人工衛星チェックアウト・システムの基本設計プロセスのプロセス・モデル HFSP による記述とその評価†

望月純夫†† 山内顯†† 片山卓也†††

ソフトウェアの設計作業は，極めて論理的かつ知的思考作業の連続であり，また設計技術者への属人的性格が強く，これを顕在化することは難しい問題である．われわれは，熟練技術者の基本設計段階における手順を明らかにすべく，一つの実用システム（DAS）の設計経験者を集め，プロセス・モデル HFSP に基づき，過去における設計手順をできる限り忠実に分析，抽出した．（手続き中心型プロセス）これを他のシステム（設計課題）の設計に再利用する観点から評価した結果，このプロセスは対象システムを設計するときの手順そのものを表現しており，規模の異なるシステムの設計にそのまま適用することは難しく，また他の技術者に理解しにくいことが判明した．これを改善するためオブジェクトの生成過程を中心にプロセスをまとめ直した結果，規模の異なる他のシステムの設計にも適用できる，ある程度普遍的な設計プロセス（オブジェクト中心型プロセス）を得た．ここに得た手続き中心型プロセスおよびオブジェクト中心型プロセスの比較を通して設計作業の実態を解析し，さらにプロセス・モデル HFSP の表現能力を評価した．その結果，その機能的側面は標準的な設計技術を顕在化して静的に表現することができるが，動的側面においてはいくつかの課題があるという結論を得た．今後，この表現能力を充実させるためには，実際の設計作業そのものをさらに深く分析し，これを関数表現に反映させる必要がある．

1. はじめに

ソフトウェアの設計業務は，論理的かつ知的な細かい作業の連続である．そのために設計作業そのものの具体的な内容および順序はそれぞれの設計者の考案と工夫に任せられており，ソフトウェア設計現場における作業の把握や管理は非常に難しいものになっている．

設計技術者の思考過程を顕在化して表現することは極めて困難なことではあるが，この数年間この課題に対して，設計作業の分析および記述を目的としたソフトウェア・プロセスの研究が活発に進められてきた[1]．しかし，これまでのソフトウェア・プロセスの研究は，設計技術者の思考の仕組みや過程の原理的な解明およびその表現のための形式的体系の研究が中心となっており，現場における設計プロセスを明確なモデルのもとに記述した例がほとんどない．ソフトウェア・プロセス研究の進展のためには，理論と実際との協調が重要である．われわれはこのような考えのもとに，プロセス・モデル HFSP を利用して，過去に開発してきた実時間処理システムの中で一つの代表的なシステムの設計プロセスを分析して記述し，さらに理論と実際との相違に主眼を置いた評価を試みた．この結果，プロセス・モデルのみでなくプロセス自身についても有用な知見を得ることができた．

なお，本論文は，文献 15) で報告したソフトウェア・プロセスの分析・評価をもとに検討したものであり，手続き中心型プロセスおよびオブジェクト中心型プロセスの記述方法を発見し，それらの比較を行い，これをもとに論文を作成したものである．

2. 階層的関数型プロセス・モデル HFSP[2]

ソストウェア・プロセスには，様々な側面があるが，そのうちで最も基本的なものは，仕事の単位であるプロセスによって生成されるソフトウェア・オブジェクト，すなわち，要求仕様書，設計仕様書，解析報告書，プログラム・コード等の関係に着目することであり，ソフトウェア・プロセスをその入出力オブジェクト間の関数として捉えるものである．HFSP では，これを機能的側面と呼んでいるが，このほかに動的側面および実働的側面があり，その三つの独立な側面によってソフトウェア・プロセスを記述しようとしている．

動的側面では，それらの関数の起動される順序やプロセス間の並行動作，プロセスの生成などのような動的な事柄を規定する．これは機能的側面がソフトウェア・オブジェクト間の静的関数関係を規定しているのと対象的である．一方，実働的側面では，機能的およ

† Describing and Evaluating Fundamental Design Process of Checkout System for Artificial Spacecraft by Process Model HFSP by SUMIO MOCHIZUKI, AKIRA YAMAUCHI (Mitsubisi Space Software Co.) and TAKUYA KATAYAMA (Department of Computer Science, Faculty of Engineering, Tokyo Institute of Technology).

†† 三菱スペース・ソフトウエア(株)
††† 東京工業大学工学部情報工学科

び動的側面で記述されたことが実際の組織やプロジェクトで実現されるための機構が規定されるが，これには，資金や人員などの資源管理の問題，スケジュール管理，また，組織やその中での情報伝達，決定機構などが含まれる[4]．

HFSPでは，ソフトウェア・プロセスの機能的側面をオブジェクト間の入出力関係，すなわち関数として定義するが，この関数関係は階層的な分割を通して定義される．

例えば，プロセスA，およびその入力オブジェクト $x_1 \cdots x_n$，出力オブジェクト $y_1 \cdots y_m$ の関係は，次のように表すことができる．

$A(x_1, \cdots, x_n | y_1, \cdots, y_m)$

Aは，関数的に実行され，$x_1 \cdots x_n$ を処理して $y_1 \cdots y_m$ を求めるが，それ以外のオブジェクトに対しては何ら変化を与えない．このとき $x_1 \cdots x_n$，$y_1 \cdots y_m$ をプロセスAの入力および出力オブジェクトと呼ぶこととする．プロセスAは，図1のように表される．

プロセスAが，そのまま実行できる程度に単純でない場合は，これをさらに細かいサブ・プロセス $A_1A_2A_3 \cdots A_k$ に分解する．オブジェクトの定義をEとし，分解の条件をCとすると次のように表すことができる．

$A \rightarrow A_1A_2A_3 \cdots A_k$ when C where E.

オブジェクトの定義のセットEは，各サブ・プロセスにどのような入力オブジェクトが与えられるか，またサブ・プロセスが完了後メイン・プロセスの結果がどのように得られるかを示す．すなわち，Eは，次の定義を含む．

①サブ・プロセス $A_1, \cdots, A_k$ への入力オブジェクト

②メイン・プロセスの出力オブジェクト

すべてのオブジェクトの定義は，次の形式で示される．

$a = f(a_1, a_2, \cdots)$

a は，ここで定義されるオブジェクトであり，a_1, a_2 … は，分解されたサブ・プロセスのオブジェクトである．f は，あらかじめ定義される補助的な関数であるが，非常に簡単な表現をとることもある．

プロセスの分解の例 ($A_0 \rightarrow A_1A_2$) を図2に示す．

このプロセス・モデルHFSPに基づいて，具体的なシステムの設計プロセスを抽出し，記述する場合，ソフトウェア・プロセスは次の手順に従って定義される．

①ソフトウェア・プロセスの入力オブジェクトおよび出力オブジェクトを明確に定義する．

②プロセスの機能が複雑である場合は，そのプロセスをさらに細かいサブ・プロセスに分解する．この作業を各々のプロセスが十分に単純化されるまで続ける．

③プロセスの分解に際しては，オブジェクトがプロセスの途中で消滅したり，急に現れたりしないように各々のオブジェクト間の関係を明らかにする．具体的には，各サブ・プロセスが，どのような入力オブジェクトを受け取り，その出力オブジェクトからプロセスの出力オブジェクトがどのように構成されるかを明確にする．

本稿は，人工衛星チェックアウト・システム設計プロセスのHFSPによる記述とその評価の報告であるが，なかでも機能的側面に重点をおいて述べることとする．これは，機能的側面が最も基本的であると同時に最も安定しており，記述や記録が行いやすいからである．また，動的側面についてもいくつかの考察を述べる．一方，実働的側面については，プロジェクト管理やリスク管理とも密接な関係があり，現実問題としては重要な側面であるが，その記述のための十分な機能が現在のHFSPにはないことおよび明確な記録が残されていないことから今回は記述の対象から外すこととした．

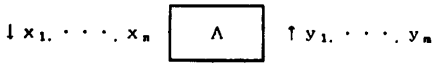
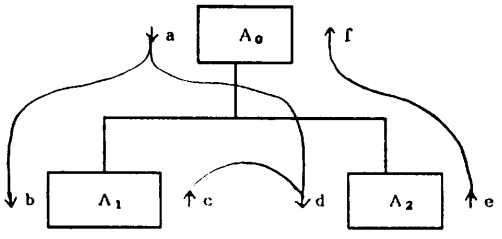


図1 プロセスAの図形表示
Fig. 1 Graphical representations of process.



when $C(a)$
where $b=F(a)$, $d=G(c, a)$, $f=e$
$A_0(a|f)=A_1(b|c)A_2(d|e)$
when $C(a)$
where $b=F(a)$, $d=G(c, a)$, $f=e$

図2 プロセスの分解の例
Fig. 2 Process decomposition.

3. 手続き中心型記述方式による実システム (DAS) 設計プロセスの分析と記述

われわれは，まずプロセス・モデル HFSP に基づいて，実用の実時間処理システムの設計手順を明らかにすることとしたが，分析の対象システムとしては，『同様のシステムの設計を何回も経験し，設計経験者が数多くいると共に設計手法が社内に醸成されており，その規模があまり大きくないシステム』という条件のもとに，人工衛星チェックアウトを目的としたデータ収集解析システム（以下 DAS(Data Acquisition System) と略称する）を選択した．

DAS は，次のような計算機システムである．

DAS の機能

人工衛星の最終的なシステム試験のうち，機能および性能試験を目的とした計算機システムである．同システムは，組立調整された人工衛星の近くに設置され，アンテナを経由して人工衛星から送出されるテレメトリー・データを受信，解析し，さらに人工衛星を制御するコマンド・データを送出することにより，衛星各部が設計どおりに正常に動作することを確認するものである．実用システムとしては，比較的小規模ではあるが，典型的な計算制御システムの特徴を備えている．

分析すべき設計フェイズとしては，システム開発のライフサイクルの中で最も重要と考えられる基本設計段階を対象とすることとした[5]．

（1） DAS 基本設計プロセスの分析

これまでに DAS の設計，開発に携わった熟練技術者6名を集めて，過去における DAS の設計プロセスを分析した．分析の作業期間としては，およそ6か月を要し，平均1か月に4回の検討会を中心として全員の中に内在する標準的な設計技術の抽出，記述を進めた[6]．次にだいたいの工程を示す．

①設計手法および手順のレベル合わせ…2か月

②上位レベルのソフトウェア設計手順およびオブジェクトのまとめ……2か月

③詳細レベルのソフトウェア設計手順およびオブジェクトのまとめ……2か月

具体的なソフトウェア・プロセスの抽出方法は次のとおりである．

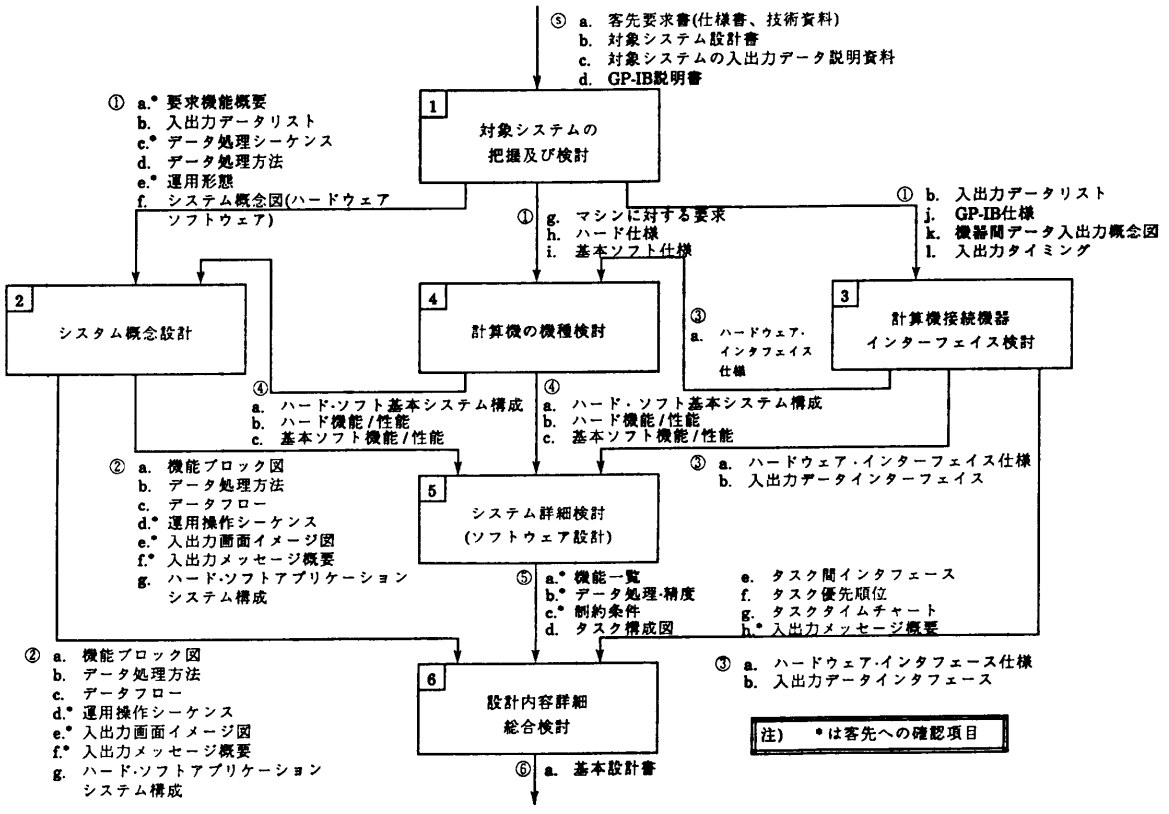


図 3 DAS 基本設計のソフトウェア・プロセス（第1レベル）
Fig. 3 Software process of fundamental design of DAS (First level).

①過去に担当したDASの設計作業をたどり，できる限り時間の経過に忠実に設計手順および各々の手順の入力および出力オブジェクトを記述した．

②最初にマクロ的に作業を記述し，それに対応するオブジェクトを記述した．（第1レベルのソフトウェア・プロセス）

③次に，第1レベルの各プロセスを詳細化して，設計手順とそれに対応するオブジェクトを記述した．（第2レベルのソフトウェア・プロセス）

④手順に技術者の個人差がある場合には，部門の標準技術として適切な手順を採用した．

この方法は，実際の設計手順を中心として現実の設計そのものを表現しようとするものであり，手続き（Procedure）に重点をおいた記述方法，すなわち，手続き中心型記述方式（Procedure-Centered Description）と呼ぶ．

その結果，第1レベル（**図3**に示す）と，それをさらに詳細化した第2レベル（例として，図3の第2プロセスのみを詳細化したものを**図4**に示す）の二つの階層からなるソフトウェア・プロセスを抽出することができた．

図中，各ソフトウェア・プロセスとその入出力オブジェクトをデータ・フロー的に表現する．各設計手順を示すソフトウェア・プロセスをブロックで示し，各手順の実行順序をブロック間の矢印で表現している．

また，各ブロックの前後には，そのソフトウェア・プロセスの入出力オブジェクトを示す．

ソフトウェア・プロセスの分解は，さらに詳細化して，第3レベル，第4レベルの階層を設定することもできる．しかし，設計作業を実行するためにはこのレベルで十分であり，またこれ以上詳細化するとソフトウェア・プロセスの安定性がなくなる（個人差がでてくる）ばかりでなく，あまり細かく規定すると個人の独創性を阻害する恐れもあると判断して，この二つの階層に止め，その先は各個人の創意工夫に任せることとした．

第3の一連のソフトウェア・プロセスの中で，プロセス3，4は，計算機の機種検討およびハードウェア・インタフェースの設計を示しており，プロセス1，2，5，6は，要求仕様のまとめからソフトウェア設計までのいわゆるソフトウェア基本設計過程を示している．

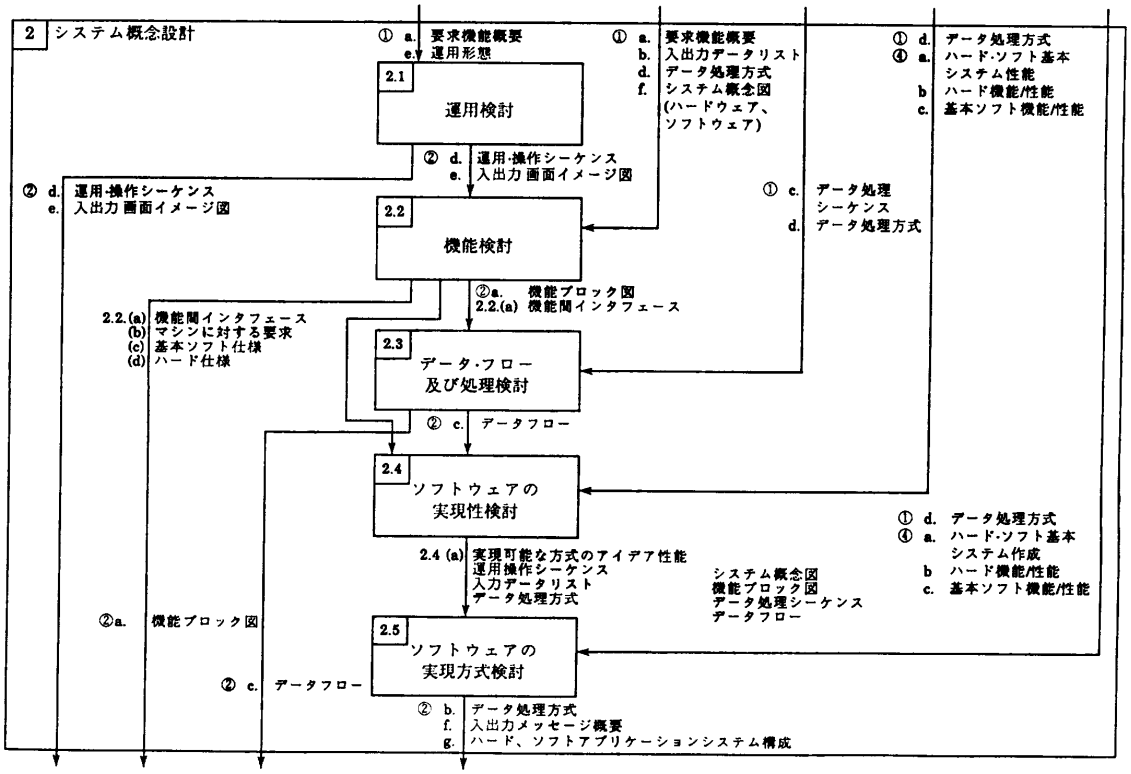


図4 DAS基本設計のソフトウェア・プロセス（第2レベル）—システム概念設計プロセス
Fig. 4 Software process of fundamental design of DAS (Second level). —Conceptual design process.

表 1 手続き中心型記述方式により抽出したソフトウェア・プロセスと各出力オブジェクト一覧表
Table 1 Software process and its output objects extracted by operation centered description.

第1レベルのソフトウェア・プロセス (図1参照)	第2レベルのソフトウェア・プロセスの数	出力オブジェクト		
		正式	中間	数
——	——	Ⓢ a. 客先要求書 (仕様書、技術資料) b. 対象システム設計書 c. 対象システムの入出力データ説明資料 d. GP-IB説明書	——	4
1. 対象システムの把握及び検討	3	① a. 要求機能概要 b. 入出力データ・リスト c. データ処理シーケンス d. データ処理方法 e. 運用形態 f. システム概念図 (ハードウェア, ソフトウェア) g. マシンに対する要求 h. ハード仕様 i. 基本ソフト仕様 j. GP-IB仕様 k. 機器間データ入出力概念図 l. 入出力タイミング	——	12
2. システム概念設計	5 (詳細は図2参照)	② a. 機能ブロック図 b. データ処理方法' c. データフロー d. 運用操作シーケンス e. 入出力画面イメージ図 f. 入出力メッセージ概要 g. ハードソフトアプリケーションシステム構成	2.2(a) 機器間インターフェイス (b) マシンに対する要求 (c) 基本ソフト仕様 (d) ハード仕様 2.4 実現可能な複数の方式	9
3. 計算機接続機器インターフェイス検討	4	③ a. ハードウェア・インターフェイス仕様 b. 入出力データインターフェイス	3.1 データ種類別分類リスト 3.2(a) スロット・アドレス (b) ビット配列	4
4. 計算機の機種検討	4	④ a. ハード・ソフト基本システム構成 b. ハード機能・性能 c. 基本ソフト機能/性能	4.1 マシンへの要求仕様 (a) CPU特性 (b) チャネル能力 (c) 主記憶容量 他13項目 4.2 プロポーザル(複数のコンピュータに関する)	5
5. システム詳細検討 (ソフトウェア設計)	4	⑤ a. 機能一覧 b. データ処理、精度 c. 制約条件 d. タスク構成図 e. タスク間インターフェイス f. タスク優先順位 g. タスク・タイム・チャート h. 入出力メッセージ概要	5.1 機能の同時並行処理の関係 5.2 機能処理の前後関係 5.3(a) タスク構成 (b) タスクの同時並行・前後関係 (c) タスク間データ・フロー	11
6. 設計内容詳細総合検討	6	⑤ a. 基本設計書	——	1

得られたソフトウェア・プロセス (32 個) およびオブジェクト (46 個) をまとめたものを**表 1** に示す.

ここでは, 設計過程で作成されたオブジェクトを次のように二つの種類に分類した.

①正式オブジェクト

設計過程で作成されたオブジェクトのうち, そのまま最後まで残って基本設計書の一部を構成するもの (オブジェクトベースに最後まで残すべき物).

②中間オブジェクト

設計を円滑に進ませるために正式オブジェクトと正式オブジェクトとの間で一次的に作成するオブジェクトであり, 図1では, 省略してある (オブジェクトベースに残す必要のないもの).

(2) DAS 基本設計プロセスの評価

前節で得たソフトウェア・プロセスを別の2名の熟練技術者 (DAS の設計経験者であるが, 前項の分析・抽出作業には参加していない) を含めて見直し,

表 2 手続き中心型記述方式によるソフトウェア・プロセス（第2レベル）とその設計作業項目の対応表
Table 2 Software process (Second level) by operation centered description and its design work items.

第1レベルのソフトウェア・プロセス（図1参照） / 設計作業項目	プロセス1	プロセス2（図2参照）	プロセス5	プロセス6
ユーザ要求の分析と整理	(1.1)	○ 2.1 ○ 2.2	——	——
運用検討	(1.1) (1.2) (1.3)	(2.1)	——	——
機能検討	(1.1) (1.2) (1.3)	(2.2)	——	——
フィージビリティ・スタディ	○ 1.2	(2.4) (2.5)	——	——
データ及び処理検討	○ 1.2 ○ 1.3	(2.3) (2.4)	(5.3)	(6.3)
制御及びタイミング検討	○ 1.2	(2.4)	(5.1) (5.2) (5.3) (5.4)	(6.1) (6.3)
フィージビリティ・スタディ及び総合チェック	——	——	(5.4)	(6.1) (6.5) (6.6)

表中の円の大きさは作業量を示す。

その再利用の面から評価した．

その評価結果を次に示す．

①『計算機の機種検討』および『計算機接続機器インタフェース検討』は，ソフトウェア設計とは別のソフトウェア・プロセスとして取り扱うべきである．

特に，計算機の機種検討は，設計技術者の経験，ユーザの好み，予算などにより左右される要素が強く，純技術的な設計プロセスとは異質のものと考えられ，一括して別のレイヤ（例えばプロジェクト管理など）として取り扱うべきである．

②いくつかの設計プロセスの作業内容に同じ種類の作業項目が重複して現れる．例えば，図4に示す第2レベルのソフトウェア・プロセスのうち，2.3『データ・フローおよび処理検討』，2.4『ソフトウェアの実現性検討』などの検討作業が，図3に示されている第1レベルのソフトウェア・プロセス5，6にも含まれている．ソフトウェア・プロセス 1，2，5，6 に対応する第2レベルのソフトウェア・プロセスとその作業内容との関連を表2に示す．表中の円の大小は，作業量の大きさを示している．これを見てわかるように，ソフトウェア・プロセス1から6までの中で同じ作業内容（後工程の作業内容のほうが詳細となる）が何度も現れるため，このソフトウェア・プロセスは理解しがたいものとなっている．

この原因は，手続き中心型のソフトウェア・プロセスは，設計手順を時間の経過に忠実に記述したものであり，その中のある手順に注目すれば，

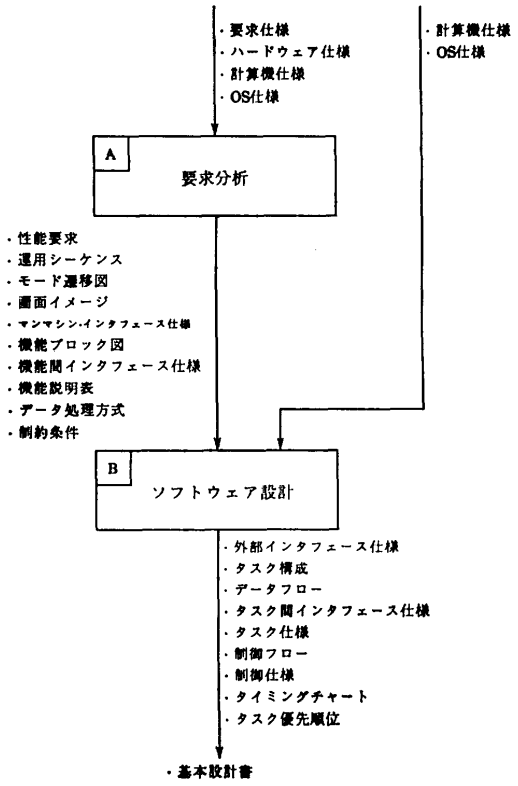


図 5 実時間処理システム基本設計のソフトウェア・プロセス（第0レベル）
Fig 5 Software process of fundamental design of real-time processing system (0'th level).

表 3 オブジェクト中心型記述方式により抽出したソフトウェア・プロセスと各出力オブジェクト一覧表
Table 3 Software process by object centered description and its output objects.

第0レベル (図3参照)	第1レベルのソフトウェア・プロセス (図4参照)	第2レベルのソフトウェア・プロセスの数	出力オブジェクト		
			正式	中間	数
	——	——	⑤ a. 要求仕様 b. ハードウェア仕様 c. 計算機仕様 d. OS仕様	——	4
A 要求分析	1 ユーザ要求の分析と整理	3	① d. 性能要求	1.2(a) 運用要求 1.3(a) 機能要求	3
	2 運用検討	4	② a. 運用シーケンス b. モード遷移図 c. 画面イメージ d. マンマシン・インタフェース仕様 e. 質問(ユーザへ)	2.3(a) 操作イメージ (b) 操作シーケンス (c) 画面遷移	7
	3 機能検討	3	③ a. 機能ブロック図 b. 機能間インタフェース仕様 c. 機能説明表 d. データ処理方式	3.1(a) 機能概要 (b) 機能間関係説明	6
	4 フィージビリティ・スタディ	——	④ a. 制約条件	——	1
B ソフトウェア設計	5 データ及び処理検討	4 (図5参照)	⑤ a. 外部インタフェース仕様 b. タスク構成 c. データフロー d. タスク間インタフェース仕様 e. タスク仕様	5.1(a) タスク構成 5.2(a) データフロー (b) タスク間インタフェース方法 (c) 共有メモリ仕様 (d) データファイル仕様 (e) 外部入出力データ仕様 5.3(a) タスク仕様	12
	6 制御及びタイミング検討	3	⑥ a. 制御フロー b. 制御仕様 c. タイミングチャート d. タスク優先順位	6.1(a) 制御フロー (b) 制御条件 6.2(a) タイミングチャート (b) タスク優先順位	8
	7 フィージビリティ及び総合チェック	——	⑦ x. 基本設計書	——	1

いくつかのフェイズにわたって実行され，はじめ粗いものから次第に詳細なものに進む過程を示すからである．したがって手続き中心型ソフトウェア・プロセスは，現実の具体的な作業を忠実に示しているといえるが，作業内容という点からは，同一の内容の作業で進度の異なるものが複数のプロセスに分かれて実行されていることになる．

4. オブジェクト中心型記述方式による実システム (DAS) 設計プロセスの分析と記述

手続き中心型記述方式によるソフトウェア・プロセスは，現実の設計作業を時間の経過に忠実に表現したという意味では重要性が高いが，内容の似たプロセスが別々に現れており他の技術者にとって理解しにくいことが判明したので，これを大幅に見直した．

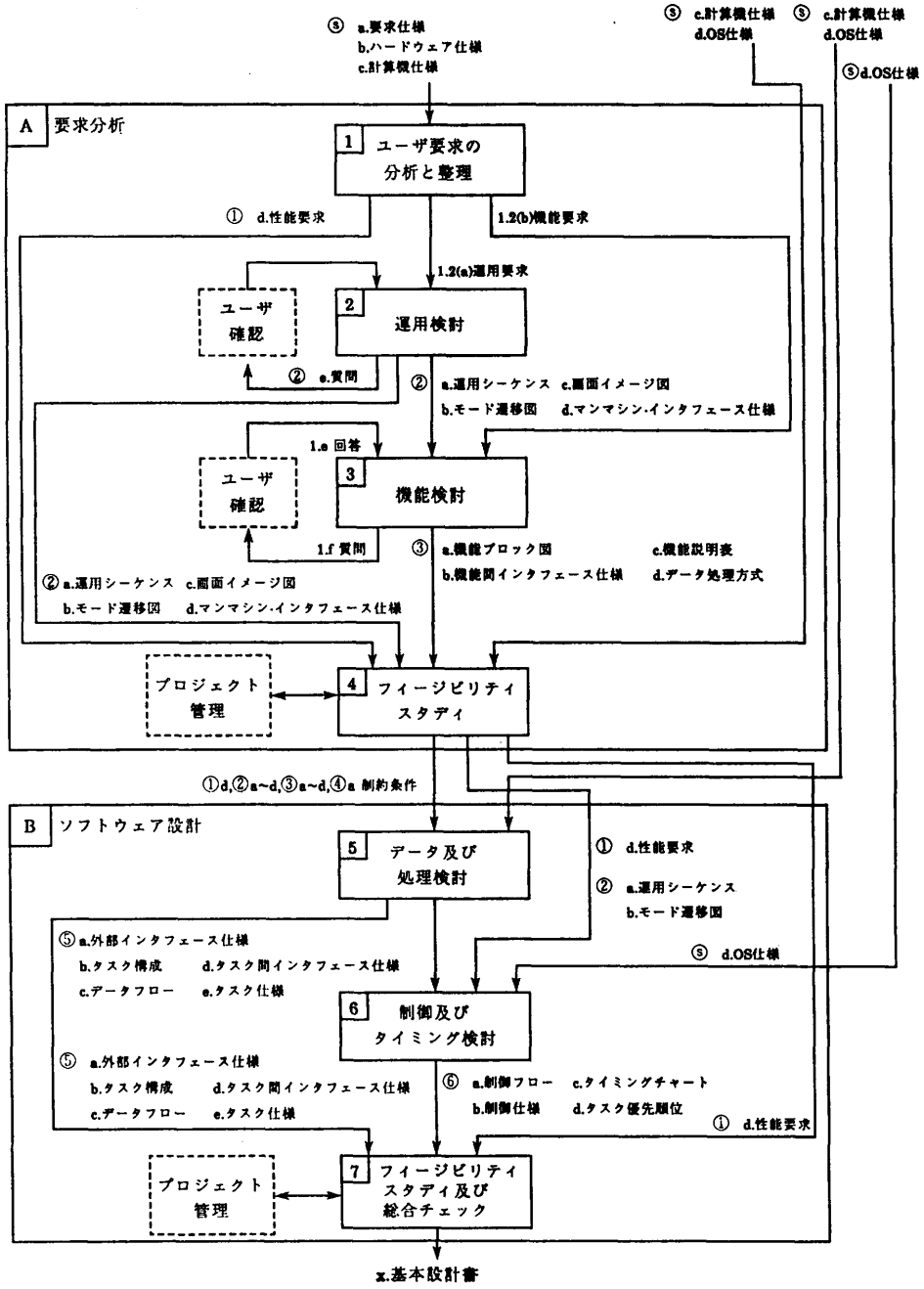


図 6 実時間処理システム基本設計のソフトウェア・プロセス（第1レベル）
Fig. 6 Software process of fundamental design of real-time processing system (First level).

設計の時間的経過をある程度無視し，各オブジェクトを生成するための作業内容を重視し，これをもとにしてソフトウェア・プロセスおよびオブジェクトをまとめ直すこととした．

すなわち，基本設計を行う過程で作成したすべてのオブジェクトを収集し，これを表2に示す設計作業項目（この項目は，基本設計書を構成する重要な要素でもある．）をもとにして分類した．さらに，このプロセスおよびオブジェクトを詳細化して，次のレベルとした．（例えば，設計作業項目のうち，“機能検討”に関係するオブジェクトとして分類したものは，表4の上から3段目（入力オブジェクト）および表3の上から

4段目（出力オブジェクト）に示されている.）

ここで述べた記述方法を第3章の手続き中心型記述方式と比較してみると設計手順の時間的経緯よりも，オブジェクトがどのように加工されていくかを重視した方式であり，これをオブジェクト中心型記述方式（Object-Centered Description）と呼ぶ.

収集した全オブジェクト（42個）およびソフトウェア・プロセス（17個）を整理した結果を**表3**に示す.

表の左側の項目は，右側に示すオブジェクトを作成するために必要な設計作業項目，すなわち，ソフトウェア・プロセス（第0レベル，第1レベル）である．これをさらに詳細化して第2レベルのソフトウェア・プロセスを求めた.

最終的に求めた3階層からなるソフトウェア・プロセスを**図5，図6，図7**（図6の中の5『データおよび処理検討』を詳細化したもの）に示す.

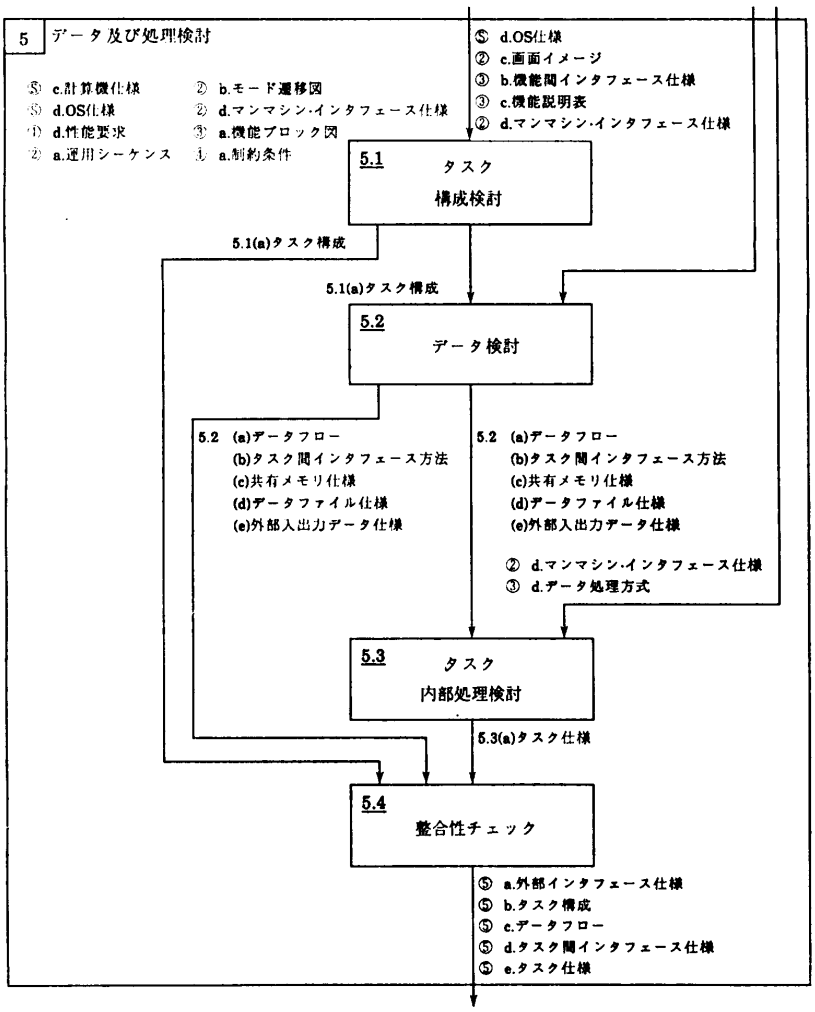


図7 実時間処理システム基本設計のソフトウェア・プロセス（第2レベル）—データおよび処理検討

Fig. 7 Software process of fundamental design of real-time processing system (Second level). —Investigation of data and processing.

5. 手続き中心型ソフトウェア・プロセスとオブジェクト中心型ソフトウェア・プロセスとの比較

手続き中心型記述方式と比較してオブジェクト中心型記述方式は，オブジェクトがどのように加工されていくかを中心に整理してまとめた形になっており，現実の設計作業を時間的な経過に沿って厳密に表現したものではない．実際には，ある時点の設計作業はこれらのオブジェクト中心型記述方式で得たソフトウェア・プロセスのうちいくつかを同時に（並行的に）実行するという形をとり，そのオブジェクトを互いに参照し，部分的な調整を重ねるというように複雑な動きを示す．この関係を**図8**に示す[8),11),13)].

ある時点 T における設計作業は，図中の T 上の断面により表現されており，また各プロセスの断面の大きさは，その作業量の大きさを示している.

手続き中心型記述方式によって求めたソフトウェア・プロセスは，設計手順を時間を追って忠実に表現したものであり，図3の中の第1レベルのソフトウェア・プロセス1，2，5，6は，図8の中の横軸（上方）および破線で示すように時間帯で作業を区切って各々をまとめた形になっている．したがって，手続き中心型記述方式によって得たソフトウェア・プロセスには，異なった時間のプロセスに同じ設計項目が何度も現れている.

また，この DAS 設計プロセスの並行動作の状況（図8）は，小規模システム設計プロセスの並行動作の状況（**図9**）とは，著しく異なっている．したがっ

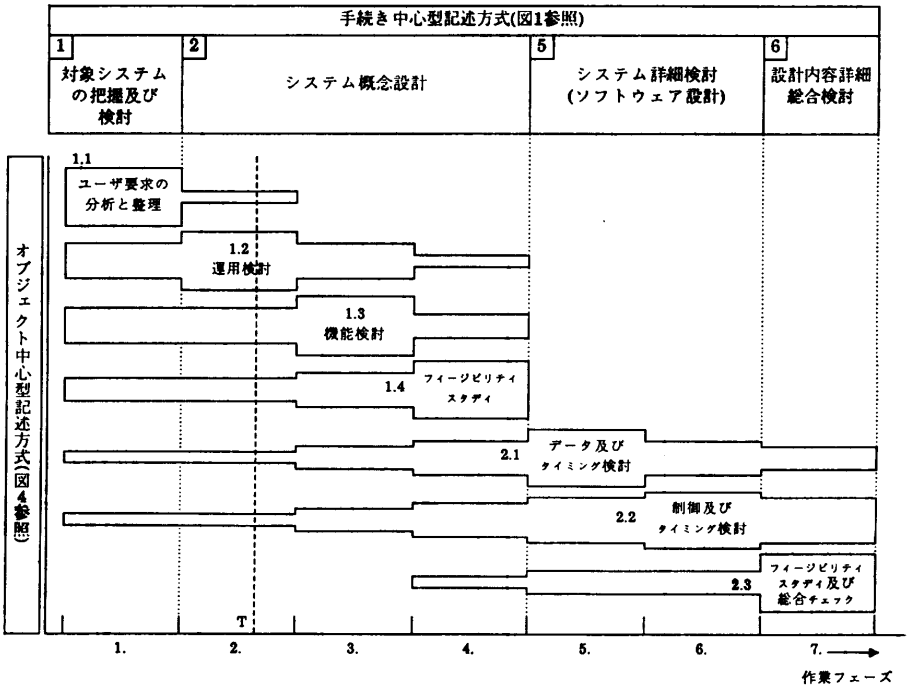


図 8 手続き中心型記述方式およびオブジェクト中心型記述方式ソフトウェア・プロセス(第1レベル)の比較 (DAS 設計プロセス)

Fig. 8 Comparison of the software process (First level) of operation centered description and object centered description. (Design process of DAS.)

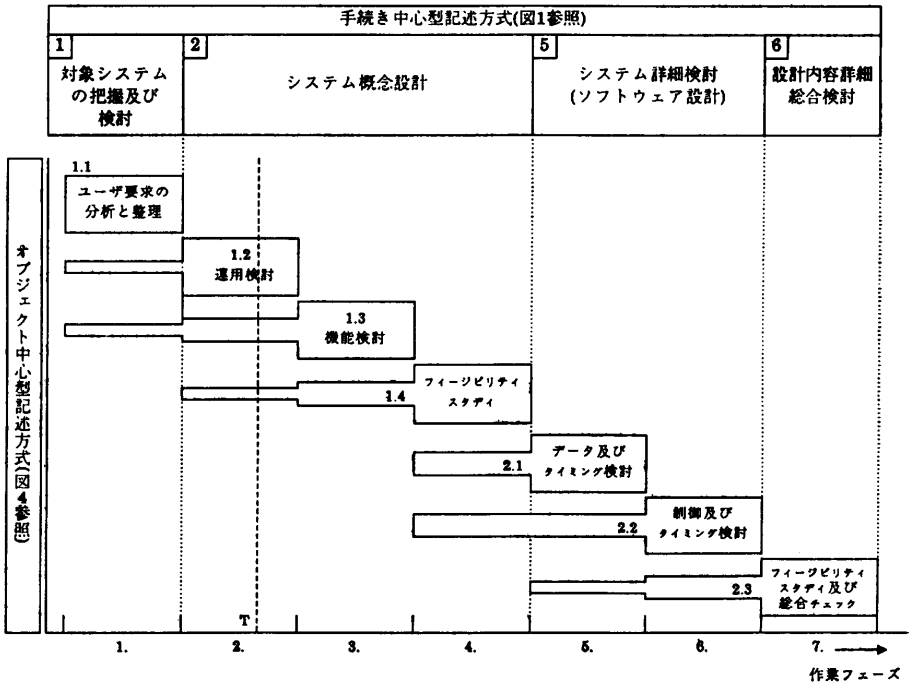
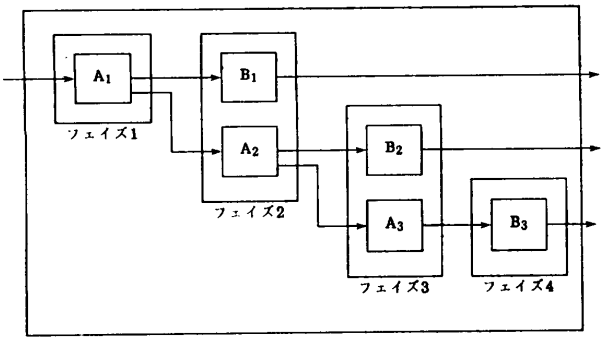
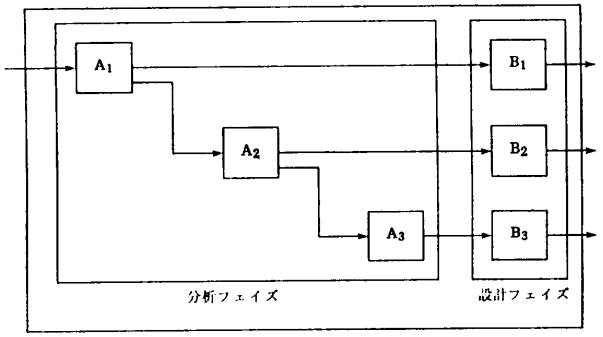


図 9 手続き中心型記述方式およびオブジェクト中心型記述方式ソフトウェア・プロセス(第1レベル)の比較 (小規模システム設計プロセス)

Fig. 9 Comparison of the software process (First level) of operation centered description and object centered description. (Design process of small scaled system.)



(a) 手続き中心型記述方式
(a) Procedure-centered description.



(b) オブジェクト中心型記述方式
(b) Object-centered description.

図 10 手続き中心型記述方式およびオブジェクト中心型記述方式の比較
Fig. 10 Comparing procedure-centered description with object-centered description.

て，このDASを対象とした手続き中心型ソフトウェア・プロセスは，規模の異なるシステムの設計にそのまま適用することのできない融通性の少ないソフトウェア・プロセスといえよう．

一方，オブジェクト中心型ソフトウェア・プロセスを実行するとき，一つの設計プロセスが複数の設計フェイズにわたって実行されることになる．

ここで，手続き中心型プロセスとオブジェクト中心型プロセスを比較するために，プロセスA（サブ・プロセス $A_1A_2A_3$ より成る）とプロセスB（サブ・プロセス $B_1B_2B_3$ より成る）を想定する（**図 10**）[16]．手続き中心型プロセスは，図 10(a) に示すように設計プロセスの実行を時間に忠実に表現しているが，オブジェクト中心型プロセスは，図 10(b) のように，設計作業の内容をもとにまとめた形になっている．

オブジェクト中心型および手続き中心型の各ソフトウェア・プロセス（第1レベル）の入力オブジェクトを対応させたものを**表 4** に示す．上の欄には，手続き中心型ソフトウェア・プロセス（第1レベル）および各プロセスへの入力オブジェクトが示されており，設計作業が時間の経過に従って左から右に進むとともにプロセスおよびオブジェクトの内容が細かくなる．左側の欄には，オブジェクト中心型ソフトウェア・プロセス（第1レベル）および各プロセスへの入力オブジェクトが示されている．ここで示されている手続き中心型ソフトウェア・プロセスは，既に述べたように過去の DAS の設計手順を分析したものであり，オブジェクト名もその当時の名称を使用している．一方，オブジェクト中心型ソフトウェア・プロセスのオブジェクトは，最近われわれが使用している SA/SD 設計手法に沿った名称を採用している．したがって，双方のオブジェクトの名称が異なっているので，ここに各入力オブジェクトの内容の対応付けを表中の丸印で示した．

表中のオブジェクト中心型ソフトウェア・プロセスの一つ『データおよび処理検討』に注目し，それが時間と共にどのように検討されるかを次に示す．

①『システム概念設計』フェイズ

a．入力オブジェクト

要求機能概要，運用形態，計算機（ハード，ソフト）に関する機能・性能データ，入出力データ・リスト，データ処理シークエンス（内容は，まだ非常に粗いもの）等．

b．処理内容

システム機能を詳細化して，システムで処理するデータの種類，数量，内容とその処理内容を検討し，その概要をまとめる．（このフェイズでは，システム機能の検討が主であり，この『データおよび処理検討』は，まだ粗いものである．）

②『システム詳細検討（ソフトウェア設計）』フェイズ

a．入力オブジェクト

機能ブロック図，データ処理方式，運用操作シークエンス，入出力データ・インタフェース，入出力メッセージおよび画面概要，システム構成図（ハード，ソフト，アプリケーション）等．

b．処理内容

ソフトウェアのタスク・レベルの設計に対応する，詳細なデータの内容，処理方法およびその流れを検討する．

③『設計内容詳細総合検討』フェイズ

a．入力オブジェクト

タスク構成図，タスク間インタフェース，機能一

表 4 オブジェクト中心型記述方式と手続き中心型記述方式の第1レベル
ソフトウェア・プロセスの入力オブジェクト対応表

Table 4 Correspondence between input objects of software process (First level) of object centered description and those of operation centered description.

ソフトウェア・プロセス記述方式			手続き中心型記述方式			
	第一レベルのソフトウェア・プロセス		1 対象システムの把握及び検討	2 システム概念設計	5 システム詳細検討（ソフトウェア設計）	6 設計内容詳細総合検討
ソフトウェア・プロセス記述方法	第一レベルのソフトウェア・プロセス	入力オブジェクト / オブジェクト入力	GPIB説明書 対象システムの入出力データ説明 対象システム設計書 客先要求書（仕様書他）	基本ソフト機能・性能 ハード機能・性能 ハード・ソフト基本システム構成 システム概念図（ハード・ソフト） 運用形態 データ処理方式 データ処理シーケンス 入出力データリスト 要求機能概要	入出力データ・インタフェース ハードウェア・インタフェース仕様 基本ソフト機能・性能 ハード機能・性能 ハード・ソフト基本システム構成 ハード・ソフト・アプリシステム構成 入出力メッセージ概要 入出力画面イメージ図 運用操作シーケンス データフロー データ処理方式 機能ブロック図	入出力データ・インタフェース ハードウェア・インタフェース仕様 タスクタイムチャート タスク優先順位 タスク構成図 タスク間インタフェース 制約条件 処理・精度 機能一覧 ハード・ソフト・アプリシステム構成 入出力メッセージ概要 入出力画面イメージ図 運用操作シーケンス データフロー データ処理方式 機能ブロック図
オブジェクト中心型記述方式	分析と整理 ユーザ要求	要求仕様 ハードウェア仕様 計算機仕様	○ ○ ○ ○	○ ○ ○ ○ ○ ○ ○ ○		
	検討 運用	運用要求	—	○	—	—
	機能検討	運用シーケンス モード遷移図 画面イメージ マン・マシン・インタフェース仕様 機能要求	—	○ ○ ○ ○	○ ○ ○ ○ ○ ○ ○ ○	—
	フィージビリティ・スタディ	運用シーケンス モード遷移図 画面イメージ マン・マシン・インタフェース仕様 機能ブロック図 機能間インタフェース仕様 機能説明表 データ処理方式 性能要求 計算機仕様 OS仕様	○	○ ○ ○ ○ ○ ○ ○	○ ○ ○ ○ ○ ○ ○ ○ ○ ○ ○ ○ ○	—
	データ及び処理検討	性能要求 運用シーケンス モード遷移図 画面イメージ マン・マシン・インタフェース仕様 機能ブロック図 機能間インタフェース仕様 機能説明表 データ処理方式 制約条件 計算機仕様 OS仕様	○	○ ○ ○ ○ ○ ○	○ ○ ○ ○ ○ ○ ○ ○ ○ ○ ○ ○ ○	○ ○ ○ ○ ○ ○ ○ ○ ○
	制御及びタイミング検討	外部インタフェース仕様 タスク構成 データフロー タスク間インタフェース仕様 タスク仕様 運用シーケンス モード遷移図 性能要求 OS仕様	○	○ ○ ○ ○ ○ ○	○ ○ ○ ○ ○	○ ○ ○ ○ ○ ○ ○ ○
	フィージビリティ・スタディ及び総合チェック	外部インタフェース仕様 タスク構成 データフロー タスク間インタフェース仕様 タスク仕様 制御フロー 制御仕様 タイミングチャート タスク優先順位 性能要求	—	—	○ ○ ○ ○	○ ○ ○ ○ ○ ○ ○ ○ ○ ○ ○ ○ ○

表 5 手続き中心型プロセス，オブジェクト中心型プロセス記述の比較
Table 5 Comparing describing method of procedure centered process with that of object centered process.

	プロセスの記述に関して重視する点	他システムの設計への再利用	理解しやすさ
手続き中心型ソフトウェア・プロセス	具体的システムの設計手順を時間の経緯に重点を置いて記述.	対象システムより大幅に規模および複雑さが異なるシステムには不適切.	対象システムが複雑になると理解しにくい.
オブジェクト中心型ソフトウェア・プロセス	オブジェクトがどのように加工されるかを重視して記述.	再利用可能（ただし，プロセスの調整が必要）.	理解しやすい.

覧等.

b．処理内容

各タスク・レベルでのデータ・インタフェースおよびデータ処理内容を総合的に見直し，詳細を調整する.

以上を整理すると次のような事項がわかる.

①オブジェクト中心型ソフトウェア・プロセスは，オブジェクトがどのように加工されていくかという観点からまとめたものであり設計の内容が理解しやすい．しかし，設計対象システムの規模がある程度大規模になると実際の設計にそのまま利用することはできず，各手順の並行動作を工夫する必要がある.

一方，手続き中心型ソフトウェア・プロセスは，時間の経緯に忠実に具体的システムの設計作業を表現したものであり，技術者への作業指示，工程管理などの活動にそのまま利用することができる.

②一つのオブジェクト中心型ソフトウェア・プロセスへの入力オブジェクトは，時間の推移（表の左から右へ）と共にマクロ的なものから次第に細かいものが与えられ，それに伴って詳細な検討が可能となる.

③一時点の手続き中心型ソフトウェア・プロセスには，いくつかのオブジェクト中心型ソフトウェア・プロセスが対応する.

最後に表 5 に，手続き中心型記述方式とオブジェクト中心型記述方式との比較を示す.

6. プロセス・モデル HFSP による表現

これまで述べてきたことにより，HFSP は，ソフトウェア・プロセスの機能的側面の記述では標準的設計技術を顕在的に表現するために十分な表現能力を持つことがわかったが，動的側面の記述の表現能力についてはいくつかの問題のあることが明らかとなった.

(1) 初期設計段階における設計とフィージビリティー・スタディ

HFSP では，各プロセスおよびその入出力オブジェクトが明確に定義されて実行されることになるが，実際の設計の初期段階においては，各プロセスへの入力オブジェクトが完全に揃うことはきわめて希である.

しかし，設計作業は，入力オブジェクトが完備していない状況においても実行されなければならない.

また，実用システムを設計する場合は目標性能などの重大な制約条件を越えないように設計の初期段階から繰り返し検討しておく必要がある．図 8 の中で，設計の最も初期段階においてもフィージビリティ・スタディを実行しているのは，このことを示している．これは，Barry W. Boehm の Spiral Model[3] で代表される Risk Management の一つである.

このような初期設計段階においては，不完全なオブジェクトのまま，粗い大ざっぱな設計検討を一通り行い，オブジェクトを補いながら設計検討を繰り返し，次第に詳細な検討に移行するが，この過程で各々の設計結果に基づいてその時点におけるフィージビリティー・スタディを行っている.

このような，不完全な入力オブジェクトに基づく設計プロセスの実行は，次のように進められる.

①不完全なオブジェクトの補充

オブジェクトが完備していないときは，残りのオブジェクトを想定してその不足を補う.

②プロセスの縮退，実行

オブジェクトの種類および数を想定し，ほぼ全体が補充できる場合には，本来のプロセスをそのまま実行するが，不完全な補充しかできない場合は，本来の設計プロセスをそのまま実行することはできない．このときは，設計技術者が，その入力オブジェクトに応じて縮退させたプロセス（通常は，プロセスの内容を簡略化したもの）を作成して，実行している.

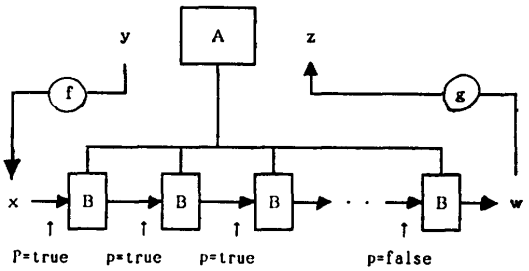


図 11 繰り返し処理 (while) の分解
Fig. 11 "while" decomposition.

(2) 繰り返し動作

繰り返しは，不十分な状況のもとに実行されるソフトウェア・プロセスとして重要な動作である．

HFSPでは，繰り返しは次のように不特定多数の同一のサブ・プロセスに分解され，表現される（図 11 参照）．

A→repeat B while C where E

A→repeat B until C where E

双方ともAは，一連のBに分解される．

A→B, B, … B

アクティビティ B は，同一の入力および出力オブジェクトを持ち，次のような形式で表現される．

$B(x_1.\text{in}, \cdots, x_n.\text{in} | x_1.\text{out}, \cdots, x_n.\text{out})$

ここで，"in" および "out" は，同一の形式のオブジェクトを区別するためのものである．

Eは，最初のBに入力オブジェクト，$x_1.\text{in}, \cdots, x_n.\text{in}$ が，どのように A の入力オブジェクトから計算され，またAの出力オブジェクトが最後のBの出力オブジェクト $x_1.\text{out}, \cdots, x_n.\text{out}$ から得られるかを定義する．

中間のBの入力オブジェクトとしては，その前のBの出力オブジェクトがそのまま与えられる．

繰り返しの動作は，Bの入力オブジェクトが，条件C ("while" 条件) を満足させず，出力オブジェクトが，条件C ("until" 条件) を満足させるときに終了する．

しかし，現実の設計作業の中における繰り返し動作としては，例えば，不完全なオブジェクトをもとにしてプロセスをいったん実行し，その後新たに追加された入力オブジェクトをもとにして設計作業を繰り返す場合等がある．このときの設計作業は，前述の HFSP 表現のような前の設計作業の単純な繰り返しではなく，新たに得た入力オブジェクトにより再検討が必要な部分およびその入力オブジェクトにより新たに検討が可能となった部分を検討し，このような動作を繰り返しつつ先へ進めるのである．

このような繰り返し動作は，共同作業の中でも存在するが，この場合共同作業を分担する作業者は，設計作業を進めながら互いに頻繁に関連情報を交換しつつ前述の繰り返し作業を行うのである．

(3) オブジェクト中心型プロセスの実行

既に述べたように，オブジェクト中心型記述方式によるソフトウェア・プロセスの表現は理解しやすく，また設計対象システムの規模の大小に関係なく利用可能であるという利点がある．

しかし，このソフトウェア・プロセスを利用して設計する場合，小規模システムに対しては，ほぼこの順序で実行することができるが，大規模システムに対しては，いくつかのプロセスを並行動作させるための工夫が必要である．すなわち，この並行作業の必要性は，対象プロジェクトの規模が大きくなるほど顕著となる傾向にある（図8，図9参照）．

DAS 設計のような大規模プロジェクトでは，一つのプロセスを実行する場合，その前後のプロセス（特に，工程的に後方に位置するプロセス）を同時に検討し，前後のプロセス間の一貫性をあらかじめ保っておかないと，先に進んで重大な手戻りの原因となる（プロセスの先読みの必要性）．

プロセス・モデル HFSP においては，各プロセスは各々一つの厳密な数学的関数によって表現されるが，プロセスの先読みは，次のように表現することができる．すなわち，一つの関数（設計プロセス）から見ると，設計の進行に伴って，次々に異なった入力データ（最初はマクロ的な入力オブジェクトで，次第に細かいオブジェクトとなる）が与えられ，この関数は各々の時点で与えられた入力データに対して可能な範囲の処理を行う．このようなプロセスの動きを表現するためには，この関数を最外置換的方法で実行する必要がある．

7. 結　び

ソフトウェア開発部門が，生産性向上，技術力向上，開発作業の管理など様々な活動を行うためには，その部門の標準技術の確立が基礎となる[7),9),10),12),14)]．このような目的のためにも標準となる技術を顕在化し，まとめることは，非常に重要な意味を持つ．

われわれは，プロセス・モデル HFSP に基づいて，現実の実用システム DAS の基本設計プロセスを分析し，抽出した．

ソフトウェア・プロセスおよびオブジェクトを抽出するに当たり，最初は，手続き中心型記述方式 (Procedure Centered Description) により，時間の経緯に忠実に設計プロセスの分析を行ったが，その結果得たソフトウェア・プロセスは，他の技術者には理解しがたいものとなった．次に，オブジェクト中心型記述方式 (Object Centered Description) により，オブジェクトを重視して分析し，これを大幅に改善した．

プロセス・モデル HFSP は，設計プロセスを明確な関数形式に表現しようとするものであり，いくつかの典型的な設計作業の表現が提案されている．しかし，実際の設計作業は複雑であり，これを定式化するためには，さらに実際の設計の進め方を分析し，これを関数表現に反映させる必要がある．

われわれは，今後ともソフトウェア・プロセスの理論と実際との差異の解析を進めると共に，プロジェクト管理についても検討も加えたいと考えている．

謝辞 本研究の機会を与えてくださった三菱スペース・ソフトウェア(株)若田和明社長，鈴木康一鎌倉事業所長に感謝いたします．

参 考 文 献

1) Osterweil, L.: Software Processes Are Software Too, *Proceedings of the Ninth International Conference on Software Engineering*, Monterey, California, pp. 2-13 (Apr. 1987).
2) Katayama, T.: A Hierarchical and Functional Software Process Description and Its Enaction, *Proceedings of the 11th International Conference on Software Engineering*, pp. 343-352 (1989).
3) Boehm, B. W.: A Spiral Model of Software Development and Enhancement, *ACM Software Engineering Notes*, Vol. 11, No. 4, pp. 14-24 (Aug. 1986).
4) Dixon, D.: Integrated Support for Project Management, *Proceedings of the 10th International Conference on Software Engineering*, pp. 49-58 (1988).
5) Balzer, R., Cheathan, T. E., Jr. and Green, C.: Software Technology in the 1990's; Using a New Paradigm, *Computer*, Vol. 16, No. 11, pp. 39-45 (1983).
6) 望月，山内，片山ほか：ソフトウェア・プロセス—実時間処理システムにおけるケース・スタディー，情報処理学会研究報告，90-SE-71, pp. 139-148 (1990).
7) 望月，山内，片山ほか：ソフトウェア・プロセスの設計教育用ツールへの適用及び評価，情報処理学会研究報告，90-SE-73, pp. 83-90 (1990).
8) 望月，山内，片山ほか：ソフトウェア・プロセスの分析及び評価，第 41 回情報処理学会全国大会論文集，2G-8 (1990).
9) 望月，山内，片山ほか：ソフトウェア・プロセスを利用した教育用ツールの開発，第 41 回情報処理学会全国大会論文集，2G-9 (1990).
10) Mochizuki, S., Yamauchi, A. and Katayama, T. et al.: Applying the Softwara Process to the Instruction Tool in System Design, *Proceedings of 6th International Software Workshop*, pp. 141-143 (1990).
11) 望月，山内：リアルタイム・システム開発のソフトウェア・プロセス，ソフトウェア・プロセス・ワークショップ配布資料 (1991).
12) 望月，山内，片山：ソフトウェア・プロセスによるシステム設計教育・訓練の試行および評価，情報処理学会研究報告，91-SE-78, pp. 49-57 (1991).
13) 望月，山内，片山ほか：ソフトウェア・プロセスの分析と評価，第 42 回情報処理学会全国大会論文集，7S-7 (1991).
14) 望月，山内，片山ほか：ソフトウェア・プロセスを利用した教育用ツールの開発（その2），第42回情報処理学会全国大会論文集，7S-6 (1991).
15) Mochizuki, S., Yamauchi, A. and Katayama, T.: Analysing and Evaluating Fundamental Design Process of Checkout System for Artificial Spacecraft, *Proceedings of the Fifteenth Annual International Computer Software and Applications Conference* (*COMPSAC '91*), pp. 507-514 (1991).
16) Katayama, T. and Mochizuki, S.: What Has Been Learned from Applying a Fomal Process Model to a Real Process, *Proceedings of the 7th International Software Process Workshop* (1991).

(平成3年9月 5 日受付)
(平成4年3月 12日採録)

望月　純夫（正会員）

昭和14年生．昭和38年東京工業大学理工学部電気工学科卒業．昭和40年同大学院理工学研究科修士課程修了．同年三菱電機(株)入社．以来一貫して，実時間処理システムの開発に従事．昭和62年三菱スペース・ソフトウエア(株)に移り，宇宙および通信関連システム部門を担当．ACM，電子情報通信学会，日本経営教育学会各会員．

山内　譲

昭和29年生．昭和51年東海大学工学部航空宇宙学科卒業．昭和53年同大学院工学研究科修士課程修了．同年三菱スペース・ソフトウエア(株)入社．現在，同社鎌倉事業所第二技術部システム技術課，課長．人工衛星の追跡管制システム開発関連の業務に従事．日本航空宇宙学会会員．

片山　卓也（正会員）

1939年生．1962年東京工業大学理工学部電気工学科卒業．1964年同大学院理工学研究科修士課程修了．工学博士．同年日本IBM(株)入社．1967年東京工業大学工学部助手，1974年助教授，1985年教授．1991年北陸先端科学技術大学院大学情報科学研究科教授，現在に至る．ソフトウェアプロセス，ソフトウェアデータベース，ソフトウェア開発環境，ソフトウェア方法論，プログラミング言語，関数型プログラミング，属性文法などの研究をおこなっている．日本ソフトウェア科学会理事長．電子情報通信学会，ACM，IEEE各会員．