェクトを表し，矢印は手続きの依頼を示す．

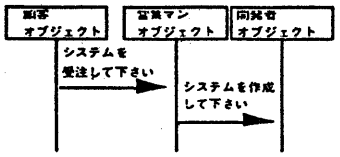

図3　システム受注作成時の事象トレース図

オブジェクト間の手続きの依頼は上から下へと時系列に示される．このような事象トレース図をシステムの主要な振る舞い（機能）の各々に対して記述する．また，同様の機能を実現する場合でも，手続きの依頼の系列（処理系列）が異なる場合は別の事象トレース図として記述する．

この事象トレース図を作成することで，オブジェクトの役割を明確にすることができる．例えば図3の事象トレース図を作成することで，"営業マンオブジェクト"は「顧客からの受注を受ける」という手続きを持つことが明らかになるし，"開発者オブジェクト"は「システムを作成する」という手続きを持つということが明らかになるからである．このようにオブジェクトの役割分担を明らかにし，その独立性を高めることによって，結果的に頑強なソフトウェアを構造を構築できることになる．

2.2　問題点

例えば，GUIアプリにおいて「コピーする」という機能について考えてみる．この場合一般的に以下の3つの手順より機能を実現することができる．

手順1 コピーする対象を選択する．
手順2 「コピー」というメニューを選択する．
手順3 コピーを実行する．

この3つの手順のなかで，オブジェクト間の手続きの依頼が図4の事象トレース図に表されるように行われている．

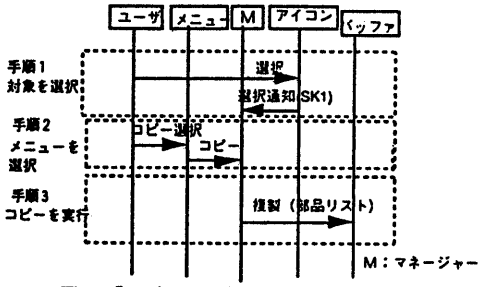

図4　「コピーする」の事象トレース図

しかし，ここで考えなければならないのは，GUIアプリの場合，手順1の「対象を選択」と手順2の「メニューを選択」が1通りではないということである．今，「対象選択法」と「メニュー選択法」に表1に表されるように，それぞれ2方法ずつあるとする．

手順	対象を選択	メニューを選択
方法	・マウスで選択 ・キーボード入力で選択	・メニューバーから選択 ・ショートカットで選択

表1　対象選択法とメニュー選択法の種類

この場合，それぞれの方法によってオブジェクト間の手続きの呼び出しを表す事象トレース図が異なる．従って，手順1～手順2までに，組み合わせとして対象選択法（2方法）×メニュー選択法（2方法）＝4通りが考えられるため，図5のように4通りの事象トレース図を記述しなければならない．

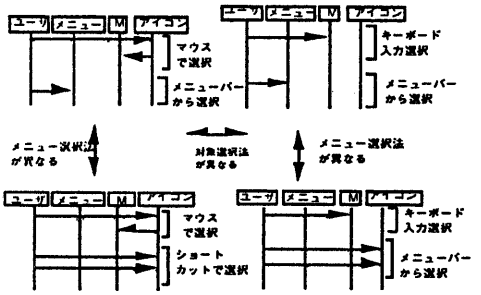

図5　対象選択法とメニュー選択法の組み合わせによる事象トレース図

この時以下のような問題が発生する．

問題1 シナリオ網羅の困難性
- 手順1と手順2にそれぞれ2通りの方法があるということが，大局的に把握できない．

問題2 グループ内での認識の不一致
- ある担当者は冗長性をさけ手順1と手順2を除いて手順3のみ記述し，別の担当者は手順1から手順3までフルに記述するといったように担当者間での記述範囲が不明確になる．また同様の手順1を記述しているにもかかわらず，担当者間で事象トレース図の記述内容が異なる場合がある．

問題3 記述の冗長性
- 手順3が重複するにも関わらず，手順1と手順2が異なるため，4通りの事象トレース図を記述しなければならない．

GUIアプリの場合，対象選択法や，メニュー選択法に多数のUIを与えている場合が多いので上記のような問題点は顕著に現れる．この結果記述の冗長性から仕様書枚数が多くなり大局的な把握が困難となるため，グループ内でのコンセンサスがはかれずかえって作業工数がかかることになる．

このため，GUIアプリにおけるUIの多様性に対応するため，事象トレース図を部品化し，この部品を組み合わせることで処理系列を表現した「シナリオ分割図」を考案した．これについて次に述べる．

3 シナリオ分割図

事象トレース図を記述する際の手順について，オリジナルのOMT法の手順を図6に，また我々のアプローチによる手順を図7に示す．オリジナルのOMT法ではシステムの主要な振る舞い（機能）をいくつかのシナリオとして言葉で洗いだし，このシナリオ毎に1枚の事象トレース図を作成する．これに対し，今回我々のとったアプローチでは，シナリオを抽出した後，このシナリオを具体的な図式表現を用いた複数の「モジュール項目」に細部化し，さらにモジュール項目に対応する具体的な「サブシナリオ項目」を抽出しこのサブシナリオ項目を単位として事象トレース図を記述する．すなわちシナリオから直接事象トレース図を記述するのではなく，図7の手順2，3のようにシナリオを図式表現により具体的に表現し細分化してから事象トレース図を記述する．図7の手順2，3に記述されている図式表現は手順1.5，2.5に記述されている内容を表す．この手順2，3により記述される図が「シナリオ分割図」である．このように，事象トレース図を部品化するとともに，部品化した事象トレース図を組み合わせてさまざまな処理系列を記述しようとするものである．

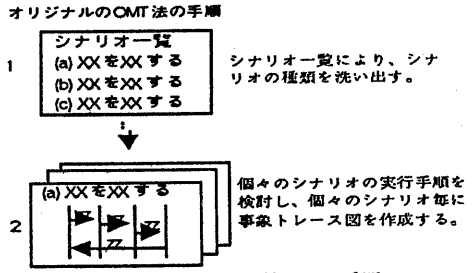

図6 既存のOMT法による手順

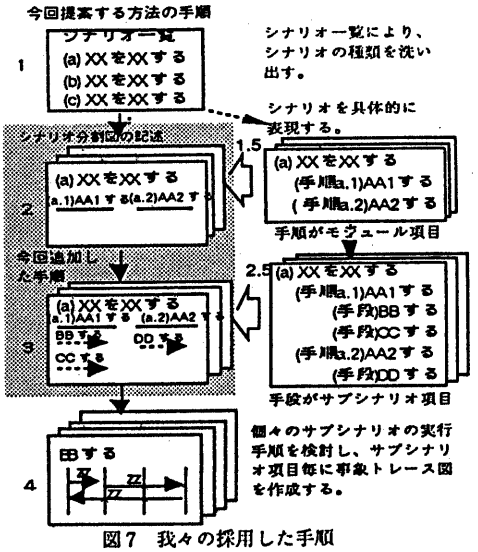

図7 我々の採用した手順

つまり，両者の大きな違いは，今回のアプローチには，オリジナルの手順に加えて「モジュール項目」と「サブシナリオ項目」という処理系列の一式をグループ化する単位が導入されているという点である．そして，「シナリオ分割図」はこの両項目の対応関係を表している．さらに我々は，システムの振舞いを大局的に把握するため，サブシナリオ項目に対して意味付けした表記法を提案した．この表記法を図8に示す。

サブシナリオ項目間の関係表記法	表記法	意味
OR関係表記法	モジュール項目 サブシナリオ項目1 サブシナリオ項目2	手段としてサブシナリオ項目1とサブシナリオ項目2の2つがあり，いずれかが実行される．
条件付きOR関係表記法	モジュール項目 □サブシナリオ項目1 □サブシナリオ項目2	事前状態によってサブシナリオ項目1とサブシナリオ項目2のいずれが実行されるか決まる．
AND関係表記法	モジュール項目 サブシナリオ項目1 サブシナリオ項目2	サブシナリオ項目1とサブシナリオ項目2は両方実行される．

図8 サブシナリオ項目の表記法

このようなシナリオ分割図を利用することによって，

(1) 個々の事象トレース図同士の関係を意味的表記法を用いたシナリオ分割図として記述しておくことでシステムレベルでの振舞いの大局的な把握を可能にする．

(2) シナリオをモジュール化し，トップダウンに事象トレース図を作成することで，事象トレース図を部品化できるので，再利用ができ，記述内容を標準化できる．

これを図4の「コピーする」というシナリオの例で説明する．まず「コピーする」という処理は，手順1として「対象を選択」，手順2として「メニューを選択」，手順3として「コピーを実行」という一連の流れからなり立つ．この各手順を「モジュール項目」として事象トレース図を細分化する．「コピーする」のシナリオ分割図と事象トレース図との対応を図9に示す．

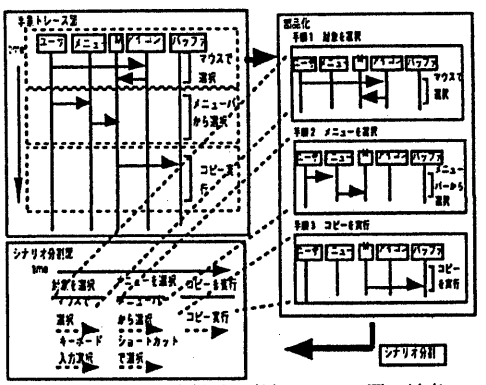

図9 シナリオ分割図と事象トレース図の対応

図9で表されているシナリオ分割図の実線が「モジュール項目」に相当するものである．時系列は左から右へとなっている．また，手順1と手順2にはそれぞれ2つの方法が存在している．これらが，実線の下に記述された点線の矢印で表されている「サブシナリオ項目」に相当する．例えば「対象を選択」するための方法としては，「マウスで選択」する方法と「キーボード入力選択」する方法があることを示している．実際には，点線の矢印で表されているサブシナリオ項目ごとに事象トレース図を記述する．また，逆向きの矢印でサブシナリオ項目を記述することによって状態の後戻り記述も可能である．このシナリオ分割図の利用により，2.2節で述べた問題に対して次のような効用が得られる．

問題1 シナリオ網羅の困難性

効用 各サブシナリオ項目のさまざまな種類の組み合わせ方式によって事象トレース図を体系化させることで，シナリオを網羅できる．

例(図10)サブシナリオ項目の表記法の例を図10に示す．

・OR関係表記法

「対象を選択」する方法として「マウスで選択」と「キーボード入力選択」の2つの方法が存在する場合，例のように点線の矢印の並列表記によるサブシナリオ項目の記述で明記できる．

・条件付きOR関係表記法

「マウスで選択」する際に同じ選択方法でも，その時の状態によって異なる処理系列になる場合，例のように□付き点線の矢印の並列表記によるサブシナリオ項目の記述で明記できる．

・AND関係表記法

「削除」を行なうには「スタック削除」と「カード削除」が同時に起こるということが，例のように実線の矢印の並列表記によるサブシナリオ項目の記述で明記できる．

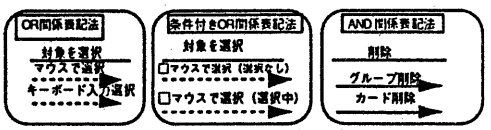

図10 サブシナリオ項目表記法の使用例

問題2 グループ内での認識不一致

効用 事象トレース図を部品化できるので，部品を単位とした事象トレース図の記述担当範囲の明確化と記述内容の標準化がはかれる．

例（図11）「対象を選択」と「メニューを選択」は開発担当者Aが，「コピー実行」は開発担当者Bがそれぞれサブシナリオ項目に対応する事象トレース図を担当する，というように開発者の明確な担当分けができる．また，サブシナリオ項目毎の事象トレース図は1つに限定できるので，開発担当者間で同様のサブシナリオ項目に対する事象トレース図の記述内容の標準化がはかれる．

問題3 記述の冗長性

効用 事象トレース図の部品化により同一のサブシナリオ項目については再利用ができ記述の重複がさけられるため，冗長性が減る．

例（図11）オリジナルOMT法では図4に示されるように4通りの一連の事象トレース図を記述しなければならなかったが，シナリオ分割図ではサブシナリオ項目の組み合わせによる事象トレース図の再利用ができ，記述量を削減できる．

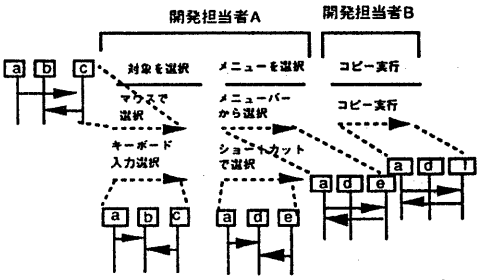

図11 シナリオ分割図と事象トレース図との対応

このように，事象トレース図を部品化するとともに部品の組合せ方に意味付けを行なうことで，シナリオ分割図によってシステムの処理系列の大局的な把握と仕様書作成枚数の削減を可能としている．

4 適用結果

提案したシナリオ分割図を実際のシステム開発に適用しその評価を行なった．本章では，適用した実システムの紹介と作成した仕様書枚数，さらにシナリオ分割図の評価結果について説明する．

4.1 適用システム

適用したシステムはハイパーエディタ[2]である．図12にそのツールイメージを示す．UI画面をデザインし，そこに仮想的な操作環境を実現しデモを行なうことが可能なGUIアプリケーションツールである．画面デザインと画面内部品に動作を設定するための編集機能と，実際に内部動作を設定した画面を実行するためのプロトタイプ実行機能がある．プロトタイプを実行することで，画面上での操作デモを行なうことができる．実装規模はC++で20KLOC程度である．

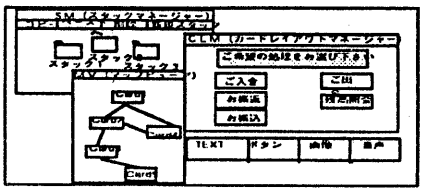

図12 ハイパーエディタ

ハイパーエディタの分析にOMT法を適用した結果作成した仕様書枚数を図13に示す．なお，機能モデルは動的モデル作成を完了した時点で，機能モデルにおける情報を十分記述していると判断できたため省略した．

ドキュメント分類	ドキュメント	対象	枚数
オブジェクトモデル	オブジェクト図	全体	1
動的モデル	事象トレース図	全体(サブシナリオ展開後)	154
	シナリオ分割図	典型的操作	10(64種類)
	事象フロー図	全体	1
	状態遷移図	個々のオブジェクト	27

図13 作成した仕様書枚数

図13から作成する仕様書のうち事象トレース図の作成枚数が最も多いことがわかる．

4.3 評価結果

評価項目は，シナリオ分割図導入によるシステムの振る舞いの大局的な把握のし易さと，事象トレース図の再利用率についてである．

(1)システムの振る舞いの大局的な把握のし易さ

図14はシナリオ分割図に対する事象トレース図の作成枚数を示している．縦軸は仕様書の種類，横軸は仕様書の作成枚数を表している．

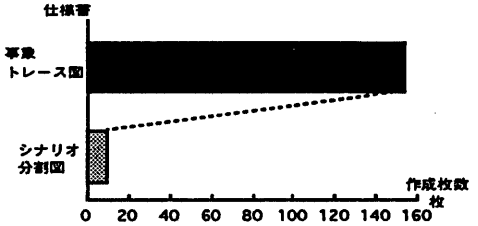

図14 事象トレース図とシナリオ分割図の仕様書枚数の対比

図14に示されるように通常事象トレース図だけで把握しなければならないシステムの振る舞いを，事象トレース図の1割以下のシナリオ分割図により把握することが可能となる．詳細レベルの大量の仕様書を多数の開発者間で理解しようとするより，グループ内でのコンセンサスをはかる手立てとして，より抽象的なレベルでシステムの大局的な振る舞いを把握しようとする時シナリオ分割図は役立つことがわかる．

(1)シナリオ分割図導入による事象トレース図の再利用率

図15は通常のOMT法による事象トレース図の記述量と，シナリオ分割図を使用した場合の事象トレース図の記述量を表した図である．縦軸は記述方法を表し，横軸は，OMT法による記述量を100%(見込) とした場合の記述量の割合を示している．

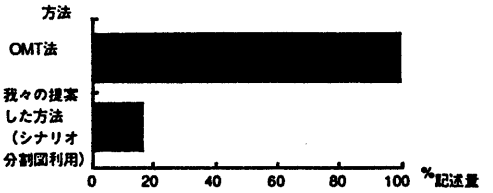

図15 事象トレース図の記述量の比較

図15から，シナリオ分割図を導入した場合，事象トレース図の記述量は1/5以下に削減されていることがわかる．すなわち，シナリオ分割図を使用しなかった場合の80%程度の記述量がシナリオ分割図による再利用でまかなえることがわかる．このように，高い再利用率が得られたのは適用システムであるGUIアプリケーションの特質によるものである．GUIアプリケーションはUIを重視するので，処理系列が「対象を選択」→「メニューを選択」→「処理」という一連の流れが多い．このうち，「対象を選択」と「メニューを選択」のためのUIには数種のものを与えている場合が多い．このため「対象を選択」と「メニューを選択」の組み合わせによりさまざまな事象トレース図が発生してしまう．この組み合わせ部分が事象トレース図の約80%を占める．この部分をシナリオ分割図による再利用でまかなえたため，このような大幅な記述量削減につながったと考えられる．

5 おわりに

OMT法による分析作業を共同作業で行なった場合に発生する問題点として，グループ内でのコンセンサスがとりにくいということがある．特に開発対象システムがGUIアプリケーションの場合，UIの種類が多種にわたり複雑にからみあうためこの傾向が顕著に現れ，単独作業で行なう場合よりも却って作業効率が悪くなる場合がある．本稿では特に記述量が膨大となる事象トレース図に着目し，グループ内でのコンセンサスと記述量減少の手立てとして，事象トレース図を部品化するための「シナリオ分割図」を提案し，その効果を確認した．本稿では分析作業における共同作業について報告したが，今後設計，実装と作業を進めるなかで，分析フェーズの位置付けを明らかにし，共同作業のコンセンサスを得るための鍵となる情報について明らかにしていきたい．

参考文献

[1]James Rumbaugh et al.，羽生田監訳「オブジェクト指向方法論OMT-モデル化と設計-」，トッパン，1992
[2]Hiroyuki Kamio et al."A UI DESIGN SUPPORT TOOL FOR MULTIMODAL SPOKEN DIALOGUE SYSTEM," ICSLP,Japan,Sep.pp.1283-1286,1994