ソフトウェアのドキュメンテーション*

角　田　盛　保**

1. まえがき

1960 年代の初期，すなわち，ソフトウェアの開発がまだ揺籃期にあった時代には，ドキュメンテーションは，プログラムに対する補助的な存在価値しかなく，プログラム・フローチャートと簡単な補足説明のみで用が足りた．しかし，その後，プロジェクトの規模の拡大とシステムの高度化・複雑化に伴って，その開発に占めるドキュメンテーション作業の比重が増大し，同時に，ドキュメンテーションの重要さと困難さがあらためて見直され始めた．

ドキュメンテーションを困難にしている原因の1つは，人間の性向としての文書化作業への抵抗感であろう．この抵抗感が最も強いのは，経験 1～4 年の比較的新しいプログラマーに多く，経験5年以上の熟練者になると，抵抗感は減少すると言われている．いずれにしても，ドキュメンテーションがソフトウェアの一部として重要な位置を占めている限り，この苦難を避けて通るわけには行かない．今後は，プログラマーやシステム・アナリストにも，ドキュメンテーション能力が必要資格の1つとして要求されてくるであろう．

また，ドキュメンテーションの困難さの他の原因は，開発担当者に対して，ドキュメンテーションに関する作業手順 (Procedures) や標準 (Standards) が十分に与えられていないことである．ある外国のレポートによると，適切な手順や標準を設定し，これに従うようプログラマーに義務づけることによって，上記のようなドキュメンテーションに対する忌避感の克服に役立てることができると言われる．円滑なソフトウェア開発のためには，プログラミングのみならずドキュメンテーションに関しても，それぞれのソフトウェアの目的に合った手順や標準を確立することが必要であり，また，実際の開発作業がそれらに従って行なわれているかどうかを常に見定めておく必要がある．

本文では，通常のソフトウェア開発において最小限必要と考えられるドキュメンテーションについて，開発の各フェーズごとに，解説を進めて行きたい．

ただし，以下の記述は，筆者のソフトウェア開発の経験および関連雑誌・著書等に示された事例をもとにして，いわば帰納的に行なった一般的，基本的なドキュメンテーションの説明であって，筆者が現在行なっているドキュメンテーション作業をそのまま述べたものではないということをお断りしておきたい．

2. ドキュメンテーションの目的

ドキュメンテーションとは，書類または書類中に記録されている情報を作成したり，編集したり，配布したりする作業である．また，このような活動の結果できあがったもの（ドキュメント）を意味する場合もある．ソフトウェアのドキュメンテーションというときは，通常，プログラムに関する技術文書またはそれの作成を指すが，広義には，ソフトウェア開発に関連する文書類，例えば，開発管理や進捗管理のためのドキュメント，プログラム・テストや教育・訓練のためのガイドブック等まで含めて考えることができる．しかし，本文では，技術文書の作成に重点を置いて解説する．

ドキュメンテーションの目的として，

- プログラミングのための設計図を提供すること
- 完成したプログラムの使用や保守のための資料を提供すること

の2つをあげることができる．ドキュメンテーション結果の良否は，これらの目的をどれだけミートできたかによって決まると言える．

3. ドキュメンテーションの種類

ソフトウェア開発中に作成される技術文書は，大別して，プログラミング目的説明書（以下，目的説明書と略す），仕様書，およびマニュアルの3種に分類することができる．目的説明書は，システム・デザインへ

---

* Software Documentation, by Moriyasu Tsunoda (Program Product Center Publications, IBM Japan)

** 日本アイ・ビー・エム株式会社 DP ビジネス計画・プログラム開発センター

のインプットとなるもので，開発すべきソフトウェアに関する全体的な記述が含まれる．仕様書は，システム・デザインの結果をまとめたもので，プログラミングのための設計図と言うべきものである．これには，アプリケーション仕様書とプログラム仕様書とがある．場合によっては，オペレーション仕様書を作成することもある．マニュアルは，プログラム開発中または開発が終了したあとで作成され，完成したプログラムの使用・修正・更新等に役立てるものである．マニュアル作成のための主な情報源は仕様書である．

4. ソフトウェア開発とドキュメンテーション

ソフトウェア開発の段階の分け方にはいくつかの方法があるが，ここでは，プランニング，デザイン，インプリメンテーション，マニュアル作成，および総合テストの5つのフェーズに区分して，各フェーズでどのようなドキュメンテーションが行なわれるかを説明する．ドキュメンテーションは，これらの開発過程を通して継続して遂行される作業であり，しかも，開発が終了したあとでも，プログラムの生命が存続する限り，ドキュメンテーション（主としてマニュアル）もまたメンテナンスされなければならない．

4.1 プランニング

このフェーズの目的は，ニーズの分析と開発すべきソフトウェアの輪郭を描き出すことである．開発の対象となるソフトウェアは，導入ハードウェアをサポートし，同時に，ニーズを満足するものでなければならない．このフェーズで作成されるドキュメントは目的説明書である．その内容はおおよそ次のようなものである．

1) プログラムの機能
   - ● 機能上の特性（プログラムの機能，プログラムを構成するモジュールの機能，各モジュール間の関連等）
   - ● 機械構成
   - ● 使用するプログラム（コンパイラー，アセンブラー，ユーティリティ・プログラム等）
2) プログラムの性能
   - ● 各プログラム・モジュールの最大・最小コア・サイズ，ステートメントの数
   - ● 入出力装置とデータ・ベース，ファイル構成
   - ● プログラム・ランの概略推定時間
   - ● コントロール・プログラムの占有コア・サイズおよび占有時間
3) 作成すべきドキュメントの種類，内容，程度等に関する記述
4) 教育・訓練に関する記述
5) 開発スケジュール
6) 必要資源（マンパワー，スキル，機械使用時間等）に関する記述

この目的説明書は，テクニカルな記述を行なうというよりも，むしろ，開発のための資源利用の可能性と開発実行の可能性について詳細に検討した結果作成されるものであり，この意味で，開発直接担当者が作成する仕様書とかマニュアルなどとは多少性格を異にすると言えよう．

目的説明書を作成する場合に困難なことは，ニーズを満たすために必要なプログラムの適用範囲をどの程度に決定するかということと，実行可能な開発スケジュールおよびそれに必要な資源をどの程度に見積るかということである．中でも，スケジュールとコストの予測は特に困難である．

予測に影響を与える要素として，次のものをあげることができる．

- ● ジョブの困難性（プログラムの複雑さ，サブ・プログラムの数，データ・フォーマットの種類や数）
- ● ハードウェア（コンピュータのタイプ，ターンアラウンド・タイム，信頼性）
- ● プログラミング・システム（利用可能期日，使用の難易度，信頼性）
- ● 開発担当者（経験とスキルの程度，人数，情報伝達の難易）
- ● その他（開発プロジェクトの優先度，プログラムを使用する部門の程度）

プロジェクト開発の失敗は，技術上の問題によるよりも，スケジュールやコストについての予測の間違いに起因する場合が多い．すなわち，予測以上のコストの増大とスケジュールの遅延である．極度に楽観的または非観的な予測は許されないし，もしこれを行なえば，プロジェクト開発運営にとって逆に大きな足枷になる．一般的には，予測の段階で，コストとスケジュールに関しては，10%～15% のコンテインジェンシーを見込んで見積りを立てておくことが必要であろう．

4.2 デザイン

デザイン・フェーズの目的は，プログラミングのた

めの仕様を決定することである．その結果は仕様書にまとめられる．デザイン・フェーズは，基本デザインと詳細デザインの２つの段階に分けられる．

基本デザインの段階では，プランニング・フェーズで設定されたプログラミング目標にそって，ニーズを最も良く満たすプログラムの仕様を決定し，その開発に関する技術上・資源上の実行可能性をさらに深く検討する．基本デザインの段階で作成されるドキュメントはアプリケーション仕様書である．これの主たる担当者はシステム・アナリストである．アプリケーション仕様書と目的説明書の内容に矛盾がないかどうかが検討され，必要であれば，目的説明書が修正される．この場合には当然マネジメントの承認を必要とする．

アプリケーション仕様書は，開発しようとしているプログラムを機能上の観点から詳述するものである．このドキュメントは，次の段階で作成されるプログラム仕様書へのインプットとなり，また，プログラム完成時に作成されるマニュアル，すなわち，アプリケーション概説書とプログラム解説書の基礎になるものである．アプリケーション仕様書に盛り込まれるのは，プログラムの機能や性能の説明だけであって，その実施（インプリメント）の手順や方法については，次の段階で作成されるプログラム仕様書に書かれる．

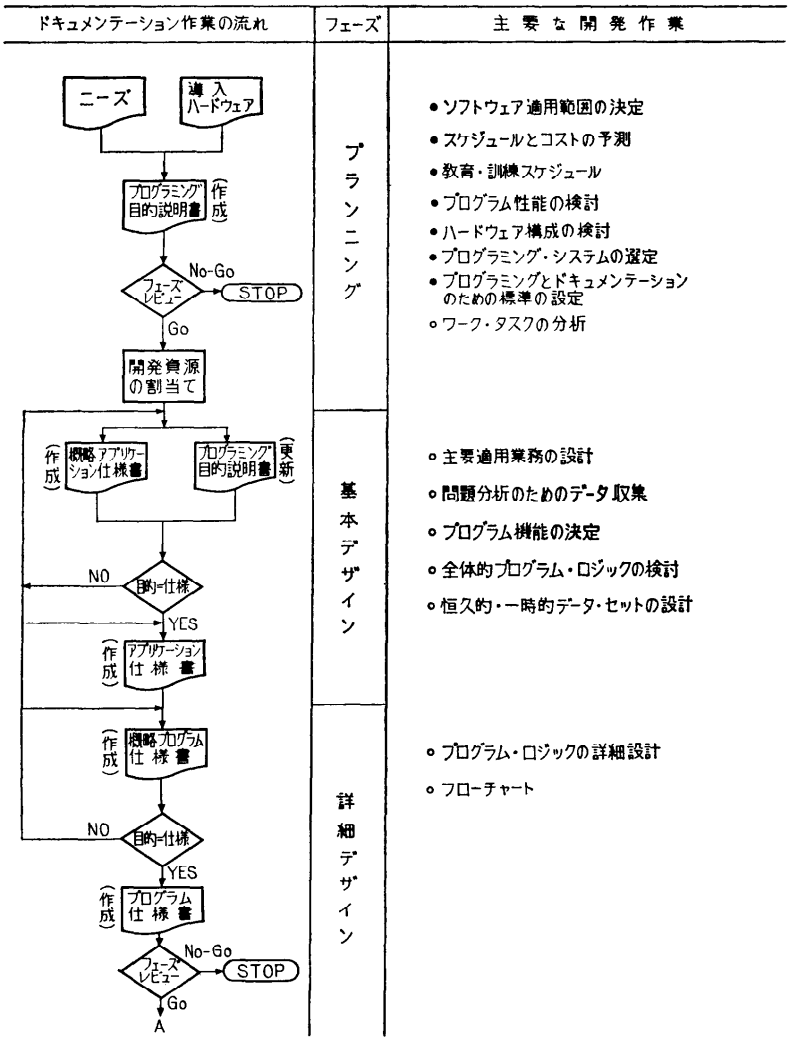
アプリケーション仕様書の主な内容は次のとおりである．

- ●プログラムの概要
  プログラムの目的，適用範囲，機能等を説明する．
- ●処理方法
  まず，プログラムを構成する各モジュールやサブルーチンについて，その機能，関連，入出力情報等を記述する．プログラムの主要な機能は，概略システム・フローチャートで表現する．次に，必要に応じて，適用する理論や方式，計算方法，必要精度，使用上の制限・範囲等を記述する．
- ●入出力データ
  プログラム処理のために必要とされるファイルやレコードについて，内容と役割を説明する．ファイルの仕様では，ファイルの媒体，編成方法，さらに，入力・出力の区別，ワーク・ファイルか否かの区別，また，レコードの仕様では，レコードの名称，内容，形式，大きさ，構造(固定長，可変長，ブロッキング・ファクター等)を記述する．このほか，必要に応じて，入出力データに関する例外処理の方法や制限事項も記載される．
- ●プログラム導入・使用上の考慮点
  完成したプログラムの導入および使用に関する必要事項を述べる．例えば，準備すべき入出力ファイルの種類，数量，内容，印刷フォームのデザイン，導入・使用の手順，導入準備に要する期間，オペレーティング・システムやジョブ・コントロール・カードの使用上の注意点等．
- ●性能・効率
  プログラムの性能や効率を記述する．プログラムが未完成の時点で，タイミングやスループットの測定を行なうことは困難であり，かつ，誤差も大きいが，しかし，プログラムが達成すべき性能目標値は，少くともこの段階でも定めておく必要がある．性能を上げるための特別な手法があればそれについての説明，また，特に考慮すべき制限・禁止事項があればそれについての記述も行なう．
- ●プログラミング・システム
  使用すべきオペレーティング・システムやその他のプログラム，使用すべき言語(FORTRAN, COBOL 等)，アクセス・メソッド，コーティリティ・プログラム等をリスト・アップする．
- ●機械構成
  導入・使用すべき機械の構成を述べる．

次に，詳細デザインの段階に入ると，このアプリケーション仕様書に基づいて，プログラム仕様書が作成される．プログラム仕様書は，アプリケーション仕様書で決定されたプログラムの機能をどのようにして処理するかについて，その手順と方法をプログラム・ロジックの面から述べたものである．プログラムの論理構造を明確にしたプログラマーのためのコーディング指示書である．

プログラム仕様書の内容は次のとおりである．

- ●プログラムのロジック
  アプリケーション仕様書に基づき，プログラムの構成や構造を詳述する．記述の順序としては，プログラムの主要な部分に関する全体的・概略的な説明から，各モジュールやフェーズについての詳しい説明書へと進んで行く．アプリケーション仕様書に規定されたものはすべて記述されなければならない．ロジックの説明はプログ



ラミングの重要なベースとなるため，プログラマーがこれによってコーディングが行なえるよう十分な詳細度で述べなければならない．

● フローチャート

上記のナラティブなロジックの説明をさらにフローチャートで表現する．ここでは，全体的なシステム設計についての概略フローチャートとプログラム・ロジックについての詳細なフローチャートが書かれる．必要であれば，このフローチャートに補足説明を加えることもある．

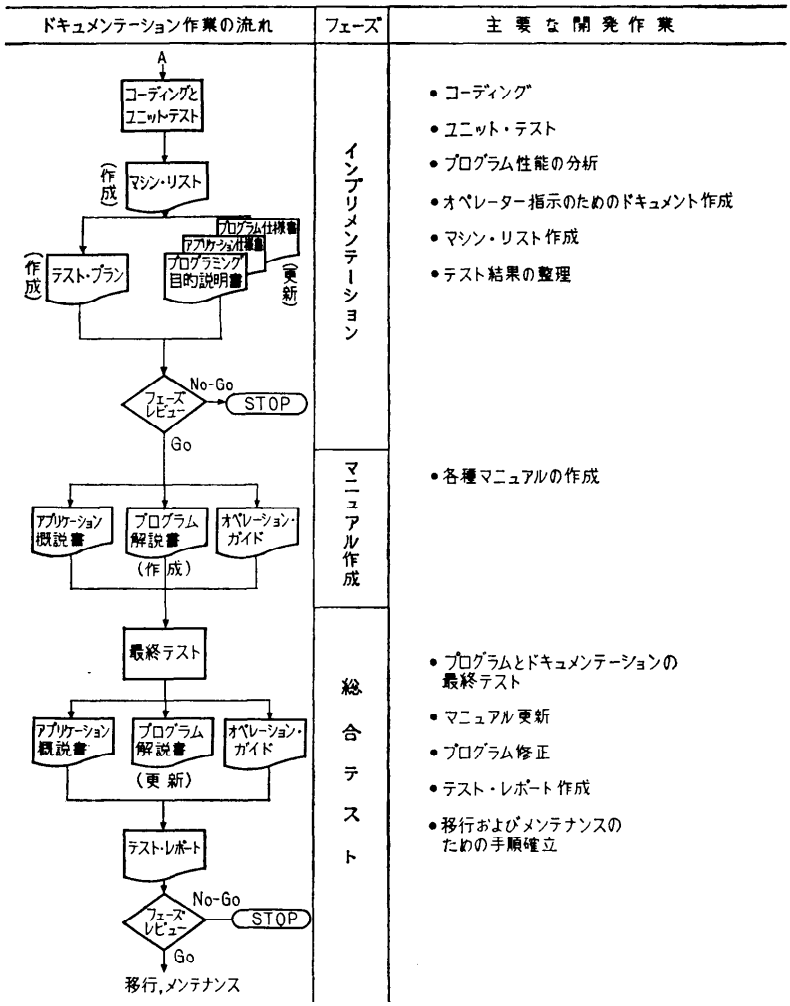
● 主・補助記憶装置内の割当て

プログラムが使用する主記憶装置と補助記憶装置について割当てを行なう．主記憶装置に関しては，オペレーティング・システム，入出力ルーチン等の恒久的なものと一時的なもの，また，補助記憶装置に関しては，ディスクやテープ上のシステム占有部分，大きさ，コントロール・プログラムが使用するワーク・スペースの大きさ等を記述する．

● その他

以上のほかに，判断表(デシジョン・テーブル)を載せたり，また，特別なプログラミングの手法があれば，それについての説明を加えることもある．

ソフトウェア開発の過程で，最も困難な作業の1つは，調査・分析の結果を仕様としてまとめあげること



である．仕様書の作成には，数人，場合によっては数十人という多数の開発担当者が関係する．仕様書の作成で常に問題になるのは，「何を書くか」よりも，「どの程度の詳細度で書くか」ということである．不適切な仕様書ができあがるのは，通常，記述の内容が各担当者によって，あるいは，同一人の場合でも，自分の得意とする部分や，興味のある部分とそうでない部分とで，詳細度がまちまちで平均化されていないことに原因があることが多い．プロジェクト・リーダーは，プログラムの大きさや性格に応じて，仕様書作成についての細かな手順や標準をつくり，さらに，これらが忠実に守られているかどうかを常に監督して行くことが必要である．

4.3　インプリメンテーション

インプリメンテーション・フェーズでは，アプリケーション仕様書とプログラム仕様書に基づいて，これらをコンピュータ言語へ翻訳する作業(コーディング)と，できあがったプログラムをテストする作業が行なわれる．このフェーズにおけるテストは，いわゆるユニット・テストであり，このあとのフェーズで行なわれる総合テストとは異る．インプリメンテーションの主役はプログラマーである．

このフェーズにおける，ドキュメンテーション作業は，仕様書の更新とテスト・プランの作成である．良くできた仕様書でも，プログラミングの進行に従って多くの部分で修正や変更を余儀なくされる．しかし，これらの修正・変更は，あくまでも，初期の段階で定められたプログラミングの目標や仕様の範囲を越えるものであってはならない．このことは，与えられた資源とスケジュールを守って開発を進めて行く上で重要

なことである．結果論的な言い方になるが，最初の仕様書とプログラム完成時での仕様書とを比較して，修正個所が少なければ少いほど，それだけ良くできた仕様書であったと言うことができ，開発スケジュールも遅延が小さくて済むと言える．

テスト・プランは，総合テスト・フェーズへのインプットとなるもので，内容としては，プログラムについての机上検査や機械による検査の結果を記録したものである．例えば，テスト・ランの回数，各テスト・ランにおけるエラーおよび修正個所の説明，テスト・データ，ジョブ・コントロール・カード，処理手順と処理時間，入出力装置，主記憶装置内の割当て，マシン・リスト，総合テストのための注意点等についての説明などが含まれる．

4.4 マニュアル作成

プログラミングが完了するとマニュアルの作成にとりかかる．本文では，説明の便宜上，マニュアル作成のフェーズを独立させたが，実際は，プロググラミングやユニット・テストと並行して行なわれる作業であるため，マニュアル作成はインプリメンテーション・フェーズに含められる．

マニュアルのタイプは，それに盛り込む情報の違いによって，次の4種に分類することができる．

1) 紹介のためのマニュアル

開発されたプログラムについての全体的な説明を行なうもので，主として教育のための情報を含む．主題については広く説明するが，細かな部分までは立ち入らない．対象読者は一応のコンピュータ知識は持っているが，アプリケーションについては精通していない．このマニュアルに書かれた内容は，プログラムの変更・修正によって影響を受けることはない．

2) ハウ・ツー (How-to) マニュアル

このマニュアルは，プログラムの使用についての案内書である．作成のためのソースは主として仕様書である．ここでは，プログラムの使用方法，手順，テクニック等が説明され，また，説明を助けるために，例示やケース・スタディなどが追加される．

3) 参照のためのマニュアル

プログラムに関する詳細な説明が行なわれる．必要な情報が容易に見出せるように，説明の順序，体裁，構成等が整理されている．読者はプログラムについて精通した知識を持っていることが前提となる．このマニュアルは，プログラムの変更・修正に影響を受ける．

4) ロジック説明のためのマニュアル

プログラムがどのようにして処理されるかについて，その論理構造の面から詳細に説明される．プログラムのエラーの原因を探し出し，その対策が立てられるよう十分の情報が盛られなければならない．プログラムの変更・修正によって受ける影響は参照のためのマニュアルよりも大きい．従って，メンテナンスの頻度も高くなる．

この分類に従ってソフトウェアとして最小限必要なマニュアルをあげれば，アプリケーション概説書，オペレーション・ガイド，プログラム解説書，およびロジック・マニュアルの4種になる．これらのマニュアルは，アプリケーション仕様書，プログラム仕様書，テスト・プラン，その他プログラムに関係するすべてのドキュメントの総まとめとして作成されるものである．

これらのマニュアルのうち，アプリケーション概説書とプログラム解説書は，それぞれ，アプリケーション仕様書とプログラム仕様書を純化(Refine) したものであって，これらのマニュアルに記述する項目は，最終仕様書のそれとほとんど同じである．ロジック・マニュアルは，プログラム仕様書を母体にして作成されるが，このマニュアルの作成はプログラムの性格に依存していて，場合によっては，プログラム解説書に含めることも可能である．

オペレーション・ガイドは，プログラムを実行させるために必要な準備事項や操作手順について述べたものである．主な内容としては次のようなものがある．

- 必要準備事項

  プログラムを走らせるために必要な準備事項について，ジョブ，さらに，ジョブ・ステップごとに記述する．例えば，オペレーションに必要なファイル，ジョブ・コントロール・カード，入出力装置の指定，印刷フォーム，データ・カード等．

- 操作手順

  プログラム実行に必要な操作手順を記述する．アプリケーション・プログラムやオペレーティング・システムから出されるメッセージの意味と応答方法，プログラム停止や待ちループの意味と必要な措置，チェックポイント／リスタートの手順，入出力データの処置，その他必要な

周辺オペレーションの説明と手順，例外処理の方法・手順等について記述する．

仕様書とマニュアルの違いは，前者が主としてプログラミングのためのドキュメントであるのに対して，後者が完成したプログラムを解説するためのものであるということである．また，マニュアルの場合は，読者のレベルや範囲が仕様書に比較して多様性がある．この意味で，マニュアルは仕様書よりもさらに読み易く，理解し易く，かつ変更・修正が容易にできるよう，体裁，構成，表現などに十分の考慮を払う必要がある．簡潔でしかも正確な記述が要求される．本格的なソフトウェアの開発，例えば，販売を目的に開発する場合とか，タイム・シェアリングのように不特定多数のユーザーを対象に開発を行なう場合には，専門のマニュアル作成の担当者が用意されていて，プログラマーやシステム・アナリストによって作成された各種仕様書またはマニュアル・ドラフトに基づいて，マニュアルが作成・編集される．

4.5 総合テスト

総合テストのフェーズは，プログラマーによって作成されたプログラムを，各種ドキュメントに基づいて，プログラマーとは全く別の立場から総合的にテストする段階である．総合テストが完了してはじめてソフトウェア開発が終了したことになり，このあとは，移行，メンテナンスのフェーズに入る．

総合テストのフェーズは，すべてのプログラミング目標と仕様がプログラムとマニュアルに，もれなく，正確に反映されているかどうかを確認するためのフェーズである．このフェーズで新たに作成される技術文書はない．作成されたすべてのマニュアルは，それぞれの目的を十分に果たしているかどうかが検査され，将来のプログラムの使用・修正・メンテナンス等に役立ち得るように仕上げられる．

5. あ と が き

以上，ソフトウェアとして最小限必要と考えられるドキュメンテーションについて述べた．もちろん，これらのほかにも，開発するソフトウェアの目的・性格に応じて種々のドキュメントやマニュアルが作成されるが，これらについての説明は多少一般性を欠くため割愛させていただいた．

筆者は開発フェーズを便宜上5つ（移行とメンテナンスを含めれば7つ）に分けて説明したが，フェーズの分け方にはバリエーションがあり，また，部分的にオーバラップするフェーズもあったりして，実際上のフェーズの区切りは難かしい．開発コストの把え易さから区分したり，アクティビティあるいはフェーズ・レビューの観点から区分したり，いろいろの方法が可能であるが，各開発部門にあっては，それぞれの開発体制およびプロジェクトの特殊性に従って，開発目標を最も良くミートするようなフェーズを設定する必要があろう．

ドキュメント作成のためのプログラムが現在いくつか利用可能であるが，これらはすべて欧米文によるドキュメンテーション用に作成されている．将来日本語によるドキュメンテーションのためのプログラムが開発されれば，ドキュメントの更新やフォーマッティングのための強力な Aid として，わが国でも盛んに利用されるようになるであろう．しかし，ドキュメンテーションの良否が，プログラムや機械ではなくて，あくまでもプログラマーやシステム・アナリスたちのドキュメンテーションに対する重要性の認識と努力の程度によって決定されることは言うまでもない．

参 考 文 献

1) 日本事務能率協会：“MIS ハンドブック”，pp. 664～667，昭 44.
2) F. M. Trapnell：“A Systematic Approach to the Development of System Prgrams,” SJCC (1969).
3) M. M. Gotterer：“Management of Computer Programmers,” SJCC (1969).
4) R. H. Kay：“The Management and Organization of Large Scale Software Development Projects,” SJCC (1969).
5) IBM：“データ処理規準書作成の手引き”，(NC 20-1670-1).
6) IBM：“電子計算組織の経営コントロール”，(NF 20-0006).

(昭和47年5月15日受付)