

ソフトウェア開発技法の教育経験に基づく実施上の急所

斎 直人

itsuki@slab.ntt.jp
NTT ソフトウェア研究所
〒180 武蔵野市緑町3-9-11

筆者らは、構造化分析設計技法のソフトウェア開発部門への導入に取り組んで来た。開発担当者の多くは構造化技法を未経験であったため、最初に構造化技法の教育を行った。教育の目的はソフトウェア開発への適用である。目的と対象者が限定されているため、教材の構成、研修方法に工夫、改善が求められた。そこで、教材や研修方法の改善を進め、テキストの構成、技法の解説、例題、演習などにおける留意点が明確になった。それは、学習後に実際のソフトウェア開発に構造化技法を適用できるような教材や研修を求める。本報告では教育への取り組みと留意点をまとめる。

Vital Points of Software Development Methodology Instruction

Itsuki Naoto

itsuki@slab.ntt.jp
NTT Software Laboratories
3-9-11 Midori-cho Musashino-shi Tokyo 180 Japan

The author has arranged the introduction of the structured analysis and design methodology to software development sections. Almost all engineers in these sections received the methodology instruction at first, because they didn't know it. The instruction aimed at the application of it to the software development. Since the aim and learners of the instruction is limited, improvement of the textbook and lessons was requested. So the text and lessons have improved, vital points of them became clear. This paper describes the instruction of the structured methodology and vital points of it.

1 はじめに

筆者らは、構造化分析設計技法（以下、構造化技法）[1][2]と、構造化技法に基づいてソフトウェア開発の上流工程作業を支援するCASEツールSoftDA[3]のNTT社内導入に取り組んできた。NTT社内には、多数のソフトウェア開発部門（以下、開発部門）がある。それらの開発部門への導入をなめらかに進める目的で、筆者らは1990年に専門家による普及支援担当を設置した。普及支援担当では、開発部門への導入に必要な技術情報の提供や、マニュアルの作成、具体的適用法の検討などを行っている。

開発部門への導入当初、開発部門担当者の多くはソフトウェア開発の初心者ではないが、構造化技法の経験はほとんど無かった。ほとんどの開発部門担当者は、それぞれの開発部門独自の開発技法を身に付けていた。それらの開発技法は、大型コンピュータを対象とした技法であり、ワープロ文書による設計処作成が中心である。ワークステーションやCASEツールの導入による、設計作業の機械化は進んでいなかった。そこで、構造化技法やSoftDAを早急に開発部門に導入し、ソフトウェア開発作業の効率化が求められた。

そこでまず、習得者がほとんどいない構造化技法とSoftDAの教育が必要となった。特に構造化技法は、SoftDAの学習に先だて行わなければならない。普及支援担当では、構造化技法とSoftDAの教育カリキュラムを作成し、導入推進の一貫として教育を行った[4]。

教育の目的や対象者は限定しており、求められる教育内容が具体的であるため、教材の構成、研修実施上に工夫が必要であった。筆者らは教育を進めながら、そのカリキュラムや教材の改善を進めた。本報告では、筆者らの構造化技法の教育への取り組み方を述べ、それから得た教育上の留意点、急所をまとめる。

2 構造化技法の導入と教育

2.1 体制

構造化技法とSoftDAの導入は図1の体制で進めた。普及支援担当は以下の業務を担当する。

- SoftDA ファイルの配布、インストール

- 構造化技法、SoftDA の教育
- SoftDA 操作の支援、障害対応
- ソフトウェア開発への導入時の問題解決

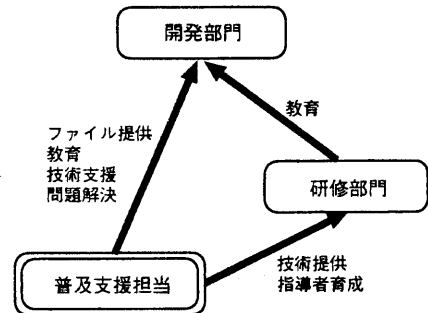


図1 構造化技法の普及体制

本報告の対象である「構造化技法、SoftDAの教育」は、各種の教育経験が豊富な社内の研修部門と協力して行った。図2に、筆者らが行った構造化技法教育の受講者数の累積を示す。毎年200名程度に研修を実施した。

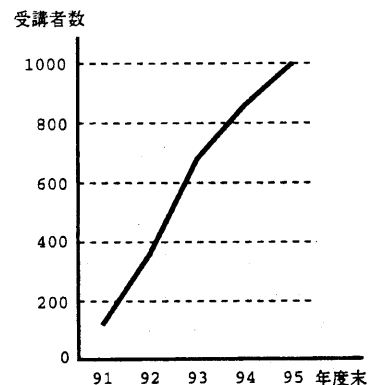


図2 構造化技法研修受講者累積数の推移

2.2 研修

研修は数名から20数名程度を1クラスとし、講義と演習を組み合わせで実施した。テキストは各受講生に配布するが、演習問題の回答は含んでいない。受講生が多い場合は、講師1名以外の講師補助者1～2名を付け、演習時の個別指導に対応した。SoftDAの操作を学習する

時は、1名に1台のワークステーションを割り当てる。

SoftDA の学習は操作法習得の後、構造化技法による分析設計作業の演習を行う。UNIX の操作が出来ない受講生は、SoftDA の操作学習の前に UNIX の操作の研修を行う。ところが、SoftDA による分析設計作業を行うには構造化技法の知識が必要である。このため、まず構造化技法の十分な習得が必須となる。

構造化技法の学習と SoftDA の学習は、この順序で実施するが、構造化技法だけを学習する場合がある。両者の学習が完了すれば、ソフトウェア開発への導入が可能になる。しかし構造化技法だけを導入し、SoftDA を使用せずにソフトウェア開発を実施することも可能である。構造化技法を適用する対象が狭い場合は SoftDA 未使用でも、設計品質の向上等が見込まれるからである。

したがって筆者らは構造化技法の教育に重点を置いて取り組んだ。

3 構造化技法教育の改善

3.1 学習者

構造化技法教育の目的は、短に技法を習得することではなく、その技法によりソフトウェア開発を効率化することである。したがって、学習者が構造化技法学習に取り組む意識は高く、目標が具体的である。

学習者は経験、年齢、技術的知識に幅がある。具体的には次のような特徴がある。

- ソフトウェア開発の経験者が多い。特に大型機による大規模システムの開発経験が多く、古典的な開発方法を身に付けている。しかし、初心者もいる。
- 年齢が20歳台から40歳台までに渡る。
- これまでに経験してきた開発対象に関する業務知識を豊富に持っている。その開発対象はさまざまである。
- 次の開発が間近にあり、構造化技法を早急に適用することが求められている。
- SoftDA の動作する UNIX の操作経験は少ない。開発現場への UNIX 普及は、当初は低かった。

- 上流工程で使用するツールはワープロ程度である。

3.2 改善の進め方

前節で述べたような特徴を持つ学習者への研修であるため、以下のように改善を進めた。

テキスト —

テキストとして市販の構造化技法解説書の使用は次の点で問題があるため、独自にテキストを作成することとした。

- 解説内容が幅広く、難しく、学習者が短期間に、開発作業に必要な項目を選択して学習するには不向きである。
- 例題、演習の内容が一般的で、開発作業で実際に記述する内容とかけ離れ、学習者が自らの問題として取り組めない。

当初のテキストは、要点だけを記述した2～30ページの少ない冊子であったが、学習者の意見を反映して追記、改訂を繰り返して、最終的には独学も可能な、解説を詳細にしたテキストとなった[5]。

カリキュラム —

開発現場での導入レベル、学習者の学習の必要度、時間的制約などにより、研修毎にカリキュラムを変えて実施する必要があった。そこで代表的な研修カリキュラムを数種作成し、研修申込時に選択できるようにした。以下にカリキュラムの種類を示す。

4日コース — ソフトウェア開発の初心者を対象に、演習時間を多くした研修。
表1に時間割の例を示す。

2日コース — ソフトウェア開発の経験者を対象に、演習時間を短縮し、短期間に習得できる研修。

半日コース — 実際の設計作業を行わない管理者などが、概要を知る研修。

練習 —

演習は理解度を確認するだけでなく、実際の開発作業への適用に自信を付けることが重要な目的である。したがって演習で取り上げる課題は、その目的に合った例となる

表1 構造化技法研修時間割例

日	時限	研修内容
1	1	構造化技法概要 構造化分析概要、DFD DFD作成手順、練習問題 DSD, 練習問題 SoftDA紹介
	2	
	3	
	4	
	5	
2	1	ERD, 練習問題 構造化分析演習
	2	
	3	
	4	
	5	
3	1	構造化設計概要、SC 構造化設計演習
	2	
	3	
	4	
	5	
4	1	グループで演習結果のまとめ 演習結果発表、討議 効果測定 まとめ、講評
	2	
	3	
	4	
	5	

1 時限は80分である。

DFD : データフロー図

DSD : データ構造図

ERD : エンティティリレーションシップ図

SC : モジュール構造図

ように問題を作成した。

演習は、基本的に個人で回答するが、複雑な問題はグループで回答するようにした。答案のチェックは、研修の中で講師が壇上で行い、学習者が互いの回答を参考にできるようにした。

講師の育成

多数の研修を実施するために、社内の研修部門でも研修を実施することとした。しかし研修部門には講師ができる者がいないため、講師の育成も必要となった。講師を育成するに、テキスト等の教材だけでなく、研修の手引きも作成した。

4 教育上の留意点、急所

本章では、構造化技法教育の改善で明確になった、教育上の留意点、急所をまとめる。

4.1 テキストの構成

テキストは学習において最も重要な位置を占める。テキストは、研修内容を定め、学習順序

に従って解説を記述する。さらに、学習の予習、復習への使用や、独学する者にも配慮しなければならない。

したがって、テキストの構成は学習の成否を左右する。構成に関して特に留意した点を以下に述べる。

【最初に全体の概要を説明する】

構造化技法では複数の図式を記述する。それらは互いに関連しており、他の図との関連を述べずに解説することは難しい。そこで、まず最初に構造化技法全体の概要、すなわち構造化分析設計作業で行うことを概説し、記述する図式を紹介することが必要である。技法の概要を示すことで、各図式の詳細を学習する目的付けができる。

また、主要な用語を最初に挙げることで、後の個別の解説でそれらの用語が使用できるため、解説が容易になる。

【用語の意味を明確にする】

構造化技法特有の用語を使用する場合は、その初出時に意味を解説する。特に、通常のソフトウェア開発で使用される用語を独自の意味で使用している場合には、最初に構造化技法での意味を明確にしなければならない。

例えば、「モジュール」とは構造化設計のモジュール構造図に記述するシステムの構成要素である。一方、通常のソフトウェア開発でもモジュールという用語を使用し、ソフトウェアを構成する要素を表すことがある。構造化技法での「モジュール」と通常の「モジュール」との差異を初出時に明確にし、テキストでの用語の使用に注意して読者に誤解を与えないように努めなければならない。

また、未定義の用語や図式を使用すると学習者の注意がそちらに向く。注意がそれないように用語や図式の使用に配慮する。

4.2 技法の解説

技法の解説は、基本概念と各図式の記述規則、分析設計手順で構成する。これらの中で、もっとも基礎となるのは図式の記述規則である。記述規則の学習は、図式の種類、使用する記号、

記号の構文則、図式の意味、図式相互の関係をなどを扱う。図式の記述規則の学習を通じて、基本概念を理解し、分析設計手順を習得する。

本節では、技法の解説の主たる部分を占める記述規則の解説に関する留意点を述べる。

【規則の背景を明確にする】

記述規則は、「こう書かなければならない」とか、「こう書いてはならない」という表現となる。しかし規則だけでは、実際のソフトウェア開発での適用は難しい場合がある。

例えば、「データフローには意味のある名前を付ける」という規則がある。これだけでは、「意味のある」とは何かが不明確である。「意味のある」を理解するには、データフローの概念、目的等の規則の背景を学習する必要がある。

規則を単に記述するだけでなく、その背景、ソフトウェア開発における意味を解説すると、規則の理解を深める。規則の背景を理解しないと、実際の適用は難しい。

【規則の適用基準を解説する】

技法の解説で「規則」と記述すると、その規則は厳守しなければならないと理解する学習者が多い。しかし、規則は常に絶対でなく、場合によって適度に使い分けが必要である。

例えば、「入力データフロー名と出力データフロー名を同一の名称にしない」という規則がある。これは、出力データは入力データを加工した結果であり、同一の内容、意味ではないため、名称で区別することを意図している。この規則を厳守すると、多数のデータフロー名が必要で、それぞれの名称が長くなる傾向となる。データフロー名が長くなると、全体として理解性が低下し、バグの原因となる。場合によってはこの規則を厳守しない方が、データフロー図の理解性や品質の向上になる。

したがって、規則の適用基準を解説し、学習者が適切に規則を適用できるようにしなければならない。しかし初心者は、規則を単に記憶しようとする。そうではなく、規則の本来の目的を解説し、学習者が実際の場面で判断できるようにする。

このために、構造化技法のテキストでは規則を表2のように3段階に分類して記述した。

表2 記述規則の分類

名称	適用基準
記述規則	図式を記述する際に必ず守りたい約束ごと
注意事項	図式を記述する際に気を付けたいこと
捕捉事項	図式を記述する際に知っておくと便利なこと

4.3 例題

例題は、技法の基本概念や図式の記述規則の理解を補助する。そのため、簡単で誰もが理解できる例題でなければならない。しかも、テキストの紙面は限られるため、例題はコンパクトにまとめなければならない。

本節では、例題の記述について留意点を述べる。

【簡単な例題は、現実感がない】

コンパクトにするため、解説の要点だけを含んだ、出来るだけ簡単な例題を採用することがある。例えば、単純な機能のシステムやその一部、あるいはソフトウェアとは無関係な作業などである。

しかし学習者は、実際のソフトウェア開発作業への構造化技法の適用を目的としている。実際の開発では記述しないような簡単な例題では、学習者が現実感を持つことができない。現実感を持つことで、より真剣に学習する意欲が学習者に起きる。したがって、例題はできるだけ実際の開発に近い内容や表現であることが望まれる。

【身近な例題が良いとは限らない】

理解性を上げるため、学習者の身近な例題を取り上げることがある。しかし、学習者に身近なため、非常に詳しく知っており、例題での表現に抜けや誤りを発見することがある。すると学習者は抜けや誤りに関心が行き、例題への信頼性が低下する。

例題は、本来の狙いに学習者が注目し、理解を深めることが重要である。他の部分に学習者の関心が移らない例題としなければならない。

【要求仕様を明記する】

例題で図式を示す時、単に例の図を記載するのではなく、その図が表現するシステムの要求仕様が文章で付記することが望ましい。特に、間違った記述をした例を示す場合、本来の要求仕様が明確でないと、間違いを理解できない。要求仕様を示すことで、学習者は要求仕様と対比してデータフロー図を読むことができ、記述方法等をより深く理解できる。

4.4 演習

演習は学習の理解度を確認することが目的である。演習で使用する課題は、例題を取り上げる時と同様の注意が必要である。さらに以下のような留意点がある。

【さまざまな答案の出る問題にしない】

例えば、要求仕様が文章で提示し、データフロー図を記述せよとした演習の場合、学習者数だけの回答ができる。研修実施時に、そのすべての回答をその場でチェックし、誤りを指摘することは難しい。また学習者の立場でも、他人のすべての回答を理解することは不可能であり、自らの回答の正誤にのみ関心が行く。理解度の低い学習者の場合は、全く回答できない場合がある。

そこで、回答がある範囲内に納まる設問とすることが望ましい。データフロー図を作成する演習では、ほとんど完成したデータフロー図を提示し、未完成な一部分を穴埋めするような問題がよい。理解度の低い学習者も、ある程度の回答ができ、理解度の向上になる。

【演習で設計を疑似体験させる】

学習者は、構造化技法を習得した後、実際のソフトウェア開発に適用する。研修では、ソフトウェア開発への構造化技法の適用に自信をつける必要がある。

そこで、ソフトウェアの設計作業を疑似体験させる演習を用意する。その演習は、与えられた要求仕様が順次分析し、設計書であるデータフロー図等を完成する課題である。筆者らはこの要求仕様の分析手順を開発し[6]、演習問題化した。演習問題は複数の課題から構成し、それ

を順次解くことで設計が進む。さらに前項で述べたように、回答がばらつかないような問題とすることで、学習者の多くが最後まで設計できる演習となる。

図3に演習問題の構成例を示す。この構成では、演習の始めに学習者に提示し、全体の見通しを明らかにして演習を進める。

【多少の間違いが出る演習にする。】

学習者全員が正解すると、回答の解説が単純になる。誤答があると、それをきっかけに解説を行い、学習者がより深く理解することができる。したがって、本節の最初の留意点とは多少矛盾するが、演習では回答がある程度ばらつき、間違いが出るようにすることも大切である。

4.5 CASE ツール

構造化技法をソフトウェア開発に適用する時、多くの開発担当はCASEツールを使用する。そのために、構造化技法に加えてSoftDAの使用法の研修を行うことがある。

【技法の教育と、ツールの教育は分離する】

技法とCASEツールの学習は分離して行うのが望ましい。なぜなら、分離しないと次のような問題が発生する。

- 構造化技法学習の途中でSoftDA使用法の学習が入ると、理解すべき項目が増加し、混乱する。
- SoftDAの学習にはUNIXの使用が不可欠だが、学習者の中でUNIXの経験度に差があり、同時に研修ができない。
- データフロー図の作成等の演習では、紙と鉛筆で記述するより、SoftDAの使用が効率的と理解されがちだが、初心者の場合、まず机上で記述し、完成した図式をSoftDAに投入する方が効率的である。

また、CASEツールによる効率的な開発作業を学習者に実感させるためにも、分離して行うのがよい。技法の教育によって、その基本概念や図式の記述法を理解し、演習により実際に図式を記述する体験があると、CASEツールの各種機能の有効性がより強く理解できる。

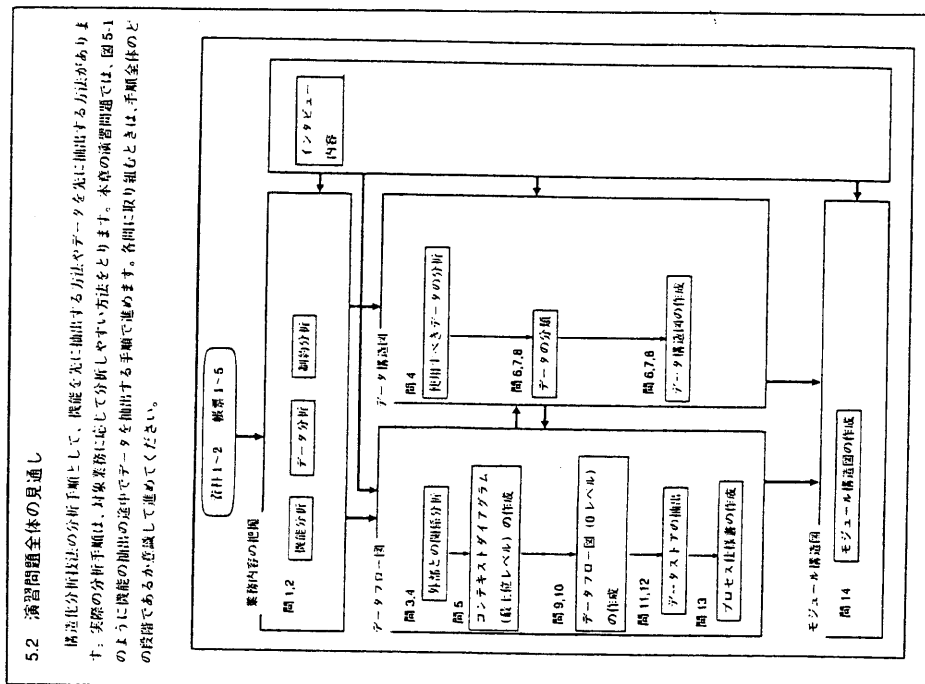


図3 設計を疑似体験する演習問題の構成

5 テキストの改善例

図4、図5にテキストの改善例を示す。この例の場合、改善項目は次のようになる。

- 例題に関する要求仕様を明記した。
- 規則に反する記述について詳しく解説した。
- この規則を学習する段階では未学習であるデータ構造図を排除した。

また、図3で示した構成の演習を実施したところ、学習者のほとんどが設計を最後までできた。

6 おわりに

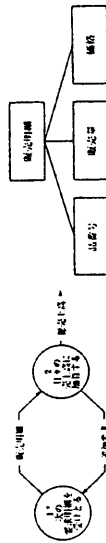
本報告では、筆者らの構造化技法教育の改善と、それにより得た留意点を述べた。留意点は、教育対象の学習者の特性により変化する。本報告で述べた留意点は、今後に行う新しい設計技法の教育で生かす予定である。

【謝辞】 本検討の機会を与えて頂いたNTTソフトウェア研究所黒田幸明主幹研究員、協力して頂いた各位に感謝する。

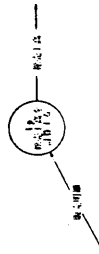
【参考文献】

- [1] Edward Yourdon: Managing the Structured Techniques, 1985 (邦訳、黒田純一郎、渡部研一訳：構造化手法によるソフトウェア開発、日経マグロウヒル社、1987)
- [2] Tom DeMarco: Structured Analysis and System Specification, 1979 (邦訳、高梨智弘、黒田純一郎訳：構造化分析とシステム仕様、日経マグロウヒル社、1986)
- [3] 磯田定宏、黒木宏明：統合化 CASE システム SoftDA の機能 — 上流と下流の統合化、コンピュータソフトウェア、Vol.10、No.28(1993)、PP.26-37
- [4] 斎 直人、田中 清、山本修一郎：構造化分析設計技法の教育と適用、利用者指向の情報システムシンポジウム、情報処理学会、1993
- [5] 斎 直人：構造化分析設計技法入門、電気通信協会、1996
- [6] 山本修一郎、斎 直人：系統的要求分析手法 DREM、ソフトウェア工学の基礎ワークショップ 94、日本ソフトウェア科学会、1994

⑤ 繰り返し処理をデータフロー図で記述しない
誤)



正)



プロセス仕様書

```

プロセス：総売上高を計算する
repeat for 全販売明細
begin
販売額 = 販売量 × 価格
総売上高 = 販売量 + 販売額
end
    
```

図 4 改善前の記述規則の解説例

データフロー図全体の規則 4
繰り返し処理をデータフロー図で記述しない

【例 16】

要求文 ある百貨店の総売上高を計算する。計算手順は、まず販売明細について1枚ずつ売上高計算する。次に、計算結果を日々の売上高に加算していく、それをある一定の期間ごとに総売上高とする。

上記のような要求文をデータフロー図にしてみましょう。
<悪い例>

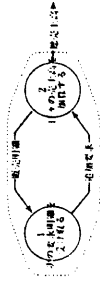


図 2.54 データフロー図として繰り返し処理になっている図

【解説】

図 2.54 では、繰り返し処理になっています。プロセス番号1「次の要求明細を受け取る」は、プロセスを制御するプロセスになっています。また、プロセス番号2「日々の売上高を加算する」は、実際に処理を行うプロセスになっており、データフロー「追加要求」は、制御にあるデータが出力されています。このようにプロセスを制御するようなプロセスはデータフローの1つのプロセスとして記述せず、その処理する内容をプロセス仕様書(「2.4 プロセス仕様書」p.97参照)で記述します(「2.3.6(4)データフロー図全体の規則3」p.66と同様)。したがって、図 2.55 のように記述します。

<良い例>

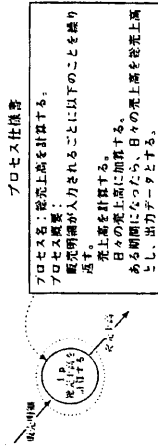


図 2.55 繰り返し処理をプロセス仕様書にしたデータフロー図

図 5 改善後の記述規則の解説例