U.D.C. [681.322.004.14:657/659]:681.32.06.002

構造化設計技法の事務管理部門における適用

Application of Structured Design Method to Business Systems

内山俊一* *Syun'ichi Uchiyama*
小沢忠夫* *Tadao Ozawa*
西　雅幸** *Masayuki Nishi*

現在，システム開発の上で各種の技法が発表されている。日立製作所の事務管理部門でも，日立式ドキュメントの確立とソフトウェア生産性向上を目指し，新しい技法の導入を検討した。

新技法の導入に当たっては委員会を設置して，公表されている数多くの技法のうち，社内の各事業所の実務に適用可能な技法を取捨選択した。その結果，構造化設計技法を主体として，図による表現方法及びストラクチャードプログラミング手法を取り入れた，「システム仕様書・プログラム仕様書・プログラム作成規準」を取りまとめ実施している。また実例編も設け，だれにでも構造化設計技法に基づくシステム開発ができるように，実務への適用面を重視した。

本論文では，この作成規準の内容と厚生年金基金オンライン開発に適用した結果について述べる。

1 緒　　言

事務管理部門では，業務量の増加と現行システムの変化に迅速に対処するために，ソフトウェアの生産性向上を図ることが重要な課題となっている。現在多くの技法が紹介されているが，そのうちから最も一般的に利用され実績の挙がっている，構造化設計とSP(Structured Programming)の二つの技法を採用し，日立製作所事務管理部門の実務に適したものを選定して，日立の規準を作成することとなった。更に，従来各事業所で独自に行なっていた仕様書の記述方法を，全社的に統一することにした。

2 構造化設計技法の導入とその背景

従来，日立製作所の事務管理部門では，処理を中心としたシステム開発を行なっていた。しかし，ソフトウェアをライフサイクルとしてとらえる場合，その生産性向上のためには開発とともに保守・拡張も重要となる。すなわち，(1)システムで実施している機能(何をやっているか)の明確化，(2)システムの保守・拡張に対処できる適切な機能分割，が必要となる。

この条件を満たす技法を模索して，昭和53年4月に委員会(運営合理化分科会)を発足させ，検討を開始した。

検討に当たっては，構造化設計技法とSPの二つの技法を選定し，モデルを用いてシステム開発を実作業と同様に行なった。モデルには日立製作所の実情にふさわしく生産管理を選び，約3箇月にわたりシステム設計以降プログラム作成までを実施した。また実施の途中で発生した問題点の解決には，他の技法も参照した。

この結果を基に試案を作成し，各事業所で試行した。

最終的にこれら試行結果を取り入れ，「システム仕様書・プログラム仕様書・プログラム作成規準(以下，作成規準と言う。)」を取りまとめ，昭和54年4月から実施してきた。

技法としては，システムの目的(機能)を達成するために，トップダウンにより機能を中心としてシステムを制御可能な大きさに分割し，その段階ごとに十分に検討を加えながら開発してゆく「構造化設計技法」を導入している。

記述に当たっては，図を多く用いる方法，すなわち，機能階層図，機能説明図及び関連図を用いている。また，プログラムはSPを導入している。

3 作成規準の概要

3.1 構　　成

作成規準は，規準編と実務編で構成される。規準編では設計手法と記述方法を規定し，実務編では実務に適用する際の参考事例として試行結果を取りまとめた。

参考事例は，前述のモデルシステムで試みたときの経過をまとめたもので，システム設計からプログラム作成に至る過程を一貫して説明した。特に規準編だけでは理解が困難な機能分割や記述方法についても具体的に説明している。

3.2 対象範囲

事務管理部門での開発体制の実態を考慮し，開発手順をシステム設計，プログラム設計及びプログラムに大別し，その範囲を対象とした。

ニーズ分析及びテストも重要であるが，今回作成した規準範囲の定着化をみたのち検討することとして，今回は対象外とした。

3.3 様　　式

システム開発には各種ドキュメントが必要となるが，本作成規準の基本となる4種類の様式について以下に説明する。

(1) 機能階層図

機能を段階的に細分化し，階層構造で表現したもの。

(2) 機能説明図

機能階層図の個々の機能を，入力，処理，出力の関係で記述したもの。

(3) プログラム関連図

プログラムのつながりを，システムフローチャートの形式で記述し，機能説明を併記したもの。

(4) セクション関連図

プログラム内の機能のつながりを，フローチャートの形式で記述し，階層構造も表現する。

各々の概要を**図1**，**3**に示す。

* 日立製作所事務管理部　** 日立製作所ソフトウェア工場

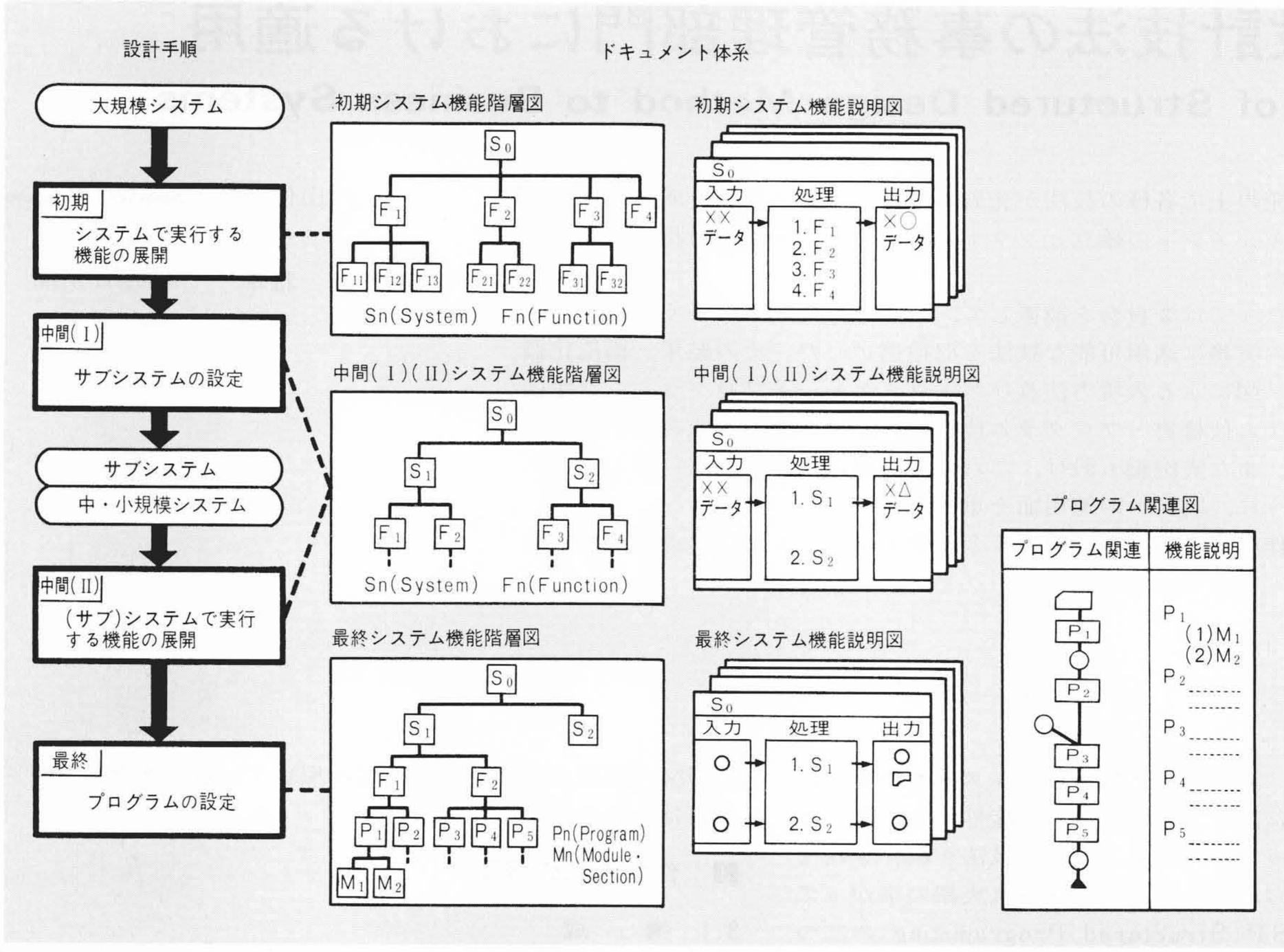


図1 システム設計手順とドキュメント体系 システム設計の基本的な手順と作成するドキュメントの概要をまとめた。最初は必要な機能を摘出し，最終的に実行可能な具体的なものへまとめてゆく。

3.4 システム設計

システムを構成する基本となるプログラムの設定を行なう。

(1) 設計手順

設計手順とドキュメント体系を**図1**に示す。設計手順は初期，中間(I)(II)及び最終に大別する。

初期段階では，既に設定されているシステムの主要機能を，サブシステムが設定可能なレベルまで段階的に分割し階層図を作る。次に，分割された個々の機能を入力，処理，出力の関係で説明し，機能の検証と入出力の設定を行なう。

中間(I)段階で機能の実行順序，入出力データのタイミング，ファイル構成など，システムを構成する外部要因を考慮しサブシステムに再編成する。

中間(II)段階では，サブシステム単位に初期と同様の方法でプログラムレベルまで機能を分割する。中・小規模システムの場合は，この段階から始める。

最終段階では，個々の機能の実行順序，システム資源などを詳細に検討し，プログラムの主要機能レベルまで分割してプログラムを設定し，最終機能階層図を作成する。この階層図を基に個々の機能を入出力に対応させて説明し，最終機能説明図を作成する。

本作成規準では，この段階にプログラム関連図を追加した。

この関連図は最終機能説明図を基に作成し，システム運営管理上必要な処理の流れ，入出力データ類の関連及び個々のプログラムの機能説明を記述する。

これら最終のドキュメントをシステム仕様書とする。

(2) 機能分割

構造化設計技法でも，適切に機能を分割するためには経験が必要となる。本作成規準では，実務に容易に適用するために**図2**に示す目安を設けた。

事務計算システムの特徴は，入出力データ及びマスタデータの扱い方，すなわちデータ処理の問題で，入出力データとそれに伴う処理条件を分析することにより，機能分割をスムーズに行なうことができる。

3.5 プログラム設計

本作成規準では，プログラム設計をシステム開発の一段階として明確に区分した。プログラム設計で，システムを構成する個々のプログラムの仕様を詳細に記述する。これにより，システムの処理構造を決定するとともに，プログラム作成を容易にする。

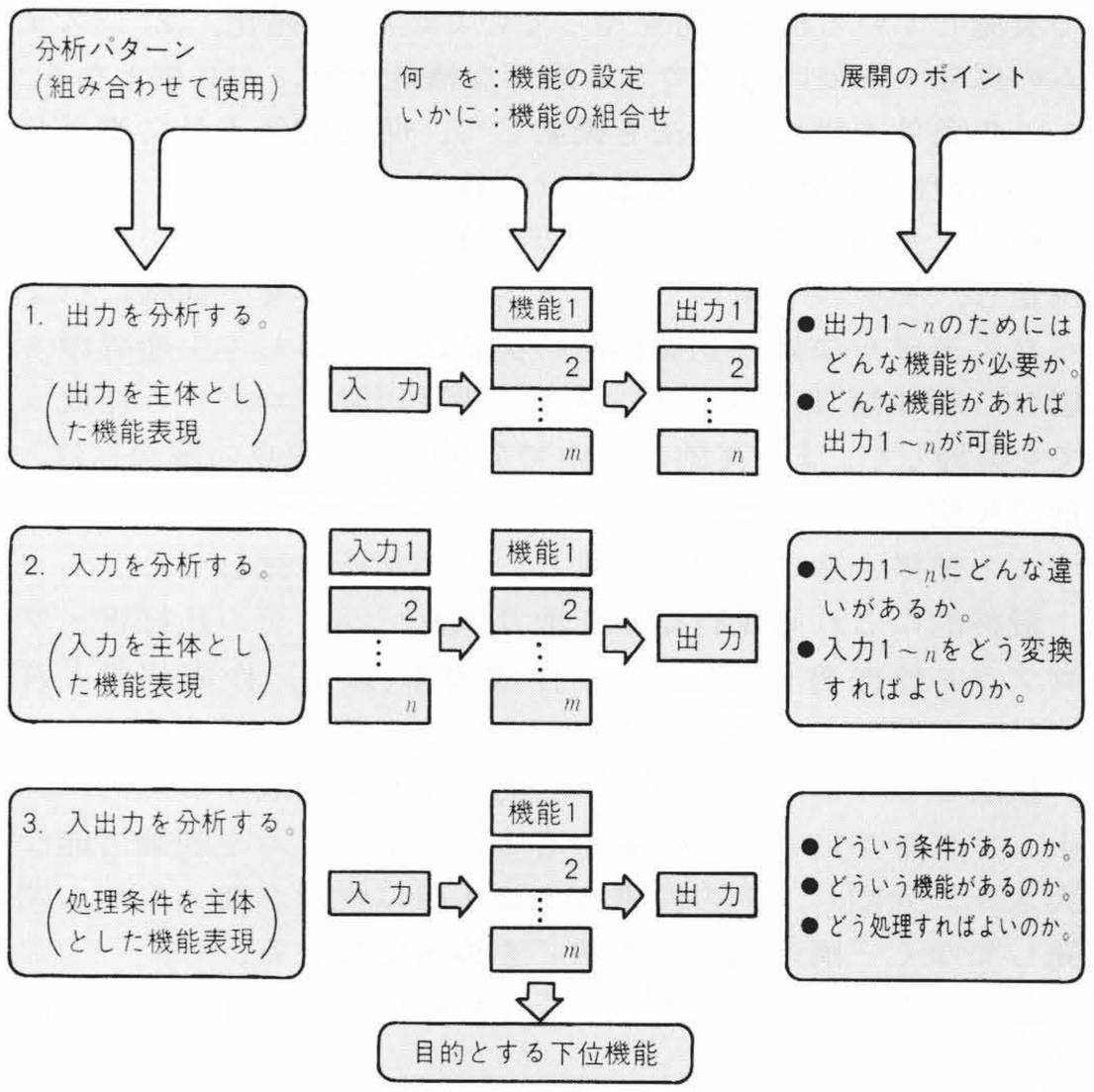


図2 機能展開と表現のための一方法 実務に容易に適用するための一方法として，入出力と処理条件の関係に着目した。

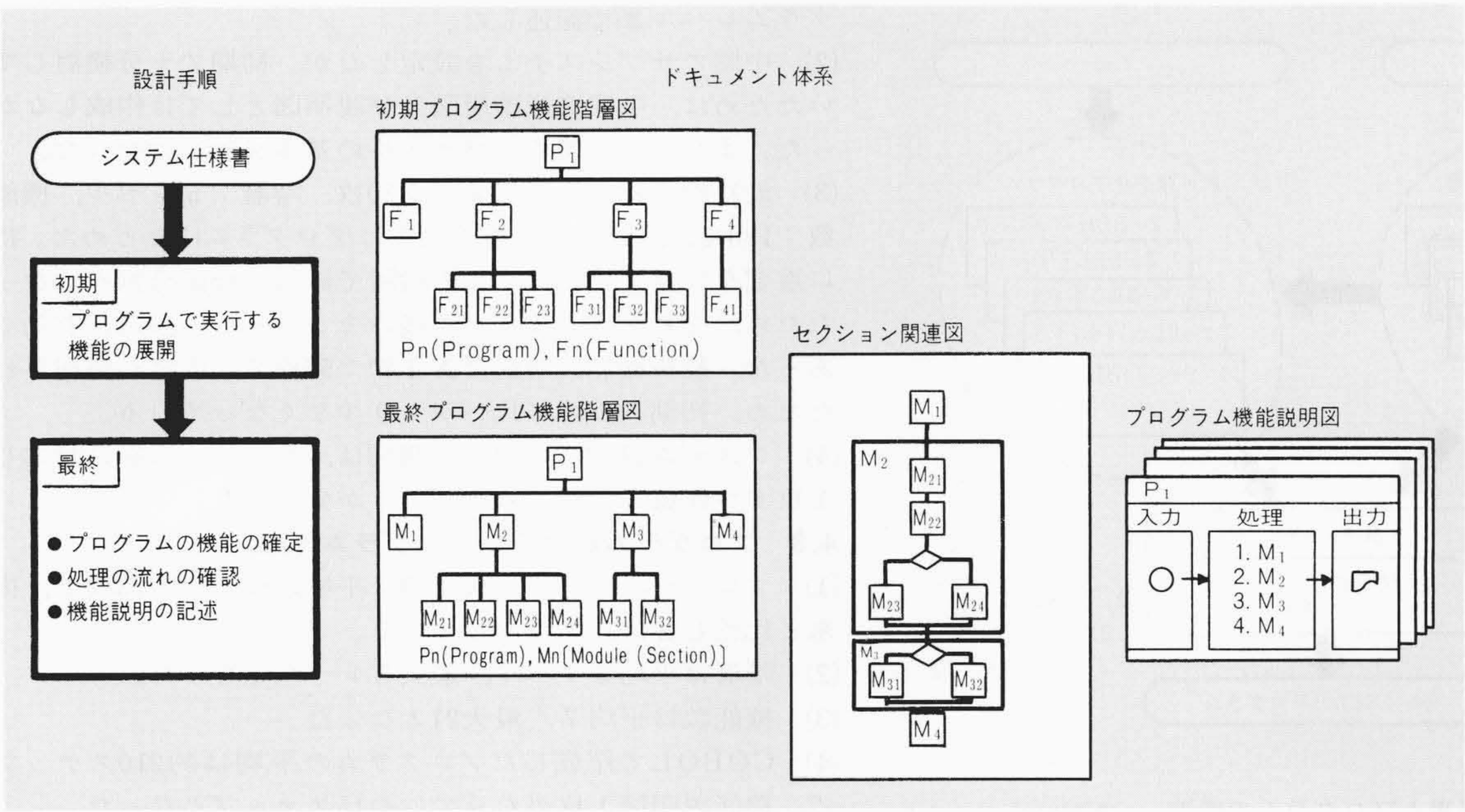


図3　プログラム設計手順とドキュメント体系　プログラム設計の基本的な手順とドキュメントの概要をまとめた。システムと比べ規模が小さいため，中間は省略した。

(1)　設計手順

設計手順とドキュメント体系を**図3**に示す。基本的にはシステム設計と同じ方法で行なう。

初期段階では機能を中心とした分割を行ない，最終的に処理を考慮して再編成する。最終機能階層図を基にセクションレベルの関連図を作成し，基本的な処理の流れを確認する。

このようにして確定した個々の機能について，入出力データを対応させて下位の機能の組合せ及び処理手続をプログラミングが可能なレベルまで詳細に記述するとともに，何をやっているのか(機能の表現)もよく分かるように記述する。また，この記述がそのままコーディングに対応するので，ＳＰができるように記述する。

これらの点を考慮しておけば，仕様書でプログラムを管理することが可能になり，開発後の保守・拡張時に適切に対処することができる。

(2)　機能分割

プログラム設計では，処理を考慮した機能の分割と再編成が重要になる。特に，本作成規準にのっとり作成したプログラム仕様書は，そのままプログラム作成につながるため，論理的な矛盾のないように十分に検証する必要がある。

分割の一方法を**図4**に示す。個々の機能は，50ステップ程度，又は機能説明図1枚を目安とした。

3．6 プログラム作成

構造化設計技法に基づき作成したプログラム仕様書の主旨を生かし，プログラムをより分かりやすいものにするために，ＳＰ手法を導入した。

すなわち，**図5**に示すように，プログラム仕様書に記述された各々の機能をCOBOLの場合はセクションとしてコーディングすることにより，プログラムを構造化するとともに，仕様書とプログラムの構造的な一致を図り，プログラムの信頼性・正確性を高める。

更に，プログラムを読みやすくするために，コメントの挿入，字ずらし及び桁合せも行なうこととした。

また，ＳＰの導入に当たり最も問題となったＧＯ　ＴＯ命令などによる飛び先の変更は，セクション内に限り使用を認め，仕様書にも明記することとした。

これにより，従来はあいまいになりがちであったプログラム仕様書とプログラムの関係を明確にし，プログラム作成を分離しても，信頼性の高いプログラムを得ることができる。

4　厚生年金基金オンライン開発における適用結果

本システムは，日立製作所の全従業員の厚生年金保険料の掛金情報を退職時まで時系列に蓄積し，年金・一時金の給付計算を行なうもので，これには正確なデータ管理と複雑な計算処理が要求される。特長としては，年金個人情報の管理と給付計算のオンラインサービスを可能とした点にある。すなわち，月次に発生する大量データをバッチ処理し，その後のメンテナンス及びその他年金試算などのサービス業務をオンライン処理している。したがって，バッチ処理とオンライン処理に同一の機能が必要となり，その機能のモジュール化がシステム開発の上で重要な課題であった。その解決に当たり，本作成規準が課した役割と適用結果について次に説明する。

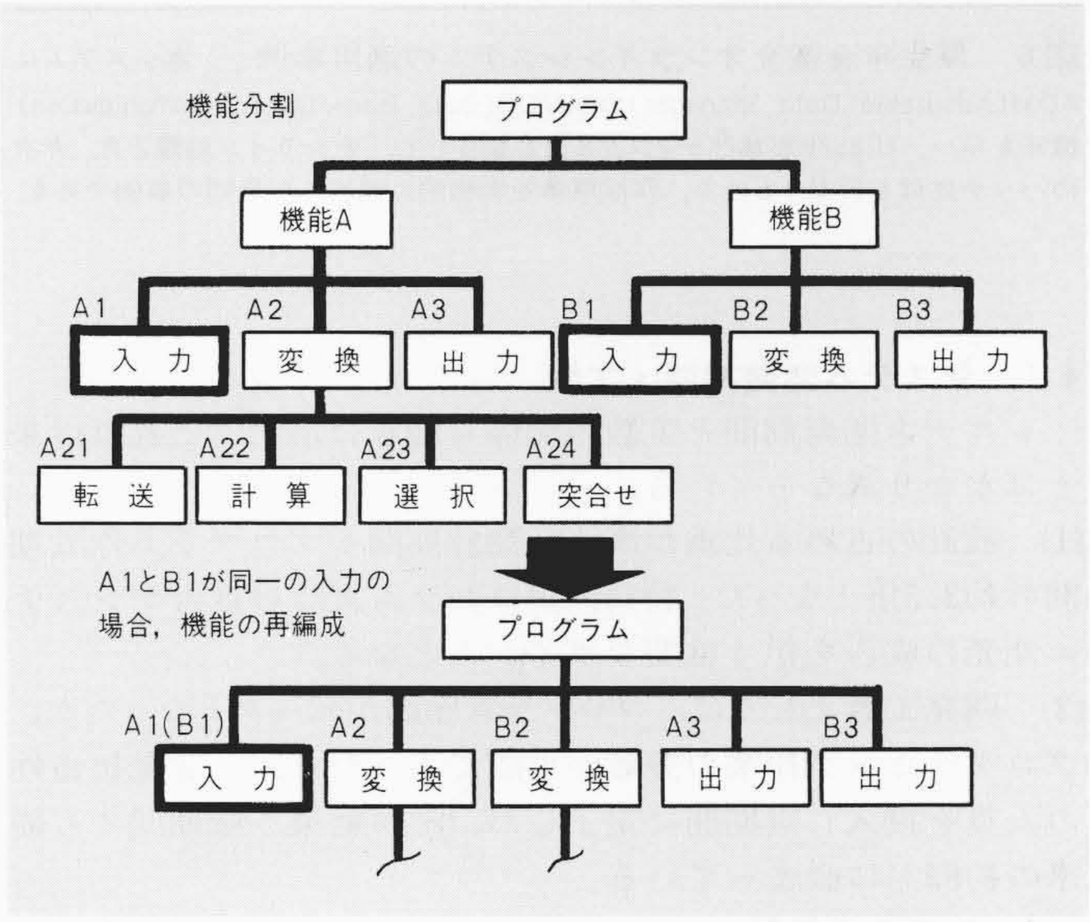


図4　機能分割と再編成の一方法　機能分割には，(1)まず入力－変換－出力の三つに分ける，(2)次に変換を転送・計算・選択・突合せに分ける。入出力は一般的にファイル処理となるため，それらの再編成を行なう。他の機能についても重複していれば再編成を考える。

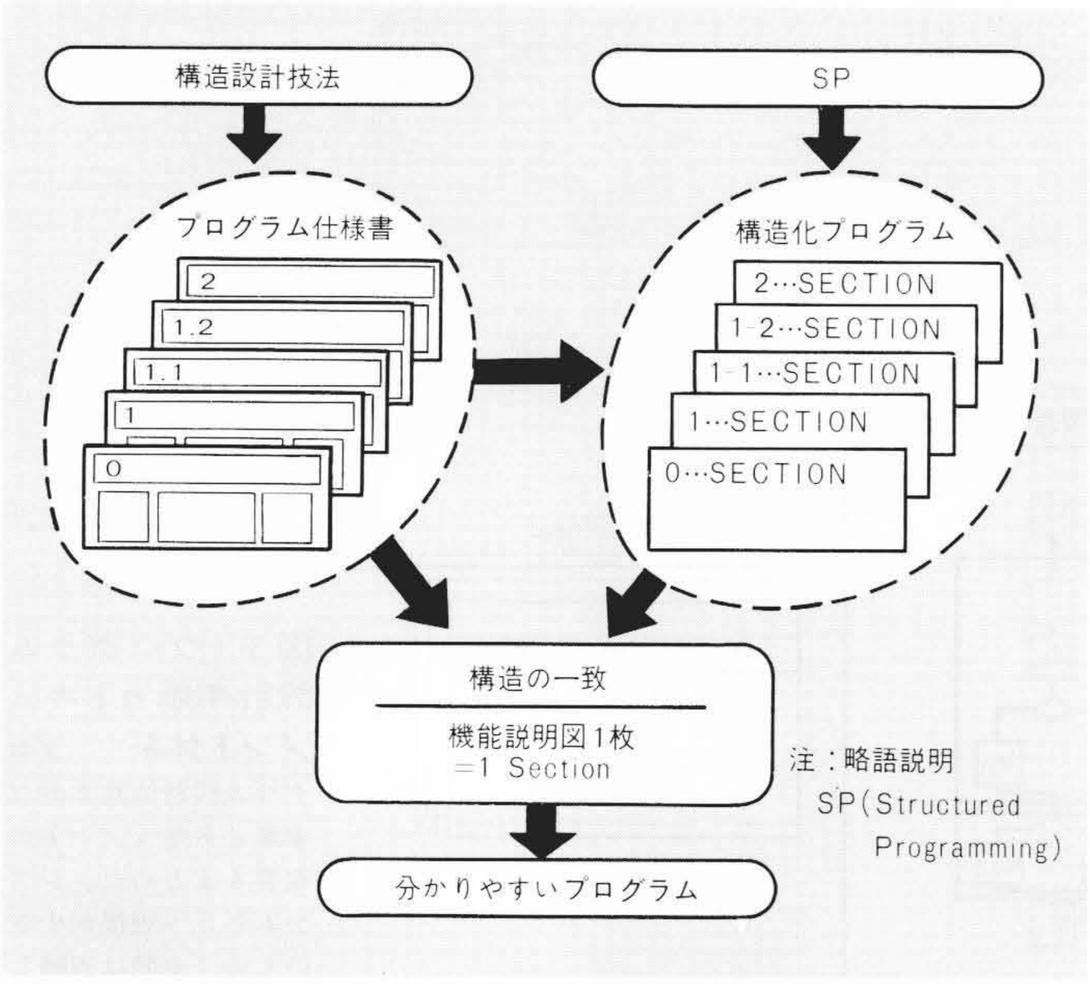
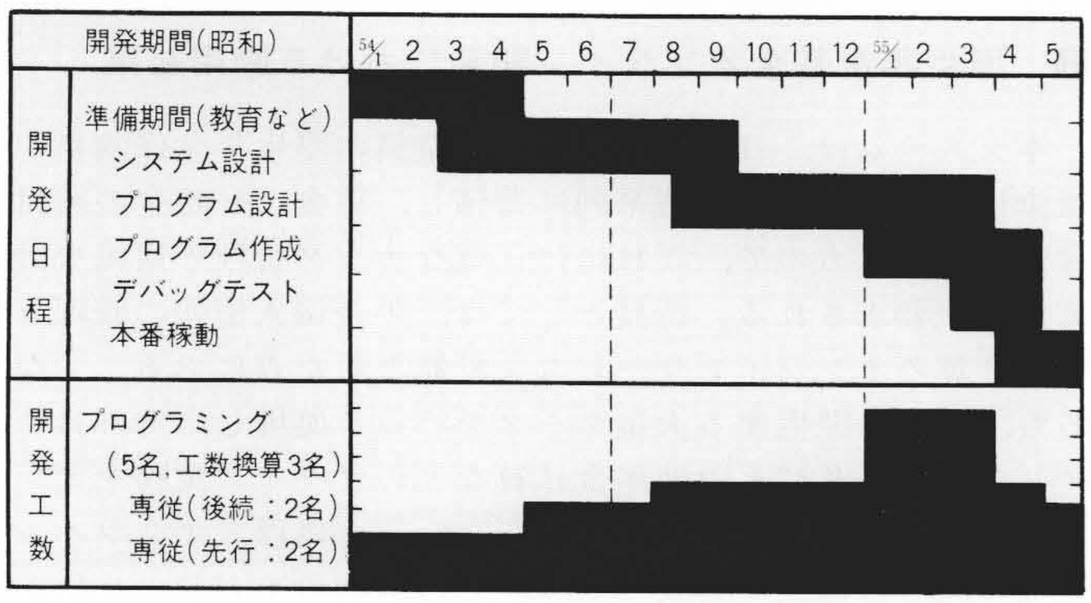


図5 プログラム仕様書とプログラムの構造 構造化設計技法により作成された仕様書をSPでコーディングし，それぞれの構造を一致させる。

1. 開発期間と工数



2. システム規模

プログラム本数：170本(モジュール，マスプを含む。)
ス テ ッ プ 数：53キロステップ

図6 厚生年金基金オンラインシステムの適用事例 本システムはADM(Adaptable Data Manager)のDB/DC(Data Base/Data Communication)機能を用い，H-8589形集団ディスク8台を駆使して，オンライン処理と月，年次のバッチ処理を行なうもので，作成規準を本格的に適用した最初の事例である。

4.1 システム開発期間・工数

システム開発期間と工数の関係を**図6**に示す。これは従来とはかなり異なっている。

(1) 設計の占める比重が高く，設計期間がプログラム作成期間の約3.5倍となった。特に，プログラム設計の良否がシステム開発の成否を担う重要なポイントとなる。

(2) 開発工数としては，プログラム作成がピークとなったが，プログラミング作業の分離が可能であったため，開発担当外の人員を投入し短期間に完了した。その結果，総期間でも従来の約85%に収まっている。

4.2 システム設計

(1) 初期機能階層図は，A3判：5枚，階層：10レベル，機能数：95で，プログラムレベルの直前まで分割した。初期機能説明図はA3判：74枚で，最下位の機能説明ではほぼプログラムレベルまで記述した。

(2) 中間でサブシステムを設定したが，初期で十分検討していたために，中間機能階層図及び説明図としては作成しなかった。またこの段階で，ファイルの基本設計を行なった。

(3) 最終機能階層図はA4判：30枚，階層：5レベル，機能数：190で，最下位の機能の大半はプログラムにとどめた。特に重要なプログラムは，この段階で詳細に機能分割を行なったため，プログラム及びその後のモジュールの設定に有効であった。最終機能説明図はA4判：56枚で，内容が整理されたため，初期のものに比べかなり少なくなっている。

(4) システム設計での機能説明図は，必ずしも機能に対応し1枚ずつ作成する必要のないことが分かった。

4.3 プログラム設計及びプログラム作成

(1) プログラム仕様書はA4判：平均15枚，最大28枚で，従来と比較し2倍以上になっている。

(2) 階層は平均3レベル，最大5レベルとなった。

(3) 機能数は平均7，最大21となった。

(4) COBOLで作成したプログラムの平均は約210ステップで，機能説明図1枚当たりでは約15ステップとなった。

(5) 特に機能の重複が多いプログラムについては，モジュール化を図った。例えば，給付計算プログラムは48本のモジュールに分割し，そのうち40本は2箇所以上で使用しており，1モジュール当たり平均120ステップとなった。これにより，総ステップ数を当初の予想の$\frac{1}{2}$以下に削減でき，オンライン下でのロードメモリの縮小が可能となった。

(6) プログラム作成にシステム開発担当外の人員投入が可能で，非常に短期間に終了した。その品質も設計者が作成したものとの差が少なかった。

(7) 設計者がデバッグ及びテストを行なったが，仕様書とプログラムの構造が一致しているためにプログラムの解読に仕様書が有効で，その工数は従来の約$\frac{1}{2}$で済んだ。したがって，今後の保守・拡張時の効果が期待できる。

(8) 仕様書が機能表現のため，ユーザー部門も仕様書で十分検討できた結果，その後の変更も少なく，また全体的にも本番稼動後のトラブルが少なかった。

4.4 その他

この作成規準について事前に教育を行なっていたこと，不明な点は実例編を参照したことにより，適用はスムーズに行なえた。

この適用事例からも分かるように，設計に対する負担は大きいが，その反面，受ける効果も大きかった。特に，仕様書によりシステムを管理できるメリットは非常に大きい。

5 結 言

以上，構造化設計技法を主体とする作成規準とその適用事例について説明してきたが，適用を開始して約2年を経過したばかりで，まだ内容的にも改善の余地があると思われる。

(1) 設計面で，当初から問題意識はあったが，機能分割に決定的な手法を見いだせないため，依然として属人性が残る。

(2) 記述面でこの手法を支援するツールがないために，手作業が多く設計者の負担が大きい。

(3) テスト方法，レビュー方法を確立する必要がある。

(4) プログラムを漢字化・図表化し，仕様書とプログラムを一体化する。

など，様々な課題があるが，適用事例にもあるとおりその効果は大きく，また，システム監査への対応にも有効で，これら諸問題を解決し，なおいっそうの充実を図る考えである。