情 報 シ ス テ ム 63－4
（1997．3．19）

言語体験による繰り返し型オブジェクト指向グループ学習

高橋 富夫　深沢 竜一　吉原 哲宏

新しい技術を学習する方法の一つにグループ学習がある．共通のテーマをもつ少人数のメンバが定期的に集合して学習する．時間や費用の制約が比較的緩やかであるので多忙なＳＥに好まれる．筆者らは，主として，事務処理アプリケーションを開発してきたメンバを対象に，オブジェクト指向技術（ＯＯ）のグループ学習会（平たくいえば，勉強会）を主催してきた．

構造化パラダイムに長年慣れ親しんできた学習者にとっては，ＯＯパラダイムへの移行は非常にタフである．学習会を開始した当初は，学習の進捗や成果の個人差は避けがたいものと考えていた．しかし，いくつかのグループの観察や自らの参加を通じて，学習者は，ＯＯ学習を滝モデルのように前工程から順次行おうとする傾向があること，実装段階では一挙に多数のクラスを実装しようとして，ＯＯ概念とＯＯ言語の「二重の未知」と格闘していることを発見した．構造化／滝モデル文化の慣性のなせる技である．

ＯＯの学習にも，ＯＯパラダイムで提唱されているプロセスモデル，つまり，繰り返し型モデルやインクリメンタル開発モデルを採用し，言語体験と組み合わせる学習方法が有効である．本稿では，事務処理アプリケーションを構造化パラダイムで開発してきた学習者が，オブジェクト指向技術をグループで学習するときの問題点を整理し，これを解決する新しい方法を提案する．

第一章　はじめに

構造化パラダイムに長年慣れ親しんできた学習者がＯＯを学習するとき，以下の傾向があることを発見した．

① 構造化パラダイムではプロセスモデルとして滝モデルが多く使われており，ＯＯ学習も滝モデルのように前工程から順次行おうとする傾向がある．全体を理解する前に詳細を理解しようとすると，分析段階で方法論の迷路に入るこむことが多い．

② 実装段階では，一挙に多数のクラスを実装しようとして，ＯＯ概念とＯＯ言語の「二重の未知」と格闘することになる．これも，構造化／滝モデル文化の慣性のなせる技である．

ＯＯの学習にも，ＯＯパラダイムで提唱されているプロセスモデル，つまり，繰り返し型モデル[05]やインクリメンタル開発モデル[06]を採用し，言語体験と組み合わせる学習方法が有効である．

第二章では，ＯＯグループ学習の方法と実施例，および，その教訓を紹介する．第三章では，オブジェクト指向の開発パラダイムをグループ学習にも取り入れた方法を提案する．第四章では，その適用事例を述べる．最後に，まとめと評価をおこなう．

第二章　ＯＯグループ学習の教訓

本章では，言語体験によるＯＯグループ学習の方法，学習者の特性を述べ，幾つかのケーススタディを通して得られた問題点と教訓を述べる．

2.1　ＯＯグループ学習の方法

ＯＯの学習方法には，集合教育，自己学習，ＯＪＴ教育，研究会，グループ学習，および，これらを組み合わせた方法がある[10]．また，グループ学習にもさまざまな形態がある．筆者たちが扱ってきたグループ学習では，初期の段階で以下の項目をグループ内で決定する．

- グループ構成　類似のテーマをもつメンバがグループを構成する．
  グループリーダを決める．
  テクニカルアドバイザ（以下，ＴＡ）が参加する．
- テーマ　学習の目的を簡潔に表現する．
- 対象 ｱﾌﾟﾘｹｰｼｮﾝ　分析，実装の対象アプリケーションを決める．
- 教材　使用する教材を決める．
- 学習方法　学習会の期間，開催周期，開催場所を決める．
- ｺﾐｭﾆｹｰｼｮﾝ　相互のコミュニケーションの方法をきめる．

類似のテーマをもつメンバが 3～5 名がグループを構成する．学習の進め方はメンバの自主性を尊重してグループで決める．グループリーダを互選で選び，グループリーダはメンバ間の調整をする．ＴＡは，技術的な助言をする．グループで学習目的や学習内容を簡潔に表現したテーマを決める．これまでのテーマ例を表 1に示す．

グループ内で一つのアプリケーション（共通問題）に取り組む場合と，メンバがそれぞれのアプリケーション（個別問題）に取り組む場合がある．これまでのアプリケーションの例，実装言語，共通／個別の種類を表２に示す．

Experience on OO Group Study through Iterative Implementation

Tomio Takahashi　Ruhich Fukasawa　Tetsuhiro Yoshihara
Fujitsu Ltd.　Kanebo Ltd.　Fujitsu Oita Software Labaratories Ltd.

表1　グループでのテーマの例

構造化技術とオブジェクト指向技術の比較
ＯＯ分析手順の作成
実装体験によるＯＯの効果的利用法
Design Pattern　の事務処理への適用研究
ＯＤＢとＲＤＢの比較
JAVAの事務処理への適用可能性の研究

表2　対象アプリケーションの例

対象アプリケーション	実装言語	問題
切符自動販売機	C++	共通
エレベータシステム	Smalltalk, IP	共通
芋虫問題（ゲームソフト）	JAVA	共通(ﾃｷｽﾄ)
販売管理システム	C++, ODQL, JAVA	共通　個別
組立工業の生産管理システム	ODQL	個別
輸送機器の障害管理システム	C++	個別
文書管理システム	ODQL	個別
機器接続管理システム	ODQL	個別
ソースコード解析システム	C++	個別

学習過程で使用する教材を決める．主に，過去のグループ学習会の報告書や集合教育の教材，市販の入門書を使用している．つぎに，学習方法として，グループ学習の期間，開催周期，開催場所を決める．期間は 6～10ケ月，開催周期は 2～6 週間が多い．場所はメンバの出身地の持ち回りとしている．メンバ間，および，メンバとＴＡとの間のコミュニケーションには，NIFTY-Serve のプライベートフォーラムを利用する．

2.2 学習者の経験

表１，表２のとおり，学習者の大部分は，事務処理アプリケーションを開発している．学習者のこれまでの開発パラダイムは，ＤＯＡ（データ中心技術）を含めて，いわゆる，構造化パラダイムである．ＯＯパラダイムで分析をするには，クラス，情報隠蔽，継承，カプセル化などの概念を学習する必要がある．プロセスモデルに関しては，ほとんど滝モデルの経験のみである．開発環境はＣＯＢＯＬを中心としたメインフレーム環境が多い．ＵＮＩＸやＰＣワークステーションなどの開発環境には当惑するメンバもいる．開発作業では，企画，分析，設計，プログラミングなどの滝モデルの開発工程単位で分業が徹底している．一般に，経験を経るにしたがって上流工程を担当し，しばしば，プログラミングを軽視する風潮がみられる．これはＯＯでの繰り返し型開発モデルで実装するときの障害になる．一方，プロジェクトの評価基準については，伝統的に，工程（納期），品質，費用（生産性）である．ＯＯでは，これらに加えて柔軟性／拡張性／変更容易性や短期開発が加わる．しばしば，後者が伝統的評価基準に優先する．学習者は，これらの新しい文化を一気に体験することになる．表３．

表3　学習者の新旧文化

	現行の文化の例	新しい文化の例
開発パラダイム	構造化パラダイム	ＯＯパラダイム
プロセスモデル	滝モデル	インクリメンタル開発
言語	ＣＯＢＯＬ８５	ＯＤＱＬ，Ｃ＋＋，ＪＡＶＡ
分業	企画，設計，製造分業	繰り返し開発
プロジェクト評価基準	工程，品質，費用	短期開発，拡張性

2.3 ケーススタディ

グループ学習は，メンバの自主性を尊重しグループで学習目的や方法を決める．当初は，自主性という言葉に囚われてグループに任せていた．スムーズにパラダイムシストを行えたグループもあるが，躓いたグループもあった．あたかも，

表4　学習者の経験

学習者	経験	ﾊﾟﾗﾀﾞｲﾑ	ﾌﾟﾛｾｽﾓﾃﾞﾙ	言語	分業
Ａ１	１２	構造化	ＷＦ	ＣＯＢＯＬ／Ｃ	企画／分析
Ａ２	１０	構造化	ＷＦ	ＣＯＢＯＬ	分析／設計
Ａ３	５	構造化	ＷＦ	ＣＯＢＯＬ	設計／実装
Ｂ１	２０	構造化	ＷＦ	ＣＯＢＯＬ	企画／分析
Ｂ２	８	構造化	ＷＦ	ＣＯＢＯＬ	分析／設計
Ｂ３	４	構造化	ＷＦ	ＣＯＢＯＬ	設計／実装
Ｃ１	１８	構造化	ＷＦ	ＣＯＢＯＬ	企画／分析
Ｃ２	１２	構造化	ＷＦ	ＣＯＢＯＬ／Ｃ	企画／分析
Ｃ３	５	構造化	ＷＦ	ＣＯＢＯＬ	設計／実装

システム開発で，プロジェクトのメンバを集め要求仕様を決めれば，納期内に自動的にシステムが完成するかの如く，学習グループを構成しテーマを決めれば，ある一定の時間が経過すれば学習成果は自動的に達成されると考えていたこともあった．以下に三つの事例を見る．なお，学習者の経験を表４に示す．

【ケースA】... 共通問題，切符自動販売機，C++
チームは三名のメンバ（A1，A2，A3）より構成された．メンバの経験が異なるため，仕様がわかりやすいという理由で切符の自動販売機を共通問題とした．主な仕様は，コインの投入，行き先の選択，切符の出庫，お釣りの返却である．一日の最後に売上日報を出力する．釦の押下やコインの投入，切符の出庫は，キーボードの入出力でシミュレートすることにする．クラス図の作成までは比較的順調に進んだ．しかし，相互作用図(Interaction Diagram[08])を書く段階で，他社券（JRからみると私鉄）の扱い，複数枚同時購入などのユースケース[08]は複雑すぎるという理由で削った．分析／実装は，実質的にはA1氏とA2氏が行った．A3氏は，実装でグループから遅れた．学習期間は８ケ月．
A1氏は余裕があったので，清涼飲料水の自動販売機も作成し，さらに，汎用販売機（お金を投入して商品を出庫する機械）を完成させた．A2氏は，クラスとインスタンスやカプセル化の概念は実装して初めて理解できた，しかし，終了時点では，ポリモフィズムはわからなかった，と報告している．A3氏は，C++のポインタの理解に時間がかかった．
【ケースB】... 個別問題，JAVA／C++
チームは三名のメンバ（B1，B2，B3）より構成された．OOの基本概念を学習した後，各メンバは個別に問題を選定し，ユースケースを書いた．TAの助言を得ながら，メンバ全員でクラス図と相互作用図を作成した．実装は，ふたたび，各メンバが各自におこなう．B1氏は，OOの効果を最大限に享受することを狙って，現行システムの販売管理アプリケーションのもっとも複雑な問題に取り組んだ．他のメンバはこの仕様の理解に時間がかかり，共同でのクラス図の作成が難航した．B2氏は，これをみて問題を変更した．集合教育のテキスト[10]の事例から，複雑な部分の仕様を削り簡単な受注業務の問題とする．集合教育のテキストにユースケース，クラス図，相互作用図などがあり，これらの一部を削除するだけなので表面上は円滑に進んだ．実装段階では，B1氏は，JAVAのシンタックスエラーに悩み，また，10を越えるクラスを一気に組み込んだため進退不能の状態に陥った．B2氏は，問題が簡単だったので一番早く完成した．B3氏は，習期間内には分析，実装が完成しなかった．学習期間は６ケ月．
B1氏は，実装段階でカプセル化がまったく分かっていないことがわかった．B2氏は，継承もないほど仕様を削ってしまった．実装はしたが，OOの効果が理解できない，という．B3氏は，概念を完全に理解しようとして，OMTのテキストの精読に時間を費やしすぎた．
【ケースC】... 共通問題，芋虫問題，JAVA／C++
チームは三名のメンバ（C1，C2，C3）より構成された．JAVAは未経験であった．まず，JAVAに慣れるという理由で，簡単なゲームソフト（芋虫問題[12]）を共通問題とした．この問題は，集合教育のテキストに掲載されており，クラス図，相互作用図があり，また，C++で実装されていた．これをJAVAに変換することから始めた．JAVAのシンタックス，アプレットの使い方でしばしば苦労したが，約二ヵ月でGUIも含めて全員が完成させた．その後，各メンバが個別問題に取り組んだが，順調な立ち上がりをみせた．Cの経験のあるC2氏の存在がグループ全体の進捗に貢献した．学習期間 10 ケ月．
C1氏は，なにも考えないで，ひたすら，C++のコードをJAVAに変換した．また，JAVAを，市販の本の例題をそのまま転記し，その一部を修正することによって学習した，と報告している．学習方法は，C2氏も同様であったがGamma たちのデザインパターン[07]が参考になった，という．C3氏は，ゲームソストの開発に夢中で，気がついたらOOの概念を学んでいた，と振り返っている．

上記三つのケースから学んだ教訓を整理する．
◆メンバが作成する共通問題の落とし穴
アプリケーション経験の異なるメンバがグループを構成して，共通問題を作成しようとすると，全てのメンバに理解しやすい問題を選ぶ．よく選ばれる問題として，自動販売機，エレベータ，コンビニエンスストアがある．グループ学習がメンバの自主性を尊重してメンバ自身が仕様を作成することは悪くはないが，OO経験のないメンバが自ら作成した問題が必ずしもOO学習の問題として適切であるとは限らない．ケースAでは，最初は難しすぎ，途中で仕様を削りすぎた．継承やポリモフィズムも含まれていない．TAの適切な助言が求められる．一方，ケースCでは，集合教育の教材から問題を選んだ．その問題は適度な複雑さで，段階的にOOの概念が学べるように慎重な工夫がされている．
◆滝モデルからの脱却
全体を理解する前に，OMTの分析フェーズを精読しても，クラス図が書けない．OOP（実装）体験がないとOOA（分析）の意味が分からない方法論（あるいは，その教えかた）に問題がある．B3氏は，分析段階で，相互作用図や状態遷移図，限定子やリンク属性[08]などの表記法や意味を理解しようとして学習が停滞した．事後的にみると，その時点では，インスタンシエートの意味が分かっていなかった．一方，B1氏は，最初から 10 を超えるクラスを実装しようとして進退が極まった．事前に，インクリメンタル開発の有効性を学習しているが，滝モデルの慣性からの脱却は難しい．
◆言語の学習曲線
全員がCOBOLの経験者である．C++やJAVAなどの言語は，自己学習に委ねているため，習得度には大きな開きがみられる．事後面接では，順調に離陸したメンバとそうでないメンバには，以下の三点で差がみられる．第一は，COBOLとCのバイリンガルのメンバは，概して順調に離陸している．第二は，EDP歴の長い（一般に，年長者）メンバには，新しい言語を敬遠し勝ちである．また，OOの概念を理解することのみを目的に言語に取り組むメンバは，学習期間内には離陸できないことが多い．第三は，市販本のコードや集合教育のテキストの事例をひたすら入力した，と報告しているメンバは順調に離陸している．
◆OO＝ポリモフィズム論
多くの学習者が，インスタンシエート，メッセージパシング，継承は，実装して初めて体感した，と報告している．ま

た，ポリモヘィズムを実装して，Aha! Experience （嗚呼，そういうことだったのか体験[01]）をした，という．また，Composite デザインパターン[07]も目から鱗がとれるような体験だった，と感想を述べている．特に，Composite では，ポリモフィズムと同時に，ＣＯＢＯＬ時代には味わえなかった操作の再帰呼び出しに感激する．一方，Ｂ２氏のようにポリモフィズムが無いと，これはＯＯではないと，悩んでしまう．技術の魅力は，学習の動機付けに貢献する．しかし，副作用もある．いったん，このような技術の魔力に取りつかれると，クラス構造を必要以上に複雑にすることがある．ＯＯの目的の一つは，変化に強い，拡張性のある robust なクラス構造を構築することである[06]．学習テーマがＯＯＡなどの上流にある場合は，一通りの概念の言語体験ができたあとは，エンティティ（ドメイン）クラスやその関係の演習に時間を重点配分したほうがよい．たとえば，Gamma たちのテキストに出てくるような抽象クラスや動的なクラスの協調の仕組みへの深入りは，初学者に混乱を与える．

第三章　ＯＯグループ学習の方法論

集合教育には長年のノウハウが蓄積されている．さらに，講師は，教育対象や学習者の特性を考えて，効果的で効率的な教育のために，さまざまな工夫を意識的にする．しかし，メンバが日常業務の空き時間を利用して自主的に参加するグループ学習については，明確な学習の方法論を意識して行うことは少なかった．我々は，あたかも，システム開発で，プロジェクトのメンバを集め，要求仕様を決めれば，納期内に自動的にシステムが完成するかの如く（この方法は，健全な方法論や進捗や品質管理などの統制を欠いており，成功は全く偶然に支配されるのだが），学習グループを構成し，テーマを決めれば，ある一定の時間が経過すれば学習成果は自動的に達成されると考えていた．そして，前章の事例に見たごとく，メンバ間で学習成果の達成にばらつきがでる．そこで，これまでの経験に基づいて，繰り返し型ＯＯグループ学習の方式を考案した．

3.1　繰り返し型ＯＯグループ学習方式の構成

グループ学習のＣＳＦ（Critical Success Factors: 本質的成功要因）は，学習プロセス，教材，コミュニケーションの方法，学習成果のまとめである．図１．学習プロセスは，標準学習パス，繰り返し学習モデル，インクリメンタル学習（の考え方）よりなる．教材は，学習プロセスを促進する触媒であり，共通問題，事例集，ＦＡＱ（Frequently Asked Questions)よりなる．メンバ間のコミュニケーション手段としてプライベートフォーラムの活用も効率に影響する．メンバの理解度の把握や適切な助言が必須である．また，学習成果のまとめも重要である．本章では，学習プロセスの要素を概説する．図 1

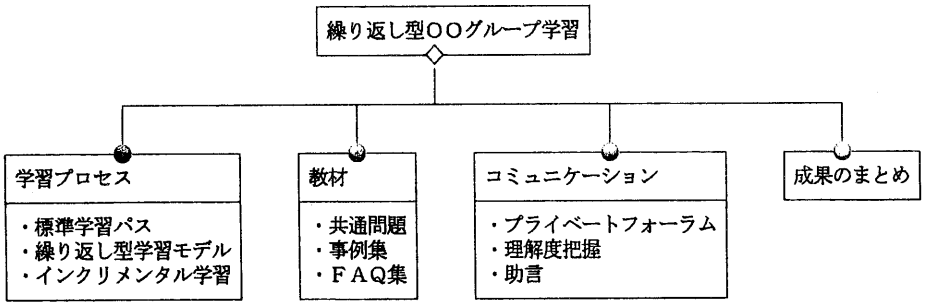

図 1　繰り返し型ＯＯグループ学習

3.2　標準学習パス

標準学習パスは，ＯＯを始めて学習するための標準的な学習順序を示している．学習テーマやメンバの経験により，適宜，カストマイズして使用する．図2．

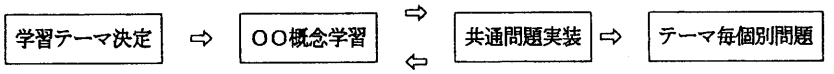

図 2　標準学習パス

たとえば，表1 の「Design Pattern　の事務処理への適用研究」というテーマでは，ＯＯの概念を学習したあと，教材として用意された共通問題を実装する．この共通問題には，クラス，インスタンシエート，メッセージパシング，継承，ポリモフィズムが段階的に学習できるように，慎重に問題が考案されている．Ｃ＋＋でのコード例がついている．Composite や Observer などの代表的な Patternも含まれているが，どこまで扱うかは，メンバの経験や理解度によって，ＴＡが適当に助言をし，メンバが決定する．この共通問題には，Abstrtact Factory や Decoratorなどの高度のパターンは入っていない．また，「ＪＡＶＡの事務処理アプリケーションの適用可能性研究」のテーマの場合は，内容は同じで，ＪＡＶＡでのコード例がついている．

基本的な考えは，ＯＯの概念の理解に，言語学習が有効である，という経験[08]に基づいている．目的は，ＯＯの概念

を理解することであり，言語の詳細を学習することではないので，難解なシンタックスは使用しない．また，クラスの数も４～６個に限定している．クラス図がコードに対応していることや，動的な動きを示す相互作用図の意味を理解する．個別のテーマ，上記の例では，Design Pattern　やＪＡＶＡ固有のテーマは，テーマ毎の個別問題の段階で扱う．

3.3　繰り返し方学習モデル

共通問題を使用して，計画（学習目的），概念，その実装，評価のサイクル（ラウンド）を繰り返してながら○○の概念を学習する．図３．

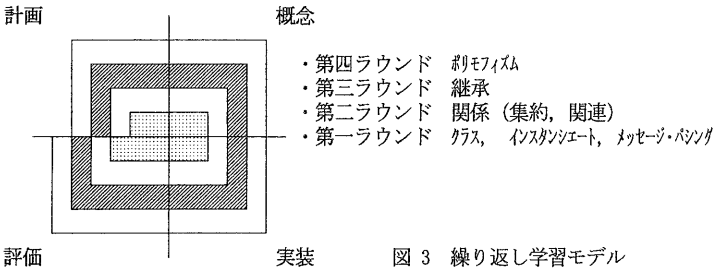

図 3　繰り返し学習モデル

第一ラウンドでは，少数の（たとえば，二つの）クラスよりそれぞれオブジェクトを生成し，一つのオブジェクトが他のオブジェクトにメッセージを送る．これだけのプログラムを実装して動作させる．○○概念としては，クラス（属性，操作）の定義を通して，カプセル化，情報隠蔽を学習する．インスタンシエートにより，クラスとオブジェクトの概念を体験し，コンストラクタの動きを理解する．オブジェクトへのメッセージをおくり，オブジェクト間の通信の仕組みを理解する．第二ラウンドでは，関係（集約：aggregationと関連：association）を学習する．第三ラウンドでは継承を，第四ラウンドではポリモフィズムを学習する．

3.4　インクリメンタル学習

個別問題ではインクリメンタルに実装する．最初はクラス数２から始める．まず，もっとも中心的で重要なクラスをえらび，オブジェクトを生成し初期値を設定する．そのオブジェクトを検索し，表示することから始める．そして，順次，段階的にクラス数を増やす．リンク属性やポリモフィズムの実装は最後に行う．

第四章　繰り返し型○○グループ学習の事例

本章では，前章で提案した繰り返し型○○学習の事例を２件紹介する．ともに，3.3 節の共通問題で○○の基本概念を学習した後，個別問題をインクリメンタルに取り組んだ．

4.1　事例　その１　販社政策ｼｽﾃﾑ

(1) システムの概要と開発順序

Ｄ氏が，分析段階で作成した販社政策システムのクラス図を示す（紙面の都合で，クラス名と関係のみ）．図 4. 商品カテゴリと販社カテゴリによって販社政策（戦略パターン）が異なる．政策マネージャは，取引（取扱）状態によって，

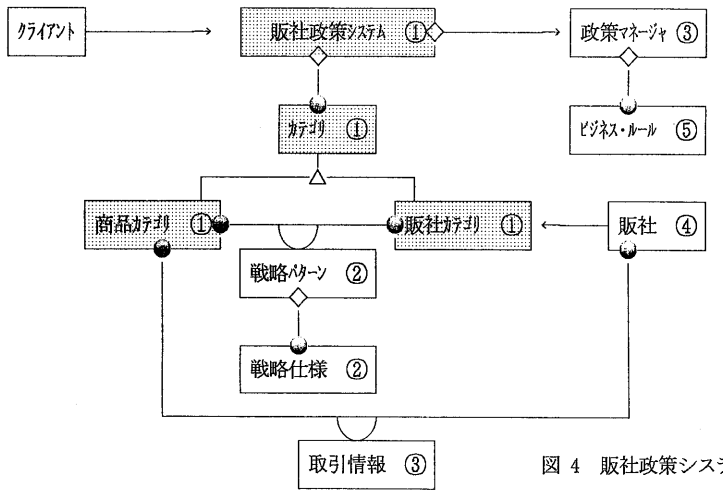

図 4　販社政策システムのクラス図と実装順序

どの戦略を用いるかを決定するためのルール（ビジネスルール）をもっている．
クラス名の右に表示した番号は，実装順序を示している．順次，インクリメンタルに実装するクラスを追加している．言語はC＋＋．
なお，D氏は，共通問題でOO概念の基本的な実装方法については事前に学習していたが，CおよびC＋＋で自由にプログラムを書ける域には達していなかった．必要なイディオムを学習するために，その都度，小さなテストプログラムを作成した．図5に，それぞれのテストプログラムでの学習したイディオムと各版の取り組み順序を示す．

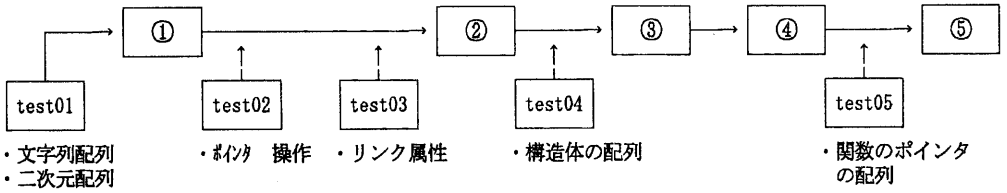

図5　イディオムの学習と各版の取り組み順序

各版ごとのクラス数，操作数，行数，クラス平均操作数，操作平均行数を示す．表5．

表5　各版ごとの主要メトリックス

クラス名	①	②	③	④	⑤
ｶﾃｺﾞﾘ	2	2	2	2	2
商品ｶﾃｺﾞﾘ	3	3	3	3	3
販社ｶﾃｺﾞﾘ	3	3	3	3	3
戦略ﾊﾟﾀｰﾝ		5	5	5	5
戦略仕様		2	2	2	2
販社政策ｼｽﾃﾑ	12	12	20	21	21
販社			7	8	8
取引情報			3	4	4
政策ﾏﾈｰｼﾞｬ					4
ﾋﾞｼﾞﾈｽ・ﾙｰﾙ					1
クラス数 c	4	6	8	8	10
操作数 m	20	27	44	47	52
行数 l	146	169	241	242	225
平均操作数 m/c	5	5	6	6	5
平均操作行 m/l	7	6	5	5	4

(2)　学習者の評価
D氏に面接をした．以下は，D氏のその体験談である．
・最初は，クラス数が 10 もあり，どこから手をつけていいか当惑したが，インクリメンタルに実装することで，常に，一つ，または，二つのクラスに集中すればよかった．
・クラスを追加する前に，必ず，小さな（10～20行）テストプログラムで言語のシンタックスと動作の確認をした．特にリンク属性の実装（商品カテゴリ--戦略パターン--販社カテゴリ，商品カテゴリ--取引情報--販社）を小さなプログラムで事前に確認できたことは効果があった．
・販社政策の戦略やその適用ルール（ビジネスルール）が複雑で，しばしば変更になる．TAの助言を参考にして，別のクラスにした．仕様やルールというクラスは，自分ではなかなか思いつかない．
・段階的にクラスを増やしたが，既存のクラスの公開インタフェースに大きな手戻りはなかった．知識としては，インクリメンタルモデルを知っていたが，経験できてよかった．

4.2　事例　その2　ソースコード解析システム
(1) システムの概要と開発順序
E氏は，C＋＋のソースコード解析システムを段階的に開発した．指定されたキーワード，クラス，操作，関数などのメトリックスを解析する．途中で，性能上の理由で，途中でクラスの構造を大幅に変更した．図6

D氏と同様，ファイルアクセスや文字列操作のために，多くのテストプログラムを作成している．各バージョンごとのメトリックスを示す．表6

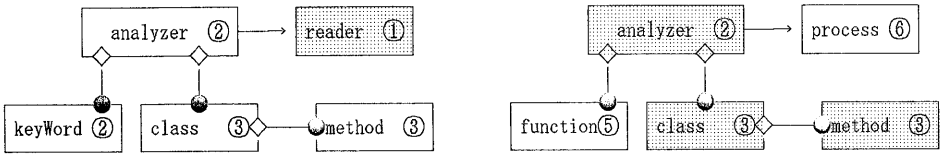

図6　ソースコード解析システムのクラス図と実装順序

表6　各版ごとの主要メトリックス

クラス名	①	②	③	④	⑤	⑥
reader	3	3	3			
analyzer		3	8	9	7	7
keyWord		3	3			
class			4	8	7	7
method			4	7	6	6
fuction					12	13
process						2
クラス数 c	1	3	5	3	4	5
操作数　m	3	9	22	24	32	24
行数　l	42	80	166	357	486	534
平均操作数 m/c	3	3	4	8	8	5
平均操作行 m/l	14	9	8	15	15	22

(2)　学習者の評価

E氏に面接をした．以下は，E氏の体験談である．

・keyWord, class, method, funtion などのクラスは比較的容易にみつかったが，その他のクラスを見つけるのが難しかった．

・最初は，ファイルアクセスの仕組みを情報隠蔽するために，reader クラスを作った．analyzer がキーワードを見つけ，クラスを発見すると，クラスオブジェクトを生成し，そのオブジェクトに操作を探せとメッセージを送る．操作が見つかると，操作オブジェクトを生成し，そのオブジェクトに行数を数えろとメッセージを送った．小さなテスト用のソースコードは優雅に動作したが，大きなプログラムは分析時間が非常に長くなり実用に耐えなかった．'reader'が一行のソースコードの情報しか持っていなくて，クラスオブジェクトや操作オブジェクトからアクセス要求がある度ごとに，ファイルの最初からソースコードをアクセスしていたためである．また，ソースコードにコメント行やストリング（たとえば，"}" や "class"）があると間違えた分析をした．そこで，実質的に getLine() の役割しか果たさない 'reader' を取り去った．また，分析に先立ってコメントを取り去る処理と，関数（クラスと一般関数）を見つけるクラス "process"を追加した．

・段階的に開発する途中で，テキスト解析の知識が増えていくことが分かった．また，事前に分析対象を単純にすることが重要であることを再認識した．

・学習終了後に，幾人から，ソースコードをクラスと考え「読み込む」や「解析する」を操作とすべきであるとコメントを得たが，開発中には気づかなかった．

第五章　まとめと課題

本章では，まとめと今後の検討課題を議論する．

5.1 まとめ

本稿では，構造化パラダイムに親しんできた学習者が，OOをグループで学習する方法とその問題点の幾つかをを事例を通して明確にした．そして，OOの開発で用いられる繰り返しモデルやインクリメンタルモデルをOOの学習にもとりいれた学習法を提案し，その事例を紹介した．それらの体験で得られた知見や教訓を整理する．

インスタンシエート，メッセージパシング，ポリモフィズムなどの概念は，実装(OOP) して初めて理解できた，と多くの学習者が報告している．分析段階で，クラス図や相互作用図（事象トレース図），状態遷移図を頭で理解できたと思っていても，インスタンシエート，メッセージパシングが理解できない状態では，それらの分析図をなぜ，どのように，書くべきか，また，どれくらい詳細に書くべきかがわからない．OMTのテキストを最初のページから精読することになりリンク属性や限定子などの細部で学習が止まる．したがって，全体を早期に理解する意味で概念の学習と平行して，言語による実装体験が有効である．ところが，構造化パラダイムで滝モデルを中心に事務処理アプリケーションを開発してき

た学習者は，ＯＯの学習に際しても，滝モデル型で上流工程から順次学習しがちである．この思考法は，ものごとの実行順序とその学習の順序を混同している[02]．

学習に際してグループで共通問題として選択するときに，二種類の共通問題があることに注意を払うべきである．メンバが作成する共通問題とＯＯの経験者が用意した共通問題である．アプリケーション経験の異なるメンバが自ら共通問題を作成しようとすると，全メンバに理解しやすい問題が選ばれる．ＯＯ経験のないメンバが自ら作成した問題が必ずしもＯＯ学習の問題として適切であるとは限らない．共通問題には，仕様を理解しやすいという利点はあるが，一種の遊びとなる危険性がある．メンバによっては真剣に取り組まないことがある．またグループの他のメンバに任せて，傍観することがある．一方，メンバの関心を高めるという点では，自分の問題（個別問題）が望ましいが，問題がＯＯ学習に適切であるかという点では同じ落とし穴がある．

我われのＯＯグループ学習では，ＯＯ経験者が助言者（ＴＡ）として参加する．しかし，ＴＡはＯＯ技術の一部についての経験はあるが，必ずしも，教育や訓練についての知識や経験をもっていないことがある．問題の適切性について助言したＴＡもいた．問題意識をもっていないＴＡもいた．集合教育は，効果と効率をもとめた時間と戦いであり，しかも，有償であるため，教育・訓練の立場から常に厳しく評価される．一方，このグループ学習は，メンバの自由意思で参加している．グループ学習の方法論を欠とく，メンバ間に大きな学習成果の差が生じることになる．

これらの観察と反省を基に，繰り返し型グループ学習の方法論をまとめた．従来，私的な勉強会として積極的には取り上げられなかったグループ学習に一定の品質を確保することを狙いとしている．これは万能のグループ学習の方法論ではなく，構造化パラダイムに慣れ親しんできた事務処理アプリケーションの技術者がＯＯパラダイムへのシフトに特化した方法論である．ＯＯの開発プロセスモデルを学習プロセスに取り入れた．メンバの経験を考慮してメンバ毎に学習パスを設定する．ＯＯ概念を実装を繰り返しながら学習する．また，インクリメンタルな実装を提案している．

この提案にしたがって開発した二つの事例，および，その実装順序と基本的なクラスや操作についてのメトリックスを示した．二つの事例とも学習者の満足度は高く，我われの提案が効果的であることを示している．

5.2　課題

Latoたちは，訓練中に解くべき問題は，明快な解のある問題ではなく，実世界の問題を解くことが重要であり，また，メンタ（本稿ではＴＡに相当する）は，単に教師としてではなく，チームのメンバとして参加させるべきである[03]と主張している．また，Booch は，70% のクラスは分析中にたやすく発見できるが，25% のクラスは設計や実装中にしか見つけられない．残りの 5% は保守段階にならないと見つけられない，と述べている[01]．4.2 節で紹介したソースコード解析の問題は，述語論理などに精通した言語処理専門家にとっては，明快な解のある問題かもしれない．しかし，事務処理アプリケーションＳＥであるＥ氏にとっては，未知の問題であった．最初は分析対象のソースコードの構造のあまりの複雑性に当惑して適切なクラスを見つけられなかった．試行錯誤の結果，分析に先立ちコメントを取り去る事前「処理」をすることにより問題が簡単になることを発見した．この事前「処理」は「手続き的」であり，これはオブジェクト指向では「忌むべき」ものと考え，クラスとすることに抵抗があったと，事後報告している．このような学習過程の開発者の心理をＴＡは事後でしか把握できなかった．また，かりに，ＴＡがチームのメンバとして参加したとして，どのような問題には助言を与え，どのような問題には学習者に「悩ませる」べきなのだろうか．今後の課題である．

参考文献：

01　Object Solutions; Grady Booch; Addison-Wesley Publishing Company; 1996
02　The First Step in Training: Analysys & Design or Implementation Language?; Luke Hohmnann; JOOP 1996.10
03　Effective Training in OOT--Learn by doing; K. Lato　ほか; JOOP 1996.10
04　OOA/OOD/OOP: What programmers and managers believe we should teach; E.F. Gehringer　ほか; JOOP 1996.10
05　Software Engineering Economics; B. W. Boehm; Prentice Hall; 1981
06　オブジェクト指向ソフトウェア工学 OOSE; I. ヤコブソン; 西岡利ほか訳; アジソンウエスレイトッパン; 1995
07　デザインパターン；Gammaたち; ソフトバンク；本位田真一ほか訳; 1995
08　オブジェクト指向ＯＭＴモデル化と設計; J.ランボーほか; 羽生田栄一訳; トッパン; 1992
09　オブジェクト指向技術の効果的利用; LS研；1994
10　OO集合教育の経験; 高橋富夫ほか; 情報処理オブジェクト指向研究会; 1996.7
11　オブジェクト指向による分析設計実践; 高橋富夫; 富士通ラーニングメディア教材
12　オブジェクト指向へのいざない; 中谷多哉子; オブジェクト指向 '96　シンポジュウム資料集; 96.7