TECHNOLOGY REVIEW

UNISYS 技報

通巻 40

1994年2月発刊 Vol. 13 No. 4

特集：TRITON

日本ユニシス

ISSN 0914-9996

銀行業界では昭和40年代の第一次オンラインシステム構築以来ほぼ10年毎にシステムの全面更改がされてきた。林秀雄は**銀行勘定系システムの動向と技術**の中で，まずシステムの全面更改が発生する要因について考察し，次に勘定系を中心としたポスト三次オンの銀行情報システム像を予想し，そしてポスト三次オン・システムとTRITONシステム技術について概観している．

金融業界は金利・商品の自由化，金融制度の緩和等による急激な変化が予想される．このような状況下のシステム開発は短期間・低コスト開発を目指し，解決策の一つとして共同・分散の開発形態が考えられる．TRITONの開発はこれらの動向を先取りし，かつ大規模・長期間にわたるプロジェクトとなった．片岡禧造・笠井潔の**大規模・共同・分散開発におけるプロジェクト・マネジメントの実際**は，その立ち上げから百五・紀陽両行同時本番までのプロジェクト・マネジメントについて，どのように考え，推進してきたか，開発工程毎の特徴とともに述べている．

「見えないソフトウェア開発」を「見えるソフトウェア開発」へ転化させる方策の一つに「管理可能な単位への分割」があるが，「全体が見えにくくなる」「分割された単位間の整合性をどうとるか」という課題を惹き起こす．今開発では「全体が見通せる成果物の作成」と「標準化の実施」という方策で対処した．藤井昭の**開発手順の標準化**は，その具体策として実施した全体工程定義書，各工程の標準化規定等の各種ガイドの設定について記述するとともに，開発手順上での品質維持に関わる工夫についても説明している．

今回のTRITON開発では，標準化やそれをもとにした機械化・自動化の推進により一定の効果を得，将来に向けての環境整備が図られた．現在はドキュメントなしのシステム開発・保守・利用はありえず，ドキュメントの重要性は論を待たない．村田豊彦は**ドキュメントの標準化**の中で，その体系や記述内容等の標準化と，ドキュメンテーション支援ツールによる機械化・自動化の工夫・考慮点について述べている．

TRITONではプログラムの保守性を高めるために，プログラムの理解のし易さ，修正負荷の軽減等にポイントをおき，階層化構造設計・漢字COBOLの採用，テーブルウェア・部品の活用等さまざまな工夫をしてきた．竹村匡弘は**保守性の良いプログラム構造**の中で，その工夫について具体的な内容と効果について述べている．

XTPAでは，複数ホスト間でのデータベース共用機能の提供により，信頼性，処理能力，可用性，拡張性に優れたシステムの構築を可能にしている．TRITONではXTPAの採用により，汎用機でのノーダウン・システムを実現させた．沢田啓の**XTPAに基づくノーダウン・システム**は，まずXTPAおよびTRITONシステムの紹介を行い，次にTRITONでのノーダウン・システムの基本的な考え方を述べ，さらにノーダウン・システムをどのように実現したか，営業店での実際の動きはどうなるかについて述べている．

TRITONを支える基盤ソフトウェアにIDESがある．TRITONは，大阪・東京・津・和歌山を拠点とする4か所分散の大規模プロジェクトでありながら，個々の開発工程でIDESを採用・適用したことにより，均質化したソフトウェアの開発が実現でき，開発手順の標準化が図られ，単一開発環境と同程度の生産性を保持できた．奥野信幸の**開発工程とIDES**は，TRITONの開発工程の中でIDESをどのように適用・活用していったかについて整理し，今後IDESが抱える課題・展望を含めて記述している．

TRITONのシステム基盤は，XTPAに基づきXISをベースに構築されている．その機能は従来以上にメーカ標準製品として整備されている．これにより，ユーザは開発・保守の負担が軽減され，アプリケーション開発に専念できた．宮村洋は**銀行システムとXIS**の中で，TRITONのシステム基盤について，XISをベースにどのように構築したかを紹介している．

特集「TRITON」の発刊によせて

藤田康範

金融業界におけるオンラインの進展をみるに，昭和40年代初期の第一次オンラインから平成3～4年代をピークとした都市銀行を中心とする第三次オンラインへと，約10年を一区切りとして大きく変革してきている．都度再構築に際してはその時々の環境要請による狙いと目的があり，実現レベルも高度である．

とくに第三次オンラインにおいては，合理化の推進，BIS規制・業容拡大による戦略システム化＝競合力の強化，取引量増大に対するシステム限界の打破，金融自由化を見込んだシステム構造の柔軟化等，精力的に取込んできた．

日本ユニシスはこれら第三次オンライン対応をさらに進め，向こう10～15年のシステムライフを維持できるシステムをポスト三次オンと位置付け，①より高度な信頼性，安全性の追及，②より柔軟なシステム拡張性の確保および銀行24時間稼働によるサービス向上，③集計レス・伝票レス等による一層の省力化を目指し，㈱百五銀行および㈱紀陽銀行との三者による共同開発にて実現し(TRITONシステム)，1年間の調査と3年間の開発期間を経て平成5年5月両行にて同時本番稼働し現在に至っている．

本号ではTRITONシステムを支えるインフラの構造と仕組み，および標準化の考え方と適用そして大規模開発におけるプロジェクト・マネジメントの実際を実例として紹介する．

システム基盤への要請として，①大規模（ハイボリュームトランザクション）システムの実現，②高信頼性/安全性の確保，③開発/保守の高生産性確保等が挙げられる．

実行環境としてはXTPA(eXtended Transaction Processing Architecture)，すなわちXPC(高速レコードロック制御)の提供により大型汎用機による並列処理を可能とし，①ハイボリュームトランザクション処理，②ホスト障害時の他ホストへの自動切換えによる無停止システム，③2～4システムまでのホスト拡張等の実現をしている．開発/保守環境に対してはリポジトリをベースとした統合CASEツールIDES(Integrated Development Environment support System)を適用し，プログラム品質の均一化，分散開発対応および開発標準のシステムへの組込み等により生産性の向上を計った．

一方，当開発は前述三者による大規模システムの共同開発であり，工程ごとに混成部隊の編成，業務の共通化，オーバーヘッドの極小化，同時2システムの本番および集中/分散時期の見極め等従来のプロジェクト・マネジメントにはない要素を持っており，とくに多数の開発要員が集中的に参画したプログラミング/単体テスト工程での運営体制については工夫を要した．

本号は金融業界基幹業務システムの実例ではあるが，システム基盤面および開発工程推進面については業界を問わず関心がもたれるところと認識し特集した．

システム化検討にあたり一助となれば幸いである.

謝辞

TRITON 開発において(共同開発パートナーとして)絶大なる御尽力を賜りました㈱百五銀行 佐々木取締役システム部長および㈱紀陽銀行 松本システム部長(現 紀陽ソフトウェアサービス株式会社 常務取締役) に厚く御礼申し上げます.

(金融システム企画開発本部 本部長)

銀行勘定系システムの動向と技術

Trends of Banking Accounting Systems and Related Computer Technologies

林　秀雄

要　約　銀行業界では昭和40年代の第一次オンラインシステム構築以来,第三次オンラインシステムに至る20年間でほぼ10年ごとにシステムの全面更改がなされてきた．本稿の目的の第1はシステムの全面更改が発生する要因について考察すること，第2に勘定系を中心としたポスト三次オンラインシステムの銀行情報システム像を予想すること，第3にポスト三次オンラインシステムとTRITONシステム技術について概観することである．

銀行情報システムの今後は分散協調型システムに移行していくと予想するが，勘定系の現インフラは今後10年は維持されるであろう．その理由はあまりにも勘定系システムが巨大になりすぎて一斉更改が困難なものになっていること，さらに勘定系システムの規模の拡大が過去と同様の伸び率であれば，ますます困難なものとならざるをえないためである．

技術的ブレークスルーの可能性は，AIや自然言語に近いコンピュータ言語の開発により，劇的にソフトウェアの生産性が向上することであるが，その道のりは遠い．したがって，今後の勘定系システムはサブシステムの部分的更改・連続的更改による発展形態をとるであろう．その場合，インフラは三次オンラインと四次オンラインのそれが混在するハイブリッド型にならざるをえない．

Abstract Starting with a surging wave of building first-phase on-line systems in the late 1960s, the banking industry saw over-all system changes take place every ten years for the following 20 years, leading to the current presence of third-phase on-line systems. This paper first points out the factors which have made it necessary to change systems on an across-the-board basis. Secondly, its focus is on what accounting-specific banking information systems will be like as a successor to phase-three on-line banking systems. Thirdly, the paper refers to the positioning of TRITON system technologies for the creation of on-line banking systems that will next come into the picture.

Although existing banking information systems are considered to turn into distributed cooperative computing systems (or distributed complex systems where general-purpose computers and downsized computing systems are all interlinked), the present infrastructure of accounting systems is expected to survive for another ten years to come. The reasons are (1) because those accounting systems have grown too huge to go through an entire systems recreation on "all-at-one-time" basis, and (2) such attempts would surely confront all the more difficulty if the scale of systems continues to expand at the same rate as in the past.

The potential technological break-through is synonymous with a dramatical improvement in software production efficiency through the use of AI technology and through the new development of computer languages very close to natural human tongues. But, for this goal to be reached, a long, distant way still lies ahead of us. Accordingly, it can be said that accounting systems at banking institutions will follow the direction in which their subsystems will be partially modified or continuously enhanced. In this case, the systems infrastructure will be forced to be of a hybrid type, which allows both third-phase on-line systems and-fourth-phase ones to co-exist.

1. はじめに

銀行業界では昭和61年以降第三次オンラインシステムの構築がなされてきた.昭和40年代初期の第一次オンラインシステムから現在の第三次オンラインシステムの構築に至るまで,ほぼ10年ごとに銀行は勘定系システムを中心に全面的なシステムの再構築を行ってきた.

本稿の目的の第1はシステムの全面更改が発生する要因について分析すること，第2に勘定系を中心としたポスト三次オンラインシステムの銀行情報システム像を予想すること，第3にポスト三次オンラインとTRITONの技術について概観することである.

2. 銀行情報システムの変遷

銀行情報システムは過去,約10年ごとに一次から三次に至る大規模な全面更改が行われてきた．一次から三次に至るオンラインシステムの特徴をまとめたものが表1である.

とくに，表1の開発プログラム・ステップ数に注目すると10年ごとに約4倍の規模へ増加している．そのままの増加率をこの後10年にあてはめれば，平成7〜8年には銀行情報システムの規模は2800万ステップに増加する程の勢いで拡大している.

ほぼ10年ごとに大規模なシステム更改が発生する要因は図1に示すフレームワークで説明できると考える.

第1の要因は業容の拡大による銀行システムの構造的劣化であり，システムの拡張性が次第に失われてしまうことに起因する．銀行における業容拡大は，付表1の〝銀行情報システムの歴史と環境の変化〟に示すように，金融環境の変化とともに多様な業務が付加され，進展してきた．当然それら業容の拡大に伴いシステムは追加，修正されていく.たとえば二次オン構築当時にはファームバンキング等の対外系や証券系，国際系等のシステムは存在しなかった．その後の金融の自由化，通信の自由化をきっかけに業容が拡大した結果，構築されたシステムである．当初システムを構築した時は予定していなかった各種処理が加わることにより，システムの制約値による処理能

表1　都市銀行におけるオンラインシステムの発展経緯
Table 1　Growth process of banking online system

	第一次オンライン	第二次オンライン	第三次オンライン
開始時期	昭和40年〜	昭和49年〜	昭和61年または62年〜
狙　い	省力化	機能サービス強化	金融自由化への対応
投資額	150〜200億円	250〜350億円	800〜1000億円
CPU 処理能力	1 MIPS	10 MIPS	200 MIPS
開発プログラム ステップ数	50万ステップ	200万ステップ	700万ステップ
特　徴	元帳のセンター集中 預金・為替の個別オンライン 本支店間オンライン	主要科目連動 総合口座 銀行間オンライン	勘定系・情報系等各系の有機的結合 顧客とのネットワーク強化

(金融情報システムセンター，金融情報システム白書 平成3年版より編集)

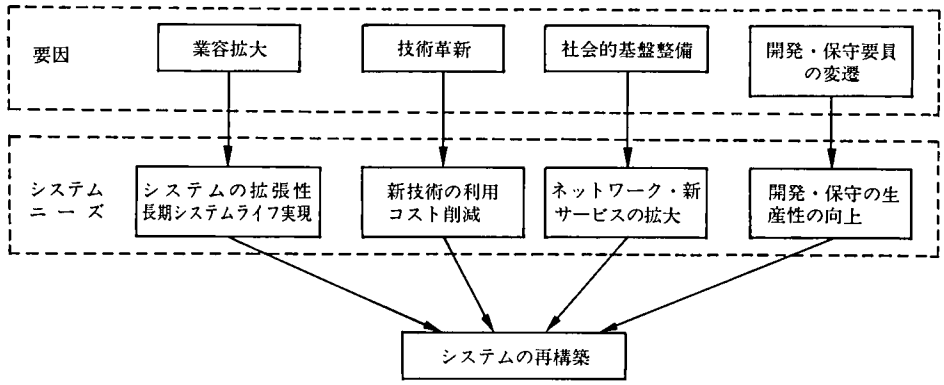


図 1　システム再構築の要因
Fig. 1　Factor of system reconstraction

力の限界が表面化したり，データベースのレイアウトに余裕がなくなったり，体系が崩れる等でシステムは劣化する．その結果，保守に多くの労力がかかるようになることがシステム更改を促す．

第 2 の要因は技術革新によるコスト低下の圧力である．現行システムを維持するよりも，新技術を適用したシステムの方が長期的にはコストがより低減することからシステムの書き換えが発生すると考えられる．技術革新によるコンピュータコスト低下は表 2 に見られるように CPU, メモリ, ディスクについて過去 20 年間著しいものがある．このコスト低下が銀行の業容拡大とデータ量の増大に対しシステムを適応させる原動力となってきた．

表 2　大型機種における価格性能比の推移(20 年前を 100 とする)
Table 2　Cost performance trend of large scale computer

機　器	20 年前	10 年前	5 年前	現　在
プロセッサ	100	20	10	5
メモリ	100	5	1	0.5
外部記憶	100	24	10	6

表 3，図 2 は大手地銀クラスのコンピュータ・リソース増加の事例である．年率 30～50 %の伸びで CPU, メモリ, ディスク容量が増加しているが，技術革新による著しいコスト低下がこのことを可能にしてきたと言える．最近はマイクロ・プロダクトの発達が営業店における分散処理を可能とし，コンピュータ間結合技術やファイル・シェアリング技術の進展によりホスト系における分散処理化もより容易になりつつある．また，コンピュータ言語面でオンラインに高級言語や第 4 世代言語を利用する動きがある．このような技術革新のもとで，新技術体系を適用したシステムの更改が長期的には低コストのシステムを実現する．

第 3 の要因は行政あるいは法律などの社会基盤の整備である．通信の自由化や NTT の民営化，第二電電の出現がファームバンキングや VAN の進展を促したごとく，社会基盤の整備は既存システムへのインパクトになる．

第 4 の要因は開発・保守要員の変遷である．10 年もすると現行システムを構築した

表3 コンピュータ使用資源の推移例
Table 3 Examples of growth of banking computer resource

	A 銀 行 例		B 銀 行 例	
	S 51～S 61 間の倍率	年率換算	S 55～S 63 間の倍率	年率換算
CPU 能力	12 倍	28%	8.2 倍	30%
メモリ容量	20 倍	35%	31 倍	54%
ディスク容量	25 倍	38%	19.3 倍	45%

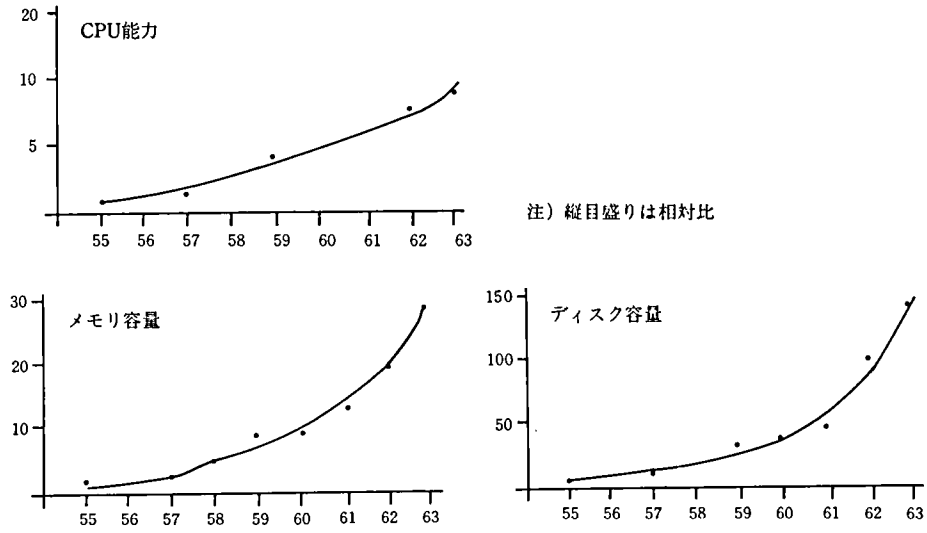


図 2 コンピュータ資源増加の状況（B 銀行）
Fig. 2 Trend of B bank computer resource

要員の多くはシステム部門から異動する．一方で業容の拡大は多くのシステム開発要員を必要とし，それは新規要員や外部委託要員によってまかなわれる（表 4）が，これらの要員は現行システムに必ずしも精通しているわけではない．全体的に見るとスキルの低下を招くこととなり現行システムを支えていくことが困難となる．要員育成の

表4 都市銀行・地方銀行のコンピュータ関連部門人員数と比率
Table 4 Manpower of city-bank and regional-bank computer section

		昭和61年	62 年	63 年	平成元年	2 年	3 年	4 年
総人員数(人)	都銀	160031	157964	154322	152122	152237	152307	155286
	地銀	165063	162692	160374	158950	158243	158825	161681
コンピュータ関連部門	都銀	3.2	3.4	3.7	3.7	3.6	3.4	3.3
社内人員比率(%)	地銀	3.2	3.2	3.1	3.2	3.2	3.4	3.6
派遣人員比率(%)	都銀	1.3	1.6	1.8	1.8	1.6	1.8	1.6
	地銀	1.1	1.3	1.4	1.3	1.9	2.9	2.9

（金融情報システムセンター，金融情報システム白書 平成 3 年版・平成 5 年版より編集）
総人員数：役員含み，パートタイマ・嘱託・臨時職員等除く
コンピュータ関連部門社内人員比率：システムの企画・開発・運用に関する部門に従事する自社人員の総人員に対する比率
派遣人員比率：総人員に対する上記部門への社外からの派遣人員の比率

観点からもシステムの更改が必要となる．

なお，銀行情報システムは一般企業の情報システムに比べて次に示す要件をもっており，これら要件を満たすためにOS，言語，データベース，データコミュニケーション等のベーシックソフトウェア以外にオンラインシステム構築支援のためのミドルソフトウェアが重要な機能を提供してきた．

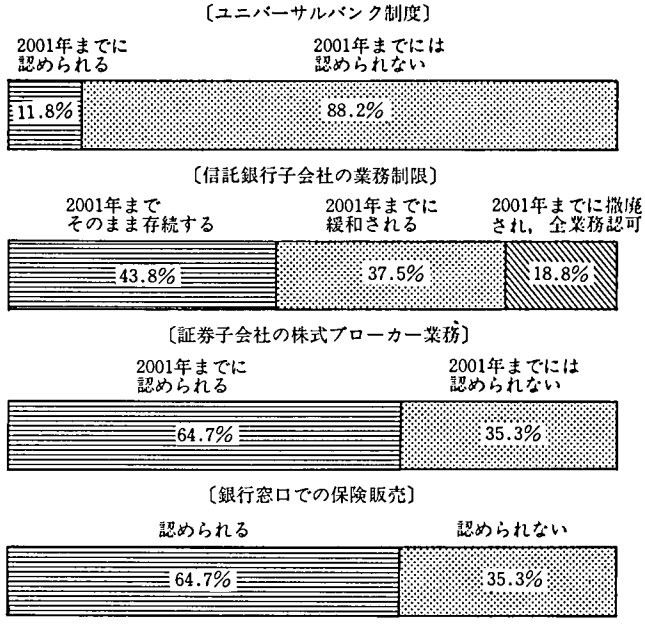
① 大量の取引をリアルタイムに処理できるハイパフォーマンス特性
② 高信頼性を求められるシステムであり，障害回復時間が極めて短いこと
③ 多様で大規模なネットワークを制御できること
④ 顧客情報等，ストックとしての情報を参照・更新するデータベース・アクセスが多いこと

弊社の例では，UNITOS，BOSS 11，AIS，XIS等の製品が10年ごとのシステム更改の際に新システムへ採用されてきた．これらベーシックソフトウェアとミドルソフトウェアのインフラソフトウェアのもとに基幹業務が標準化され構築された．

インフラソフトウェアの立場から見ると，過去の一次から三次のオンラインシステムはインフラソフトウェア変更のために全面更改が発生してきたという見方もできる．

3. 今後の金融環境の変化と情報システム

図3に示すアンケート調査にも表れているように，金融自由化は着実に進展し，金利の自由化，商品設計の自由化とともに今後も業容は拡大し，サービス競争の激化が



（富士総合研究所「2001年の金融機関の経営環境」より）

図 3　業際問題はどうなるか（有識者アンケート）（回答数構成比％）

Fig. 3　Expectation of banking business (questionaries for well-informed people)

予想される.

金融の自由化の進展により，各金融機関は「どの領域で強みを生かすか」という経営戦略が重要なものとなる．このことは，従来のように他行が実施するから一斉に同一サービスを提供するということが必ずしも発生するとは限らないことを意味する.

しかし一方で，他と差別化した商品・サービスを提供することにより，競争優位を築こうとする動きも活発になるものと考えられる.

金融機関の情報システムの今後を予想すれば，依然，システムは拡大傾向にあるものと考えられる.過去20年間における金融機関のシステムの保有プログラムステップ総量は10年間ごとに4倍，20年では16倍の増加をみている．今後も業容の拡大が続く予想であり,過去20年間と同じような割合で保有プログラムステップ総量が増加していく可能性は高い.

4. 勘定系を中心とした今後の銀行情報システムに求められる要件

本章から6章にかけてシステムニーズである「今後の銀行情報システムに求められる要件」とシーズである「コンピュータシステムおよび通信技術の進展，社会的基盤の進化」の両面から分析を行い，西暦2000年頃に至る銀行情報システムの方向性について考察する.

金融自由化のうねりの中で銀行が挑戦しなければならない経営課題は次の三点に集約される.

① 省力化・合理化によるコスト削減努力
② 競争力強化のためのマーケティング志向経営
③ リスク管理の充実

今後の銀行情報システムは，これら三つの課題の解決に向けて役立つものでなければならない．銀行情報システムに求められる要件を，これら三つの課題に対応するものと，さらに銀行システム自体に根ざすシステム要件の四つに分類するとき，次のように考える.

1) 省力化・合理化要件
- 収益構造の改善と計画・管理支援
- 集計レス，伝票レス等の営業店事務の省力化推進
- レスペーパ化による合理化推進
- システムコストの削減

2) マーケティング機能と顧客サービス機能要件
- 新規業務への進出が容易であること
- 自己完結型ビジネスからネットワーク型ビジネスへの移行を推進できること
- 商品設計の自由化に迅速に対応可能なこと
- 24時間365日稼働等の顧客サービスが充実されること
- 総合的な顧客情報管理ができる営業店支援機能が充実されること——情報の鮮度を保てる仕組み
- さまざまな切り口でのマーケティング分析が可能なこと
- 手数料ビジネスの拡大支援機能が充実されること

・応答時間が短くてユーザフレンドリな使い勝手のよい営業店の情報武装を推進できること

3) リスク管理要件

・リスクの把握，予想，防止と対策，管理等の支援

4) システム要件

・開発・保守の生産性向上

—大規模システムへの対応が可能なこと

—段階的なシステム更改が可能な柔構造システムであること

・オープン化・標準化への対応

—メーカ/ベンダの自由競争のもとでのコストダウンが可能な移植性の高い業界標準にのっとったシステム

—マルチベンダ対応の可能なこと

・大量トランザクション処理，大量バッチ処理への拡張性確保

・コンピュータ運用の自動化・無人化の推進

—レスバッチ運用

・高信頼性システム

表5　西暦2000年へ向けての技術動向

Table 5　Computer technical trend for 2000 's

技　術	動　向
CPU	汎用機は2000年頃に単一プロセッサで現在の4倍の能力に達する． UNIX/PCで使用されるマイクロプロセッサは2000年頃には現在のUNIX/PC能力の10〜40倍の能力向上が可能となる．
メモリ	現在の主流は1〜4 Mbit DRAMであるが2000年には64〜256 Mbitの素子となる．コストはbit当たり単価で1/19に低下する．
ディスク	記憶密度は現在の1平方インチ当たり100〜150 Mbitから2000年頃には1〜数Gbitへ拡大しディスクの小型化が進行する．汎用機で使用されている8〜10インチは2000年頃には2.5〜3.5インチHDへ移行する．1インチHDも使用されるようになり装置の軽・薄・短・小化が進行する．GB当たりの価格は現在の3/100〜8/100程度になることが予想される．また，2000年頃にはディスクアレイ商品が主流となり信頼性が向上する．
通　信	フレームリレー・サービス(2 Mbit/秒までのパケット交換サービス)が1994年より開始． ディジタル携帯電話サービスも1994年より開始． 156 Mbit/秒の広帯域統合ディジタル通信網(B-ISDN)サービスが一部地域で1998年より開始． 2000年に向けてLAN普及，さらに移動体通信機能と統合化された携帯型情報機器が出現． マルチメディア技術が進展．
ソフトウェア開発・保守技術	2000年に向けてオープンOSが普及しマイクロカーネル化進展． UNIX用OLTP製品普及． CASE/第4世代言語が普及しエンドユーザ・コンピューティングも進展． RDB，ODB，オブジェクト指向技術が利用されるようになる．
システム形態	分散処理を中心としたシステム形態へ移行(クライアント・サーバモデル，分散協調型システム) 汎用機は疎結合構成をとり，システムとしては単一プロセッサの数十倍の最大能力． CMOSプロセッサの並列処理形態．

5. コンピュータ・通信技術と社会的基盤の進化

銀行情報システムに求められる前述の要件を満たす技術的シーズと社会的基盤の側面の進展を本章で述べる.

西暦2000年へ向けての技術トレンドを表5に示すが,特徴的な事象をまとめると以下のようになる.

1) 技術革新の動きは引き続き力強いであろうと予想できる. とくに, マイクロプロセッサを中心とした技術革新はめざましく, UNIX*/PC等のマイクロ系機器のCPU処理能力向上は2000年に向かって現在の10～40倍へと向上する予想である.

一方,汎用コンピュータは単一プロセッサでは2000年までに4倍の能力向上であり, 複数プロセッサを組み合わせた疎結合システム構成（システム全体として最大性能は数十倍）および数十～数百台程度のプロセッサによる並列処理プロセッサへの進化の道を辿るものと予想され, マイクロ系機器と複合した分散協調型システム技術が進展するものと考えられる.

記憶装置も大容量化・小型化・低価格化が進行し, マイクロプロセッサの能力向上と相まって営業店・本部各部門への分散処理を可能とすることが見込まれる.

また, 高信頼性システムを築く上で重要なフォールトトレラント技術が一般化していくと思われる.

2) 通信技術は2000年に向けてLANが普及し, LAN間のデータ伝送を効率よく行えるフレームリレー方式のサービスが1994年より開始され普及していく見込みである.

一部地域では1998年から156 Mbpsの広帯域統合ディジタル通信網（B-ISDN）の利用が開始され, コンピュータ間・オフィス間・企業間を結ぶ大量データの転送が可能となる.

また, EDI等の標準化も進展し, 通信に関する社会的基盤の整備がなされていく見通しである.

3) ソフトウェアはオープンOSが2000年に向かって普及していくものと予想される.

CASE/第4世代言語の使用も普及し, エンドユーザ・コンピューティングが利用されるようになると考えられる.

6. 勘定系を中心とした今後の銀行情報システム像

現在, 第三次オンラインと呼ばれるシステムの機能の特徴的なものを付表2に記述する. 1章で述べたフレームワークから, 将来ポスト三次オンラインシステムの構築はいずれかの時点で行われることは必然であると思われる. 銀行情報システムに求められる業務上, システム上の各要件を今後進展する技術と社会的基盤の進化の上で考察すると, アプリケーションは図4に示す体系のもと, 次に述べる銀行情報システムの方向性が予想される.

1) システム形態は各アプリケーション系の処理特性を最も低コストで処理できる

* UNIXオペレーティング・システムはUNIX System Laboratories, Inc.が開発し, ライセンスしている.

図 4　銀行アプリケーション体系図

Fig. 4　Banking application system

分散協調型が主流になっていくものと考える．

2) 勘定系は大量トランザクションを高速で処理できる疎結合システムで構成され，高信頼化要求からフォールトトレラント化された機能を備えることになる．

疎結合システムは24時間365日稼働を可能とするダイナミックな構成変更や縮退運転機能を備えたものになる．

なお，超大規模な勘定系の場合，バックアップセンタの設置が検討される．

3) 勘定系以外の各系はダウンサイジング化されたマイクロ系機器の導入が進み，コストダウンを図ったシステム構成となることが予想される．A銀行における営業店情報系システムのダウンサイジング化に見られるように，すでに一部ではその動きが始まりつつある．

情報系基礎データベースのプラットフォームは大規模データベース・サーバ化の方向が予想される．それらデータベースの更新は勘定系その他の系からのディレード処理による鮮度の高い情報の受け渡しが必要であり，そのためにファイルシェア技術または大量データ転送可能なコンピュータ間LAN技術が適用される．情報系のアプリケーションは，クライアント・サーバモデルを利用した大規模データベース・サーバ外の機器でなされることが予想される．

4) 事務処理の合理化とマーケティング能力向上の必要性から，営業店系システムにも大幅な処理能力が付与された分散処理形態のシステムが構築されるものと予想される．このシステムはUNIX/PC等マイクロ系機器を利用してコストダウンを図った構成で高性能のCPU能力と情報記憶能力を保持しており，営業店ローカルデータベースを構築した上でセンターホストと連携してペーパレス，収益管理，顧客管理，マーケティング分析が可能なシステムとして利用される．

5) ネットワーク系は営業店～ホスト間の大量データ伝送が可能な低コストのISDNへ切り替わっていくことが予想される．

また，OSIや業界標準のプロトコルによる通信が一般化するであろう．

6) ファームバンキング，ホームバンキング，金融VAN等の金融機関外部ネットワークとの結合が通信インフラの整備進展でより拡大するであろう．したがって，対外接続システムは大規模化すると予想される．また，対外接続システムもフォールトトレラント化による高信頼性と24時間365日稼働を前提としたシステム構成が必要となり，かつ高度なセキュリティ機能に重点を置いたシステムが構築されるものと考えられる．

7) ソフトウェアに関しては，制御系ソフトウェアはメーカ標準品または業界標準品を利用し，ユーザはアプリケーションシステム構築に専念することになる．いくつかのシステムではUNIX等のオープンOSをベースとした構築がなされ，パッケージの利用やソフトウェア流通が本格的に行われるようになると考えられる．とくに，新規ビジネスはパッケージ化のニーズが高く，流通ソフトウェアまたは共同開発が多くなるものと予想される．

一方で金融自由化の結果，金融機関がどのような差別化した戦略をとるかにより，個別開発に対するニーズも種々発生するであろう．差別化を有効なものにするためにも迅速にシステム開発を行う必要がある．CASEツール，第4世代言語

等を利用して開発・保守の生産性を向上させた開発環境が構築される．

8) 業務系ソフトウェアの構造はサブシステム分割されて互いに影響を及ぼすことを極小化した．したがって，今後の環境変化によって受ける影響を極小化した柔構造が必要になる．

従来の一〜三次オンのように一斉に全システムを再構築することが不可能なほど，システム規模が拡大しており，次期システムでは連続的・部分的システム更改が可能なサブシステム分割構造にならざるをえない．

なお，これらの方向性にそって想定したポスト三次オンの機能を付表3に示す．

7. TRITONの技術と銀行情報システム要件

前章までの議論を踏まえ，本章以降で今後のバンキングシステムのあるべき姿がTRITONシステムでどのように実現されているかを述べる．

7.1 TRITONシステム構成

TRITONシステムのハードウェア構成とソフトウェア構成およびアプリケーション構成をそれぞれ図5，図6，付表4に示す．

ハードウェア構成はUNISYS 2200シリーズ汎用コンピュータを複数利用し，データベース，回線を共有しあう疎結合構成である．

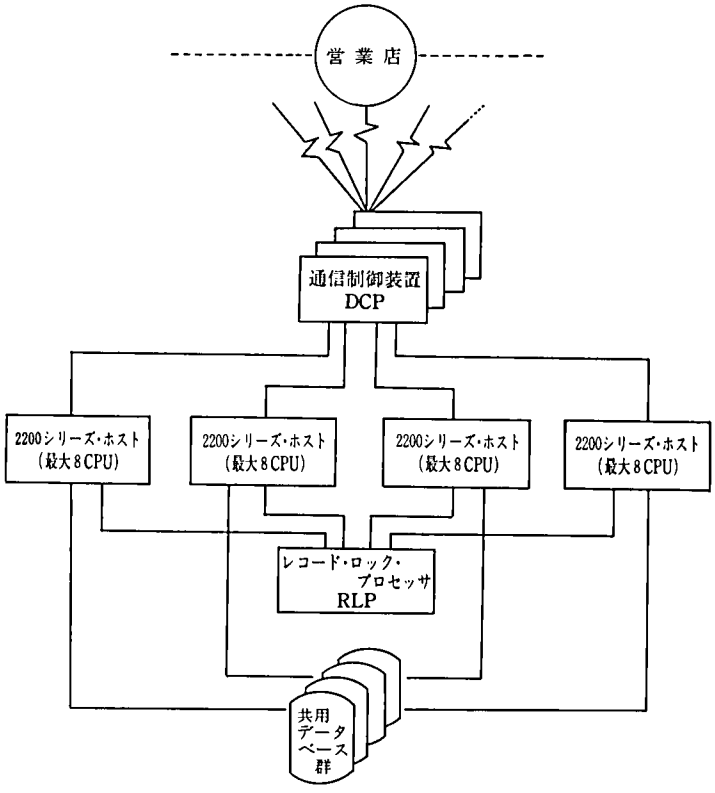


図5　TRITONハードウェア構成

Fig. 5　TRITON hardware configuration

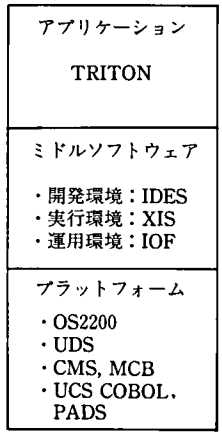


図6　TRITONソフトウェア構成

Fig. 6　TRITON software configuration

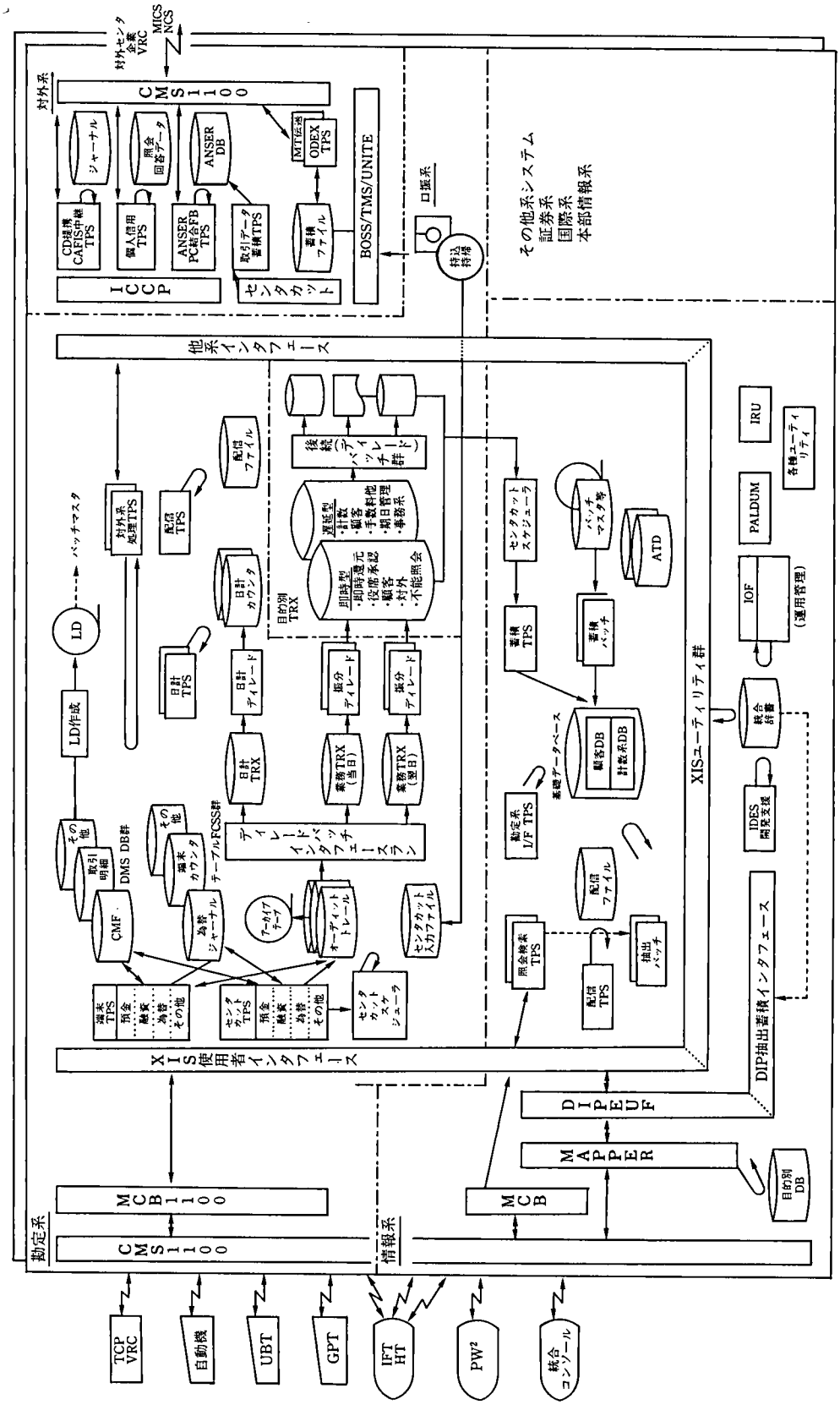


図 7 TRITON システム概要

Fig. 7 Outline of TRITON system

ソフトウェアは最新鋭 UNISYS ベーシックソフトウェアのもと，開発環境支援ソフトウェア “IDES” (Integrated Development Environment support System)，実行環境支援ソフトウェア “XIS” (eXtended Information System)，運用環境支援ソフトウェア “IOF” (Integrated Operating Facility) の UNISYS ミドルソフトウェアを採用している．

アプリケーションは勘定系システムを中心に情報系や対外系の一部機能および他系システムとのインタフェース部分をも含むものである．図7に TRITON システム概要図を示す．

7.2 開発環境技術

TRITON システムは図8に示すような地域分散でウオータフォール型の開発を実施した．大規模開発の生産性と開発後の保守性を向上させるために金融機関のシステム開発・保守に向いたドキュメントの整備，工程・作業手順の標準化（表6）を実施し，新プログラミング環境のもとで統合開発支援ツール IDES を使用した．

新プログラミング環境の特徴は次の通りである．

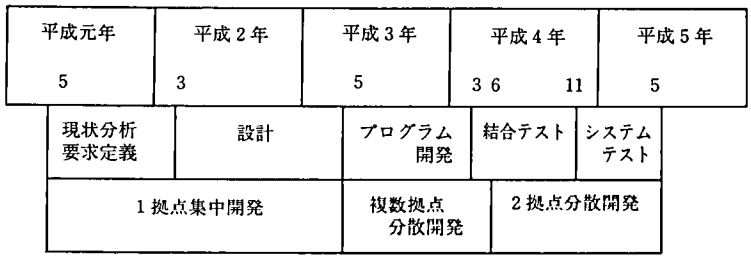


図 8　TRITON の開発工程

Fig. 8　TRITON development process

表6　主な TRITON 標準化マニュアル

Table 6　TRITON standardization manual

工　程	標準化マニュアル
設　計	基本設計ガイド 基本設計便覧 バッチ標準化資料
プログラミング	分散運用ガイド 分散運用ガイド管理詳細編 モジュール仕様書記述要領～リアル編 バッチ編 コーディング規約　～リアル編 バッチ編 エラー文言ファイル作成要領 入力編集テーブル作成要領 出力編集テーブル作成要領 単体テスト実施要領
テスト	結合テストガイド 端末打ち込みテスト実施要領 センターカットテスト実施要領 トラブル追求ガイド ユーティリティ使用ガイド

1) UCS（Universal Compiling System）言語によるアドレス空間の拡大，構造化機能を取り込んだCOBOL 85の活用による日本語機能充実

2) リンキングシステムによる動的・静的連結編集機能を利用してモジュールの独立性を実現

3) PADS（Programming Advanced Debugging System）による対話型デバッグ

4) 第4世代言語MAPPERとDIP* を利用した情報系プログラミング環境

また，統合開発支援ツールIDES（図9）の特徴は以下の通りである．

1) リポジトリを中心とした開発・保守……データ項目，ファイル，プログラム等のシステム開発・保守過程で発生する設計・開発用データを一括して管理するために開発用リポジトリを利用する．

設計・開発用データはリポジトリを使い体系的に保守することによって一貫性と完全性を保証し，大規模開発における膨大な設計・開発データの収集・参照を可能とし，かつ保守工程において生産性を向上させる．

2) 開発・保守作業とその管理の機械化……UNIX/PCというマイクロプロダクトを有効活用しホストと連携して適用することにより，開発・保守作業とその管理の機械化ならびに開発環境投資の低減を図っている．低価格PCとLAN環境を利用した一人一台の端末環境，UNIX/PCの持つ操作容易性は開発要員の生産性を向上させる．

プログラム以外に設計文書を電子ファイル化し，作業の機械化，仕様書・プログラムの標準化を推進することが可能になる．従来の紙と鉛筆による成果物では作業の合理化の余地は少ないが，設計書をはじめとして成果物を電子ファイル化する事により機械化による合理化の可能性が出てくる意義は大きい．

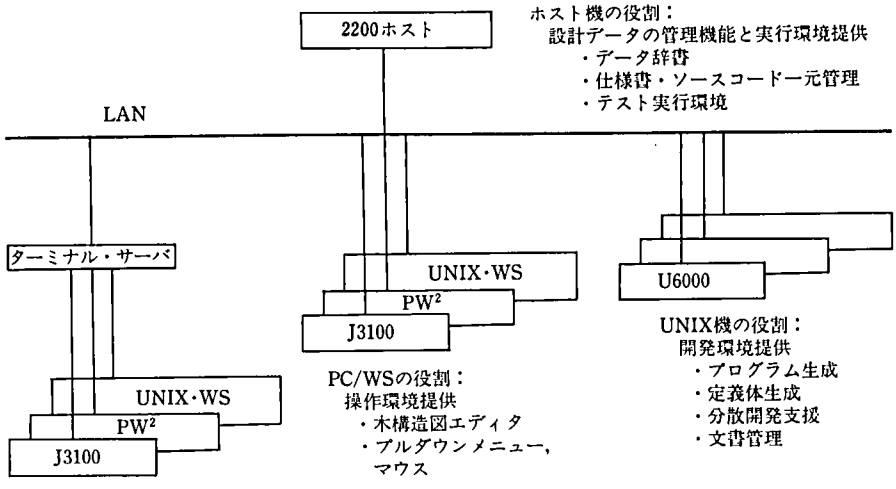


図 9 IDESのハードウェア構成

Fig. 9 IDES hardware configuration

* DIP (Dynamic database Interface Package)：情報系基礎データベース・目的別データベースへのデータ蓄積・検索・加工を支援するミドルソフトウェア．

3)　分散開発……地域分散開発を可能とするための高速回線を利用した分散リポジトリの同期機能，ファイル転送機能，ホストアクセスパスが整備されている．分散開発機能を利用して TRITON では 4 拠点 9 地域に分かれた開発が行われた．

7.3 実行環境技術

TRITON システムにおける実行環境技術の主な特徴は以下の通りである．

1)　分散処理技術……UNISYS の分散処理技術 XTPA（eXtended Transaction Processing Architecture）を適用した．

XTPA は最大四つの独立したホストを疎結合してデータベース，回線をそれぞれのホストから共有することができる．XTPA により大量リアルタイム処理への拡張性と無停止連続処理を実現できる．

2)　総合オンラインシステム支援技術……ユーザが総合オンラインシステムを構築するための支援を行うミドルソフトウェアとして図 10 に示す XIS を活用した．XIS の特徴は次の通りである．

①　OS の持つ分散処理技術 XTPA や統合リカバリ機能をユーザが容易にかつ効率よく利用できるための環境を提供する．

②　実行環境上の情報をリポジトリへ一元管理することにより保守性向上を図ることができる．

③　ハードウェア，OS 環境等，プラットフォームの変化からアプリケーションを保護するための仮想化を実現する．今後，汎用機とオープン機器との連動機能が重要さを増すであろう状況では，このシステムの仮想化の考え方がシステム更改上の大きなポイントとなる．

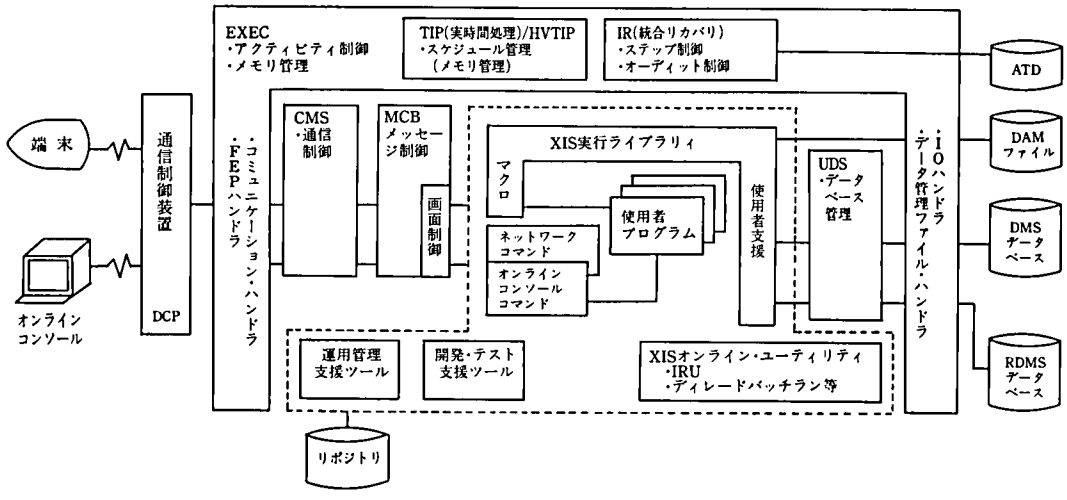


図 10　XIS とオンライン機能構成

Fig. 10　XIS and structure of online system

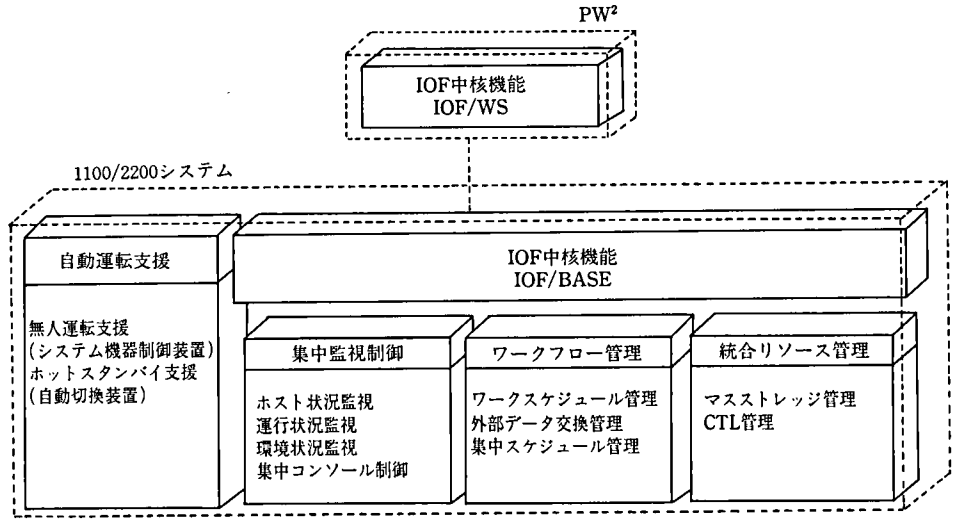


図 11　統合運用システム IOF の構成

Fig. 11　Function of IOF

④　総合オンラインシステムを構築するための機能を標準提供する.

●センターカット運行機能，ディレード処理支援機能，
データベース/データコミュニケーション・インタフェース
オンラインシステム監視・運用機能等

7.4　運用環境技術

TRITON のバッチ運行，オンライン監視等の運用環境構築を支援するミドルソフトウェアとして IOF を採用している. IOF は分散化したコンピュータ資源を統合して管理・運用することができる. したがって，複数コンピュータの運用をシングルシステム・イメージで行うことができる.

IOF の構成および機能は図 11 に示すが，主な特徴は次に示す通りである.

1)　複数ホスト・コンピュータの運行を集中して監視・制御することができる.

2)　バッチ運行の無人化・自動化をすすめることができる.

3)　バッチジョブのスケジューリングや外部データ交換をリポジトリを介して制御し，効率良く運行管理することができる.

7.5　金融アプリケーション基盤技術

TRITON の金融アプリケーション基盤として特徴的なものを以下に示す.

1)　24 時間 365 日稼働への考慮……今後の金融取引時間帯の拡大に備えて業務の共通事項として，

①　オンライン終了を意識しない

②　静的なファイル状態を要求しない

③　先日付処理等による処理の平準化が可能

④　センターカットの実行順序の制約のないこと

⑤　日付の考え方の明確化

を考慮している．また，システム的側面でも

① ノーダウン

② オンライン稼働中のハードウェア保守・ソフトウェアバージョンアップ

③ 休日縮退運転

④ 動的ファイルガベージ

等の考慮がされている．

2) 集計レス・システム……事務合理化のための機能として受付単位貸借の自動照合や係別（テラー，出納，現金自動管理装置等）現金有高管理のオンライン化を実現している．

3) 構造化・部品化された保守効率向上のためのプログラム構造……階層構造設計によるデータエリア，プログラム構造であり，表7に示されるテーブルウェアを活用することにより，プログラム変更・追加の発生を低減している．また，マクロ命令を多用することによりデータベース，テーブルレコード処理を部品化している．

表7 TRITON テーブルウェアの例
Table 7 Examples of TRITON tableware

テーブルウェア対象	テーブル内容
端末電文の編集	入力編集テーブル 出力編集テーブル
業務処理対象商品属性	預金商品ファイル 融資商品ファイル
センターカット関連情報	走行制御情報テーブル 流量制御情報テーブル 処理決定情報テーブル センターカット統合管理ファイル
帳表情報定義	帳表定義データベース

7.6 TRITON システム評価

TRITON システムを第三次オンライン機能および今後予想されるポスト第三次オンラインシステム像と比べるとき，評価は次のように考える．

第1にアプリケーション機能面では第三次オンライン機能の特徴をすべて満たしている．また，ポスト三次オンライン機能の一部，集計レス機能や24時間365日稼働機能等も先取りしていて，現時点で最も進んだ勘定系パッケージの一つと評価できる．

第2にシステム機能面も第三次オンライン機能のすべてを満たすと同時にポスト第三次オンライン機能の一部を取り込んでいる．すなわち，ポスト第三次オンライン機能として，開発環境面ではCASEツール・第4世代言語・分散開発環境の機能が取り込まれており，実行環境面には分散協調型システム・完全無停止・コンピュータ負荷の平準化等の機能を備えたシステムとなっている．また，将来のオープン・システムとの連動も，TRITONの仮想化されたシステム構造のもとでメーカ提供の実行環境支援システムXISがその機能を提供可能な構造である．

8. おわりに

本稿では銀行業界における業容拡大，技術革新，社会的基盤の整備，開発要員の変遷がシステム更改の要因になることを述べ，金融機関の環境変化の中におけるニーズと技術革新の予測から今後の銀行情報システム像を予想した．

今後の銀行情報システムは分散協調型システムへ移っていくであろう．ただし，勘定系の現インフラは今後10年は維持されるであろうと予想する．その理由はあまりにも勘定系が巨大になりすぎて一斉更改が困難になっていること，加えて過去20年間の傾向から10年で総プログラムステップ保有量が4倍になるとすれば，ますます一斉更改は困難なものとならざるをえないからである．

技術的なブレークスルーの可能性は，西暦2000年をすぎてAIの利用や自然言語に近いコンピュータ言語の開発が完成し，劇的なソフトウェア生産性の向上が実現することにかかっているが，その道のりはまだ遠い．

したがって，今後の勘定系を中心とするシステムは部分的更改・連続的更改によって徐々にシステムのインフラが変化・並列置換していかざるをえない．すなわち，三次オンラインと四次オンラインのインフラが混在するハイブリッドシステムが予想され，その実現へ向けての対応が課題となる．

TRITONシステムはポスト三次オンライン機能をかなり取り入れ，かつ将来における変化への対応として仮想化された構造で作られているが，今後も新技術の適用と金融アプリケーション変化への対応に向けて不断の努力が必要であると認識している．

付表1　銀行情報システムの歴史と環境変化

App. Table 1　History of banking online system and change of banking environment

年　月	銀行の機械化	金融制度・商品・サービス・行政
昭和40年2月	三井銀行預金オンライン開始	
昭和43年7月	全国地方銀行データ通信システム開始	
昭和46年8月 9月	オンラインCD設置(自動支払機)	 公衆電気通信法改正(第一次データ通信回線解放)
昭和47年1月 8月		財形貯蓄制度開始 総合口座取扱い開始
昭和48年4月 7月 12月	第一次全国銀行データ通信システム開始 オンラインAD設置(自動預入機)	 2年定期預金取扱い開始
昭和49年11月	富士銀行第二次オンライン稼働	
昭和50年11月	NCS開始	
昭和51年10月	全国信用金庫データ通信システム開始	
昭和52年1月 4月	オンラインATM設置(自動預入・支払機)	割引国債(期間5年)発行～国債の多様化へ 通産省「電子計算機システム安全対策基準」策定
昭和54年2月 5月 12月	第二次全国銀行データ通信システム拡充	 譲渡性預金取扱い開始 外国為替および外国貿易管理法一部改正(原則禁止から原則自由へ)
昭和55年3月 4月 10月	SICS開始(6都銀キャッシュサービス) TOCS開始(都銀オンラインキャッシュサービス) SCS開始(全国相銀CDネットサービス) ACS開始(地銀CD全国ネットサービス) SNCS開始(信金ネットキャッシュサービス)	
昭和56年3月 5月 6月 8月 10月	 ANSERサービス開始(音声自動応答システム)	邦銀58行SWIFT加盟 新銀行法成立 新型期日指定定期預金取扱い開始 信託ビック取扱い開始 長信銀ワイド取扱開始
昭和57年1月 4月 10月	ファクシミリ振込通知サービス開始	 金取扱い業務開始 公衆電気通信法改正 (第二次データ通信回線解放) 財形年金貯蓄制度開始
昭和58年4月 5月	SOCS開始(信託銀行オンラインキャッシュサービス)	公共債窓販取扱い開始 変動金利型ローン取扱い開始
昭和58年7月 8月 10月	銀行・企業間オンライン接続開始 FBの全銀協プロトコル制定	 国債定期口座取扱い開始 地銀バンクカードサービス開始

昭和59年1月	BANCS開始(都銀CDオンライン網一本化)	
2月	CAFIS開始(クレジット情報システム)	
4月	クレジットカードによるCDキャッシングサービス開始	海外CD/CPの国内販売開始
5月		大蔵省「金融機関のコンピュータと顧客端末機等とを通信回線で接続したオンライン処理による資金移動取引について」通達
6月	ペイバイフォンサービス開始	公共債ディーリング開始
10月		一括支払いシステム取扱い開始
11月	CNS開始(地銀データ伝送システム)	CAPTAIN端末による資金移動サービス開始
12月	銀行POS実験開始	
昭和60年1月		通産省「システム監査基準」策定
3月		市場金利連動型預金(MMC)取扱い開始
10月		大口定期預金の金利自由化
昭和61年5月		パソコン資金移動認可
7月	CD稼働時間延長	
8月	CD/ATM土休稼働	
昭和62年4月	都銀共同CNSセンタ開始	
5月		電気通信事業法一部改正(国際VAN自由化)
7月	ファミコンによる「ホームトレード」開始	
11月	第三次全国銀行データ通信システム稼働	
昭和63年2月	全銀協ICカード標準仕様制定	
10月	日銀ネット開始 第三次全銀MTデータ伝送，個人信用情報サービス開始	
平成元年2月		相互銀行の普通銀行転換
6月		小口自由金利商品の取扱い開始(スーパーMMC)
		債券先物取次業務開始
10月	ファミコンによるホームバンキング開始	
平成2年2月	MICS開始(全国キャッシュサービス)	
3月	都銀懇，オフライン銀行POSの標準化決定	
5月		金融機関のラジオ広告解禁
10月	多機能電話を用いたEBサービス開始	
平成3年11月	全銀協が全銀手順の適用回線としてISDN追加	
平成4年6月		金融・証券制度改革法成立
		新型貯蓄預金取扱い開始

(金融情報システムセンター，金融情報システム白書より編集)

付表2　第三次オンライン機能
App. Table 2　Function of 3rd banking online system

分　類		機　能
アプリケーション機能		
	合理化・省力化機能	① 新端末機・自動機器拡充による事務のスピード向上・無人化推進 ② FB, HB 等の外部接続機能充実とネットワーク拡大による営業店事務の削減 ③ センターカット処理拡大による営業店事務削減 ④ 勘定締め上げ処理の簡素化，または集計レスシステムの導入による勘定締め上げ時間の短縮 ⑤ 連動取引拡大・ネット扱い対象科目の拡大によるオペレーションの簡素化 ⑥ 地区センター等後方事務処理機能の充実 ⑦ 伝票・帳票種類の削減とシステム伝票の導入 ⑧ 一線完結処理の推進 ⑨ オンライン対象科目の拡大 ⑩ 融資の稟議～審査～実行の一連の流れのオンライン化 ⑪ 予約取引の拡大による事務の平準化 ⑫ 本部 OA 推進 ⑬ 手数料管理システム・経費管理システム
	マーケティング機能と顧客サービス機能	① 自動機新サービスとサービス時間延長対応 ② 新端末による漢字化推進，顧客情報の漢字化 ③ FB，HB の休日稼働・時間延長 ④ 名寄せ機能の充実と顧客情報内容の充実，およびディレードバッチを利用した鮮度の高い情報の保持 ⑤ 取引情報の長期保存 ⑥ 渉外支援システム充実 ⑦ 相談業務支援システム充実 ⑧ 窓口セールス機能充実 ⑨ 営業店からの照会機能強化
	リスク管理機能	① 稟議～審査機能の充実 ② 役席管理支援機能の充実 ③ ALM，原価計算システム ④ 営業店計数管理システム
システム機能		
	開発環境支援機能	① 第三世代高級言語採用 ② メーカ標準制御系ソフトウェア導入 ③ パッケージソフトウェアの積極的採用 ④ 開発支援ツールの充実 ⑤ 開発作業の標準化・ドキュメントの整備 ⑥ 設計仕様書の電子ファイル化 ⑦ システム構造の階層化・部品化・仮想化
	実行環境支援機能	① ホットスタンバイ機能 ② ファイル・回線の二重化 ③ CPU 間 LAN，またはファイルシェア技術による業務各系の有機的結合支援機能 ④ 業中センターカット支援 ⑤ ディレードバッチによるレスバッチシステム支援
	運用環境支援機能	① バッチシステムの高速化機能 ② 無人運転支援機能

付表3　ポスト三次オンラインシステム機能

App. Table 3　Function of 3 rd banking online system

分　類		機　能
アプリケーション機能		
	合理化・省力化機能	① 第三次オンライン機能の更なる拡充による合理化・省力化の推進 ② とくに処理能力が向上したマイクロ系機器導入による営業店OA・本部OAによるレスペーパ化推進 ③ 同じく能力向上したマイクロ系機器利用による審査事務の省力化・省脳化 ④ 勘定系以外のイメージ情報・映像情報への対応 ⑤ 携帯型情報機器利用による渉外活動支援
	マーケティング機能と顧客サービス機能	① 24時間365日稼働による金融サービス ② FB，HB等外部接続メニューの充実と金融VANへの取り組み推進 ③ 新規業務充実 ・証巻，信託，保険，不動産… ④ データベースマーケティング・システム ⑤ マイクロ系機器を利用した営業店情報システムの構築
	リスク管理機能	① 総合利益計画管理システムの構築 ② 顧客別採算管理システム ③ トレジャリシステムの構築
システム機能		
	開発環境支援機能	① CASEツールの全面利用 ② 第4世代言語のオンラインソフトウェアへの適用 ③ エンドユーザ・コンピューティング機能の普及 ④ 分散開発環境 ・垂直分散機能(マイクロ系機器利用を主体とした分散開発) ・水平分散機能(組織的・地理的分散開発) ⑤ ISO 9000-3への取り組み
	実行環境支援機能	① 分散協調型システム ・複数ホストをネットワーク化し同一端末から単一システムビューで使用 ・サーバ，WS/PCへの処理機能分散 ・オープン機能による異機種マシンの結合機能 ② 完全無停止システム ・高信頼化技術による障害耐性 ・計画停止不要の設計 ～システム構成変更，プログラム変更，ファイル管理 ③ 外部接続増に応じた暗号化等セキュリティ機能充実 ④ 並列処理システムによる大量トランザクション・大量バッチ処理の効率化 ⑤ ISDNを利用した接続機能，ネットワークOSI対応 ⑥ コンピュータ負荷の平準化機能
	運用環境支援機能	① 自動運用システム ・ダイナミックリソース割当機能 ② 遠隔地無人運用システム ③ 異機種間統合運用システム ④ バックアップセンタ運用支援システム

付表4　TRITON アプリケーション構成
App. Table 4　Detail of TRITON application

カテゴリ	科　目	機　能
システム共通		障害管理，セットアップ/ターム，オンライン運用，研修
勘定系共通		24時間(先日付，入金待ち)，サンデーバンキング(ミニ元帳，本元帳)，センターカット，期日管理，不能管理，金利管理，重要用紙管理，残高証明書，事故・注意，役席承認・役席照合，窓口セールス支援地区センター処理
顧客管理 融資先管理		CIF，採番管理，軒先管理，名寄せ 稟議(受付，稟議書作成，支払口管理，顧客情報管理，信用情報管理) 担保(預金担保，有価証券担保，不動産担保，保証，その他担保)
預金共通		自動機取引，自動訂正，カード発行，雑益編入，端末予約，移管，㊙㊢管理
流動性預金	当座 普通 別段 納税 従業員 新型貯蓄	一般，㊙，パーソナル口 一般，総合口座 個別管理，残高管理 納税一般口，納税貯蓄組合 一般，住宅積立口 I型(40万型 or 30万型)， II型(20万型 or 10万型)
固定性預金	定期 通知 積立型定期 定期積金 財形 譲渡性	一般(通帳式，証書式)，社員預金，スイング定期 自由金利型定期，市場金利連動型定期，変動金利定期
融資	商業手形 手形貸付 証書貸付 代理貸付 住宅金融公庫 新型当座貸越 カードローン 支払承諾	
為替		直送，打溜(当日，翌日)，振込機，公金送金隔地払，資金入金待対応
機能サービス		登録総振，登録給振，登録集金，定額自動送金，資金集中配分，住民税納付サービス，ペイバイフォン，ファームバンキング(PC サービス)，バンクカード
日計・精査		精査，日計登録，収支ペア，回付処理，現金オンライン管理
決算		月次決算，期末決算
口座振替		契約管理，受付管理，口振照会，不能管理，返却処理
その他科目・業務		債券窓販，仮受仮払金，利子諸税，消費税，諸手数料，経費，県証紙
外部接続インタフェース		ANSER，データ伝送(CPU 結合/PC 結合) 全銀，CD 提携 CAFIS 接続(BANK POS，キャッシング) 東証データ，海外店データ(MARK III)

サブシステム接続インタフェース		情報系インタフェース ・本部情報系：データ量，事務量，原価計算，ALM ・営業店情報系：顧客管理，計数統計，渉外支援，期日管理 国際系インタフェース，金利表示ボード 個人信用情報登録・照会，COSMOS II， 自動機集中監視
即時還元バッチ		不能照会，重要用紙，役席承認・役席照合要処理明細主要取引明細
統計バッチ		計数更新，端末照会，統計出力
情報バッチ		顧客管理
事務バッチ		預金系バッチ，融資系バッチ，為替その他バッチ，DM

参考文献 [1] 金融情報システム・センター編，金融情報システム白書，昭和63年版～平成4年版
[2] 富士総合研究所編，2001年の金融機関の経営環境
[3] 原田公敬・城代優二，“リポジトリシステムのあり方とその実現に向けて”，UNISYS技報，Vol. 13, No. 2, 1993. 8.
[4] 儀間哲雄，“IDESのリバース機能”，UNISYS技報，Vol. 13, No. 2, 1993. 8.

執筆者紹介 林　秀雄（Hideo Hayashi）
昭和46年早稲田大学理工学部数学科卒業．同年日本ユニシス(株)入社．58年慶應大学経営管理研究科卒業．TRITON開発等の金融システム開発および，都市銀行・地方銀行のSEサービスに従事．現在，金融システム第一本部都銀システム一部部長．日本システム監査人協会会員．

大規模・共同・分散開発における プロジェクト・マネジメントの実際

Project Management for Developing a Large-scale Distributed Cooperative Computing System

片岡禧造, 笠井 潔

要 約 これからの金融業界は金利の自由化，商品の自由化，金融制度の緩和等により急激な変化が予想されている．このような状況下でのシステム開発は，短期間，低コスト開発を目ざすことになり，その解決策の一つとして共同・分散という開発形態をとることも多くなると想定される．

次期金融機関向け勘定系システム（TRITON）の開発はこれらの動向を先取りし，かつ大規模，長期間にわたるプロジェクトとなった．

本稿では，プロジェクトの立ち上げから2銀行同時本番開始までの各開発工程ごとのプロジェクト・マネジメントについて，どのように考え，どのように推進してきたか，を開発工程ごとの特徴とともに述べている．

本稿は，大規模システムにおいて共同・分散開発を担当するマネージャが本稿で述べているプロジェクト・マネジメントの実際を理解し，その上に改良・工夫を加えて失敗のないシステム開発ができる一助となることを目的としている．

Abstract Under the pressure of deregulated interest rates as well as financial goods and loosened finance systems, the banking circle is expected to undergo drastic changes. In such situations, the creation of new systems, mostly distributed cooperative computing systems to the authors' way of thinking, will have to be done in search of less time and lower costs required for development efforts.

With those trends foreseen, the development of the TRITON system, a next-generation accounting system for financial institutions, started and has proved to be a large-scale project which has consumed a great deal of time. This paper describes what the authors thought out in quest of the appropriate ways to manage the project, process by process, during a long-term period ranging from the project's birth to the simultaneous starting of the system's production runs at two banking businesses, and also mentions how the authors dealt with the perceived needs, additionally referring to the characteristics of each development process.

The objective of this paper is to help banking systems development project managers, if slightly, to avoid failures in their efforts by fully learning about what has really been done, as described here, and by newly adding on their own ideas for further improvements in project management.

1. はじめに

弊社の金融機関向け勘定系システムの商品開発を(株)百五銀行殿，(株)紀陽銀行殿と三者共同で検討開始したのが平成元年5月であった．

以来4年間にわたる開発のもと，平成5年5月6日に両行は目標通り本番稼働を開始した．この開発はプロジェクト・マネジメントの分野から考えると，以下の大きな

特徴を持っていた.

- 大規模開発
- 三者共同開発
- 一か所集中開発→四か所分散開発→二か所併行開発→二行同時本番
- CASE ツール（IDES*）の本格採用
- 次世代インフラの先取り採用（XTPA, UDS, UCS, XIS)*

今後の金融機関におけるシステム開発は，共同・分散を前提条件とした開発が多くなることが予想される．このため今後の TRITON の導入適用の中で，至らない部分の改善を進めながら弊社のプロジェクト・マネジメント技術の向上に寄与しなければならないと考えている．本稿では，筆者等が両銀行のマネジメントと一体となって実践したプロジェクト・マネジメントを紹介する．今後の大規模システム開発に役立てていただければ幸いである．

2. 開 発 経 緯

2.1 TRITON 開発の背景

金融業界は，昭和 60 年から始まった金利の自由化，商品の自由化等金融制度の緩和により，急激な変化の時代，競争の時代に入り，抜本的なシステム対応が急がれていた．そして，これらのニーズと革新的に進歩したコンピュータ技術を先取りした大手都市銀行等においては第三次オンライン・システムが稼働し始めていた時期でもあった．この情勢を踏まえて，昭和 63 年，弊社内に次期勘定系システム検討プロジェクトが発足した．

一方，(株)百五銀行殿，(株)紀陽銀行殿においても，今後の量とスピードを重視した大規模トランザクション処理に加え，金利自由化対応，迅速な新商品・新サービス対応等，自由化に対応できる次期システム構築の必要性が認識されていた．しかしながら先行都市銀行等の第三次オン開発実績によると，その開発規模（工数）は 20,000 人月前後，地銀でも 10,000 人月前後となっており，単独開発は困難な状況になっていた．

ここで上記プロジェクトは共同開発を強く意識し，バンキングシステムの共同開発の事例として，信託銀行五行による共同開発，都市銀行二行の共同開発，および地方銀行二行の共同開発の事例を調査した．

これらの事例では，共同開発の狙いとして共通的に以下の点が挙げられていた．

- 開発コストの削減
- 開発要員の確保
- 開発期間の短縮
- 相互ノウハウの提供
- 良質なシステム

---

* IDES (Integrated Development Environment support System)：統合開発環境支援システム
XTPA (eXtended Transaction Processing Architecture)：拡張トランザクション処理アーキテクチャ
UDS (Universal Data System)：汎用データ・システム
UCS (Universal Compiling System)：汎用プログラミング・システム
XIS (eXtended Information System)：統合オンライン・システム

・事務処理の標準化

また，共同開発の問題点としては共通的に以下の点が挙げられていた．

・調整の手間による開発の長期化

・独自性の低下

規模を同じくする地銀二行による大型の共同開発は銀行界でも例がなかったが，共同開発のメリットを高く評価し，問題点については開発上存在するリスクと割り切り両行にもご賛同いただき，三者間で「共同開発検討確認書」を交わした上で平成元年5月に共同開発をめざしたNBS（New Banking System）開発検討プロジェクトが発足した．

2.2 三者共同開発プロジェクトの発足

NBS開発検討プロジェクトの目的は「本当に三者共同開発が期待通りにできるか否か」の結論を出すことであり，開発工程上は，要求定義の概要設計を進めることであった．

概要設計の具体的作業は，①両行の現行システム対比調査と共通化検討，②次期システムの特徴となるべき項目についての目標設定討論，課題検討書作成であり，この検討結果をとりまとめ，次期システム概要書，次期システム開発計画書を作成した．

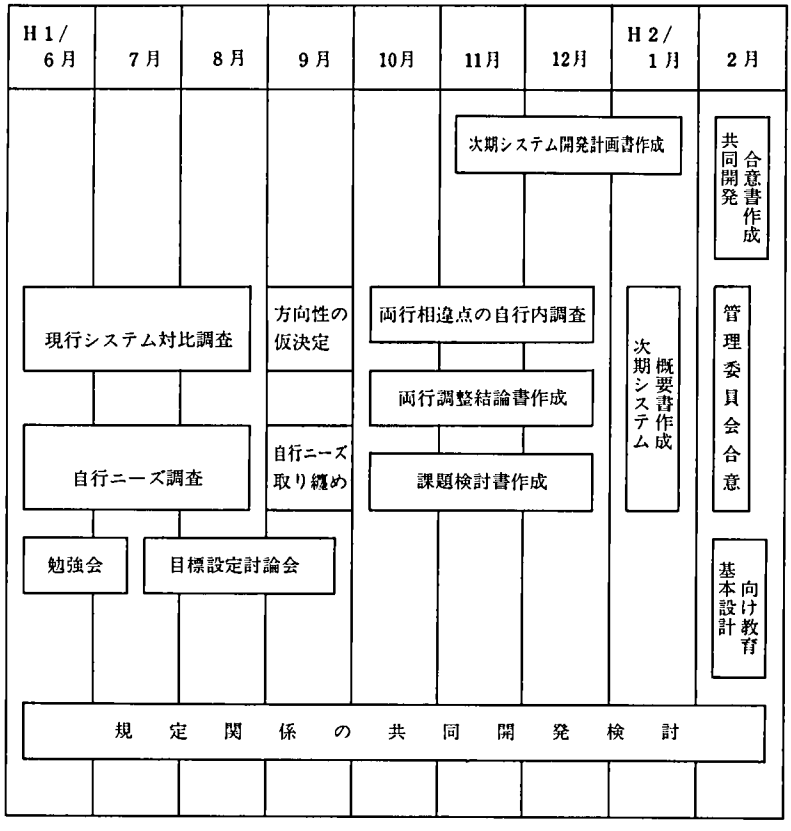


図 1　要求定義工程（NBS開発検討プロジェクト）作業項目とスケジュール

Fig. 1　Schedule on phase of defining requests

これらの主作業および関連作業項目の実施時期は図1の通りである.

しかしながら,最初は銀行ごとで使用する言葉の意味の違い,標準化ルールの違い,開発工程定義の違い,およびドキュメント体系の違い,等が各討議の場で明らかになり,さらには,労働条件の違い,初対面の人達との共同作業等,共同開発の課題が一度に出てきた.

これらの課題に対しては,各企業の中堅・代表者により,「共同開発運営ルール」を作成し,決定権を持っている人の参加を要請し三者合意を取りつけるという方法で乗り切った.

今後の共同開発プロジェクトにおいても,概要設計工程では小数精鋭で臨み,共同開発のねらいおよび新システムのねらいを参加者に優先度も含めて十分徹底させるとともに,上位マネジメントの直接参加を要請することを勧める.

2.2.1 メンバとスケジュール

メンバは両銀行11名ずつ(内訳はシステム担当7名,業務担当4名),弊社13名,協力企業12名の計47名でスタートしたが,当工程終了時は両銀行17名ずつ(内訳はシステム担当13名,業務担当4名),弊社41名,協力企業37名の計112名になっていた.

共同開発の最大のポイントは共通化であり,共通化作業は要求定義と同義であり,この共通化の徹底を図るためには十分な期間が必要であるとの考えに基づき,検討期間は9か月間という長期間を設定した.

2.2.2 共通化検討

すでに多数の稼働実績のある弊社勘定系総合オンラインパッケージであるFAST 1100をベースとして約1500項目を抽出し,これらの項目に関して現行システム対比調査を実施した.その中から両行で相異があるが共通化しなければならない(事務の統一化をしなければならない)項目が170項目抽出され,9月以降両行担当者による共通化作業が行われた.

共通化作業(仮決定)は,プライオリティ順に以下のルールで実施した.

① どちらか良いものを採用する.

② 良いものがなければ双方を捨てまったく新しいものを作る.

③ 共通化不可項目はやむをえず二つ作る.

この結果を「共通化項目検討書」としてまとめ,各行のバックオフィスの承認をとりつけたところ,40項目程度が承認されず再検討となった.

最後まで共通化できず,結果的に両行分の開発が必要になったのは,営業推進分野であり,システム的には情報系につながる分野であった.

2.2.3 次期システム課題の検討

共通化検討は現行システム機能の共通化であり,次期システムには,さらに5年以上先を予想した新機能の取り込みが重要である.

そのため,「目標設定討論会」(図1参照)にて以下の課題を討議し,次期システムの目標を設定した.

- 24時間稼働(レスバッチ・システムを含む)
- 集計レス,伝票レス,印鑑レス

・レスバッチ，レスペーパーシステム
・期日管理とセンタカット
・口振，自振システムの改善
・当座貸越
・コード体系の見直し
・勘定系と情報系の機能分担
・勘定系と対外系システムとの連動
・ファイル体系
・ソフトウェア構造，プログラム構造
・非オンライン科目のオンライン化
・ターミナル・インディペンデント
・開発の方法
・開発ツール

また，上記課題のうち以下の項目については，さらなるブレークダウンが必要との共通判断に基づき詳細討議を実施し，その結果を「課題検討書」としてまとめ，次期システムの基本設計の入力とした．

・24時間稼働（含む期日管理，C/Cシステム）
・集計レス，伝票レス
・レスバッチシステム（レスペーパーシステム）
・勘定系と情報系の機能分担
・ファイル体系とプログラム構造
・新規業務
・コード体系
・ターミナル・インディペンデント
・開発工程
・開発ツール
・規程，マニュアル

上記の共通化検討結果，課題検討結果を「次期システム概要書」としてまとめた．

また，共同開発の範囲，共同開発作業工数予測，日程計画，工程別工数計画，工程別開発体制，工程別開発分担等を検討し，今後の開発指針となる「共同開発計画書」を作成した．

1)　共同開発効果のより一層の向上……業務の共通化・ソフトウェアの共通化を第一義とし，できる限り共同開発範囲の拡大を図った．

2)　独自性を生かす分野の限定……共同開発工数，工程別工数計画については，共通化徹底によるコスト削減は計るものの，独自性の強い以下の分野については当初から100％の共通化は困難と考え，開発工数算出時に上乗せした．

・融資（稟議）リアル
・口振バッチ
・原価計算インタフェース

また，結合テスト，システムテスト工程が両銀行併行実施となること等，共同

開発なるがゆえの要素も工数計上時考慮した.

3) 公平な開発分担……工程別開発分担については，開発予想ステップをベースに均等となるように決定したが，基本設計終了時点で開発工数の再見積りを行い，分担調整を行うこととした.

参考までにこの時点での開発分担を以下に示す.

- 百五銀行殿担当……融資リアル，融資バッチ，預金バッチ
- 紀陽銀行殿担当……預金リアル，為替/口振/外接インタフェース(リアル，バッチ)
- 日本ユニシス担当……リアル共通，住公/代理貸付/債券/日計/情報系インタフェース（リアル，バッチ)，
  標準化作業，開発管理

これらの成果物を受け，著作権の取扱等を含め，三者の契約書として「共同開発合意書」が調印され，平成2年3月より「TRITON 開発プロジェクト」がスタートした. この時期に，共同開発範囲をできるかぎり広くとらえたことと，やみくもに共通化せず各業務の特性を分析し，共通化を徹底的に行う分野と，共通化を図りながらも独自性を生かす分野とに区分したことが，今回の共同開発成功要因の一つであったと評価している.

2.3 システム開発スケジュール

TRITON プロジェクトの大工程と開発場所の展開は図2の通りである.

大規模・分散・共同開発である TRITON 開発スケジュール設定にあたっては，以下の点が課題となった.

1) 共同開発体制から両行独立した開発体制への切替時期の設定

TRITON は共同開発ではあるが，本番稼働は百五銀行，紀陽銀行それぞれ独立に行われるため，本番稼働までの間に両行ともにカスタマイズの実施，および関

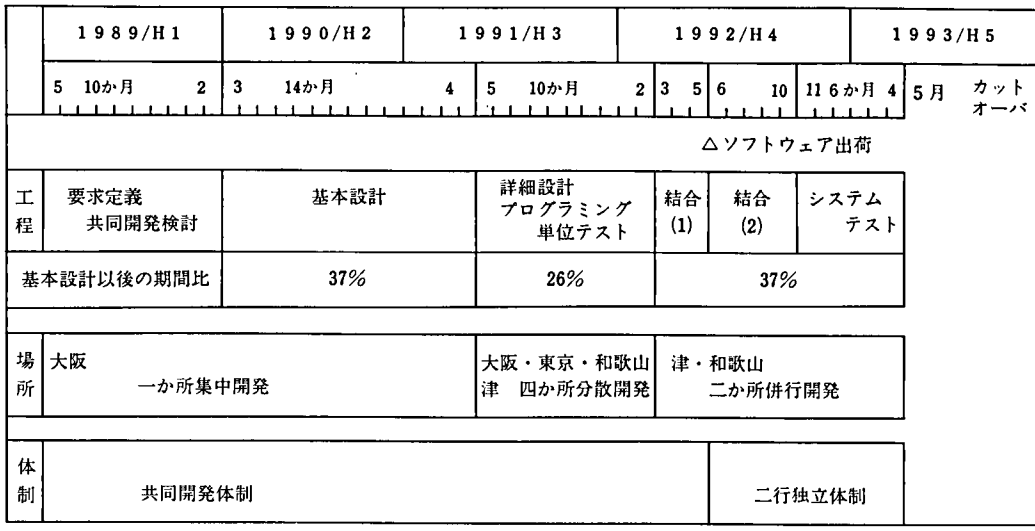


図 2 TRITON プロジェクト大工程表

Fig. 2 Phase of TRITON project

連システムとの連動確認が必要であった．このため，共同開発体制から銀行ごとの開発体制への切替が必須であり，これをいつ実施するかが最大の課題となった．

検討のポイントは三点であり，その第一はTRITONシステムの基本機能の確認が終了し，両行および弊社がその開発分担分のソフトウェアとドキュメントを責任をもって交換できる時期の見きわめであった．第二は体制切替後，本番稼働まで開発作業期間が何か月必要かであった．第三は両行独立体制とすることは，銀行ごとで見るとTRITON設計経験のある開発要員が半減することであり，このインパクトの最少化を図れる時期はいつなのかであった．各種検討を実施し，切替時期は結合テスト開始3か月後とした．

ここで，共同開発体制下での結合テストを結合テスト(1)とし，体制切替後の総合テストを結合テスト(2)とした．

2)　基本設計工程期間の拡大

要求定義工程では，総論として合意の傾向も見られたが，各論分野である基本設計工程では，細かい分野での共通化オーバーヘッドが予想されたため，2〜3か月の余裕を見込み期間を拡大した．

3)　システム・テスト工程期間の6か月以上確保

営業店参加のシステム・テストおよび新システムにおける営業店側のリハーサル等休日，祭日でのテスト実施が不可欠であり，両行および弊社ユーザにおける事例からも最低6か月は必要であると評価した．

4)　各工程実施期間の重複回避

通常のシステム開発では，詳細設計・プログラミング工程を中心に実施期間を重複して設定することが可能であり，実際にも実施されているが，今回の開発では，図2で示すように，各工程の終了時期に合わせて，開発場所の移転，開発体制の切替等の実行が不可欠であり，複数の工程を同時併行で設定することは不可能であった．

5)　システム移行に最低3日間のオンライン休止日の確保

システム移行時に必要とするプログラム開発本数の最小化を考えると，新システムへの全店一斉切替に要する期間は，移行確認作業も含めて過去の実績からオンライン休止日で3日間は必要と見積った．

現在，銀行のオンラインシステムが完全停止するのは祝祭日しかなく，本番稼働日は平成5年5月6日がシステム移行の面からは最適な日となっていた．

上記課題について，プロジェクト内での検討ばかりでなく，両行のバックオフィスおよび弊社内での検討を合わせて実施し，図2のスケジュールが決定された．

モジュールの詳細設計，プログラミング・単体テストは，5,000本/400万ステップを10か月間で完了させることになり，非常に厳しい工程となった．結果的には，各工程とも期間的にぎりぎりであったことから，詳細設計・プログラミング・単体テスト工程を26%という少ない期間に押さえ込んだことは正当であったと評価できる．

2.4　システム開発規模

結合テスト(1)が完了した平成4年5月末時点での，三者共同開発成果物は以下の通りである．

表1 開発モジュール数とステップ数
Table 1 The number of modules and their steps

種別		総ステップ数 (Kステップ)	本数
オンライン	百五固有	97	137
	紀陽固有	104	151
	共通	2,062	2,532
	計	2,263	2,820
バッチ	百五固有	127	167
	紀陽固有	136	175
	共通	1,473	1,795
	計	1,736	2,137
合計		3,999	4,957

2.4.1 モジュール数とステップ数

TRITON 開発プロジェクトとしてのプログラム開発総量は表1の通り約400万ステップである.

また, TRITON として開発したモジュールについて, 百五銀行と紀陽銀行がそれぞれ必要とするモジュールをまとめると, 両行ともに380万ステップ弱の規模である.

なお, PDP (COBOL データ定義) については約10000本/155万ステップである.

2.4.2 その他のパラメタ類

TRITON システムでは保守効率の改善のために, テーブルウェア化を推進した. とくに両行の営業店端末が異なっていたこと, また今後ともセンタシステムの再構築とは非同期で, 営業店端末が更改されていくこと等から, ホスト内の業務処理プログラムに, 営業店端末用の入出力編集処理等を持ちこまず独立させ, そのインタフェースを入出力編集パラメタとしてテーブルウェア化している.

入出力編集パラメタのステップ数を表2に示す.

その他, 用語テーブルパラメタ, システムテーブル・パラメタおよびフォームオーバレイパラメタ等, 両行分合わせて170万ステップである.

表2 入出力編集パラメタ
Table 2 The number of input/output message parameters and their steps

		ステップ数	本数
百五銀行	入力	81.968	1.393
	出力	327,915	1,305
	計	409,883	2,698
紀陽銀行	入力	91.948	1,211
	出力	358,349	1,345
	計	450,292	2,556
合計		860,175	5,254

以上をまとめると, TRITON としての全開発量はプログラム400万, PDP 155万, パラメタ256万, 合計811万ステップであり, また各行単位で見てみると, プログラ

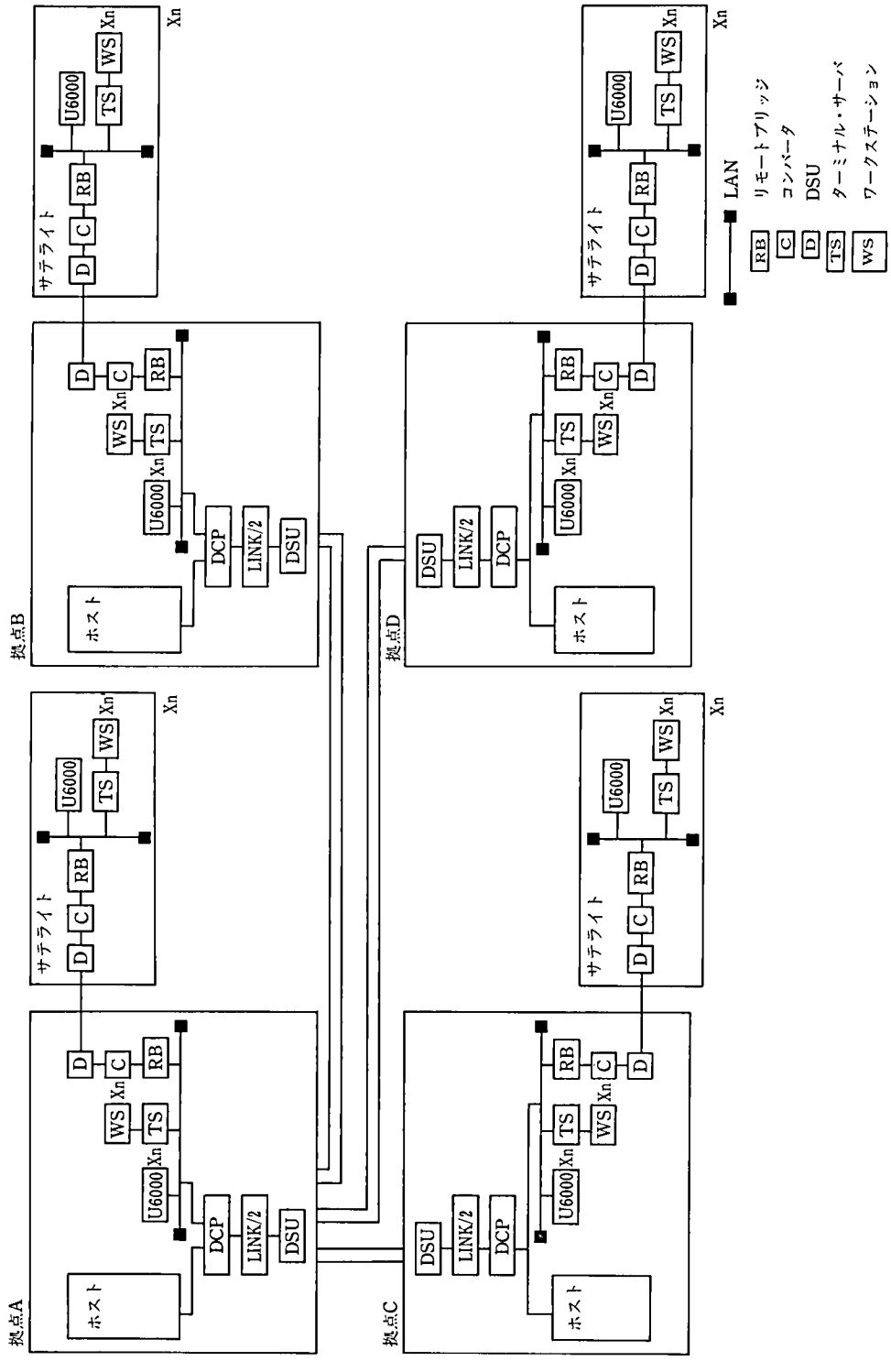


図 3　分散開発ネットワーク

Fig. 3　The network system in programing phase

ム 380 万，PDP 155 万，パラメタ 140 万，合計 765 万ステップの規模となる．

2.5 共同・分散開発の展開

開発工程中，最大要員を必要とする詳細設計，プログラミング，単体テスト工程を大阪，東京，津，和歌山の四か所分散開発形態で実施した．

プロジェクト・マネジメント面からは一か所集中開発が望ましいが，

- 要員の収容場所がない
- 協力企業は東京，大阪地区に集中している

等の理由から不可能と判断し，まず東京，大阪を拠点とした．また，

- 主力ホストマシンが設置されている
- 両行のバックオフィスのメンバが活用できる

等の理由から，津，和歌山も拠点とし，四拠点となった．

四か所に分散するのは一部不安があったが，使用 CASE ツール (IDES)，四か所ネットワーク，他の開発環境もとくに問題がなく，実施に踏み切った．

構築した分散開発ネットワークは図 3 の通りである．図中のサテライトは，拠点のスペース上の都合で四拠点から一部ソフトウェア内に設けたサブ拠点である．

3. プロジェクト運営と開発管理

3.1 開発組織と役割分担

三者共同開発を円滑かつ効果的に推進するため，以下の方針のもとに共同開発期間内における開発組織を考えた（図 4）．

- 開発責任は三者による業務系列ごとの分担

表 3 三者主担当一覧
Table 3 The company in charge

	設計		検証(全工程)	詳細/プロ/単	結合テスト	システム・テスト
	業務	基盤				
百五銀行 紀陽銀行	◎	○	◎	○	◎	◎
日本ユニシス	○	◎	○	◎	◎	○

◎主担当　○従担当

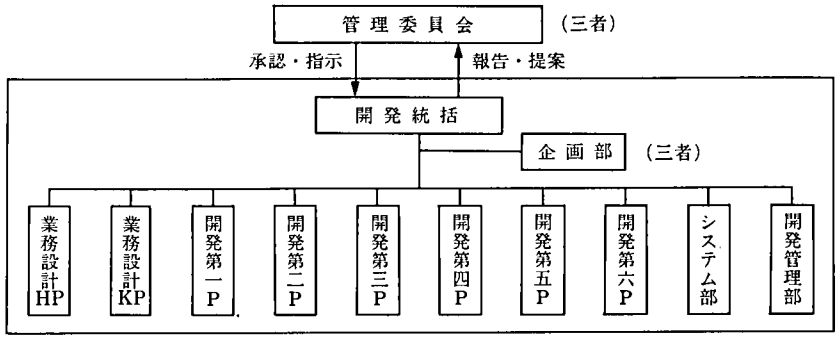


図 4 基本設計工程から結合テスト工程 I までの開発組織
Fig. 4 The structure of the TRITON project

・三者要員は保守に備えて全業務系列へ配置
・要員育成を計るため若手SEの積極的を登用
・三者の強みを発揮させるため，工程別分野別主担当の明確化（表3）

なお，図4に示す各部・プロジェクトの役割は表4の通りである．

また，諸会議は表5の通り定め実施した．これらの会議の中で共同開発プロジェクト運営の要となった会議は，上位マネジメントと現場のリーダの間に位置した企画会議であった．

3.2 プロジェクト・マネジメントの方針

さまざまな問題に直面する大規模システム開発プロジェクトであることに加えて，プロジェクト内のほとんどの部門が三者要員の混成であり，さらに協力会社10数社の要員で編成された共同開発プロジェクトを拠点分散で運営するにあたって，「情報の共有と三者共通認識の維持」を最重要課題とした．一企業内の開発プロジェクトでも，各チームのマイナス情報は隠されてしまう傾向が強いにもかかわらず，開発にかかわるすべての情報を本番稼働前日まで公開・相互交換を継続できたことは，三者をはじめプロジェクト要員の相互信頼感，連帯感の維持に寄与し，システムの品質向上，生産性向上に有効であったと確信している．

具体的方針は次の通りである．

1) 開発推進専任部門の設置

以下の役割を持った開発第五P(表4参照)を設置し，各プロジェクトの各工程実行のためのガイド作りと環境構築を行った．

・工程の詳細定義
・工程の計画ガイドおよび実行ガイドの作成
・パイロット工程実施によるガイド類の検証と諸計数把握
・進捗管理システム構築
・教育，訓練および工程立上げ支援

等々を各工程ごとに繰返し行い，標準化の推進，全拠点における開発作業の整合性維持，および管理方式の統一を実現した．

2) 計画レビュの実施

各工程開始の一か月前には作成された実行計画を，三者合同で主に要員面・予算面からレビュを実施し，前工程の残作業の見通しと合せて調整を行った．プロジェクトの当初計画との差異を共通認識とし，三者合意の後，その作業計画を作成し，進捗管理システムに登録し工程を開始した．

3) 実績管理の徹底

毎週月曜日には各拠点ごとに週次レビュを実施し，進捗状況の確認，課題の整理を行い，

・計数管理表
・各プロジェクトの週報
・拠点での評価報告書

を三者相互に交換し，情報の共有と三者共通認識の維持を図った．相互の問題点，および課題を参考にし，作業項目の抜け防止と共通課題の解決の効率化を図った．

表4 役割分担表
Table 4 Divided roles of each project team

部・P	役　割	担当
開発統括	・開発全体統括責任者 ・二行のマネージャは企画部に席を置き統括責任者との連係により開発の円滑推進に努める.	三者共同
企画部	・全体開発計画調整(進捗，要員，範囲) ・総コスト統制，管理委員会への報告・提案事項協議，対外折衝，広報活動	三者共同
開発管理部	・開発に係わる全管理業務，運用および情宣活動を行い，健全な開発運営を計る.	ユニシス
システム部	・次期システム実現(構想段階から適用,リリースに亘る)に向けて日本ユニシス提供プロダクト(基盤ソフトウェア)面からの支援活動を実施する.	ユニシス
開発第一P	・預金業務開発	紀陽銀行
開発第二P	・融資業務開発	百五銀行
開発第三P	・為替，口振，外接インタフェース	紀陽銀行
開発第四P	・情報系インタフェース，住公，代理貸付，新規業務，日計業務開発，債券窓販	ユニシス ⇩ 百五銀行 紀陽銀行
関発第五P	・共通AP開発および開発第一P～開発第六Pの開発を円滑推進するために広義の開発環境構築，技術的課題の先取り対応	ユニシス
開発第六P	・バッチ業務(預金，融資)開発	百五銀行
業務開発HP	・基本設計工程における業務面からの検証と伝票・帳表・画面等の設計	百五銀行
業務開発KP	同　上	紀陽銀行

表5 会議一覧
Table 5 Conferences

会　議　名	目　的	主　催	事務局	参加メンバ
NBS管理委員会 (3か月ごと)	三者意思決定機関として審議・調整	管理委 メンバ	担当営業部	管理委員(三者部長) NBS：開発統括 三者マネージャ
NBS月例報告会 (第2水/月)	NBS開発進捗報告	三者部長	企画部	三者部長 NBS：開発統括 三者マネージャ
NBS月次レビュ (第1水/月)	NBS開発進捗レビュ(課題の抽出と対策案)	開発統括	開発管理部	開発統括 企画部　部長 各プロジェクトリーダ システム部　部長 開発管理部　部長
企画会議 (毎週火PM)	全体開発計画調整	企画部長	企画部長	開発統括 三者マネージャ
PL会議 (第1/第3木)	プロジェクト間調整 案件検討	開発統括	開発第5P	各プロジェクトリーダ 企画部　部長 開発管理部　部長
SEサービス関連部門会議 (第2木/月)	NBS開発計画に基づいたSEサービス部門との調整	開発第5P	開発第5P	開発第5P 企画部　部長 システム部　部長

三者マネージャは当初，紀陽銀行，百五銀行，ユニシスの開発部門の代表者三名，結合(2)以後は両拠点の銀行代表とユニシス代表の四名.

(NBSは平成2年4月以後TRITONと改称)

なお，工程ごとの開発管理の詳細は 3.3 節の開発管理で述べる．

4)　品質に関する情報の交換

テスト内容，障害内容も文書化するとともに，実施テストケース数，障害発生件数および解決状況も計数管理し，相互に交換し，品質向上に努めた．必要に応じてプログラムコードの突合わせも実施する等，相手拠点の成果をもれなく活用し，品質確保の作業負荷軽減を図った．

開発当初から開発終了まで開発推進専任部門を維持したことが，この最大四か所にまで分散配置された共同開発プロジェクトを可視的な状態で運営できた最大の要因であった．

3.3 開 発 管 理

開発管理については，三者それぞれ過去の開発経験にもとづくノウハウを持っていたが，基本設計工程以後は表 6 に示す全拠点統一基本原則を定め，IDES 開発管理システムに機能追加・改善を行い，開発管理システムを構築した．

データ入力の締切時刻も同期させることにより，共同管理を実施する三者が均一な情報を把握し，評価・判断をし易くするとともに，上位マネジメントから担当者までが共通認識を持つことができた．

各拠点に開発管理グループを設置し，データ入力および運用に専従させた．管理帳表は，プロジェクト/グループ/チームの管理層に合わせた帳表と，時系列帳表を用意し，各リーダの資料作成負荷を軽減した．

表 6　進捗管理の七つの原則
Table 6　Principles of progress management

原則その 1：フォーマットはシンプルにする	(切り口多種用意)
原則その 2：機械出力必須	(二重帳簿の禁止)
原則その 3：進捗管理データ入力の厳守	(同　　上)
原則その 4：計数ベースでの中告	(虚偽の中告の予防)
原則その 5：現物に基づく中告	(同　　上)
原則その 6：過去 2 か月の実績生産性平均値に基づく翌月予定生産性の検証	
原則その 7：遅延発生時は，そのリカバリだけではなく，再発防止策も必要	

3.3.1 基本設計工程

基本設計工程は，工程内を要求定義工程の終了状況に合わせて，以下の 3 サブフェーズに区分けした．

① サブフェーズ 1：要求定義残作業（ニーズ調整，未決事項検討）
　　基本設計準備作業（設計基準，開発環境）
　　基本設計先行作業（共通機能定義）

② サブフェーズ 2：ファイル設計
　　入出力設計

③ サブフェーズ 3：処理設計

サブフェーズ 1, 2 ではアクティビティ単位の管理とし，進捗率を％で表示することとしたが，

・作業項目が多岐にわたっていたこと

・項目単位の生産性が捕らえにくかったこと

・項目単位進捗率の評価基準が定められなかったこと
・設計初期作業のため手戻りが多発したこと

等で混乱をきたし，作業計画の調整，作業完了分と未着手作業分の把握には有効であったが，仕掛中の作業状況評価には残念ながら十分に機能しなかった．

サブフェーズ3では，管理単位を設計作業項目から最終成果物である設計書単位に変更し，管理ポイントを設計書作成作業の開始・終了と検証作業の開始・終了に定めた．

設計書が完成したか否かを管理基準としたことにより，各グループ，チーム共に進捗状況の評価が定量化された．また，単位期間の生産性および生産能力が把握でき，チーム別の工程終了時期の予測が可能となり，プロジェクト全体がいわゆる「見える」状態となり，的確な対策が実施され工程の立て直しに寄与した．

3.3.2 詳細設計工程

TRITONの詳細設計工程とは，一般的な表現をすれば，プログラム仕様書作成およびテスト仕様書作成（以後ＭＩと呼称），プログラミングおよび単体テスト（以後ＭIIと呼称）である．

当工程の命題は，10か月間で5000本400万ステップのプログラムを4拠点が総力を挙げて製造することであった．

前工程の残作業，手戻り等により，東京・大阪両センタへのメンバの集結は遅延し，両センタが本格的な体制になったのは工程開始日より，すでに2か月過ぎた6月であった．

この工程において工夫したポイントについて以下に記述する．

1) 拠点ごとの役割の定義……目標達成に向かって拠点間の役割を定義するとともに，進捗管理単位を責任分担の範囲に合わせた（表7）．

2) 発注管理……津・和歌山拠点から東京・大阪両センタへのＭＩ，ＭII作業の発注計画を拠点で作成．両センタはこの計画にもとづいて要員計画を作成し，作業調整を行った．センタ要員の手空きを防止するために，センタ側でその実績を管理し，遅延分の督促を行うとともに，さらにセンタ内の作業調整を実施した．

3) 開発作業の納期管理……ＭＩ，ＭII作業の納期管理と検証指摘事項への反映作業管理を行う．センタとしては週次を原則としたが，一部は日次レビュを実施した．

レビュ単位はサブシステムごと，協力会社ごとで行い，遅れ分については遅延理由書を提出させ，課題解決に向けて進捗管理者が具体策を実施するとともに必要に応じて他拠点へ要請を行った．

主な具体策は次のとおりである．

① 協力会社からの開発作業上の問題点および要望事項を，週次レビュ，アンケート調査，ヒアリング等で吸い上げ，改善・変更可能な項目を東京/大阪の両センタ内で実施すると共に，他拠点へも改善要請を行った．

② 設計時にキーマンであった協力会社リーダが管理業務に手を取られ，開発作業推進に支障を来していたため，主力協力会社へ開発管理専任者の常駐を要請した．

表7　詳細設計役割分担表
Table 7　Divided roles in proglamming phase

拠点名	要員数	役　　割
津拠点	60	・2P融資 ・8P日計，精査　　設計MⅠ，MⅡ検証 ・6Pバッチ　　Q&A，説明会対応 ・開発環境運営(2200/9111) ・週次，月次拠点レビュ ・最終納品成果物管理
和歌山拠点	60	・1P預金(流動性，定期性)　　設計MⅠ，MⅡ検証 ・3P為替，口振，機能サービス　　Q&A，説明会対応 ・7P情報系 ・開発環境運営(2200/9111) ・週次，月次拠点レビュ ・最終納品成果物管理
大阪拠点	50	・プロジェクト統括 } 統括部 ・月例報告会，管理委員会 } 企画部 ・全体コスト管理 ・外工コスト管理，ファシリティ管理，エントリ支援 } 管理部 ・開発管理Gによるデータ入力，運用 } 管理部 ・システム共通，業務共通　設計MⅠ，MⅡ検証 ・最終納品成果物管理 } システム部 ・開発環境改善対応 } システム部 ・次工程準備，パイロット実施 } システム部 ・開発環境運営(2200/403) ・DD反映，PDP同期とり ・センタ要員教育訓練 } 5P ・各種ガイド作成 } 5P ・週次，月次拠点レビュ } 5P
東京・大阪開発センタ	550	・進捗管理と諸対策実施 ・MⅠ，MⅡ実施 ・各種基準に応じた成果物形式検証 ・最終成果物納品 ・センタ要員数育訓練 ・週次，月次拠点レビュ
計	720（人）	

③　協力会社各社の管理者は，各々独自の作業管理方式で進捗管理を行っていたため，TRITONプロジェクトの進捗管理システムの利用を勧め，津・和歌山・東京/大阪両センタおよび各協力会社すべてが同一の情報をもとに開発管理を行い，管理作業の重複と締切時点の差異等に起因する計数の誤差発生を防止した．

④　一部のサブシステムは，作業着手前に発注元拠点への基本設計書の補足説明を要請し，サブシステムおよびモジュール仕様の理解を深め，ミス発生の事前防止を行った．

⑤　開発サポート要員を配置し，デバッグサポート，および開発環境の維持・改善を図った．

⑥　生産性実績を週次・月次に把握し，最終納期予測を常に行い，協力会社要員数と担当分野を調整し，この工程の最優先事項である納期厳守に努めた．

⑦ 生産性向上のため，主管部の協力を得てIDESツールの機能改善，追加に努めた．

4) 検証作業の納期管理……東京/大阪両センタから検証責任拠点である津・和歌山両拠点へ検証依頼した成果物への検証終了の管理を両センタ側から行い，ＭＩからＭIIへの作業推進と，ＭII完了を推進させた．

主な具体策は次のとおりである．

① 拠点側の検証状況をセンタから常時チェックし，遅延が回復しない場合はセンタ側担当者の手空き防止のため作業分担の調整をし，別のサブシステム担当へ振り替えた．

② 進捗遅延の理由が難度の高さであったり，基本設計書の記述水準の浅さである場合，コード生成直後の設計担当者によるコードレビュを実施し，単体テストの繰り返しとなる無駄を防いだ．

③ 検証依頼回数をトレースすることにより，平均回数を大きく上まわるモジュール，サブシステムに対し，設計者による処理内容の補足説明会等の個別対策を実施し，再検証回数を削減した．

1回目の検証依頼で合格となるモジュールが多いサブシステムは品質良好と判断せず，品質確保の観点より十分検証を行っているかどうか検証責任拠点へ確認した．

5) 納品管理……完成したモジュールの，納品作業（成果物原本管理責任の移管作業）の進捗管理もMAPPERシステム化し，堅確化を図った．

東京/大阪両センタの作業負荷調整，センタ側と津・和歌山拠点とのスケジュール調整を行うと共に，目標達成のため四拠点が一致協力し各種改善を積み重ね，平成3年10月から平成4年1月の間は一か月当たり平均50万ステップ以上のＭＩ，ＭII作業を完了させた．

3.3.3 結合テスト工程

結合テスト工程は，この工程の途中で分散開発体制から拠点別開発体制への切り替えを予定していたため，二つのサブフェーズに区分けした．

結合テスト(1)：プログラムを4段階にランク分けし，主要プログラムは責任拠点環境で基本機能のテストを実施するとともに，テスト終了後，自拠点において，ネットワークを経由し相手拠点環境で代表的なテスト・ケースを実施し，作業確認を行い結合テスト(2)以後のパイロットとした．

結合テスト(2)：自拠点で使用するすべてのプログラムの機能確認をサブシステム内の縦のテストとして実施し，サブシステム間にまたがる機能確認を，横串テストとして実施した．

当工程では，テスト実施のみでなく，共同開発特有の作業として，自行用のカスタマイズ作業と追加プログラム開発等，独自作業も並行して実施した．後半は，現行システムからの毎月末データのファイル移行，オンライン打ち込み，および日次バッチ処理までのスルーテストを3回実施し，システムテスト工程のパイロットとした．

結合テスト(1)では,進捗管理単位をモジュール単位から実行プログラム単位に切り替えるとともに，テストケース数の予実も管理対象として，テストケース当たりの作業負荷を把握し，結合テスト(2)以後の計画作成に反映した.

また，体制切り替え時に次工程へのノウハウ継承のために，全サブシステムにわたって約1,000ページにわたる相手拠点への引継文書を作成し，交換を行った.

結合テスト(2)では，分散開発が終了し，百五銀行，紀陽銀行ともに，独自作業も含めた独立した拠点体制になったが，当初の方針通り，開発管理方式は同一のものとして，三者の共同レビュを継続し，共同開発の長所を発揮した.

主要進捗管理項目は，次の通りである.

- 懸案項目
- AP（アプリケーション・プログラム）障害記録票　（表8）
- プログラム単位の個別テスト　（表9）
- 結合テスト（横串テスト）
- カスタマイズ項目　（表9）
- 独自開発　（表9）
- 関連作業　（表9）

リアル系のプログラムは，個別テストケース数の管理も実施するとともに，積数計算・先日付処理等の重要モジュールについては200〜300ケースを通過させる集中テストを実施した．しかしながら，バッチプログラムの個別テストケース数については，途中から大量の実データを利用したことにより，通過テストケースの把握が苦しくなり，管理しきれなかった.

3.3.4 システムテスト工程

今回の開発における当工程の主なアクティビティは，表10に示す通りである.

運営体制として，システムテストグループ(以後STGと呼称)を設置し，当工程のイベントであるセンタ内システムテストおよび営業店テストの計画，推進，実施，管理，報告をその責務とした．STGは各イベントのガイドを作成し，各グループにその詳細計画を作成させ，そのとりまとめを行うとともに環境設定指示等，当日の手順書を作成し，テスト当日の指揮を行った．また，各グループからテスト実施状況を報告させ，計数把握も合わせて報告書を作成した.

一方，各グループによるシステムテストの進捗管理は，工程の立ち上げ時，全グループのテストケースと日程を一括登録し，当工程内での各種テストのいずれかで，実施の都度，終了登録することにより進捗状況を把握した.

前工程に引き続いての個別プログラムの機能確認テストの進捗管理は，ブラッシュアップテストとして追加テストケースを再設定して，前工程同様の管理を行った.

センタ内システムテスト，営業店テスト実施ごとのテスト状況の計数管理のシステム化は，オンライン打ち込みデータのプログラム別件数分析についてのみ実現できたが，共同開発対象外の各行独自プログラムを含め2,000本を超えるバッチ・プログラムの実行状況および700帳表を超える出力帳表の出力状況等が人手でしかできなかったことは，管理システムの対応遅れであり，問題点見落としの原因ともなり，反省点となった.

表 8　結合テスト他グループ依頼懸案一覧表（サマリ）

Table 8　Correstion requests accurred in test phase

依頼先		累計	内訳（依頼元）																																		
		件数	1G1	1G2	1G3	1G4	2G1	2G2	2G3	3G1	3G2	3G3	3G4	4G1	4G2	4G3	5G1	5G2	5G3	6G1	6G2	6G3	6G4	6G5	6G6	7G1	7G2	8G1
共通	合計	531	22	229	11	51	49	48	17	13	11	8	6	12	4	13	8	10	7	0	0	2	1	5	0	3	1	0
	未対応	66	8	30	1	0	1	2	5	2	3	0	0	1	0	0	0	4	2	0	0	0	0	5	0	1	1	0
預金	合計	1461	1	3	0	0	481	551	379	2	3	4	3	12	2	2	10	2	2	0	0	3	0	0	0	0	1	0
	未対応	134	0	2	0	0	97	26	2	0	0	1	1	3	1	0	0	1	0	0	0	0	0	0	0	0	0	0
融資	合計	1816	0	3	0	0	3	5	2	401	336	501	555	8	1	0	0	0	0	0	0	0	0	0	0	1	0	0
	未対応	49	0	0	0	0	2	1	0	15	14	5	9	2	0	0	0	0	0	0	0	0	0	0	0	1	0	0
口振	合計	738	0	0	0	0	0	0	0	2	0	0	0	228	201	294	0	0	5	0	0	0	0	8	0	0	0	0
	未対応	59	0	0	0	0	0	0	0	1	0	0	0	1	7	42	0	0	0	0	0	0	0	8	0	0	0	0
日計	合計	671	0	0	0	0	1	0	2	1	0	4	9	3	2	1	259	194	195	0	0	0	0	0	0	0	0	0
	未対応	23	0	0	0	0	0	0	1	1	0	0	2	0	0	0	10	7	2	0	0	0	0	0	0	0	0	0
事務B	合計	311	0	0	0	1	2	0	4	0	0	0	1	0	0	0	0	0	0	18	0	245	2	33	0	4	1	0
	未対応	89	0	0	0	0	2	0	3	0	0	0	1	0	0	0	0	0	0	3	0	54	1	21	0	4	0	0
情報	合計	155	0	1	0	0	0	1	1	0	0	0	0	0	0	0	2	0	0	0	0	0	1	6	0	20	123	0
	未対応	68	0	1	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	1	4	0	9	52	0
移行	合計	64	0	0	1	0	8	34	1	1	8	5	0	0	1	0	1	0	0	2	0	1	0	0	0	0	1	0
	未対応	25	0	0	0	0	3	11	1	1	5	2	0	0	0	0	1	0	0	0	0	0	0	0	0	0	1	0
システム	合計	186	3	53	17	8	5	12	3	5	4	0	6	12	3	8	5	9	0	3	0	2	0	11	0	8	9	0
	未対応	32	1	9	5	2	0	0	1	0	2	0	0	1	3	0	1	1	0	0	0	0	0	3	0	2	1	0
業務	合計	1	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	未対応	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
合計	合計	5934	26	289	29	60	549	651	409	426	362	522	580	275	214	318	285	215	209	23	0	253	4	63	0	36	136	0
	未対応	545	9	42	6	2	105	40	14	20	24	8	13	8	11	42	12	13	4	3	0	54	2	41	0	17	55	0

..... END REPORT .....

表 9　結合テスト(2)状況サマリ表 (全体)

Table 9　Summary of progress management

ｸﾞﾙｰﾌﾟ=9　　ﾁｰﾑ=99

テスト作業						システム・テスト					カスタマイズ			独自			関連		
テスト種別	テスト検証状況					総数 (142)		総ケース数	(0)		総数 (507)			総数 (484)			総数 (2456)		
		期間内開始数	期間内終了数	テスト-ス数	終了工分数	期間内開始数	期間内終了数	テスト-ス数	終了工分数		期間内開始数	期間内終了数	終了工分数	期間内開始数	期間内終了数	終了工分数	期間内開始数	期間内終了数	終了工分数
リアル	計画	1215	1198	43985	333.52						211	211	23.35	23	23	9.71			
	累計	1215	1198	43985	333.52						211	211	23.35	23	23	9.71			
総ケース数	実績	1217	1191	96.3%	331.19						211	211	23.35	23	23	9.71			
(44632)	累計	1217	1191	42965	331.19						211	211	23.35	23	23	9.71			
総テスト	累計差	2	-7	-1020	-2.33						0	0	0	0	0	0			
モジュール数 (1278)																			
バッチ	計画	835	525	2534	128.04						292	292	50.82	388	343	98.66			
MAPPER	累計	835	525	2534	128.04						292	292	50.82	388	343	98.66			
総ケース数	実績	784	481	47.1%	119.98						292	292	50.82	383	344	98.66			
(3172)	累計	784	481	1495	119.98						292	292	50.82	383	344	98.66			
総テスト	累計差	-51	-44	-1039	-8.06						0	0	0	-5	1	0			
モジュール数 (1664)																			
合計	計画	2050	1723	46519	461.56	23	7	0		7.5	503	503	74.17	411	366	108.37	2021	1834	530.1
	累計	2050	1723	46519	461.56	23	7	0		7.5	503	503	74.17	411	366	108.37	2021	1834	530.1
総ケース数	実績	2001	1672	93.0%	451.17	9	7	0%		7.5	503	503	74.17	406	367	108.37	1981	1820	526.63
(47804)	累計	2001	1672	44460	451.17	9	7	0		7.5	503	503	74.17	406	367	108.37	1981	1820	526.63
総テスト	累計差	-49	-51	-2059	-10.39	-14	0	0		0	0	0	0	-5	1	0	-40	-14	-3.47
モジュール数 (2942)																			

差異説明(懸案状況説明)

..... END REPORT .....

表10　システムテスト工程　主アクティビティー覧
Table 10　Main activity's list in system test phase

アクティビティ名	内　　容
センタ内システムテスト	拠点センター内におけるシステム移行，日次運用，月次運用までの総合テストを両行共に，30数回実施.
営業店テスト	全店ネットワークの確認，部分店打ち込みテスト，全店テスト，日次運用およびシステム障害等のテストで両行共に，10数回実施.
各グループによるシステムテスト	グループ内，およびグループ間で設定したテストケースを実施するテスト.
ブラッシュアップテスト	結合テスト残を含め，新たに品質向上のためテストケースを追加再設定して実施した個別テスト.

3.4　システム監査

3.4.1　監 査 の 目 的

TRITON の開発には

- 銀行の基幹業務である勘定系システム開発
- 長期間にわたる大規模開発
- 分散/共同開発
- 先進技術の採用

等の特徴があり，開発には高度な開発技術，管理技術が要求された．このため稼働時の安全性や信頼性の確保とともに，納期，品質，コスト等の開発目標を達成するため開発管理が重要であると考え，「開発方法・開発技術について，第三者からの見方，考え方を関係者に勧告することにより，スムーズな開発の一助になること」を目的にシステム監査を依頼した．

3.4.2　監査実施体制

共同開発には共同で監査することが最も重要と考え，三者の内部監査の担当部署(両銀行は検査部，弊社は技術監査室)から1～2名の監査人を選定し，計5名の要員で構成する監査チームを発足させ，「TRITON 管理委員会」からの指示で実施し，報告も当委員会に行う形態とした．

3.4.3　監査種類と実施時期

1)　計画監査……工程計画が立案される1～2か月前に，工程計画に必要な作業項目または管理項目にもれがなく，適切な手順で設定されているかを監査した．

2)　実行監査……工程着手1～2か月後に，主要作業の実施上の問題点の有無等を監査した．

同じ内部監査でも，銀行とメーカではその方法，監査のポイントも異なる．それぞれの長所を取り入れ，自社内の視点から脱して，三者の見方/考え方を包含した複眼的な視点による監査を実施できた．その報告は，日々の開発作業に追われている者にとって，基本動作に立ち戻るきっかけになり有効であった．

4. 課題と対策

1) 設計品質の確保……システム開発プロジェクトの成否を決める最大の要素は基本設計にあると言っても過言ではない．機能要件の正確な反映と記述水準がその利用者である次工程担当者にとって必要かつ十分であることが重要である．本来的には第三者による成果物レビュの徹底が不可欠と考えていたが，担当者検証で時間切れとなりレビュはできなかった．記述レベルが担当者の裁量となるのを防止するには，最低限サブシステム単位の抜き取り検査が必要である．とくに大規模システムの場合，形式的にはなるが，処理説明書の単位当たりの記述ページ数のガイドを定め見積り規模との突合わせをすることも歯止め策として有効である．

2) アクティビティ進捗管理……仕掛中アクティビティの進捗率把握は数が多くまた主観的要素が強いため，的確な把握が困難である．この点については，以下のように考える．

作業計画ベースのアクティビティは個人単位とし，1アクティビティの期間を通常のレビュ・サイクルである1週間5日以下までにブレークダウンすることにより，レビュ時の仕掛中アクティビティ数の削減が可能である．また進捗評価については，着手予定実績と終了実績によって，遅延分を明確にし，その対応策を実施することにとどめる割り切れば進捗管理は可能と考える．

3) 開発管理の総合的なシステム化……大規模なシステム開発を全工程にわたって正確に管理し続けるには，管理システムを構築しても膨大な入力負荷がかかるし，タイムリに正確な情報を得るには困難がつきまとう．開発支援ツールIDES配下で行う工程は，IDESと開発管理システムの連動による自動入力が，また結合テスト工程以後はシステムの各種ログによる連動入力の実現が望まれる．

4) 教育・訓練の合理化……当プロジェクトの教育・訓練はニューインフラ，新開発支援ツールを使用しての勘定系全業務の一斉開発であることもあって，クローズドコース，OJTを工程ごと，要員追加のつど実施するとともに，その後もスキル向上，および業務知識向上のために繰返し実施した．教育側，受講生側含めての総コストは両行，弊社，協力会社の延参加者700名強で推定工数700人月程度と多大な負荷になった．

開発要員のスキル不揃いは今後とも制約条件と考え，教育用マニュアルキットを用意するとともに，インストラクタ側の負荷軽減と受講者側の自由度拡大のため，開発支援ツール，システム構造，各業務処理基礎等のカテゴリ別のビデオ作成を提案する．

5. おわりに

共同開発検討から1年を経，さらに共同開発開始より3年を費やしたこのプロジェクトの運営は，まさに悪戦苦闘の連続であった．この間，何回も危機的状況に直面したが，そのたびに三者経営層のご理解，関連各部所のご協力とプロジェクト・メンバの頑張りに助けられ，なんとか目標に向かって前進することができ，カットオーバの喜びを分かち合えたことを感謝している．

振り返ってみると，プロジェクトのマネジメントとは，開発規模が大きくなればなるほどごく当たり前の基本動作を確実に実行することでさえ苦しくなってくるという事実にプロジェクト・マネジメントの課題が集約されるが，それに対する特効薬はなく，今の苦しみは後で楽を生むと信じ，苦しい時こそマネジメントの基本[PLAN, DO, CHECK, ACTION]に立ち戻り，それを忠実に守ることこそがプロジェクト・マネジメントであると実感している．

大規模・共同・分散開発を終えてみると，分散開発についてはネットワークの発展，UNIX* 系システムの本格的な展開等もあり，開発管理の統制と情報伝達時間の短縮化を工夫すれば容易に採用できる状況になっている．また，共同開発については，参加企業がその狙いを共通の目標としてしっかり定めることができ，開発スケジュール上で要求定義期間および基本設計期間を通常の開発に比べて2〜3割多くとることが可能であれば，調整の苦労はあるものの確実に実行可能であり，大きな効果が期待できると確信している．また，共同開発体制から参加企業ごとの個別開発体制へ切り替える時期の見極めと，実施時の諸対策を注意深く行うこと，および個別開発体制切り替え後も本番稼働に向けて共同関係を維持・継続すること等が，共同開発実行上のポイントであると評価している．

一方大規模システム開発は，今後とも解決困難な課題を残している．大規模システム開発では，ソフトウェア開発におけるさまざまな問題点が著しく増幅され，大きな障壁となってプロジェクトの前に立ちはだかることになる．

このように難しい道程ではあるが，筆者はプロジェクト・マネージャが以下の三点を継続実施すれば，プロジェクトを成功に導けると考えている．第一点は最大の情報源である開発管理システムから提供されるレポートを分析・評価し，問題発生の徴候を早期に発見し，これらの早期解決に全力を傾注すること，第二点はプロジェクトの状況をプロジェクト全体および関連部署に公開すること，第三点は結果管理から脱却し，結果を予測する先行管理型マネジメントの実行である．

また，忘れてはならないことは，大規模システム開発では開発作業および管理作業の改善はたとえ些細なことであっても，その開発規模ゆえにコスト面で大きな効果が期待できるということである．このため，マネジメント自ら関心を持つと同時に，広く要望事項・意見を積極的に求め，常に改善活動を実施することである．

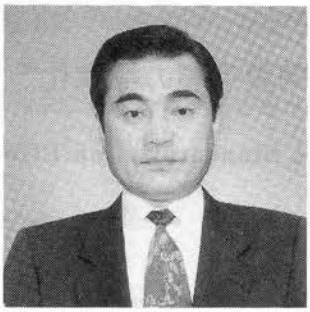
今回の経験と反省を活かすとともに，マメジメントとは何か，リーダシップとは何か，人とは何かを改めて自らに問い直し，もう一歩進んだプロジェクト・マネジメントを目指して努力していきたい．

---

* UNIX オペレーティングシステムは，UNIX Laboratories, Inc. が開発し，ライセンスしている．

執筆者紹介　片　岡　禧　造 (Kizoh Kataoka)

1943年生．1967年金沢大学 法学部法学科卒業．1969年日本ユニシス(株)入社．製造業のシステムサービスに従事後，金融部門のシステムサービス，開発を担当．現在，金融システム企画開発本部 統括部長としてTRITONシステムサービスを担当．



笠　井　　潔 (Kiyoshi Kasai)

1945年生．1969年日本大学 理工学部物理学科卒業．同年日本ユニシス(株)入社．北越銀行，三井銀行，全共連，農林中央金庫，住友銀行のシステムサービス，開発を担当．現在，金融システム企画開発本部に所属．

開発手順の標準化

A Standardization Effort for Software Development Procedures

藤　井　　昭

要　約　「見えないソフトウェア開発」を「見えるソフトウェア開発」へと転化させる方策の一つに「管理可能な単位への分割」がある．しかしこの方策は，「全体が見えにくくなる」，「分割された各単位間の整合性をどうとるか」という二つの課題を惹き起こす．今回の開発では，これら二つの課題に対し「全体が見通せる成果物の作成」と「標準化の実施」という二つの方策によって対処した．

本稿では，その具体策として実施した全体工程定義書の作成，各工程の標準化の規程，開発手順に対する各種ガイドの設定について記述する．あわせて開発手順上で品質維持に関しどのような工夫を払ったかについても説明する．

Abstract　One of the ways to turn what is called "invisible software development" into what is termed "visible software development" is "to decompose software development elements into the level of controllable units." This method, however, stirred up two new challenges: "accelerated invisibility of the whole" for one and "how to make compatibility exist between individual decomposed units" for the other. In the author's latest experience, those two challenges have been dealt with through the pursuit of two alternatives: the "making of the documents which help see the whole through" and the "implementing of standardization."

This paper refers to what have actually been done for those two alternative measures, such as newly-made descriptive forms for defining total development procedures, criteria for standardizing each process and different varieties of guidelines for software production procedures. Also mentioned here is what steps have been taken to keep the product's quality high in the entire process of the development.

1. はじめに

大規模システム開発を開発管理の観点からみた時，克服すべき最大の課題は，「見えないソフトウェア開発」をいかに「見えるソフトウェア開発」へと転化させていくかということである．

考えられる方策としては，「困難は分割せよ」との格言に従った「管理可能な単位への分割」――システムの分割，サブシステムの分割，プロジェクトの分割，工程の分割等――がある．

しかし管理可能な単位への分割は，

・全体が見えにくくなる，

・分割された各単位間の整合性をどうとるか，

という新たな二点の課題を引き起こす．

TRITON の開発では，これら二つの課題に対し，「全体が見通せる成果物の作成」と「標準化の実施」という二つの方策によって対処した．

本稿では，2章で開発工程の全体定義について記述し，とくに大規模・共同*・分散**開発という点でいかなる考慮・工夫を図ったか，を解説する．

3章，4章で開発工程における作業の流れの標準化とシステム内容を規定する標準化について説明し，5章では品質維持の観点からの考慮点について説明する．

2. 開発工程の全体定義

2.1 TRITON における開発工程

現在では弊社の全社標準の開発工程が定められているが，TRITON 開発時点では定められておらず，このため TRITON としての開発工程を定義した．

本稿では TRITON の開発工程に基づいて記述するため，TRITON の開発工程と日本ユニシス・システム開発標準工程との対比を図1に示す．

図 1　TRITON 開発工程と日本ユニシス・システム開発標準工程

Fig. 1　Contrast between "Development procedure of TRITON" and "Nihon Unisys standard procedure of system development"

2.2 開発工程定義書の目的

開発工程全体について，必要な主作業とその時期・期間・体制・開発場所・責任分担を明確にすることにより，その作業が円滑に実施されるための準備および対外部(バック・オフィス，協力会社）との折衝時期の目途を立てることを可能としている．また同時に，各工程ごとの準備作業の要である実行計画書を立案するための参照資料となることも，あわせて狙っている．なお，今回の開発の最大の特徴は，システム基盤の全面更改を伴なった大規模・共同・分散開発という点にあり，開発工程定義書も，それらの特徴が十分反映された内容となっている．

---

* (株)百五銀行殿，(株)紀陽銀行殿，弊社の三者による共同開発．

** 津，和歌山，大阪，東京の四か所．

2.3 開発工程定義書の内容

開発工程をもう一段下の作業レベルまでブレークダウンし，各々の作業ごとにその目的・概要，組織案，作業手順（前提・入力情報・成果物），留意点を提示している．

因みに各工程ごとにブレークダウンされた作業レベルの数は以下の通りである．

- 現状分析・ニーズ分析…… 3
- 要件定義 ……11
- 基本設計 ……33
- 詳細設計・プログラミング ……19
- 結合テスト…… 7
- システムテスト…… 8

2.4 共同・分散開発における考慮点

TRITON は大規模開発であったため，短期間に大量の工数を必要とする詳細設計・プログラミング工程以降は，分散開発という開発形態を採らざるを得なかった．

共同開発のもとでの分散開発形態の採用は，

- 責任所在のあいまいさ
- 他行担当分のブラック・ボックス化

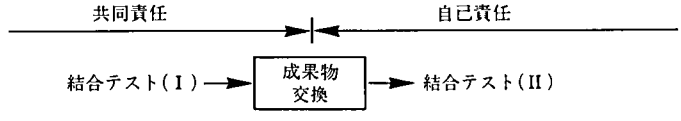
という二つの問題点を双方ともに悪化させる危険性をはらんでいる．

今回の開発においては，この危険性を除去し，問題点を回避するために以下の考慮を払った．

1) 共同責任と自己責任の分岐点の設定

次の三点の共通認識のもと，図2で示されるように，結合テストを結合テスト（I），結合テスト（II）の二つに分割し，結合テスト（I）以前を共同責任，結合テスト（II）以降を自己責任と定めた．

- 自行としての運用を確立するためにも，システムテストは自己責任の下で実施すべきである．
- 単体テスト完了という品質レベルでは共同責任を全うしたとは言えないし，その後の結合テスト工数が両行二重計上となり，共同開発の効果も薄れる．共同開発者間での成果物交換に耐えられる品質までブラッシュ・アップするためには結合テスト完了まで共同開発が必要である．
- 一方自行カスタマイズは結合テスト以前に反映しておかないと結合テストの



結合テスト(I)：自行テスト環境下でのテストが中心となるが，相手行テスト環境下のテストも拡張テストとして実施する．
（自拠点から相手行ホストを使用してテストする）

結合テスト(II)：カスタマイズ後自行テスト環境下で自行テストのみ実施する．

図 2 結合テスト（I）と結合テスト（II）

Fig. 2 Linkage-test (I) and linkage-test(II)

手戻りが発生し，テスト工数が増加する．

2) 共同開発時の責任所在の明確化

① 各プロジェクト単位に責任拠点を明確化した．

(例) 預金プロジェクトは(株)紀陽銀行殿，融資プロジェクトは(株)百五銀行殿

② 詳細設計・プログラミング工程については東西二つの開発センターを設立し

・設計・検証はプロジェクト責任

・プログラミング・単体テストはセンター責任

とした．

3) 他行担当分のブラック・ボックス化の回避

結合テスト（I）までは，各プロジェクトは三者混合体制を堅持することにより，他行分担分のブラック・ボックス化を回避した．また，この三者混合体制の採用は「プロジェクト内で承認された案件は，自動的にプロジェクト・レベルでの三者共通承認案件となる」ことを意味しており，相手行検証・承認手続きの簡略化にも寄与したと言える．

以上 1)，2)，3)の考え方に基づいて開発形態，作業場所を定めた．

時間軸の流れに伴う開発形態および作業場所の変遷状況を図 3 に示す．

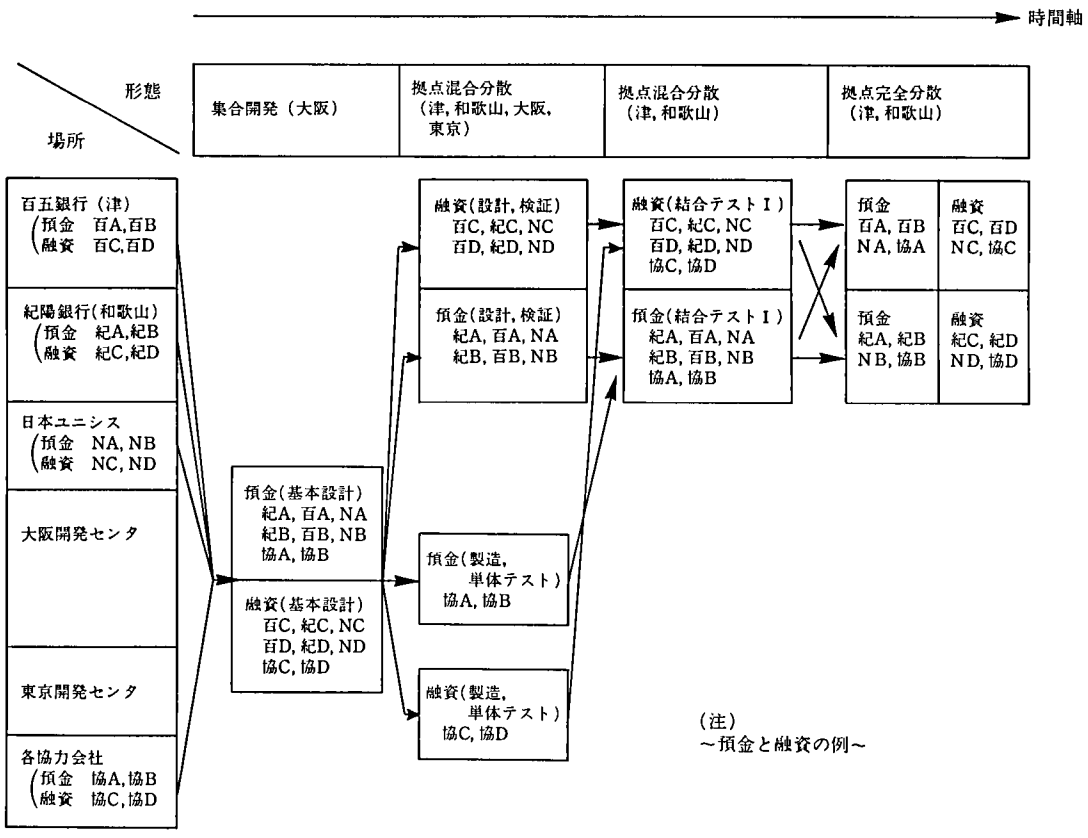


図 3　開発形態および作業場所の変遷状況

Fig. 3　Transition of development form and working place

3. 開発工程における作業の流れの標準化

開発工程における作業の流れの標準化については，以下の三点の方策にもとづいて実施した．

・組織間の情報ルートの規定
・各工程の標準化
・ドキュメント体系図の明示

3.1 組織間の情報ルートの規定

1) 委員会・会議体の設営……各プロジェクト間を横断する委員会・会議体としては以下の六つの会議体を設営した．詳細については，本誌別稿の“大規模・共同・分散開発におけるプロジェクト・マネジメントの実際”を参照されたい．

・管理委員会
・月例報告会
・月次レビュー
・企画会議
・プロジェクト・リーダ会議
・SE サービス関連部門会議

2) プロジェクト間連絡表の設定……自プロジェクト内でクローズしない問題点・変更要望・通達・指示についてはすべて，このプロジェクト間連絡表をもって行った．

なお，大量のプロジェクト連絡表が相互にやりとりされるため，管理部門が原本管理と未回答状況管理を厳密に実施した．

3.2 各工程の標準化

各工程の標準化は，工程の基本型を定めることによって実施した．

工程の基本型を定めるにあたり，以下の 4 点を基本的な考え方とした．

・パイロット・モデルの実施
・教育の実施
・実行計画・作業計画の作成
・前工程残作業の次工程への繰り延べ

1) パイロット・モデルの実施……前工程の期間内に次工程のパイロット・モデルを実施する．

パイロット・モデルの実施によって得られる効果としては，

・作業の阻害要因の明確化
・想定作業手順の正当性確認
・準備作業の洩れの有無確認
・次工程の生産性数値の妥当性（大幅な乖離の有無チェック）の検証→妥当性の検証のために使用することにとどめるべきあり，パイロット・モデルでの生産性をそのまま採用することは非常に危険である．

が挙げられる．

2) 教育の実施……前工程の期間内に次工程向けの教育を実施する．

とくに，詳細設計・プログラミング工程向けの教育については，協力会社要員

の中途参画・メンバーの入れ替え等が発生することを考慮し，

・詳細設計・プログラミング工程期間中も必要に応じて教育の場を設定する．

・協力会社の上級SEは自分が講師となれるレベルまで教育を通じてスキルアップを図り，ある程度は協力会社内で対応可能とする．

という二つの策で対処した．

3)　実行計画・作業計画の作成……前工程の期間内に次工程向けの実行計画/作業計画を策定する．

実行計画書および作業計画書は図4のプロセスで作成され，レビューを受け承認される．

①　実行計画書策定要領

開発工程定義書に実行計画書の策定要領はおおむね記述されている．

しかし，開発工程定義書は初期段階に起案されたものであるため，

・予想と現実とのギャップが発生している．

・後工程になる程つめが甘くなっている．

という不具合をもっている．この不具合を埋める役割を受けもっているのが実行計画書策定要領であり，言わば，開発工程定義書の各工程単位の更新版という位置づけで，企画部門から提示された．

②　実行計画書

パイロット・モデル実施結果および前工程残作業を反映して当工程の実行計画書を作成する．

実行計画書の主たる記載内容は以下の通りである．

・プロジェクトの役割

・アクティビティ（大項目）とラフスケジュール（月単位）

・アクティビティ（中項目）と担当・期限

・組織体制

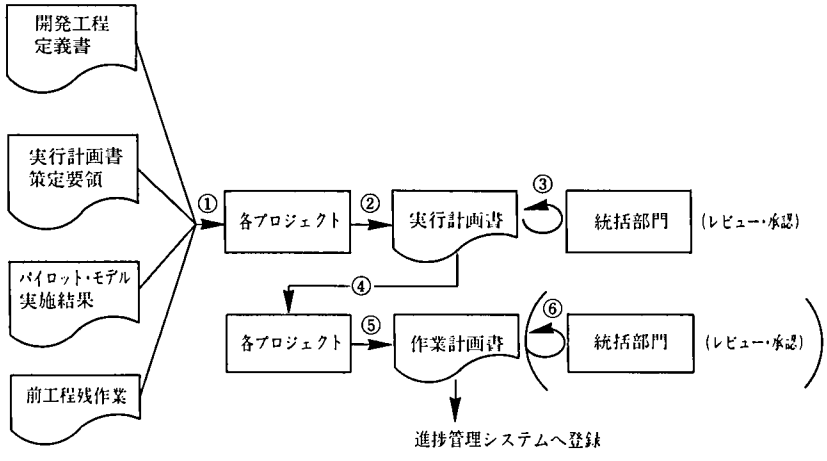


図 4　実行計画書・作業計画書作成プロセス

Fig. 4　Process of generating execution-planning list and action-planning list

・見積り工数

③ 作業計画書

作業計画書にはアクティビティ（小項目）と担当と期間を記載する．

必要に応じてレビュー・承認を受けた後，この作業計画書に基づいて個人ベースの予定を進捗管理システムへ登録する．

4) 前工程残作業の次工程への繰り延べ……共同開発では各拠点・各プロジェクトが各工程ごとの区切り目を同期させることが必要であり，このため各工程の区切り目は斜め線ではなく，直線とならなければならない．

しかし，現実の世界では各拠点・各プロジェクトが各工程ごとの区切り目を完全に同期化させることは不可能である．

この問題への対応として以下の具体策で臨んだ．

・あくまで工程の区切り目は縦の直線とし，各拠点・各プロジェクトで同期させる．

・現工程で未消化となる部分については，現工程内でそれを整理・評価し，工数・体制を策定した上で次工程向けの作業計画に盛り込む．

以上の考え方に基づいて定義された工程の基本型を図5に示す．

図 5 工程の基本型

Fig. 5 Standard works between the neighboring procedures

3.3 ドキュメント体系図の作成

各工程内および各工程間における作業の相互関連・前後関係の概要，つまり全体としての作業の流れを理解するための道具としてドキュメント体系図を作成した．図6にドキュメント体系図の一例を示す．

4. システムの内容を規定する標準化

4.1 各種設計基準

階層化構造設計手法に基づいて表1に示す9種類の設計基準を設定した．

4.2 設計・開発手順にかかわるガイド

ガイド・便覧・規約・作成要領など20種類のガイド類を設定した．その目的と概要を表2に示す．

5. 品質維持の工夫

5.1 IDES の採用

品質維持の工夫の中で，最大の特徴は，CASE ツール IDES（Integrated Develop-

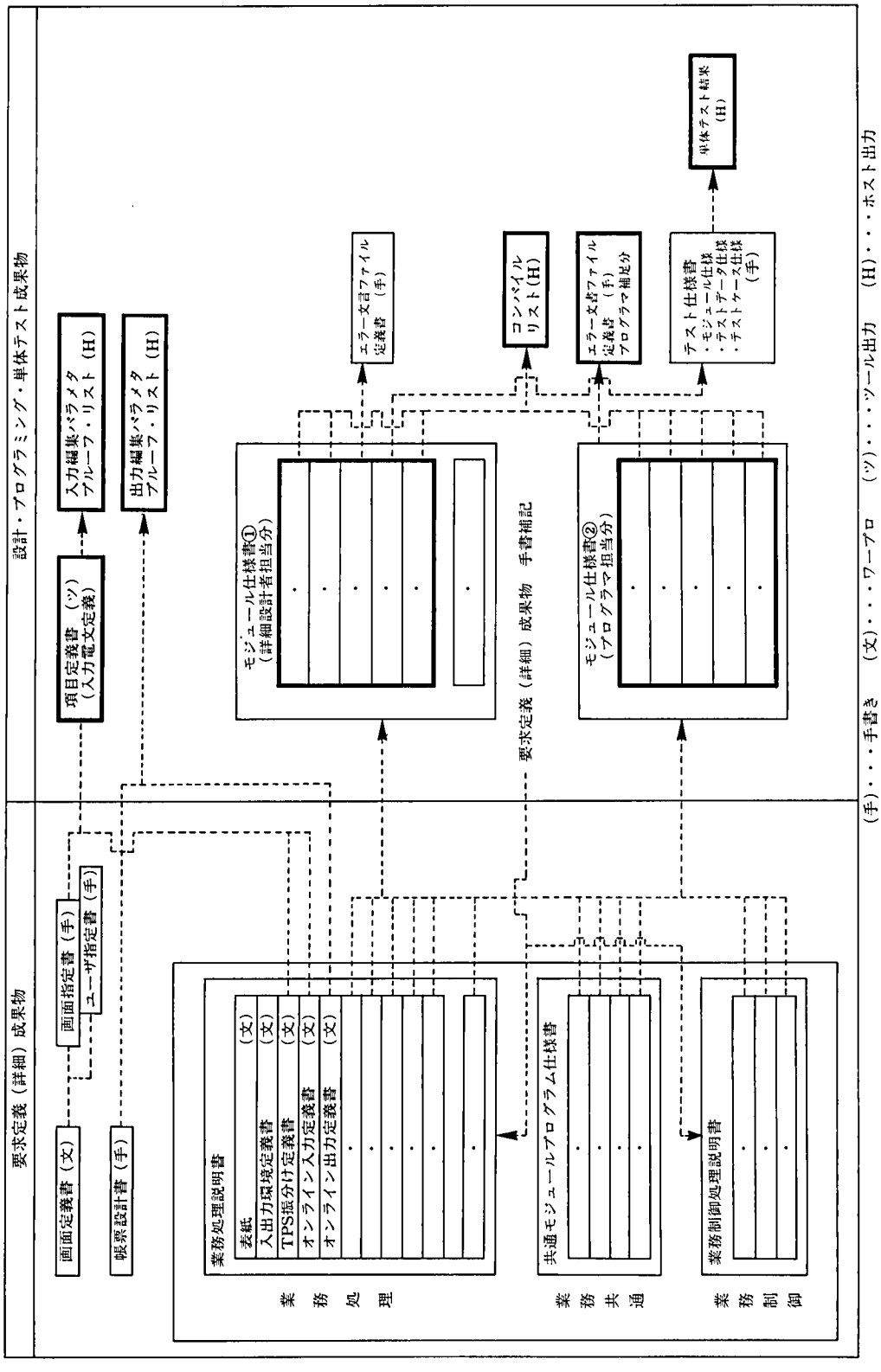


図 6　工程間のドキュメント体系図の一例

Fig. 6　Example of document system between the neighboring procedure

表1 設計基準
Table 1 The design standard

No	設計基準	内　　容
1	システム分割基準	業務種類ごとのシステム分割・サブシステム分割
2	データ項目設計基準	属性基準(演算型二進数，非演算型二進数，表示型，特殊データ) データ統一名基準(金額，残高，レート，サイズ，TRX 等) レイアウト上の留意点，基本項目のデータ定義(残高は S 9(12)とする等)
3	ファイル設計基準	ファイルの種類(種類とアクセス方式と適用の目安の対応) 論理レコードと物理レコード データベース・TIP ファイル・システムテーブルの設計基準とアクセス基準
4	トランザクション設計基準	トランザクションの種類の定義 入力トランザクション・出力トランザクション・業務トランザクション・日計トランザクションの作成単位・フォーマット・作成方法
5	コード設計基準	コード化の範囲(順番コード，区分コード，桁別コード，十進コード，表意コード，合成コード) コードの分類(営業店使用者からの視点，システム担当者からの視点) コード設定の基本的な考え方
6	プログラム設計基準	プログラム言語とプログラム構造 オンラインプログラム設計基準(構造と各階層ごとの機能，処理形態，処理上の考慮点，データエリア)
7	モジュール設計基準	モジュール機能 モジュール分割 業務制御・業務処理・共通モジュール作成上の留意点
8	取引パターン基準	取引パターンの決定 個別基準(入力形態の多様性，論理取引パターンと機能の関連等)
9	ネーミング規約	データ名，手続き名，データベース関連，ファイル名，パラメタ関連，MAPPER 関連，帳表関連，ログイン名

ment Environment Support System) の全面採用である．

IDES の採用によって得られるメリットのうち，品質維持に関するメリットには以下のものがある．

1) 分散開発時には，各分散ホスト間・各分散 UNIX*間の各種整合性確保が品質維持の観点から必須となるが，IDES の分散開発環境支援機能により容易に実現できた．

2) データ設計・モジュール設計の手順を IDES の操作手順として精緻なレベルで規定することができた．このため，この操作手順に従うことにより自動的に開発手順の標準化が遵守されることになったと言える．

また，IDES の自動チェック機能・自動生成機能が，標準化遵守のための歯止めとなっていることも評価できる．

自動チェック機能・自動生成機能の代表的なものを列挙しておく．

IDES の詳細については，本誌別稿の“開発工程と IDES”を参照されたい．

① 自動生成機能

* UNIX オペレーティング・システムは，UNIX System Laboratories, Inc. が開発し，ライセンスしている．

表2　ガイド類一覧表
Table 2　List of guides

No	資料名	目的	概要
1	基本設計ガイド	システム全体を同一思想の下で開発するための標準化内容の明確化	システム設計方法と規約，業務設計基準，端末設計基準
2	基本設計便覧	個々のオンライン業務処理設計を効率的に行うための共通事項と固有の考慮点を明確化	オンライン処理の入口～出口までの流れを切口として具体的な実現方法を記述
3	バッチ基本設計基準書	バッチ設計の均一化	システム設計方法と規約
4	バッチ標準化資料	バッチの詳細設計で必要になるシステム廻りの標準化内容の理解	バッチの詳細設計で必要になるシステム廻りの標準化内容
5	分散運用ガイド	分散開発環境における詳細設計～単体テスト工程の作業手順の確実な理解	分散開発作業における発注～納品までの手順，各種連絡方法，開発関連会議の運営方法
6	分散運用ガイド管理詳細編	開発環境の運営・維持の確実化	分散マシン運営，ライブラリアンの役割他
7	モジュール仕様書記述要領（オンライン編）	IDES 環境下で均一なオンライン業務処理モジュールの作成	SE，プログラマの具体的作業内容と基準
8	モジュール仕様書記述要領（バッチ編）	IDES 環境下で均一なバッチ業務処理モジュールの作成	SE，プログラマの具体的作業内容と基準
9	コーディング規約(オンライン編)	TRITON で使用する UCS-COBOL 命令語の種類と使用ルールの明確化	TRITON で使用する UCS-COBOL 命令語の種類と使用ルール．IDES のプログラム作成支援ツールと整合性をとっている．
10	コーディング規約(バッチ編)	同上	同上
11	入力編集テーブル作成要領	作成手順と作成時のルールの確実な理解	入力電文定義の方法，入力編集パラメタの作成方法とルールを記述
12	出力編集テーブル作成要領	同上	出力電文の構成，コマンド・ファンクションの使用方法とルールを記述
13	用語テーブル作成要領	同上	パラメタの作成方法とルール． 使用方法はオンラインとバッチに分けて記述
14	XEROX パラメタ作成要領	同上	XEROX パラメタ作成指示の記述方法とルール，発注～成果物登録までの手続き
15	単体テスト実施要領	モジュールごとの単体テスト手順の明確化による確実なテストの実施	モジュールごと単体テストの作業の全体の流れ，単体テストの準備作業，実施時の作業等
16	バッチ単体テスト実施要領	バッチ単体テスト手順の明確化による作業の円滑化	バッチ単体テスト作業の全体の流れ，環境変更手続等
17	結合テスト(1)ガイド	結合テスト(1)の位置づけ，全体計画の共通認識化．実際の作業手順の明確化による確実なテストと管理の実施	結合テスト工程定義，成果物定義，各業務単位の計画の基本的な考え方，作業手順と基準，進捗・障害管理等
18	センター・カット・テスト実施要領	センター・カット・テストの円滑な実施	センター・カット関連バッチ業務プログラムおよびセンター・カット TPS のテスト方法と留意事項
19	トラブル追求ガイド	迅速なトラブル追求と対応	トラブルの切り分けとトラブル種類ごとの追求方法，XIS 関連エラーの追求方法，各種エラー・コード一覧
20	ユーティリティ使用ガイド	各種ユーティリティの確実な利用による円滑なテストの実施	各種ユーティリティの使用方法（TRITON 作成ユーティリティ，XIS ユーティリティ，ファイル・サポート・プログラム）

・PICTURE 句の型・積数や VALUE 句等の書き方の統一
・登録集のエレメント名とエントリ名の統一
・JSP 法による構造化プログラミング
・モジュール仕様書における木構造図の書き方の標準化

② 自動チェック機能
・データ定義名の同音異義語チェック
・データ定義名の使用不可文字チェック

5.2 品質チェック

1) 各工程ごとの開始直後の成果物チェックの実施……各工程ごとに開始直後の成果物の内容をチェックし,必要に応じて前工程の作業結果の見直し作業を行った.

とくに注力したのは基本設計評価であり，評価のチェックポイントとして下記の二点を設定した.

① 現在のドキュメントから次工程へスムーズに入れるか.
② 分散開発を前提にして，新メンバ（協力会社中心）が，基本設計書をもとにプログラム仕様書を作成できるか.

2) 形式チェックの実施（詳細設計・プログラミング工程）……協力会社から成果物が納品された時，検証依頼の前に開発センタ内にてチェックリストに基づく形式チェックを実施した.

また，形式チェックでエラーになったものについては，協力会社ごと・個人ごとに統計値をとって傾向を分析し，同一の誤りを防止するよう努めた.

3) 検証依頼回数のまとめとチェック（詳細設計・プログラミング工程）……合格までの検証依頼回数の統計情報を月別に集計しチェックを実施した.

① 検証依頼回数が平均値に比べて異常に多いプロジェクト/チームについて，警告を発すると同時に，不合格理由をもとに原因を追究.
② 検証依頼回数が平均値に比べて異常に少ないプロジェクト/チームについては，月次比較，検証残件数との相関比較を行い，検証が甘くなってきていないかを検討.

5.3 カットオーバ・クライテリアの策定

カットオーバ•クライテリア（本番実施基準）は，以下の目的をもって策定された.

① 本番実施の判断を客観的・定量的基準に基づいて行うことが可能となり，上位マネジメント層と開発担当者共通の判断材料となる.
② システム・レビュー時のチェック・ポイントとして使用でき，全員の共通努力目標となる.
③ 当基準が達成できない場合の阻害要因・必須作業の摘出が容易になる.

カットオーバ•クライテリア策定時のキーポイントは基準値の設定の仕方であるが，犯しやすい間違いが二点あるため，その回避策を含め留意点として記述する.

① 目標値としては設計目標値と実現目標値の 2 種類がある.

基準値として設定すべき値は実現目標値であるが，往々にして設計目標値が設定されやすい.

⟶カットオーバ基準値の欄とは別枠で設計目標値の欄を設定する.

② アプリケーション障害残件数がゼロというような非常に厳しい基準値を設定してしまい，レビューの都度現実にあわせて基準値を下方修正する必要性にせまられる．

⟶あくまで本番実施できるかどうかという観点で基準値を設定すべきであり，かつ一度設定された基準値は変更しないという原則を貫くべきである．

表3　カットオーバ・クライテリア
Table 3　Cutover criteria

システムの品質
- プログラム単体テスト……リアルプログラム，バッチプログラム
- プログラム品質向上……リアルプログラム，バッチプログラム，機能別個別要件
- システムテスト品質……機能テスト，営業店打込テスト，障害回復テスト
- 基盤ソフトの整備・安定……最終バーション出荷状況，障害残件数，パラメタ設定，DISC 配置
- アプリケーション・ソフトの整備・安定…ソフトウエアの本番運用，ドキュメントの本番運用，障害残件数
- 端末ソフトの整備・安定……出荷状況(メーカ→銀行，センタ→営業店)，障害残件数
- サブシステムソフトの整備安定……他系を中心とした障害残件数
- 研修，練習，ユーザ部門テスト

キャパシティ
- ホスト系
- レスポンスタイム……営業店端末，自動機，対外系，センターカット

稼働・運用
- 通常運用……運用時間帯・本番ソフトの確定，オペレータ勤務体制と人員
- 障害運用……リカバリ時間(XTC，HSB，自系)，DCP 障害回復時間，T/C 切換

移行
- 体制
- 作業手順……移行フロー作成，稼働前の確認作業確定
- テストケース
- 処理時間
- 移行資源
- 移行延長対応……戻し基準の確定

マニュアル・ドキュメント
- 仕様書
- 各種手順書……通常運用手順書，障害運用手順書
- オペレーションマニュアル
- 一覧表の整備
- 採番表(ライブラリ)

稼働後保守・拡張要件
- リグレッションテスト環境の整備
- ドキュメントとライブラリ管理
- カットオーバ後の開発保守体制
- 販売支援・導入支援体制
- 稼働後 2〜3 カ月間でのスタート機能組み込み

このためには，アプリケーション障害残件数等は総件数ではなく，科目別/機能別に小分類し，重要度に応じて個別設定する等のプライオリティ・コントロールも必要である．

表3に今回策定されたカットオーバ・クライテリアを示す．

6. おわりに

大規模システム開発を「管理可能な単位へと分割」した時に派生する問題点に対して，「開発工程全体を見通せる成果物」と「標準化の実施」という二つの対策を講じ，それをいかに具現化してきたかという点について記述してきた．今回の開発において，これら二つの対策が発揮した効果については大概評価できると思われるが，今後の課題として大きく三点残されている．

まず，“バッチの標準化”である．バッチはリアルと異なり，その処理パターン・使用ファイル・関連システム・使用システム基盤・運用の考え方等，多岐にわたり，なかなか標準化できにくいということが背景にあり，今回の開発においてもチャレンジはしてみたものの，部分的な標準化にとどまっているというのが実態である．今後はこの“部分的な標準化”をベースに各ユーザ固有の状況にあわせた肉付けを行っていくことが現実的な対応であると考える．

次に，“結合テスト（II）および総合テストのガイド”である．結合テスト（II）以降は拠点完全分散となったため，各拠点ごとにその作成が委ねられたという経緯はあるものの，今後の適用を考えたとき必須とされるガイドであり，早期に整備が必要であると考える．

そして，“コード設計基準の遵守”である．数百万ステップにのぼるシステム開発の場合，設定されるコードの数も膨大なものとなる．

それらが二重定義・三重定義されることのないようネーミング基準を設定し，データ辞書に登録し，同音異義語（homonym）をチェックし，データ辞書のサーチツールを用意しても皆無の状態にすることは，なかなか難かしいというのが現実である．この問題については文献検索の技法（同義語（synonym）検索，前方一致・後方一致検索）の適用が解決策の一つになりうると想定される．

執筆者紹介 藤　井　　昭（Akira Fujii）

1971年名古屋大学理学部数学科卒業．同年日本ユニシス（株）入社．SEサービス部門（信用金庫），応用ソフトウェア部門（BIMS保守，DSS 1100-J開発・保守），SEサービス部門（地方銀行，都市銀行）に従事．現在金融システム企画開発本部TRITON企画推進部部長．

ドキュメントの標準化

Document Standardization

村 田 豊 彦

要 約 金融機関におけるコンピュータ・システムの規模は，昭和40年代の第一次オンラインから昭和60年代の第三次オンラインにかけて，「10年で約4倍」のスピードで拡がりをみせている．

このシステム規模の増大は，ソフトウェア開発・保守を，さまざまな形での作業の分割や要員の役割分担の規定によって，はじめて成り立たせることを余儀なくした．

昔は，ドキュメントがなくても，限られたスペシャリストによる人的な対応が可能であったが，現在は，書き物なしでシステムの開発・保守・利用をすることはあり得ない状況であり，ドキュメントの重要性は論を待たない．

筆者はドキュメントを人と人との間のコミュニケーション手段であると定義し，その本質は正確さとわかり易さにあるととらえている．ドキュメントに関して重要なのは，その体系や記述内容をはじめとした標準化と，ドキュメンテーション支援ツールによる機械化・自動化にあると考え，本稿ではこれらについて工夫あるいは考慮した点を解説する．

今回のTRITON開発においては，標準化やそれをもとにした機械化・自動化の推進により，一定の効果を得，また将来に向けての環境整備が図られたと考えている．

ドキュメントはそれを維持し続けることによって有用な情報となり得る．そのためにはドキュメンテーション支援ツールの対象範囲の拡大や操作性・高速性の更なる向上，ならびに開発・保守に携わる人間のドキュメンテーション作業に関する強力な手順遵守の姿勢と指導力が重要である．

Abstract Computer systems at banking institutions expanded at the rate of "almost four times in ten years" in terms of the scale and size of computing systems during a time period from first-phase on-line systems in the late 1960s to third-phase counterparts late in the 1980s. The systems scale that multiplied to this extent in the past has now resulted in forcing such work as software development and maintenance to exist only by means of laying down the rules whereby divided jobs are described and the roles of individuals are defined in varying ways.

It used to be possible for a limited number of specialists to respond to the needs, even though no documents were available to them. Nowadays, however, there is no one who never recognizes the importance of documents because any type of systems development, maintenance and utilization always requires references to the related writing.

The author defines the role of documents as a means of interpersonal communications, with accuracy and ease of comprehension being its essence. Based on the thought that the keystone of document-making is an effort to standardize the structure and contents of documents, including computerization (or automating), through the use of documentation support tools, this paper reports on what the author has done and considered in this respect.

The author's involvement in the recent development of the TRITON system has supposedly contributed to a move toward the standardization and consequent computerization (or automation) of docu-

ments, bringing about a certain amount of good effect, which has helped straighten up the related environment for the future. Documents can be a provider of useful information if they continue to be kept. That is why the author sets store by further efforts to extend the coverage of documentation support tools and to improve their operability and performance, in addition to the solid attitude and strong leadership of those who work on software development and maintenance in promoting loyalty to document-making procedures.

1. はじめに

TRITONシステムは，(株)百五銀行殿(以下百五銀行と略記)・(株)紀陽銀行殿(以下紀陽銀行と略記)・弊社の三者による大規模共同開発(投入工数約10,000人月)のもと，400万ステップに及ぶプログラムを保有する大規模勘定系システムである．企画段階および開発のシステム設計は大阪で集中して行ったが，プログラム開発の段階からは大阪以外に東京および両行の所在地である津，和歌山を加えた4箇所での地域分散開発を行った．

平成5年5月，システムの本番稼働を無事に行い，現在は保守フェーズに突入している．

このような大規模・共同・分散開発においては，次のような要件が通常開発時以上に求められる．

- 正確かつ確実な情報伝達
- 品質面での均質性
- 効率的な保守環境の具現化
- 成果物の確実な管理

また，開発環境面では，ホスト，サーバ，クライアントの3層の機器構成とし，ホストによる集中開発からサーバ，クライアントによる分散型の開発形態にしたため，分散機器を有効に活用することも重要な要件である．

以上をドキュメンテーション面からとらえ，次のような課題として整理する．

- 開発における手戻りを防止するための正確かつ洩れのないドキュメンテーション
- 保守における対象システムの理解を容易にするためのドキュメントの均質性
- 保守作業の機械化・自動化を可能にするためのドキュメントの標準化ならびに電子ファイル化の推進
- 共同・分散開発環境下でのドキュメントの確実な収集とデリバリ運営の推進
- ドキュメンテーションにおけるクライアント，サーバ機器の十分な活用

筆者としては，効率的な開発作業・保守作業を可能にするという観点でドキュメントをとらえた場合，重要なのはドキュメントの体系と記述内容ならびに標準化をもとにしたドキュメンテーション支援ツールによる自動化であると考えており，3章，4章で上記課題に対する工夫・考慮を含め解説する．

なお，TRITONシステムは，日本ユニシスの商品パッケージ・ソフトウェアであり，開発・保守ドキュメントに関する解説だけでなく，商品ドキュメントについても解説を加える．

2. ドキュメントの定義と目的

1章でも述べたようにドキュメントの種類は，開発用ドキュメント，保守用ドキュメント，商品用ドキュメントの三つに大別してとらえるが，ソフトウェア・ドキュメントの定義については総合的な定義として次のように位置づける．

> システムの管理者，開発者，保守者，利用者，運用者，販売者
> ならびに導入者間の情報伝達を確実かつ効率的に行うための手段

目的については，大別したドキュメントの種類ごとにその目的を明らかにする．そして三者間の関連についても整理してみる．

2.1 開発用ドキュメントの目的

1) 上位担当者とその配下要員との情報伝達……システム開発は開発の全体計画策定からカット・オーバに至るまでさまざまな分担作業によって行われる．個々の分担作業においては指示者と実作業担当者が必ず存在し，ドキュメントはこの両者の情報伝達手段となる．

- 期間計画・工数計画・要員編成・全体進捗状況(企画・管理面)
- 作業手順書・規約等のガイド
- 現工程から次工程につなぐ情報(設計情報が主体)
- 工程内の情報(仕様変更内容，障害状況内容，進捗状況等)
- その他各種連絡用情報(システム仕様の確認，作業依頼と完了報告等)

2) 品質保証……個々の成果物に対する品質規定，システム全体としての品質規定を明らかにするドキュメントで，開発にかかわる人員の認識の一致を図る手段となる．

- 検証基準書(各種チェックリスト等)
- 評価基準書(本番実施可否基準書等)

3) 承認……開発時は次工程に進んでよいかどうかという観点で各種設計成果物そのものがすべて承認対象である．なお，次に示すような機能の基本仕様にかかわる内容，開発工数の変化にかかわる内容についてはとくに重要で，開発統括者の水準で確実な承認を行うための手段となる．

- システム概要定義
- ソフトウェア要求仕様
- 基本仕様の変更管理情報

4) 教育

① システムの利用者向け教育

システムの利用者に対し，システム化された業務の取扱い方法を明らかにし，誤りのないシステム運営を可能にする手段となる．

- 業務取扱い要領，操作要領

② システム開発・保守担当者向け教育

システム開発・保守に携わる要員に対するシステム全体概要，個別機能別の解説あるいはシステム規約等の教育用手段となる．

・各種概説書・解説書

③ システム運用者向け教育

システムの運用・操作を担当する部門に対し，誤りのないシステム運用方法を説明するための手段となる．

・運用操作説明書

2.2 保守用ドキュメントの目的

保守用ドキュメントの最も重要な目的は，「効率的な保守を行うための情報伝達」であり，システム保守要員に対し，対象システムの理解および対応箇所の特定を容易にさせることである．

もちろん開発用ドキュメントの大半が保守用ドキュメントになり，2.1節で述べた次の開発用ドキュメントの目的はそのまま保守用ドキュメントの目的にもなる．

・上位担当者とその配下要員との情報伝達
・品質保証
・承認
・教育

ただし，開発時のドキュメントの中には，作業の効率的な手順との関連で一時的に作成はするが，最終的には後の工程で作られるドキュメントに置き換わるものが存在する．いわゆる保守対象外のワーク・ドキュメントであり，現工程から次工程につなぐ設計情報の一部がこれに相当する．これらについては，いつどのような条件で廃棄するかが重要である．TRITON 開発では不要となる成果物の廃棄時期は大工程の次々工程を基本とした．

なお，保守段階に入ってからの修正履歴情報や障害履歴情報等の各種履歴情報は，効率的な保守作業に有効なドキュメントである．

2.3 商品用ドキュメントの目的

1) 商品機能説明用……ソフトウェア商品の購入顧客に対し，対象システムの理解やカスタマイズが必要な箇所の特定を容易にさせるための手段となる．開発用ドキュメントの目的で述べた「上位担当者とその配下要員との情報伝達」および「教育」のドキュメントが対象ドキュメントである．また，販売促進用のパンフレット等のセールス・ツール，ならびに実体験により商品機能の理解をより効率的に行うことが可能なため練習教材も対象ドキュメントとして考える．

2) 商品使用方法説明用……ソフトウェア商品の利用者あるいは運用者に対し，その使用方法・操作方法を伝達するための手段となる．

開発用ドキュメントの目的で述べた「教育」のシステム利用者向けドキュメントやシステム運用者向けドキュメントが対象ドキュメントである．

3) 効率的な導入を行うための情報伝達……ソフトウェア商品の購入顧客に対し，商品の導入方法の理解を容易にさせるための手段となる．

・導入手順書

4) 効率的な保守を行うための情報伝達……導入後の保守を行う要員に対し，対象システムの理解および対応箇所の特定を容易にさせるための手段となる．開発用ドキュメントの大半が対象ドキュメントである．

3. ドキュメント体系と記述内容

ドキュメントの体系と記述内容は，ドキュメントの標準化項目の中でまずベースとして定めるべきものである．そこで4章のドキュメントの標準化に先立ち，別章として述べることをお断りしておく．

本章では2章をもとに，まずドキュメントの体系を整理し，次に開発工程とドキュメントの関連について述べる．

3.1 ドキュメントの体系

2章で述べたドキュメントの目的からドキュメントの体系にかかわるキーワードを抽出してみると次のようになる．

企画/管理/ガイド/設計/連絡/検証・評価/教育/履歴/導入

このキーワードと2章のドキュメント種類との関連をもとにし，表1にドキュメント体系を示す．なお，体系内の細目は代表的な項目にとどめている．

3.2 開発工程とドキュメントの関連

開発を手戻りなく効率的に行うためには，各工程で作成するドキュメントの内容に次工程の作業に必要な情報が正確に盛り込まれていなければならない．

表1　ドキュメントの体系
Table 1　Category of documents

			ドキュメント種類		
			開発用	保守用	商品用
ドキュメント体系	企　画	計画(全体計画，工程別計画)	○	○	
		新システムに求める機能要件	○		
	管　理	進捗	○	○	
		コスト	○	○	
	ガイド	作業手順書	○	○	○
		規約	○	○	○
	設　計	設計情報	○	○	○
		仕様変更情報	○		
	検証・評価	検証基準と実施結果	○	○	○
		評価基準と実施結果	○	○	○
	連　絡	システム仕様確認(QA)	○	○	
		作業依頼と完了報告	○	○	
	教　育	各種概説書・解説書	○	○	○
		業務取扱い要領	○	○	○
		操作要領	○	○	○
		運用操作説明	○	○	○
	履歴情報	修正履歴		○	○
		障害履歴		○	○
	導入情報	導入手順			○
		練習用教材			○
	販売促進用	パンフレット他セールスツール			○

TRITON 開発時に作成した主なドキュメントの種類，記述内容ならびに作成順序を示すために，開発工程とドキュメントの関連を表2にまとめる．なお，各開発工程の作業内容イメージの誤解を防止するために，工程名は TRITON 開発の工程名ではなく日本ユニシス標準工程名を使用して表にまとめたことをお断りしておく．

表2　開発工程とドキュメントの関連

Table 2　The relation between "development process and document"

大工程 / 中工程	日本ユニシス標準成果物	TRITON ドキュメント名称	記述内容
要求定義			
概要定義	システム概要定義書		現状の問題点/課題を抽出し，その対応(解決)方法とシステム化の範囲を明確に記述する．そして，そのシステムに要求する機能，入出力を定義(種類・媒体・データ量・タイミング)する．
		システム現状調査書 業務処理対比表 相異点検討書	TRITON システムは共同開発という事情があるため左記のドキュメントにて記述を行った．
		重要項目目標設定 討論会報告書 重要項目タスクフォース・グループ課題検討書 共通化項目検討書 ↓ 次期システム概要書	
詳細定義	ソフトウェア要求仕様書		確定したシステム化の範囲について詳細化を進め，かつ各種制約条件下での実現性を確認するために，入出力/業務機能の詳細とデータ処理の概要や基本的システムの仕組みを記述する．
		担当システム一覧表(☆) 入力定義書 出力定義書 業務処理定義書 ファイル体系図	・開発対象領域 ・各入出力の詳細(項目，サイズ) ・処理論理(機能) ・ファイル体系，データベース論理構造
	システム開発計画書	工程定義書 開発見積書	開発計画，開発規模の見積り，生産性基準等を記述する．
設計	設計計画書(☆)	実行計画書(☆)	当該工程の目的，基本方針，要員編成，当該工程の総工数・大日程を記述．
		作業計画書(☆)	作業項目ごとの詳細日程・担当割り等を記述．
	設計基準(☆) システム設計書	設計基準書(☆)	設計上の規約，手順上の規約を記述． 業務処理の機能，入出力，ファイル，インタフェース等について詳細記述を行う．
		システム概説書	各処理形態のプロセス，リカバリ単位/方法等の記述．
		個別機能別概説書	エラー処理，入出力，センタ・カット，システム運用他のデータ処理機構(仕組み，共通仕様)の詳細記述．

(注)ドキュメント名称の☆印はその工程以前に作成される．

大工程 中工程	日本ユニシス標準 成果物	TRITON ドキュメント名称	記述内容
		帳票/画面定義書	帳票レイアウト，画面レイアウト，項目属性，画面制御情報等を記述.
		データ定義書	各種データの物理設計で，項目名，型，サイズ，レイアウト等を記述.
		共通モジュール使用説明書	共通モジュールの使用者インタフェースを記述.
	プログラム外部仕様書	業務処理説明書 (オンライン) プログラム処理説明書 (バッチ)	次の内容を記述. ・プログラム概要と管理情報 ・使用するファイル名称，画面名称・番号，出力帳表名称 ・入力条件に基づいた入力→出力の関係 ・入力レコードの各項目から出力レコードの各項目をどう得るかを記述. ・メッセージ条件，番号，内容
プログラミング			
プログラム設計	プログラム作成計画書(☆)	実行計画書(☆)	当該工程の目的，基本方針，要員編成，当該工程の総工数・大日程を記述.
		作業計画書(☆)	プログラム単位・作業項目単位の詳細日程・担当割り等を記述.
	プログラム内部仕様書	モジュール仕様書	プログラム単位に，内部構造，処理手順を記述. ・プログラム概要と管理情報 ・木構造図 ・機能説明 ・呼出系列説明 ・処理詳細 ・テストケース指示書等
	テスト基準(☆)	単体テスト実施要領(☆)	プログラム種類ごとのテストケース設定基準，テスト結果の確認方法等を記述.
	テスト仕様書	テスト仕様書	プログラム単位に単体テストケースを記述. ・プログラム間インタフェース仕様 ・テストデータ仕様 ・テストケース仕様
プログラム作成	プログラム・リスト 単体テスト結果	プログラム・リスト 単体テスト結果マーキング・リスト	コーディング 単体テスト実施結果
テスト			
結合テスト	結合テスト計画書(☆)	実行計画書(☆)	当該工程の目的，基本方針，要員編成，当該工程の総工数・大日程を記述.
		作業計画書(☆)	作業項目ごとの詳細日程・担当割り等を記述.
	結合テスト結果	各種報告書	次の内容を記述. ・プログラム間整合性確認結果 ・サブシステム間整合性確認結果 ・他システムとの整合性確認結果等

(注)ドキュメント名称の☆印はその工程以前に作成される.

大工程 / 中工程	日本ユニシス標準 成果物	TRITON ドキュメント名称	記述内容
テスト			
総合テスト	総合テスト計画書(☆)	実行計画書(☆)	当該工程の目的，基本方針，要員編成，当該工程の総工数・大日程を記述.
		作業計画書(☆)	作業項目ごとの詳細日程・担当割り等を記述.
		カット・オーバー・クライテリア	本番実施可否基準であり，システムの品質，キャパシティー，運用，移行，マニュアル，稼働後保守・拡張要件等についてそれぞれの可否基準を記述.
	操作マニュアル(☆)	事務取扱要領(☆)	業務分類ごとに事務の流れにもとづいた事務手続きを記述.
		端末操作要領(☆)	画面レイアウト，入力項目説明，オペレーション注意点，出力印字見本等を記述.
		各種運用操作手順書(☆)	セットアップ・ターム，ワークフロー等の操作手順，チェック項目等を記述.
	総合テスト結果	各種報告書	開発システムが本番構成で稼働できる状態になっていることを確認するために次の内容を記述. ・システム全体の運用確認結果 ・全日，繰り越し，月次，期末処理等の確認結果 ・システム性能の評価結果 ・対外接続システムとの確認結果等
導入	導入計画書(☆)	各種導入計画書(☆)	新システム稼働準備と現行システムから新システムのハードウェア，ソフトウェアへの切り替えを行うために，次の内容を記述. ・移行計画内容 ・新システム作動確認内容
	教育計画書(☆)	各種教育計画書(☆)	集合研修，営業店研修，運用部門研修の計画を記述.
保守/評価	システム開発完了報告書	各種報告書	業務内容，波及効果，開発期間，開発規模，開発工数，開発費用，開発環境，ソフトウェア品質，評価と今後に向けての提言等を記述.
		保守手順書	稼働したシステムの維持を行うために，必要な手続きの設定(システム変更，バグ対応)や，手順の見直し結果等を記述する.

(注)ドキュメント名称の☆印はその工程以前に作成される.

4. ドキュメントの標準化とドキュメンテーション支援ツール

1章でも述べたとおり，効率的な開発作業・保守作業を可能にする要件の一つとして，ドキュメントの標準化とそれをもとにしたドキュメンテーション支援ツールによる自動化が重要であると考えている．ドキュメントの種類は開発用，保守用，商品用の三つに大きく分かれるが，いずれの場合も標準化項目は次のようなものである．

① ドキュメント・フォーマット

様式の管理情報の統一やサイズ等

② ドキュメント保存媒体

紙，電子ファイルの区分

③　ドキュメント保有形態

ドキュメントの綴り方(順番等)

④　ドキュメント管理

管理組織や管理手続き

⑤　ドキュメンテーション支援ツールの利用

支援ツールの利用範囲，利用方法等

本章では上記項目に対し，TRITON 開発時に工夫・考慮を図った点を，共同開発・分散開発という条件を加味しながら解説する．

4.1　ドキュメント・フォーマット

図1に様式の基本形式を示す．上段の管理情報は要求定義(詳細定義)工程以降のドキュメントについては基本的に同一である．要求定義(概要定義)工程までは後述のシステム分割・サブシステム分割ができていないため，図1のような管理情報ではない．

形式の標準化のねらいは次の点にある．

・ファイリングのし易さと確実さ

パート要員でも可能．コピー時や発送時のミスをチェック可能．

・ドキュメンテーション支援ツール対象ドキュメント追加時の容易性

管理情報は同一部品が利用可能．

・押印義務付け等による開発担当分野の責任明確化

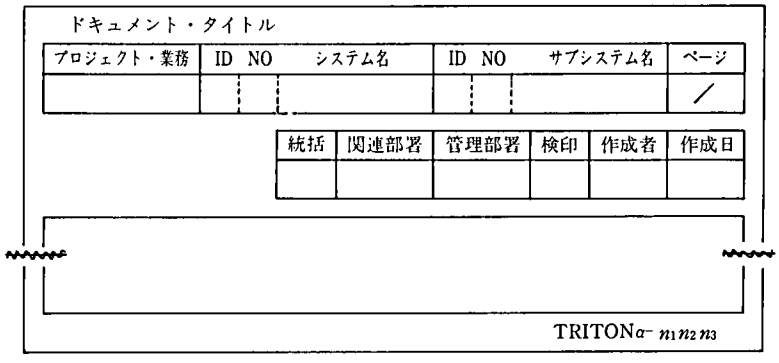


図1　様式の基本形式

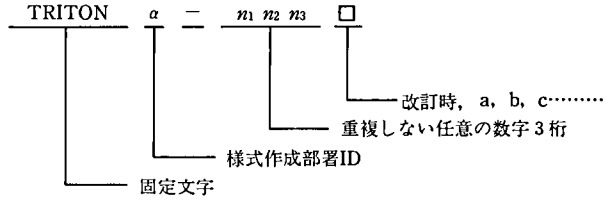
Fig. 1　Standard form of document

システム名およびサブシステム名の欄にある ID はそれぞれシステム ID(英字1桁)，サブシステム ID(英字2桁)である．

TRITON においては，要求定義(概要定義)工程にてシステム分割・サブシステム分割を行い，オンライン，バッチ合わせて，19個のシステム分割，約130個のサブシステム分割になっている．ID はユニークであり，各種ネーミングに使用される．システム No，サブシステム No はともに2桁の数字であり，ファイリング時のバインダにナンバリングする際の構成要素として用意したものである．

統括欄～作成者欄は印鑑枠である．TRITON 開発では，開発担当分野の責任を明確にするため，印鑑押捺を義務づけた．

図1の右下のTRITONα-$n_1n_2n_3$は様式の管理番号で採番は次のとおりである.



以上のような様式上の管理情報の標準化により，開発の直接部門以外の部門でもチェック・管理が可能になり，共同・分散開発下でのドキュメントの確実な収集・交換に役立ったと評価している(4.4節ドキュメント管理参照).

なお,様式のサイズはファイリングのし易さを考慮し,基本的にはA 4縦版である.ただし，後述のドキュメンテーション支援ツールの出力ではホストプリンタ印書のためA 4横版になる.手書きあるいはワープロのドキュメントと支援ツール出力ドキュメントの関係は図2のようになる.

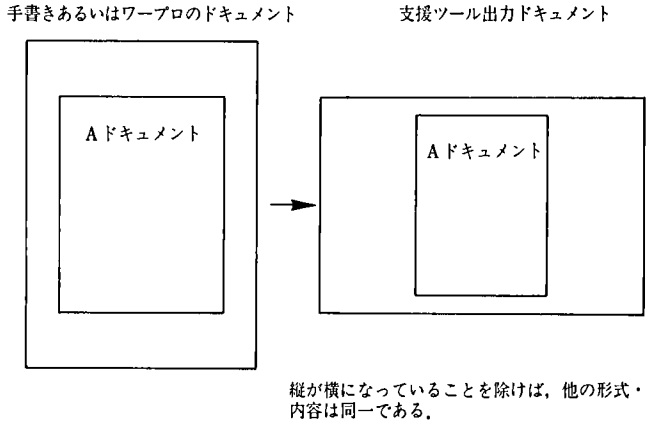


図2 手書きあるいはワープロのドキュメントと支援ツール出力ドキュメントの関係

Fig. 2 The relation between 'handwritten document' and 'document made by documentation tools'

4.2 ドキュメント保存媒体

保存媒体は，ホスト上の各種ファイル，サーバ上のファイルあるいはフロッピディスケットのような電子ファイルが基本である．一部については，前後関係の見易さと検索時の早さから紙による保存も行っている.

電子ファイル保存の代表例を表3に示す.

表3の中で，保存媒体として，ホスト上のデータ・ディクショナリ，ホスト上のテキスト・ファイル，サーバ上のファイルがあるが，これらについては4.5節のドキュメンテーション支援ツールで説明する.

電子ファイル化により，次のような効果があった.

・見やすさの飛躍的な向上を図ることができた.

(機械印字になると誤りでも本当らしく見えてしまうという問題があるが)

表3　電子ファイル保存の代表例
Table 3　Examples of electronic filing document

保存媒体	ドキュメント代表例	入力ツール	備　　考
ホスト上のデータ・ディクショナリ	・各種レコード定義書 ・各種レコード・レイアウト ・コード索引簿 ・プログラム管理情報	CASE ツール	
ホスト上のテキストファイル	・モジュール仕様書	CASE ツール	表紙・木構造図～処理詳細までのセット
ホスト上の MAPPER ファイル	・各種一覧表	MAPPER	テーブル一覧表・振り分け定義一覧表等
サーバ上のファイル	・共通モジュール使用説明書 ・業務定義書	ワープロ CASE ツール (文書管理システム)	プログラム外部仕様書の一部としての位置付け
フロッピ・ディスケット	・××計画書 ・マクロ定義書	ワープロ	

MAPPER は UNISYS の代表的な第4世代言語で，プログラマレス・コンセプトのもと，プロトタイピングでの開発・運用に適している．メインフレームだけ（UNIX, PC にも搭載可能）でも，ワールドワイドで2500セット以上の設置数となっている．また UNIX オペレーティング・システムは，UNIX System Laboratories, Inc. が開発し，ライセンスしている．

・手書き成果物に対比すると，修正時の書直し部分の極小化を図ることができた．
（とくに，追加・削除による再レイアウティング）
・一元管理の効果として，以降の成果物作成に再入力することなく，そのまま，利用することが可能になった．

保存媒体ごとの個々の効果は，4.5節のドキュメンテーション支援ツールを参照願いたい．

4.3　ドキュメントの保有形態

ドキュメントの保有形態は，機能横断的な（言いかえれば共通的な）ドキュメントと機能単位ドキュメントで大きく異なる．

前者は共通利用のし易さという観点から単独で綴り，後者は作業担当者の理解のし易さという観点から同一機能単位で複数のドキュメントを綴る．以下に機能横断的なドキュメント例，図3に機能単位ドキュメントの代表としてモジュール単位成果物の綴り方を示す．

・システム概要定義書
・設計基準書
・開発手順書
・共通モジュール使用説明書
・レコード定義書
・コード索引簿
・何々計画書等

なお，図3の例は，TRITON 開発時の保有形態のサンプルである．今後，制度改訂等で業務横断的な開発が発生した場合は，別バインダ綴じが理解のし易さという観点で有効と思われる．

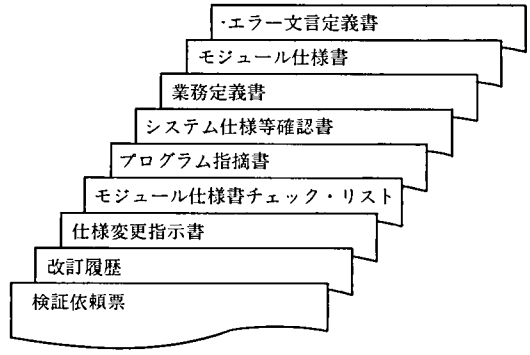


図3　モジュール単位成果物の綴り方例（オンラインの例）
Fig. 3　Filing of program module documents

4.4　ドキュメント管理

TRITON は，結合テスト前半工程までは，体制面で拠点混合分散*の形態をとりながら共通の成果物を作成した．共同開発・分散開発における工程の中には成果物交換という大イベントがある．体制面で拠点混合分散の形態をとることにより，相手拠点の担当業務の成果物を受け取った後，中味がわかる人員がいずれ自拠点に戻ってくるので，最終的な開発作業がスムーズに行われる．

しかし，拠点完全分散**以降は業務単位の有知識者が半減するのは事実で，十分に理解している分野とそうでない分野の両方を抱えながら最終ゴールを目指すことになる．

したがって，交換するドキュメントの正確さと確実な交換が以降の作業を効率的に行うための重要事項になる．

以上を踏まえ，TRITON のドキュメント管理は次のような方式を採用した．

・集中部門での成果物管理による収集とデリバリ
・ドキュメント登録・整備状況の進捗管理システム投入
・仕様変更情報起票運営の徹底

4.4.1　集中部門での成果物管理による収集とデリバリ

確実なドキュメント管理を行うために，直接開発部門以外の部門がドキュメントの集中管理を行う運営とした．図4に設計・プログラミング工程の成果物管理運営を示す．

C拠点での成果物収集・デリバリは管理部署にて実施している．管理部署でドキュメント管理に携わる要員は男性1名とパートの女性数名である．

収集したドキュメントのデリバリは基本的に工程の切れ目で実施する．

図4で示した運営は，ドキュメントをはじめとする成果物の正確さと確実な交換に十分に貢献したと評価している．

4.4.2　ドキュメント管理と進捗管理システムとの連携

TRITON では，工程ごとに管理すべき情報項目が異なるため，工程別に進捗管理シ

* 拠点混合分散とは，A拠点がX業務，B拠点がY業務を担当する際に，A拠点，B拠点ともに，両拠点の混成部隊で要員編成する形態を意味する（つまりA，B両拠点とも要員の半数近くが転勤状態である）．

** 拠点完全分散とは，A，B両拠点とも個別に全業務を担当し，自拠点要員のみで開発を行う形態を意味する．

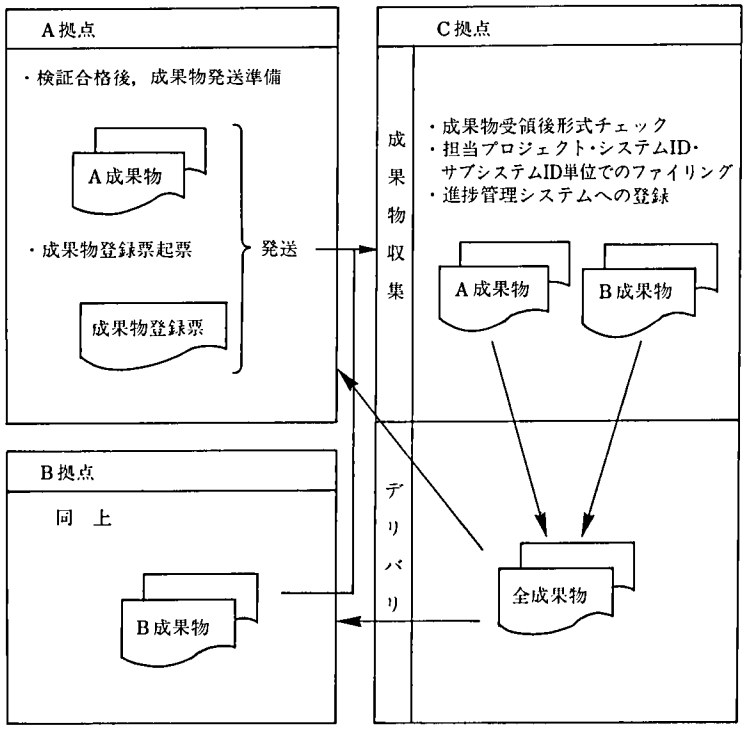


図4　集中部門での成果物収集とデリバリ

Fig. 4　Gathering and delivery of documents

ステムを用意したが，現物に基づく管理という概念は共通である．

したがって，進捗管理システム・メニューには，成果物登録や整備状況に関する更新系あるいは照会系のメニューを作成した．

詳細説明は省略するが，成果物登録に関する更新系と照会系の画面の一例を図5および図6に示す．

この他，進捗管理システムでは仕様変更に関する更新系・照会系メニューを用意したが，残念ながら実運営はされなかった(行政指導不足が原因ではないかと思っている)．

仕様変更については，成果物登録と違って，仕様変更情報を記述しない限り，洩れなく管理するのは不可能である．したがって，仕様変更情報起票運営の徹底が本質的な問題である．TRITONでは図7に示すような仕様変更運営を行った．

仕様変更は，大きく次の二つに分けてとらえた．

・明らかなバグや考慮洩れ

・基本仕様の変更につながる変更

統括部署における承認は，前者の場合は事後承認(仕様変更対応作業(a)のルート)，後者の場合は事前承認(仕様変更対応作業(b)のルート)の運営とした(図7参照)．

4.4.1項および4.4.2項で述べてきたように，ドキュメントの管理には手続きや要員面でかなりのオーバヘッドがかかるが，ドキュメントの正確さや確実な収集を行うために，共同開発・分散開発環境を配慮した必要最低限のシステム運営を行ってきた

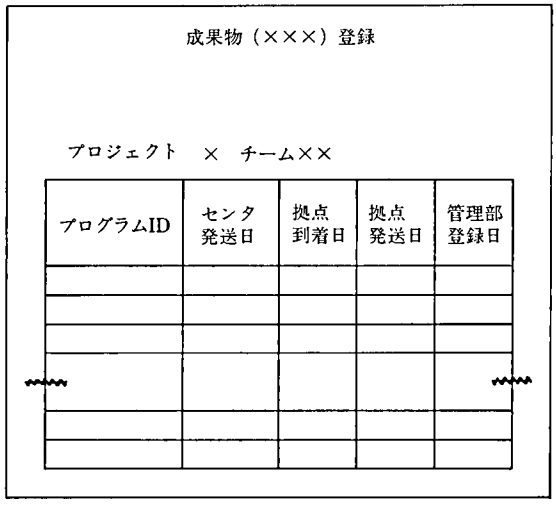


図5　成果物登録更新系画面例

Fig. 5　Document registration menu

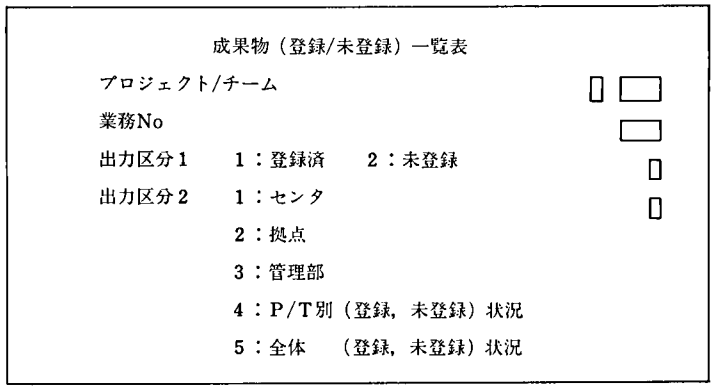


図6　成果物登録照会系画面例

Fig. 6　Document search menu

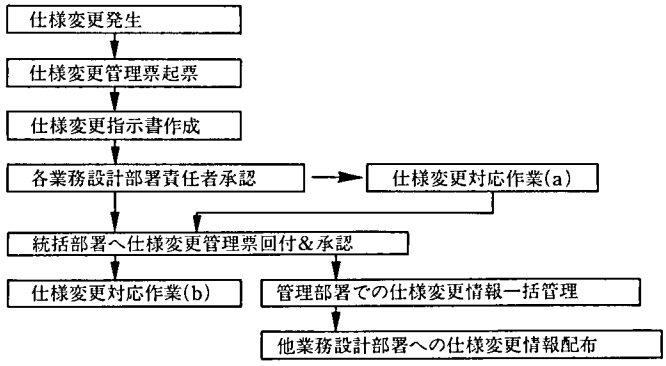


図7　仕様変更時の作業の流れ

Fig. 7　Flow of jobs when change in specifications occurred

と考えている．

4.5 ドキュメンテーション支援ツールとその適用

ドキュメントはその内容がソースコードを中心とした現実のソフトウェアと整合性のとれたものでなければならない．すべてのドキュメントが現実のソフトウェアから“注釈行”によらず自動生成されることが理想であるが，現在の技術では不可能である．とくに上位ドキュメントの自動生成は困難で，人手によって保守せざるを得ないのが実情である．

一方，ソースコードこそ最良のドキュメントであるという考え方も根強く存在している．これはドキュメンテーションにかかわる投資・労力が大きく，それに見合う効果が定量的に把握(定量的な把握は現実的に困難)されていないからであろう．

このような状況のもとで，TRITON が適用した基本技術やドキュメンテーション支援ツールのラインアップは次のとおりである．

① 日本語ワード・プロセッサの活用
② 図形エディタの活用
③ 日本語 COBOL による漢字の積極的活用
④ MAPPER の活用
⑤ データ辞書の活用
⑥ 文書管理システム
⑦ プログラム作成支援ツール
⑧ リバース・エンジニアリング
⑨ ドキュメント出力機能

これらのうち，②，⑤，⑥，⑦，⑧，⑨は，TRITON 開発で使用した CASE ツール“IDES”の機能の一部である．

図 8 にドキュメンテーション支援ツール全体図を示す．

4.5.1 日本語ワード・プロセッサおよび図形エディタの活用

1) 日本語ワード・プロセッサ……開発準備段階で先ずはワープロソフトの選定作業を行った．当時はワープロの定着期を迎え，各メーカのワープロソフトも出揃った状況であったため，売れ筋ソフトのうちの一つを採用した．ワープロ専用機を使用せずワープロソフトを採用したのは，開発用ターミナルを多目的に利用し，投資抑制を図るためである．

開発用ターミナルはスペース問題等の理由から，Dynabook*あるいは J3100 を使用した．

日本語ワード・プロセッサの活用とその成果物を文書管理システムへ登録することにより，CASE ツールが他のドキュメントとマージして出力することが可能になり，開発担当者のドキュメント編綴の負担が軽減された．また，ファイル転送にて容易にサーバやメインフレームに取り込むことができるので，ワープロ入力内容を再入力することなく，メインフレーム上でさらに加工・編集し，他の用途に利用することができた効果もあった．

2) 図形エデイタ……ドキュメントの中には構造図等のように図形で記述するもの

* Dynabook は（株）東芝 の登録商標である．

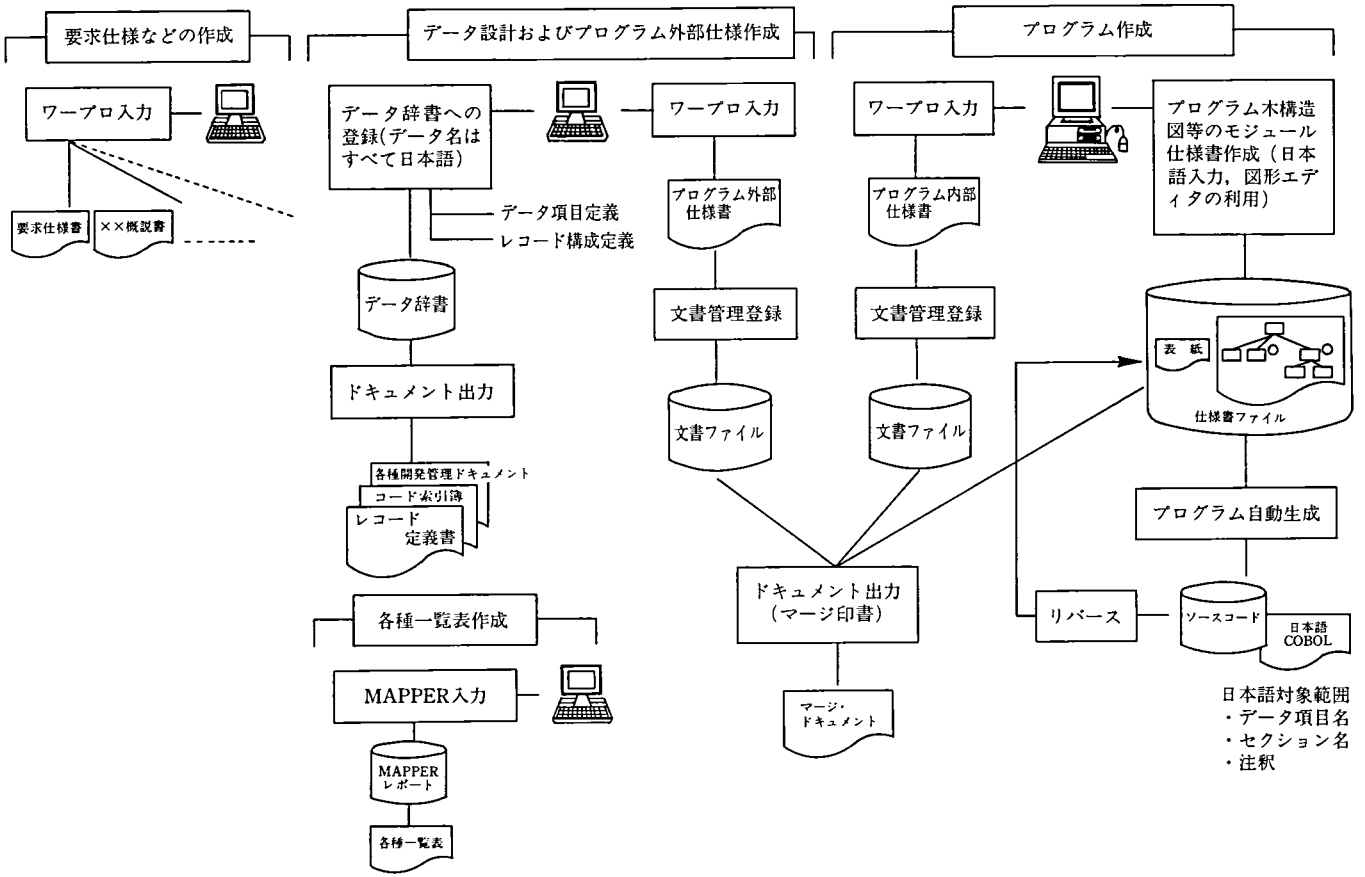


図8　ドキュメンテーション支援ツール全体図

Fig. 8　Structure of documentation support tools

が存在するため，電子ファイル化を推進するための基本機能として図形エディタが必要になる．図8のモジュール仕様書作成における木構造図は図形エディタ利用の代表例である．木構造図はIDES提供の図形エデイタをマウス・インタフェースで利用することによって作成している．

4.5.2 日本語COBOLによる漢字の積極的活用

ドキュメントの充実により，プログラムの理解をし易くし，保守時の修正を容易に行うことが可能になるが，プログラマにとってはプログラムそのものが理解し易いものになっていることも重要である．プログラムそのものの理解のし易さという観点では，プログラム構造や各種標準化等が重要な要素になるが，言語の表記は基本的な要素であるがゆえに影響度は他に比べ大きい．TRITON開発ではUCS COBOLのもとで日本語を全面的に使用し，従来のカナCOBOLから漢字COBOLの世界へと切換えを行った．漢字の積極的活用をしたUCS COBOLの採用により，ソースコード行数は約25%(同一プログラムを旧カナCOBOLと漢字を積極的に活用したUCS COBOL両方で作成し，比較調査した結果)増加したが，理解のし易さという面では十分に効果的であったと実感している．

漢字化の対象範囲は，すべてのデータ項目名(約12万項目からなる基本項目をはじめとし，集団項目名，01レベル名のすべて)，セクション名，注釈名である．

4.5.3 MAPPERの活用

数多くあるMAPPERの特徴のうち，ドキュメンテーションに関する特徴には次のものがある．

- 抽出，マッチングあるいはソート処理等の加工・編集が非常に容易にできる．
- レポート内容そのものが実行環境生成情報等への入力としてそのまま使用できる．

このような特徴を利用して，TRITON開発でMAPPERレポート上に作成した各種一覧表のうち代表的なものを表4に示す．

表4　MAPPERを利用した各種一覧表代表例
Table 4　Examples of lists made by MAPPER

一覧表名称	一覧表以外の利用
プログラム一覧表	進捗管理システムへの入力
テーブル・FCSS一覧表	実行環境生成情報への入力
帳表一覧表	帳表定義テーブルウェアの入力
トランザクション振り分け定義一覧表	実行環境生成情報への入力
エラー文言一覧表	エラー文言テーブルへの入力

MAPPERレポート化したことにより，一覧表ドキュメントを各種用途ごとの切り口で自由に加工編集したり，実行環境生成情報等に連動する基本データとして一元管理することが可能になった．

4.5.4 データ辞書を利用したドキュメント出力機能

次にデータ辞書を利用したドキュメント出力のうち，代表的なものを示す．

- レコード定義書

・レコード・レイアウト
・コード索引簿
・各種開発管理ドキュメント

レコード定義書，レコード・レイアウトあるいはコード索引簿等のデータ設計にかかわるドキュメントは従来からも辞書より出力していたが，今回はデータ定義そのものが日本語定義になっているため，カナ文字データ名(プログラムで使用されるデータ名)と漢字データ名の対応表を辞書に登録する作業が不要になり，省力化や信頼性の面で効果があった．

データ辞書上にはデータ設計にかかわる情報の他，プログラム管理に関する最新の情報が逐次蓄積されている．主な蓄積情報は次のものである．

・開発作業単位での各成果物(プログラム，仕様書，登録集，パラメタ等)の新規開発/修正に関する作業ステータス(開発予約・開発開始・開発完了)とリリースに関する情報
・プログラムとプログラム間の関連情報，データとプログラム間の関連情報

これらの情報をもとに開発作業一覧表，予約登録一覧表等の開発管理ドキュメントならびに各種影響度調査レポートが出力され，効果的かつ確実な保守作業のために活用している．

今後はプログラムやデータと画面・帳表等との関連情報をもたせてデータ辞書の内容を充実させ，影響度調査の範囲を拡大し，保守作業をより効率的・確実に行えるようにしたいと考えている．

4.5.5 文書管理システム

文書管理システムへの登録対象とした主なものは次のとおりである．

・各種概説書・解説書
・各種ガイド
・プログラム外部仕様書
・共通モジュール使用説明書
・プログラム内部仕様書

これらの文書を図9に示すような全体イメージで運用するようシステム構築をしたが，実態としては，後述のスピード面の問題のため，個々の直接開発部門は文書登録せず，集中部門にて登録を行った．

1グループは1人の管理者と複数の利用者で構成し，管理者の役割は利用者や書庫の登録ならびに各利用者の文書の登録・削除に対する承認とした．文書を検索する時は，利用者および書籍IDのグループとセキュリティ・レベルをチェックし，参照可能な文書か否かを決定する運用とした．共通書庫の文書はセキュリティ・レベルさえ合えばどのグループからも登録・参照可能になる．

登録対象とした文書のうち，各種概説書・解説書，各種ガイド，共通モジュール使用説明書等はサーバ上のみでの文書としているが，プログラム外部仕様書やプログラム内部仕様書はメイン・フレーム上にも転送し，仕様書ファイルとのマージ印書を可能にしている(図8)．

今まで述べてきたように，文書の電子ファイル上での一元的管理や，CASEツール

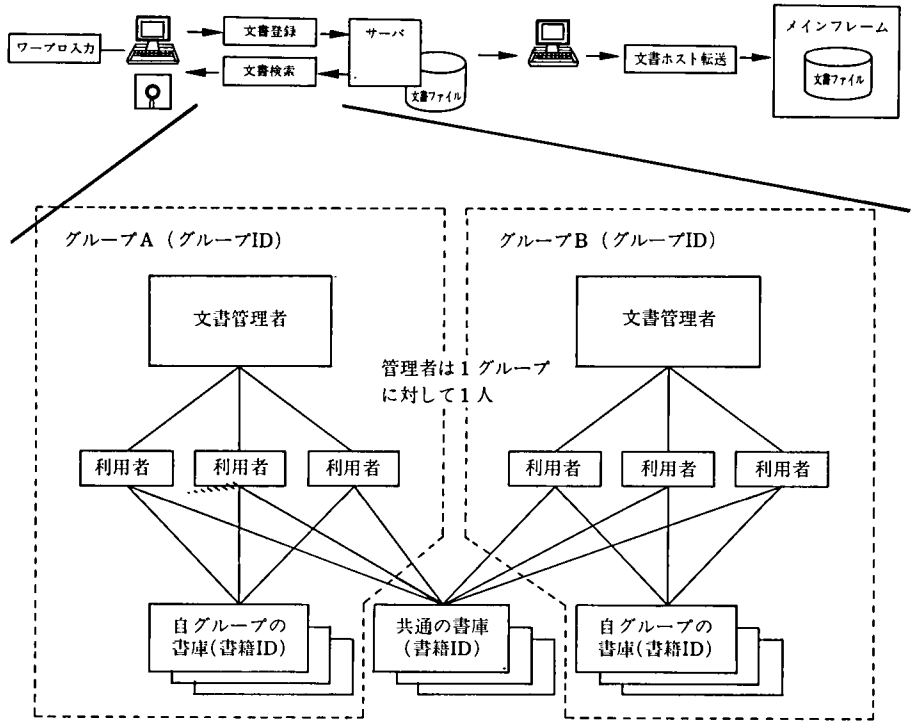


図9　文書管理の全体イメージ図

Fig. 9　Use of document management system

を利用した仕様書等とのマージ出力が可能になったが，文書管理システムは操作時のアクセス・スピードが速くなければなかなか定着しない．文書管理システムの適用途上で，PCの論理ドライブにサーバ上の文書ファイルをリモート・マウントする機能改善を行った．

その結果スピードを6倍程度速くすることができた．今後はMS-Windows*を使用して，ワープロ・ソフトウェアとその他の基本ソフトウェアのタスク・スイッチングを可能にすることによる操作スピードの更なる向上を計画している．

4.5.6　プログラム作成支援ツールとリバース・エンジニアリング

本節の冒頭でも述べたとおり，プログラム作成支援ツールおよびリバース・エンジニアリングはCASEツール“IDES”の機能の一部である．

プログラム作成支援ツールにおいてドキュメンテーションと密接な関係をもつのは木構造図である．保守作業において木構造図はプログラムの骨格を容易に読みとり，修正すべき箇所の特定を確実かつ短時間で行うことを可能にする．ちなみにTRITONシステムのオンライン機能の中核となる“業務処理モジュール”では，木構造図で示される処理単位は30～50個前後に分割されており，その一つの処理単位は平均20～30ステップ前後となっている．IDESはJSP(Jackson Structured Programming)に基づいた木構造図を使用して，モジュールの処理詳細を定義し，これをモジュー

* MS-Windowsはマイクロソフト社の商標である．

ル仕様書ファイルへ登録し，プログラム・ソースコードの生成を行っているため，木構造図ドキュメントとソースコードの一致が基本的に保証される．

IDES 適用当初はできなかったが，現在はソースコードから木構造図を逆コンバートするリバース機能も提供されており，正方向・逆方向の両方行で整合性をとることが可能になった．この機能により，木構造図ドキュメントとソースコードの一致はより確実なものになったと言える．

リバース機能についてはもう一つのサブ機能があり，ソースコードから日本語のプログラム内部仕様書を作成するものである．保守作業においては前述の木構造図とこの日本語プログラム内部仕様書に対し修正内容を明示し，プログラマの作業を効率的に実施することが可能である．リバースされた日本語プログラム内部仕様書については，現状では読み易さの面で若干の難点があるが，今後，日本語的抽象度をより高めるための改良が計画されている．

なお，IDES については本特集号別稿“開発工程と IDES”および文献[4]の“IDES のリバース機能”を参照されたい．

5. 効果および評価

1章で整理した課題を中心に振り返ってみることにする．

開発を手戻りなく効率的に行うためには，ドキュメントの記述内容・作成順序が重要である．とくに要求定義(詳細定義)から設計工程にかけて作成されるドキュメントこそ，システム開発の命運を握っているといっても過言ではない．今回の TRITON 開発では，次のような事情から“性急に結果をもとめる”開発になる傾向が強く，その結果，とくに処理機能に関するドキュメント定義では試行錯誤的な部分が発生したことを反省している．

- 銀行勘定系システムはシステム化以降 25 年の年月を経ており，既存機能についてはすでに確立(人の頭の中で内容が十分に把握されていという意味で)したものであること．
- 共同開発という事情の中で，銀行システムのスペシャリストが数多くシステム開発に参加していること．

実態としては，長年の保守作業の中では，保守対象外であった部分についてはなかなか理解し得ないことや，要員の人材育成面から必ずしもスペシャリストを充てることができない等の事情があり，ドキュメント定義面でも“急がば廻れ”が結果的に落ち着きどころになったと考えている．

保守における対象システムの理解を容易にするドキュメントの均質性については，次の二点から通常開発ではなかなか達成できない均質性を得ることができたと評価している．

- 三者共同開発という条件のもと，成果物交換という義務を互いに負っていたことによる要員のモラルの向上．
- ドキュメンテーション支援ツールによる標準化・手順遵守の歯止め．

標準化や電子ファイル化による作業の機械化・自動化については，4章で述べたとおり，次のような効果を得ることができた．

- フォワードおよびリバース技術による仕様書とソフトウェアの一致が図れる環境が整備されてきたこと.
- データ辞書とドキュメント出力機能によるドキュメントの確実さと出力の容易性が図れたこと.
- 保守情報の充実により，確実な保守を可能にする環境が整備されたこと.
- 日本語 COBOL での全面的な漢字採用によるデータおよびソースコードの理解のし易さが図れたこと.
- ワード・プロセッサと文書管理システムにより，ワープロ・ドキュメントの一元管理を行う環境が整備されたこと.
- MAPPER システムの活用により，各種索引的ドキュメント(一覧表の類)の一元管理ならびに実行環境生成情報との連動を可能にしたこと.

以上のような効果を得ることはできたが，今後改善すべき項目が残っているのは 4 章で述べたとおりである.

共同・分散開発環境下でのドキュメントの収集とデリバリについては，専担管理部門の設置，精緻な開発手順書，管理ツール等により，確実に行うことができたと評価している.

6. 今後の展望

今回の TRITON 開発では，5 章にまとめたような効果を得ることができたが，今後各企業で定着化を図るためには，ドキュメンテーション支援ツールの更なる改善と，各企業での開発作業や保守作業に関する強力な手順遵守の姿勢と指導力が重要であると考えている.

なお，今回の TRITON システムでは検討されなかったが，多量のドキュメントの関連付けをどうするかという課題がある. 現在，TRITON 開発で使用した CASE ツール IDES は，上流工程の支援機能を開発中であり，ER ダイアグラム*，データフロー・ダイアグラム**等を対象としている. これらとモジュールの構造図やソースコードあるいはワープロ・ドキュメント等の関連付けを行い，データベース化することにより，効率的な情報検索が期待できる.

振り返ってみると，以前は数人に 1 台であった開発用ターミナルが，ワードプロセッサ・ソフト，LAN 環境ならびに，ラップトップ型ターミナルの普及により，1 人 1 台の文化が根づいてきたというのが実感である. 現在，通信分野の技術的進展はその高速性と同時多重通信性において目ざましく，前述のオフィス内の環境整備と組み合わせ，高速 LAN 間接続やグループウェア***によるドキュメント・システムへと変遷していくのではないだろうか.

7. おわりに

目標に対する到達点や反省点，あるいは今後の展望については 5 章，6 章で述べた.

---

* システムが関わる実体間の関係を明確にする関連図.

** システムの中をデータがどのように流れるかを論理的，抽象的に図で表現するもの.

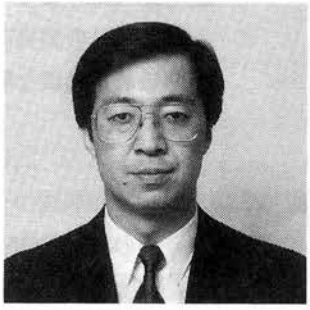
*** 会社等の組織ではひとりで仕事を行うことは希であり，作業の多くは複数の人間(グループ)による共同作業として行われる. このグループの作業を効率的に行えるように支援し，生産性を高めるためのコンピュータ・システムのこと.

各企業におけるシステム化対象業務の内容や過去の歴史の違い等により，それぞれに合ったドキュメントの標準化を設定するべきと考えるが，今回のTRITONシステムでの工夫・考慮が多少なりとも参考になれば幸いである．

最後に，TRITON開発時にさまざまな助言をいただいた百五銀行，紀陽銀行の皆様に心から感謝の意を表する．

参考文献
［1］ E. Yourdon著，黒田純一郎，渡部研一共訳，“構造化手法によるソフトウェア開発”，日経BP社，1989年2月，pp. 149～165.
［2］ 菅野文友編集，“ソフトウェアの品質管理”，日科技連，1988年6月，pp. 203～222.
［3］ 花田収悦編集，“ソフトウェアの仕様化と設計”，日科技連，1988年6月，pp. 89～142.
［4］ 儀間哲雄，“IDESのリバース機能”，UNISYS技報，Vol. 13 No. 2，1993年8月，pp. 71～83.

執筆者紹介 村田豊彦（Toyohiko Murata）
昭和49年慶応大学工学部管理工学科卒業．同年日本ユニシス(株)入社．金融機関のSEサービス，システム開発に従事．現在 金融システム企画開発本部TRITON企画推進部課長．



保守性の良いプログラム構造

Program Structure Providing High Maintainability

竹 村 匡 弘

要 約 TRITONシステムではプログラムの保守性を高めるために，プログラムの理解のし易さ，修正負荷の軽減等にポイントをおき，さまざまな工夫をしてきた．たとえば，階層化構造設計の採用，漢字COBOLの採用，テーブルウェアや部品の活用等が挙げられる．また，CASEツール"IDES" (Integrated Development Environment support System) の使用により，データ設計・保守時の階層化構造設計遵守ならびにプログラム作成・修正時の構造化手法遵守の歯止めがかかっており，データやプログラムの均質化による保守性の高さが維持されていくと確信する．

本稿では，上記の工夫について具体的な内容とその効果について述べる．なお，プログラム構造については，オンラインとバッチでプログラムの特性が異なるため，個別に解説する．

今後は，リポジトリの充実や，リバース・エンジニアリングの利用により，プログラムやリポジトリから保守に役立つドキュメントを出力して活用する等のプログラムの保守性向上が考えられる．

Abstract For the sake of its boosted program maintainability, the TRITON system has been embedded with various enhancements in search of higher program understandability and a greater reduction in loads involved in program correction efforts. They include, to name a few, the adoption of hierarchical structure designing and UCS COBOL, in addition to the employment of tableware and parts. The author is confident that the CASE tool —— "IDES" (Integrated Development Environment support System) used in developing the TRITON system —— helps software people a great deal to observe (1) hierarchical structure designing for tasks to design and maintain data, and (2) the structural method for programming and program corrections; thus resulting in gaining high maintainability through the improved homogeneity of data and programs.

This paper discusses the details and effects of those enhancements in addition to mentioning an individual program structure each for on-line and batch-processing applications because program characteristics differ between the two applications. Through his further enhancement of data dictionaries and use of reverse engineering, the author intends to keep on improving program maintainability by making it feasible to use maintenance-assisting documents printed out of programs and data dictionaries.

1. はじめに

一般的に数百万ステップに及ぶ巨大なシステムになると，複雑化して保守が非常にむずかしくなり，保守工数も膨大なものとなる．そして動いているシステムである以上，プログラムステップ数は増大する傾向にあり，プログラムの保守性向上が非常に重要な課題となってくる．そこでTRITONではプログラムの骨格を定めるプログラム構造に着眼して，「保守性の良いプログラム構造」を目指して開発を行った．

「保守性の良いプログラム構造」とはどういうものか．「保守性の良い」すなわち「保

守のし易い」とは，修正が容易であり，かつ確実に行われるということである．そのためには，理解し易い，検証し易いといったことが必要になってくる．

このようなことを兼ね備えたプログラム構造にするためには，構造化，モジュール化，標準化といった工夫が必要になってくる．

また，基盤ソフトウェアの拡充を図り，アプリケーション・プログラムの作成・保守の負荷を軽減するということも重要なことである．

本稿では，2章で基盤ソフトウェアとアプリケーション・プログラムについて記述し，3章で保守性という観点から見て工夫したアプリケーション・プログラムの特徴を概観し，4章以降でそれらの特徴について詳述する．

2. 基盤ソフトウェアとアプリケーション・プログラム

保守性の良いプログラム構造にするためには，プログラムを構築するための基盤ソフトウェアの機能が重要になってくる．その理由の一つは，勘定系の要点である高信頼性・高効率性・高負荷対策といったシステムの仕掛に関わる機能をアプリケーション・プログラムから取り除くことにより，シンプルで保守のし易いプログラムを容易に作成することができるようになることである．もう一つの理由は，プログラムを開発・保守し易い環境構築および保守性の良いプログラムを作り出す機能が重要であるが，このために日本語処理やデバッグ支援機能およびCASEツール等の基盤ソフトウェアが必要になってくることである（ここでは，CASEツールも基盤ソフトウェアに含めることにする）．本章では，TRITONで採用した基盤ソフトウェアについて，保守性という面から捉えて，その特徴を簡単に紹介する．

2.1 基盤ソフトウェアとオンライン・プログラム

1) HVTIP（大量トランザクション制御機能ソフトウェア）……HVTIP（High Volume Transaction Interface Package）は，使用者プログラム間の連動を容易にし，大規模トランザクション処理を行うための基本ソフトウェアである．

HVTIPでは，モジュール1本1本がLINK(プログラムの連結編集)の単位であり，あるモジュールが修正になった場合，そのモジュールだけを再LINKして入れ換えればよいようになっているため，従来のようにそのモジュールを使用している全プログラムを再LINKする必要がなくなった．

2) XIS（統合実行環境支援システム）……XIS（eXtended Information System）はオンライン・トランザクション処理のプラットフォーム構築に関わる開発から運用・保守に至るまでを総合的に支援するソフトウェア群である．

XISの中にはプログラムの保守性向上に寄与する機能が数多くあるが，その代表的なものを以下に記述する．詳細は本特集号別稿“銀行システムとXIS”を参照されたい．

① 従来はファイルあるいはテーブルをディスク上に持つか，メモリ上に持つかにより，プログラム上でインタフェースを使用者が使い分ける必要があった．XISでは，両者のインタフェースを統合し，使用者は媒体(メモリ，ディスク)を別途定義するだけでよい．このため，ファイルやテーブルについて媒体変更があってもプログラムの修正が不要になった．

② ファイル番号（物理番号）はシステム内でユニークにする必要があり，この変更は，従来プログラムの修正を伴った．しかし，XIS では論理番号と物理番号との変換テーブルを持ち，アプリケーションは論理番号を指定することに変更された．このため，システム変更時にファイルの物理番号が変わっても変換テーブルを変更するだけでよく，プログラム変更の必要がなくなった．

③ 従来は，為替のジャーナル・ファイル等の処理効率を上げるためにオンライン・ファイルを使用する場合，店ごとにファイルのレコード番号を管理したり，オーバーフロー対応等の仕掛けを作成する必要があった．XIS では，この仕掛けを標準提供しており，アプリケーションの開発・保守性を向上させた．

④ ファイル・アクセス時にレコード内容やパケット（XIS プリミティブの引数）のトレースを採取する機能が準備され，デバッグ時は外部より指示（コマンド入力）するだけで，トレース情報が採取できるようになった．これにより，プログラムの変更を伴わずにデバッグが可能となり，テストがやり易くなった．

3) UCS-COBOL（漢字 COBOL：ANSI 85 版）……UCS-COBOL の採用により，データ項目名やプログラムの段落名に日本語が使えるようになった．これにより，コメント等の付加を行わなくてもプログラムが非常に読み易くなり，プログラムの解読時間が短縮された．

4) IDES（統合開発環境支援システム）……プログラムを開発・保守し易い環境構築および保守性の良いプログラムを作り出すために，TRITON で採用した CASE ツールが IDES である．IDES の各ツールのうち，プログラムの保守性にとくに関係する設計支援ツール，プログラム作成支援ツール，単体テスト支援ツール，プログラム管理ツールについて述べる．なお，これらの IDES ツールはすべて日本語対応がされており，保守する上で非常に解り易いものとなっている．詳細は本特集号別稿“開発工程と IDES”を参照されたい．

① IDES 設計支援ツールは，データ項目やレコード項目等の設計情報をリポジトリ（辞書）としてデータベース化し，一元管理することにより，保守上次のような特徴をもっている．まず，データ項目の変更があった場合，影響範囲を容易に得ることができ，関連する集団項目や登録集を自動生成できるので保守が容易かつ確実にできる．次に設計情報に変更があったとき，レコードレイアウト等の設計ドキュメントを電子ファイルであるリポジトリから自動的に出力できるので，ドキュメントの保守が非常に容易であり，設計情報とドキュメントの乖離をなくし，整合性の維持を可能とした．

② IDES プログラム作成支援ツールを使用することにより，JSP 木構造図表現によるモジュール仕様書が作成され，そこからプログラム・ソースが自動生成される．プログラムの変更は，プログラム作成支援ツールを使用し，木構造図の変更にて行えるため，プログラムの構造を視覚的に捉えながら保守できるようになった．また IDES により，構造化プログラミングやコーディング規約を遵守するための歯止めがかかり，プログラムの均質性が保証されるようになった．なお，プログラム作成支援ツールの中にはマクロの機能もあるが，これについては，6 章で述べる．

③ IDES単体テスト支援ツールは，モジュールに変更があった時の確認テストを容易に行い，検証をすべて日本語のデータ項目名で行うことを可能にした．また，会話型デバッグ・ツールである PADS (Programming Advanced Debugging System) をこの単体テストの中で使用できるので，COBOL のソースライン指定により任意の箇所で実行を中断させ内容を確認でき，非常にプログラムの保守がし易くなった．その他に網羅率の計測機能もあり，保守時の確認テストをより完全なものにできる．

④ IDES プログラム管理ツールは，リポジトリでモジュール・登録集・集団項目・データ項目等の関連情報を管理しており，データ項目・集団項目・登録集・モジュール等に変更があった時のインパクト（影響範囲）情報を容易に得ることができる．また，影響のあるモジュールを自動的にコンパイルしたり，再 LINK したりすることもでき，プログラムの保守が容易になった．とくに，TRITON のような大規模システムでは，このような機能は保守上必須であると考える．

2.2 基盤ソフトウェアとバッチ・プログラム

1) NPE（NEW プログラミング環境）

ダイナミック・リンキングが可能になったことにより，サブモジュールが修正になっても，そのモジュールを使用しているすべてのプログラムを再 LINK する必要がなく，そのモジュールの入れ替えだけで済むようになった．これにより，リアル・プログラムによる HVTIP 構造同様の形態が実現された．

2) XIS（統合実行環境支援システム）

2.1 節の 2) XIS の機能以外に，以下の機能を使用できる．

ハイレベル・データベース・インタフェースの提供により，プログラムを物理的媒体から解放し，エリアを意識することなくプログラミングできるようになった．

代表的な例を以下に列挙する．

- 基本検索機能により，配置モードをとくに意識しなくても，データベースを検索するプログラムが作成できる．
- エリア自動決定機能により，使用者はエリアを意識する必要がない．
- ロジカルセット機能により，セット関係を持たせることなしに親子関係を表現できる．
- インバーテッド検索機能により，データベース上のレコードを，配置モードキー以外の項目をキーとして検索できる．
- マルチ・ユーザ・レコード機能により，一つのレコード型に，複数のユーザレコード型を含めることができる．

3) UCS-COBOL

2.1 節の 3) UCS-COBOL と同様である．

4) IDES（統合開発支援システム）

2.1 節の 4) IDES の機能以外に，以下のバッチ帳票設計支援ツールが使用できる．

- IDES バッチ帳票設計支援ツールを使用することにより，視覚的な帳票設計(画

面上でレイアウトを作成するイメージ）が可能となった．また，設計結果はリポジトリに登録され，リポジトリからのCOBOL報告書登録集の自動生成や，リポジトリによるデータ項目影響調査も可能としている．

3. TRITONアプリケーション・プログラムの特徴

理解のし易さ，修正のし易さのために工夫した点のうち，主なものを以下に列挙する．

1) 階層化設計によるデータ構造，プログラム構造の採用により，修正の範囲が局所化された．なお，データ部分はIDES設計支援ツールによりリポジトリとして管理・保守され，プログラム部分はIDESプログラム作成支援ツールによりモジュール仕様書（木構造図）として管理・保守されている．

2) テーブルウェアの活用により，プログラムの変更を伴なう修正が減った．端末やコードに依存する部分のように，金融アプリケーションの中でとくにプログラムの構造に大きく影響するものを中心にテーブルウェア化することにより，保守性の向上を目指した．

3) マクロやひな型といった部品の利用により，プログラムの均質化と作成量の軽減を図った．なお，この両者ともIDESプログラム作成支援ツールの機能を利用したものである．

4. 階層化設計によるデータ構造・プログラム構造

プログラム構造はデータ構造によって規定される．本章では，まずデータ構造について記述し，次にプログラム構造とモジュール設計基準についてオンラインを中心に保守性という観点から記述する．

4.1 データ構造

4.1.1 ファイル構造

顧客情報の一元管理のし易さ，複合商品の取引が多くなってきている点を考慮し，科目間相互参照や科目間連動（一つの取引で複数科目の処理を行うもの）の処理効率を重視してCMF* 体系を採用した．また，CMF体系を採用することにより，科目別ファイルに比べデータの一元管理がし易いため，システム全体でデータ項目の重複が少なくなるという副次効果も狙った．

4.1.2 レコード構造

1) データの階層化……図1の例に示すように，レコード(TRITONではこれを物理レコードと言う）のデータエリアは階層化構造をとり，同一種類の物理レコードは固有部分以外，同一構造を持つ．各々の階層化されたデータエリアは部品化され（この部品化されたものをレコード部品と呼ぶことにする）一元管理されているため，新規レコードの作成時は網かけの部分のレコード部品を作ればよい．また，データ構造とプログラム構造を図2のように対応させているため，データ項目の変更に対してプログラムの変更箇所が特定され，保守し易くなっている．

---

* CMF (Customer Management File)は，CIF (Customer Information File)，各科目レコードが物理的，論理的に統合されているファイル体系のことである

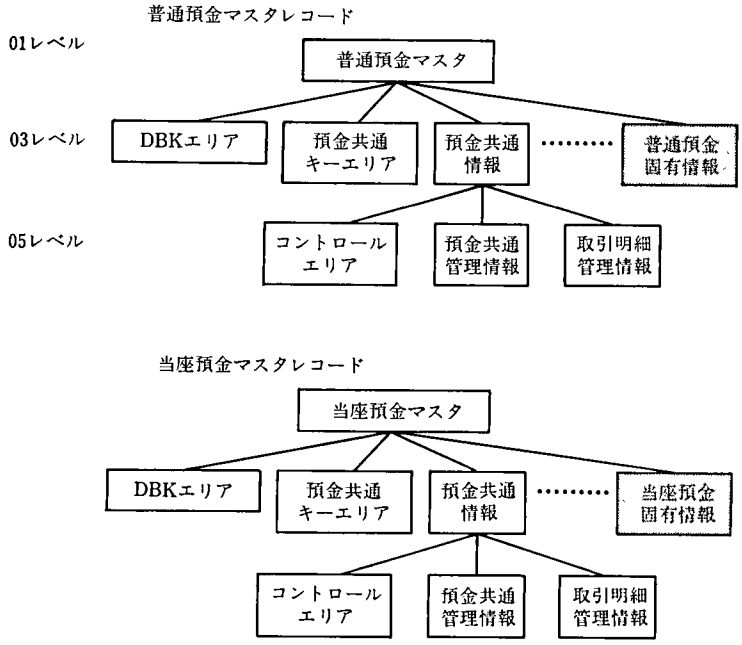


図 1　階層化されたデータ構造の例

Fig. 1　Sample of stratiformed structure on data

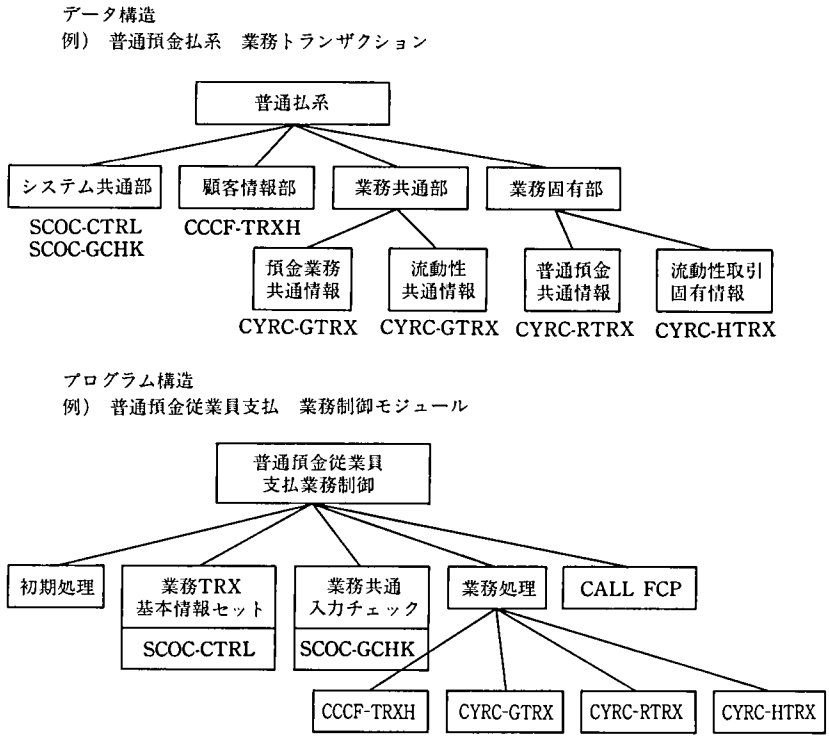


図 2　データ構造とプログラム構造の例

Fig. 2　Sample of data and the structure of program

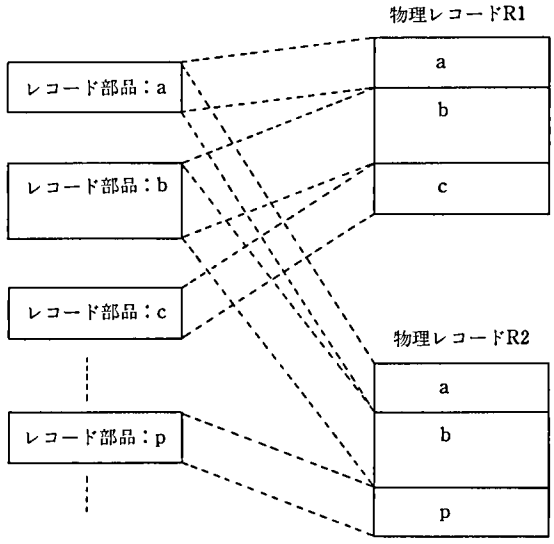


図 3　物理レコードとレコード部品の関係
Fig. 3　Relation between physical records and parts of records

階層化されたデータエリアの部品化とその効果については，2)で詳述する．

2)　階層化されたデータエリアの部品化と効果……物理レコードは，COBOL 登録集の 01 レベルに対応する．そして，物理レコードは複数のレコード部品(03 レベル）から成る．

図 3 の例では，物理レコード R 1 はレコード部品 a，b，c を組み合わせて作成されている．また，物理レコード R 2 はレコード部品 a，b，p を組み合わせて作成されている．たとえば，新商品追加等により物理レコードを新規作成する場合は，図 1 で示されるように，既存のレコード部品と新設したレコード部品を組み合わせて作成する．

ここで，このレコード部品は，下位の集団項目またはデータ項目から成る．また，集団項目は，さらに下位の集団項目またはデータ項目から成り，最下位はデータ項目である．これらが，図 1 の例に示すような階層構造となっている．図 1 では，01，03，05 レベルまでしか書いていないが，実際は，さらに 07，09，…レベルへと続き，データ項目が最下位レベルとなる．

TRITON では，このレコード構成を，IDES 設計支援ツールにて，リポジトリとして管理している．管理方法は，最小単位であるデータ項目の項目名とその属性（サイズとタイプ）を全体一意として管理し，このデータ項目の集まりをレコード部品としてリポジトリに持つ．使用者は，物理レコードを作るにあたり，この部品を選択することにより，COBOL 登録集を作成する．またこの時，同じ部品であっても，物理レコードごとにユニークな接頭語を，IDES により自動的につけることにより，プログラムの中で扱い易くしている．

このようにして，IDES により，データ項目定義やレコード定義等の設計情報をデータベース化し一元管理することにより，データ項目を修正したとき，関連する集団項目や登録集の自動生成およびプログラムへの影響範囲の特定ができ，保守性向上が図

られている．

4.2 プログラム構造とモジュール設計基準

4.2.1 プログラム構造

1) オンライン・プログラム構造……オンライン処理を行うプログラムを機能単位に図4に示す5階層とした．階層分けにより，プログラム作成や保守における役割分担および機能分担が明確化された．たとえば，端末仕様に依存する部分を1階層目および5階層目に吸収し，アプリケーション処理を2・3・4階層目と位置付けた．この結果，端末変更時の影響範囲の特定化とオンライン業務処理からの端末仕様に依存する部分の排除が可能となり，アプリケーションの開発が容易に行えるようになった．

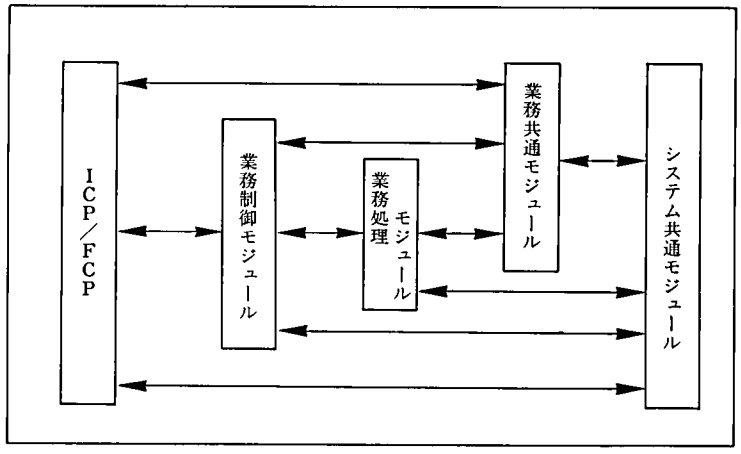


図4 階層化されたオンライン・モジュールの関連図

Fig. 4 Connection of stratiformed module

① 1階層目：ICP/FCP

- ICP（Initial Control Program）は業務処理開始モジュールのことで，入力電文の受信および編集処理・各種初期処理・業務制御モジュールの決定および分岐処理等を行う．なお，ICPは，電文仕様の違いを電文編集として吸収するモジュールであり，電文の種類単位に作成され，XISが提供している．
- FCP（Final Control Program）は業務処理終了モジュールのことで，出力電文編集および送出処理・各種終了処理等を行う．なお，FCPはシステム全体で一個であり，XISが提供している．

② 2階層目：業務制御モジュール

業務ごとの類似取引パターン単位に作成し，複数の業務処理モジュールの呼び出し制御と共通的処理を行う．また，科目間の連動処理の制御も行っている．

③ 3階層目：業務処理モジュール

入力チェック，マスタ読み込み，トランザクションとマスタの突き合わせ，マスタ更新，出力作成等の業務処理そのものを行う．

④　4階層目：業務共通モジュール

利息計算，トランザクション作成等の共通モジュール群の階層である．

⑤　5階層目：システム共通モジュール

ファイル/テーブルのハンドラ，XIS プリミティブ等のシステム全体で共通に使用されるモジュール群の階層である．

2)　バッチ・プログラム構造……オンライン・プログラムは，入力電文の取得から出力電文の送出に至るまで一定の流れがあり，定型化し易いため，モジュールを階層構造に規定した．しかし，バッチ・プログラムは，その処理形態が多岐にわたるため，むしろ業務処理モジュールの基本パターンの充実に注力することとし，プログラム構造としては業務処理モジュールと業務共通モジュールおよびシステム共通モジュールの3階層を基本とするにとどめた．

業務処理モジュールの基本パターンについては6.1.2項で記述する．

4.2.2　モジュール設計基準

モジュールの柔軟性・拡張性を確保し保守性を向上させることを目的として，オンラインの各階層に分けられたモジュールに関して設計基準を設定し，プログラムの均質化・標準化を図って見易く解り易いものとした．

なお，ここではオンライン・モジュールを中心に説明するが，業務制御モジュール等オンライン・プログラム固有の部分に関する記述を除けば，バッチ・モジュールについても基本的に同様である．

1)　全般基準……モジュールの設計基準として，「システム設計基準や設計便覧等の規約集」(基本設計段階)，「モジュール仕様書記述要領(オンライン編，バッチ編)やコーディング規約（オンライン編，バッチ編）等」(詳細設計段階）を作成し，開発におけるプログラムの均質化を図った．

たとえば以下のような基準がある．

①　1モジュールの有する機能は，システムでユニークなものとし，機能変更時の修正対象範囲の局所化をはかる．

②　モジュールは，そのモジュール内に唯一かつシステム内でユニークなエントリ名を持ち，唯一の出口を持つ．

③　モジュール間インタフェースとして，パケット（いくつかのデータ項目をまとめて一つの集団項目としたもの）の使用，レコード単位の引渡しを基本とすることにより，引数を減らす．

④　システム全体の共有データをシステム共通領域に配置し，共通データを一元管理する．

⑤　COBOL の一つの SECTION は見易さのために，1ページ(約50行)に納まる程度にモジュール内を分割する．

⑥　木構造図があまり複雑にならないように，わかり易さを目指して，頭で覚えられる範囲の4階層以内で表現する．

2)　業務制御モジュール設計基準……業務制御モジュールについては設計基準を規定せず，より具体的に単一取引形態および連動取引形態で3パターンのひな型を用意し，これを流用して作成することを基本とした．ひな型については6章で記

述する．

3) 業務処理モジュール設計基準……業務処理モジュールは個々の取引を構成するメインプログラムであり，設計基準としては，システム設計基準や設計便覧等の規約集，モジュール仕様書記述要領やコーディング規約およびIDES木構造での標準化ルールがある．

(例)

●ファイルのリード・ライトに関して

① 親子関係を持つ子レコードの入出力を行う場合は，先に親レコードの読み込みをロック付きリードで行うこと．

② 子レコードについてはロック付きリード命令は使用せず，リード命令を使用すること．

③ ファイルのリードは，口座変換ファイル（ロックなし）→CIF（ロック付き）→各マスタ（ロックなし）のように決められたシーケンスを守ること（デッドロックを回避するため)．

●連動処理に関して

① 被連動側の業務トランザクションおよび入力トランザクションに必要となる基本情報は，連動側の業務処理で連動引継ぎエリアにセットする．

② 被連動側の処理が必要な場合，連動側業務トランザクションのコントロール部のフラグ情報の連動要否表示をオンすること．

③ 業務処理モジュールは，単独・連動・被連動を意識して処理を行う．

④ 被連動側の入力トランザクション可変部の情報で，不足している項目があれば，マスタリード後，情報を補完して処理を続行する．

4) 共通モジュール設計基準……TRITONにおける共通モジュールは，システム共通モジュールと業務共通モジュールに大別され，これらはプログラムの階層モジュール構成における部品として扱う．

システム共通モジュールは，業務処理的機能を可能な限り排除して業務の変更に伴う修正が発生しないようにする．

業務共通モジュールについても，固有の業務処理機能を盛り込みすぎて複雑な構造にならないようにする．これは保守が困難になるのを防ぐためである．

共通モジュール全般として，共通モジュール内でのテーブル・ファイルの読み込み・更新については，同一レコードが複数箇所での操作とならないようにする．また，類似した機能を有するモジュールが複数発生することを避けるために，テーブルウェア化等により柔軟性を確保する．

生産性を向上させるために，できるだけ共通モジュール化を図ることは重要であるが，あまり共通モジュールに機能を盛り込みすぎて複雑にしてしまうと，保守性という面で共通モジュールの保守負荷大という問題が発生する．TRITONでは共通モジュールをできるだけシンプルにし，保守性の向上を目指した．

ちなみに，TRITONと従来の勘定系パッケージとを比較してみると，共通モジュールの全モジュールに対する本数比率は両者共だいたい同じであるが，ステップ数比率は約15％と約30％というように大きな開きがあり，TRITONの方が

共通モジュールのステップ数比率が小さい（処理が簡潔である）という結果がでている．

5. テーブルウェア化の範囲と効果

従来のテーブルに加えて，処理に伴う識別・判別も含めたテーブルウェア化の一層の推進により，システム全体の保守性・拡張性の向上をはかることができた．

TRITON における主なテーブルウェア化の対象範囲とその効果について以下に記述する．

5.1 端末依存部分のテーブルウェア化

プログラムから端末編集に関わる部分をテーブルウェア化することにより，アプリケーションから独立した位置付けとした．この結果，新規画面の追加がし易くなった．

なお，テーブルの元データである編集パラメタは，従来のような単純な編集指示パラメタではなく，条件判定による個別のデータ処理等も可能となるような疑似 COBOL 言語で記述されている．

・対象テーブル名：入出力編集テーブル

・効果　　：画面の追加/変更がテーブルの追加/変更でほとんど対応できるようになった．

5.2 営業店事務取扱い上の外部コード・用語のテーブルウェア化

外部コードの違いをアプリケーション・プログラム内に持ち込まず，違う部分はテーブルウェアとして入出力編集テーブルに吸収させ，外部コードと内部コードの変換をして，プログラム内では内部コードで処理している．

ここで外部コードとは端末機の入出力帳票やセンタからの還元資料等，利用者が利用する各種帳表に使われるコードのことであり，内部コードとはコンピュータ・システム内部で使用されるコードのことで，利用者が意識しないものである．

また，出力用語の違いをアプリケーション・プログラム内に持ち込まないために，テーブルウェア化し，プログラム内部でもつ用語 No と用語名との変換をオンラインでは端末出力編集ルーチンで，バッチでは帳表出力編集処理の中で行っている．

・対象テーブル名：入出力編集テーブル，用語テーブル（オンライン，バッチ）

・効果　　：パッケージ導入ユーザのカスタマイズ負荷軽減をはかることができる．現在使用中の外部コード・用語に容易に合わせられ，それらの変更時の保守性が向上した．

5.3 処理条件・処理パターンのテーブルウェア化

金融商品属性や事故チェック，役席承認・照合取引チェック等のテーブルウェア化を図った．また，委託者単位のセンタカット属性情報および委託者情報をテーブルウェア化し，センタカットの受付処理時のチェックやセンタカット入力ファイル作成時の補完情報として使用している．

・対象テーブル名：商品テーブル，委託先管理テーブル，事故テーブル，役席承認・照合取引テーブル

・効果　：商品属性変更，新商品追加やセンタカット処理における委託先別の変更・追加等がプログラムへ与える影響を極小化できる．
また，委託者情報テーブルに委託者の契約内容を保持することにより，口座振替のプログラム作成本数を削減できた（20本程度のプログラムで4000種目～5000種目の処理を行っている）．

5.4 バッチ処理日付のテーブルウェア化

通常はバッチ日付テーブルの内容をそのまま使用するが，基準日を変更する場合は，バッチラン ID 日付テーブルにラン ID と固有の基準日をセットし，その基準日を元にバッチ日付テーブルを再作成したものを使用する．

・対象テーブル名：バッチラン ID 日付テーブル
・効果　：この機能を使用することにより，アプリケーション・プログラムの中でとくに考慮しなくても，バッチ処理日の任意化が容易に図れるようになった．

5.5 バッチ帳表に関する情報のテーブルウェア化

帳表定義に関する詳細情報をテーブルウェア化し，IDES バッチ帳表定義支援ツールと組み合わせて使用することにより，バッチ帳表作成プログラムの作成を容易にしている．

このテーブルの中には単純な編集指示だけではなく，加工・計算等の指示情報も持ち，共通モジュールを介して編集処理ができるようになっている．

・対象テーブル名：バッチ帳表定義テーブル
・効果　：プログラムから編集情報を外出ししたことにより，バッチ帳表の変更・追加が容易になり保守性が向上した．

6. プログラムの部品化の範囲と効果

プログラムの生産性と保守性の向上を目指して，以下の2種類の部品化（ひな型とマクロ）を行った．

6.1 プログラム（木構造図）のひな型

6.1.1 オンライン業務制御モジュールのひな型

オンラインの業務制御モジュールは，図5に示すように，大部分が共通のコーディングとなる．そこで，木構造図（モジュール仕様書）のひな型を提供し，それを流用して作成することを基本とし，以下の3種類のひな型を用意した．

① 単純版1：1業務制御モジュールが1業務処理モジュールを制御するパターン用（図5のパターン）
② 単純版2：1業務制御モジュールが複数業務処理モジュールを制御するパターン用（ただし，連動処理はなし）
③ 複雑版　：1業務制御モジュールが，連動処理により複数業務処理モジュールを制御するパターン用

ひな型を使用することにより，プログラムの生産性が上がり，プログラム構造やコ

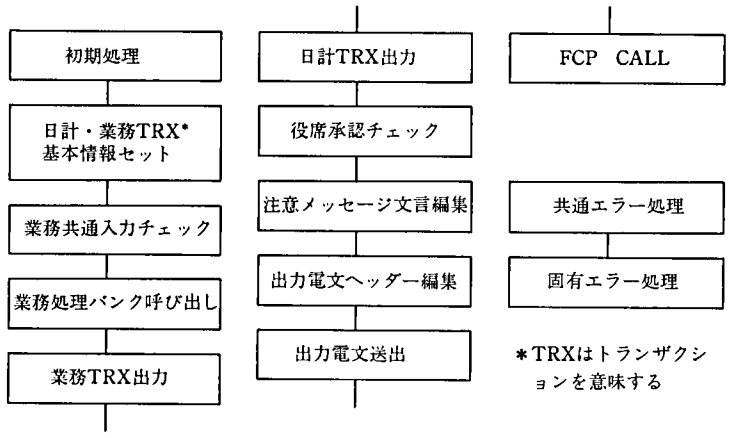


図 5　業務制御モジュールの内容（単純版1の例）
Fig. 5　Content of control module

ーディングが均質化された．

6.1.2　バッチ業務処理モジュールのひな型

バッチの業務処理モジュールは，個々のプログラムの処理機能によって木構造の作り方が様々になるので，いくつかの基本パターンについてひな型を提供し，プログラム作成時は極力このひな型を流用している．

以下に TRITON で用意した7種類の木構造のバッチ用ひな型(基本パターン)を示す．また，図6に「マッチングによる更新処理と照合処理の木構造図のひな型」を例として示す．

1)　分配のパターン……一つの入力ファイルを，正当データとエラーデータとに振り分ける例をひな型としている．

2)　集計のパターン1……一つの入力ファイル（ソート済み）から合計の集計を行い，合計のレコードを出力するパターンで，小計・中計・大計と最後に総合計のレコードを出力するパターンをひな型としている．

3)　集計のパターン2……集計のパターン1と同様の集計処理を COBOL の報告書機能によって行うパターンをひな型としている．

　キー割れ処理は，報告書機能によって行うため木構造図は集計のパターン1に比べてずっとシンプルである．

4)　マッチングによる更新処理のパターン……マスタファイルをトランザクション・ファイルによって更新するパターンをひな型とする．

　ここでは，マスタファイルに該当するレコードが存在しないときは，トランザクション・ファイルから新規にマスタファイルを作成している．

　なお，ファイルはすべてシーケンシャル・ファイルとする．

5)　マッチングによる照合処理のパターン……マスタファイルとトランザクション・ファイルを照合し，次のようにマッチ/アンマッチ処理を行うパターンをひな型とする．

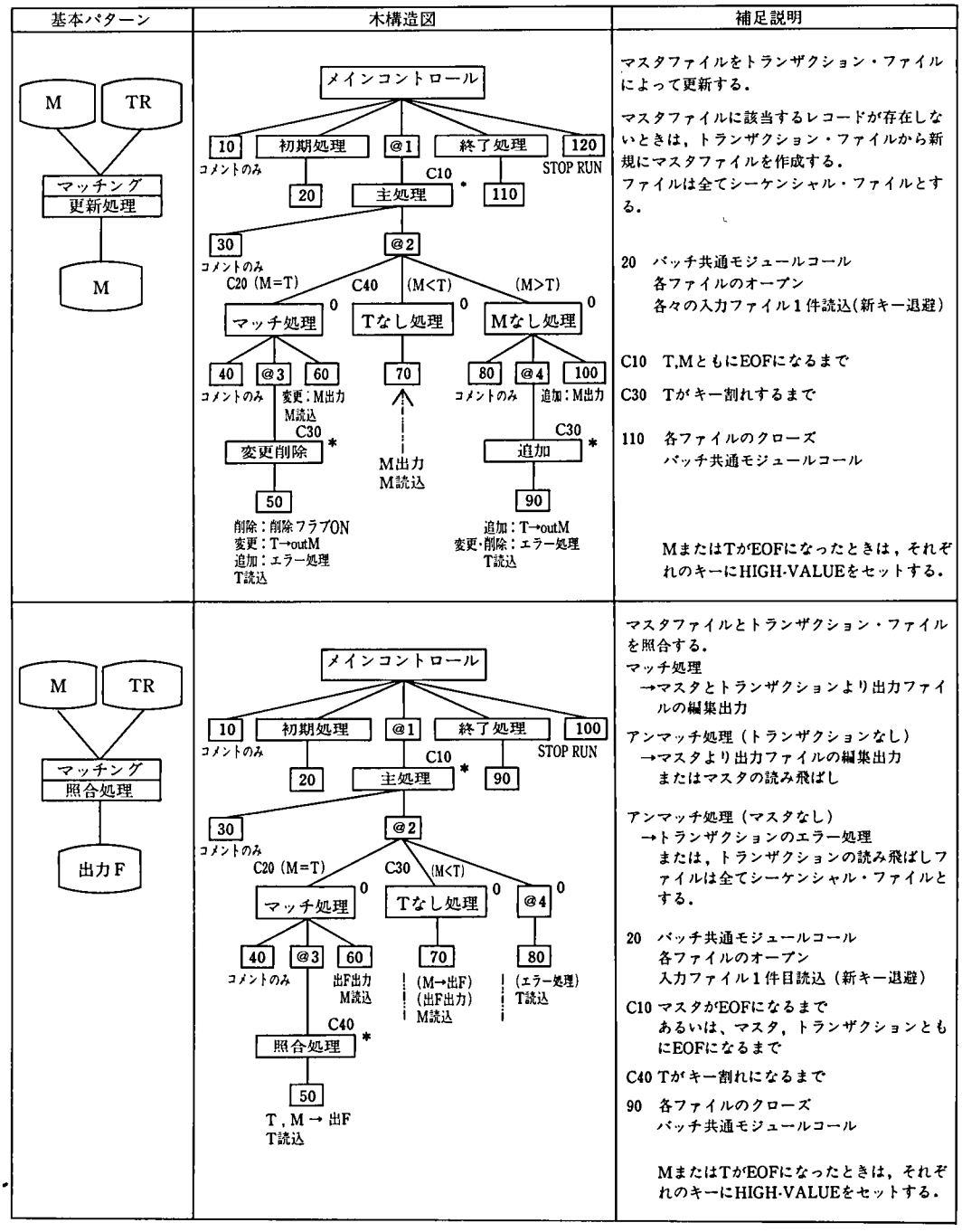


図 6 マッチングによる更新処理と照合処理の木構造図のひな型

Fig. 6 Models of tree structure of update and inquiry process by matching

・マッチ処理→マスタとトランザクションより出力ファイルの編集出力をする．

・アンマッチ処理（トランザクションなしの場合）→マスタより出力ファイルの編集出力またはマスタの読み飛ばしをする．

・アンマッチ処理（マスタなしの場合）→トランザクションのエラー処理または，トランザクションの読み飛ばしをする．

なお，ファイルはすべてシーケンシャル・ファイルとする．

6) 整列（ソート）処理のパターン……一つの入力ファイル（ランダム）から対象データを抽出し，ソート後一つのファイルに出力するパターンをひな型とする．

ただし，単純なソートの場合，ソート用ユーティリティを使用する．

7) 併合（マージ）処理のパターン……複数の入力ファイル（ランダム）から対象データを抽出し，ソート/マージ処理をして，一つのファイルに出力するパターンをひな型とする．

ただし，単純なマージの場合，ソート用ユーティリティを使用する．

6.2 マ ク ロ

TRITON では，データベースやテーブルのインタフェース部分およびステータス判定のようなパターン化されたコーディングの固まりを，IDES のマクロ機能を使用

```
                                        マクロの引数
                マクロ名
                  ↓
¥ML  ROC流動性口座変換F検索K    %P1    %P2    %P3    %P4    %P5    =
           MOVE    LOW-VALUE
             TO      XPDMSB
           MOVE    CDUR1－流動性口座変換M
             TO      D口変流M－UレコードID
           MOVE    %P1
             TO      D口変流M－システム店番
           MOVE    %P2
             TO      D口変流M－科目
           MOVE    %P3
             TO      D口変流M－口座番号
           MOVE    CXISF－DMSキーEQ
             TO      XPDMSB－タイプ
           MOVE    R－RINVM
             TO      XPDMSB－レコード識別子
           MOVE    CXISF－DMSULR転送
             TO      XPDMSB－ロック転送
           MOVE    %P4
             TO      XPDMSB－マルチエリア情報
           CALL    "XDMSRFPK" USING    XPDMSB
                                       D口変流M
        EVALUATE       XPDMSB－ステータス番号
          WHEN    ZERO
          WHEN    CXISF－NOTFOUND
             CONTINUE
          WHEN  OTHER
             CALL    "SCOC-XISE"    USING    XPDMSB
                                             D口変流M
                                             WSTAT－ステータス
             GO    TO    %P5
        END-EVALUATE
¥MEND
```

図 7　マクロ定義されたマクロの例

Fig. 7　Sample of defined macro

```
〈モジュール仕様書内でのマクロ使用例〉
  &ROC流動性口座変換F検索K ―――――――――――マクロ名の指定
                    L定期払系-被仕向システム店番
                    I 定期払訂-連動科目
                    I 定期払訂-連動口座番号 ――引数の指定
                    L定期払系-被仕向システム店番
                    CIF読込チェックE
```



```
〈上記のマクロ指定から生成されたプログラム・ソースコードの例〉
    MOVE    LOW-VALUE
      TO    XPDMSB
    MOVE    CDUR1－流動性口座変換M
      TO    D口変流M－Uレコード ID
    MOVE    L定期払系－被仕向システム店番
      TO    D口変流M－システム店番
    MOVE    I 定期払訂－連動科目
      TO    D口変流M－科目
    MOVE    I 定期払訂－連動口座番号
      TO    D口変流M－口座番号
    MOVE    CXISF－DMSキーEQ
      TO    XPDMSB－タイプ
    MOVE    R－RINVM
      TO    XPDMSB－レコード識別子
    MOVE    CXISF－DMSULR転送
      TO    XPDMSB－ロック転送
    MOVE    L定期払系－被仕向システム店番
      TO    XPDMSB－マルチエリア情報
    CALL    "XDMSRFPK"  USING  XPDMSB
                               D口変流M
  EVALUATE        XPDMSB－ステータス番号
    WHEN          ZERO
    WHEN          CXISF－NOTFOUND
      CONTINUE
    WHEN      OTHER
      CALL      "SCOC-XISE"  USING    XPDMSB
                                      D口変流M
                                      WSTAT－ステータス
      GO  TO  CIF読込チェックE
  END-EVALUATE
```

図 8　マクロの使用例と生成されたソースコードの例

Fig. 8　Sample of using macro and formed source code

してマクロ定義し部品化した．これをマクロと呼ぶ．使用者は，モジュール仕様書の中で該当するマクロ名と必要な引数を指定するだけで，後は IDES により自動的にプログラム・ソースが生成される．図 7 にマクロの例，図 8 にマクロの使用例とそれから生成されたソースコードの例を示す．

マクロの機能を使用することにより，プログラムの記述量の削減とインタフェースのコーディング・ミス防止が行える．

なお，TRITON では約 1000 個のマクロが使用されている．マクロにより自動生成されたコードのステップ数がプログラム総ステップ数に占める比率は，約 12％である．

7. おわりに

以上のように，部品化やテーブルウェアの活用によりプログラムの保守性を向上させ，階層化構造設計および漢字 COBOL の採用によりプログラムを非常に読みやすいものとした．

また，IDES の利用により，データ設計・保守時の階層化構造設計遵守ならびにプログラム作成・修正時の構造化手法遵守の歯止めがかかっており，データやプログラムの均質性による保守性の高さが維持されていくと確信する．

今後は，リポジトリの充実や，リバース・エンジニアリングの利用により，プログラムやリポジトリから保守に役立つドキュメントを出力して活用する等のプログラムの保守性向上に取り組みたい．

執筆者紹介　竹　村　匡　弘（Masahiro Takemura）

昭和 27 年生，48 年奈良高専機械工学科卒業，同年日本ユニシス入社．金融機関の SE サービス，システム開発に従事，現在，金融システム企画開発本部 TRITON 企画推進部 2 課課長として TRITON における IDES 適用を担当．

XTPAに基づくノーダウン・システム

An XTPA-based No-down System

沢田　啓

要　約　XTPA (eXtended Transaction Processing Architecture；拡張処理アーキテクチャ) では，複数ホスト間でのデータベース共用機能を提供することにより，信頼性，処理能力，可用性および拡張性に優れたシステムの構築を可能としている．TRITONではこのXTPAを採用することにより，汎用機でのノーダウン・システムを実現させた．

本稿では，まずXTPAおよびTRITONシステムについて紹介を行う．次にTRITONでのノーダウン・システムの基本的な考え方，とくに複数ホストでの疎結合分散処理であることから従来と大きく異なっているホストおよび端末の考え方について述べる．そしていかにノーダウン・システムを実現したか，営業店での実際の動きはどうなるかについて述べる．

Abstract The eXtended Transaction Processing Architecture (XTPA), which provides an inter-host database sharing facility, has made it possible to create systems of higher reliability, performance and availability as well as higher expandability. With the use of the XTPA, the TRITON system has become a no-down system based on general-purpose mainframe computers.

The first mention of this paper is about the XTPA and the TRITON system, followed by the principal ideas of a no-down system supported by the TRITON system, with a special focus placed on what the hosts and terminals should be like because they are significantly different from traditional ones due to its typical form of transactions processing; that is, loosely-coupled distributed processing on multiple hosts. And finally, the author describes how the no-down system has been built and how it is actually being operated at financial branches.

1. はじめに

銀行間の提携，自動機の普及，時間延長等によって，どこでも，いつでも，容易に取引が可能となってきている．今や利用者にとってはそれが当然のことであり，24時間365日稼働も目前に迫っているといえる．したがって，コンピュータ・システムとして連続運転，ノーダウン・システムは緊急の課題である．

とくに，バンキング・システムはその性格上，耐障害性が非常に重要視されてきた．冗送/欠送方式*，ビフォアルック/アフタルック** というリカバリ・メカニズムに始まり，システム全体の二重化へと進展し，現在はホット・スタンバイ・システムが主流となっている．一部ではフォールト・トレラント・コンピュータによるノーダウン・システムが具現化されているが，情報系を含めたトータル・システム化等の面で評価されるに至っていない．

弊社では，従来からリカバリ・システムを中心としてBOSS-11 (Banking Oriented Support System under OS1100；独自のリカバリ・メカニズム)，AIS1100 II (Advanced Information System for UNISYS Series 1100 II；IR-Integrated

* 冗送/欠送方式：メッセージのリカバリ方式
** ビフォアルック/アフタルック方式：データベースのリカバリ方式

Recovery による統合リカバリ)とシステム基盤を提供してきた．これらのノウハウを引き継ぎ集大成するとともに，XTPA/EM 等のニュー・アーキテクチャ対応を行い XIS (eXtended Information System) を開発した．TRITON ではこの XIS を全面的に採用することにより，汎用機によるノーダウン・システムを容易に実現するとともに，処理能力，可用性および拡張性についても優れたシステムを実現している．

2. XTPA 概要

複数ホスト間でファイルを共用するため，従来からマルチホスト・ファイル・シェアリング (MHFS: Multi Host File Sharing) 機能が提供されていた．しかし，ホスト間のロック制御は各ホストの OS が通信装置 (IPCC: Inter Processor Channel Coupler 等) を介して行うため，オーバヘッドが大きく，ロックの単位をレコード・レベルに局所化できなかった．このためデータベースをホスト間で共用し，トランザクション処理を複数ホストに分散することは不可能であった．

XTPA ではこの根本的な問題を解決し，大量トランザクション処理，ノーダウン・システムを疎結合された分散ホスト上で実現可能とした．

XTPA システム全体構成を図 1 に示す．

2.1 XTPA を構成するハードウェア

2.1.1 RLP (Record Lock Processor)

RLP は XTPA の中核となるハードウェアであり，ホストとの接続のためのチャネル・モジュール，レコード・ロック制御を行うロック・モジュール，ファームウェアのロード等に使用する保守モジュール，および電源装置からなる．RLP 障害が発生すると，XTPA を構成する全ホスト停止となるため，ロック・モジュールは四重化，チャネル・モジュール，電源装置は二重化され，全面障害が極めて起きにくい構造になっている．

RLP はレコード・ロック制御の他に，データの主記憶領域上のコピーが更新された時に他ホストの主記憶領域上のコピーの無効を通知する無効化通知機能，ホスト間で同期をとる必要があるシステム・テーブルが更新されたとき，ただちに他ホストに更新情報を通知する機能を有する．

RLP はデータベースの更新の時間順に一意のコミット順序番号を発番する．この番号はホストごとに存在するオーディット・トレイルに更新履歴として書き込まれ，データベースのリカバリが必要になったとき，各ホストのオーディット・レコードはコミット順序番号順にマージされる．

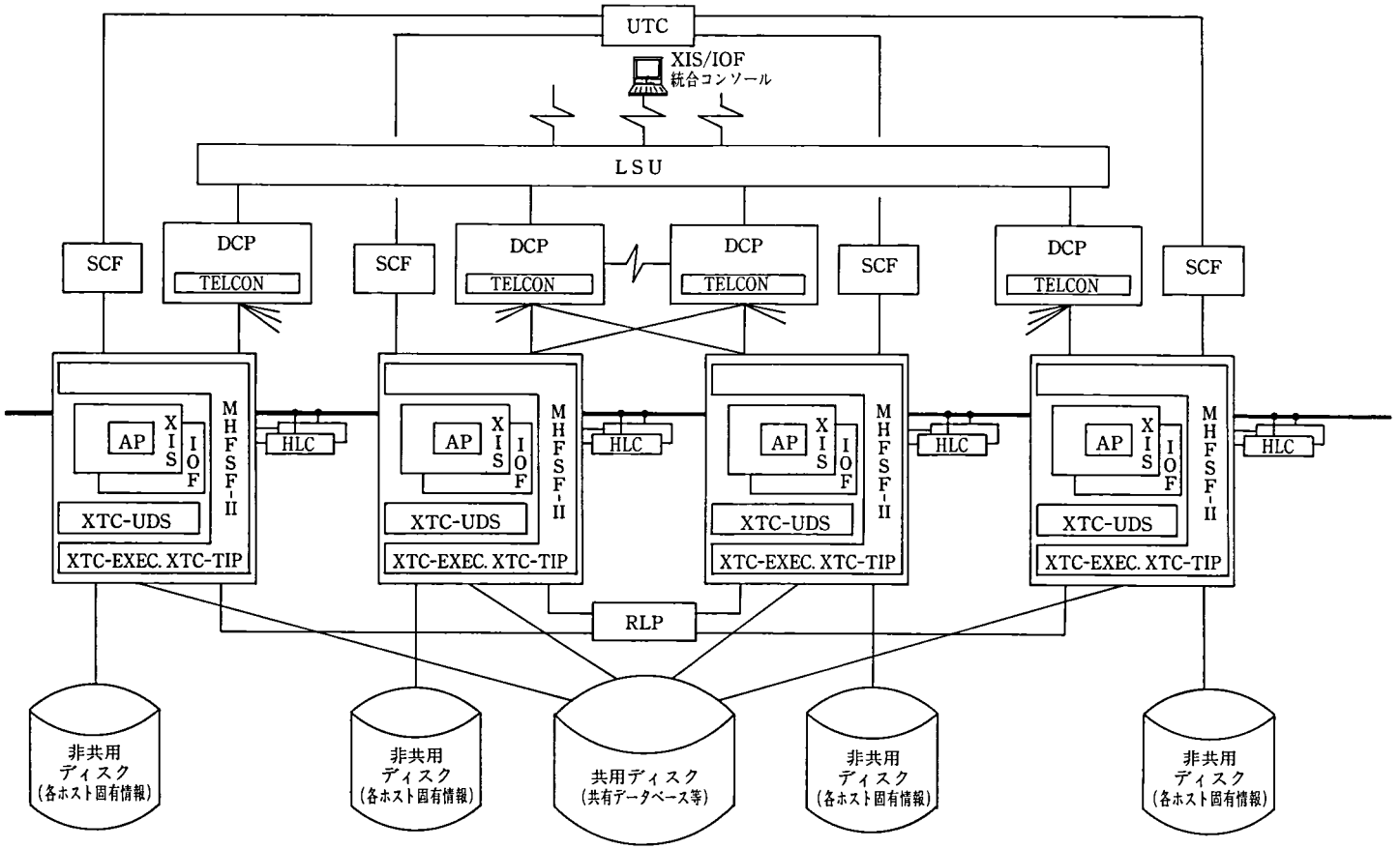
2.1.2 HLC (Host Lan Controller)

マルチ・ホスト・ファイル・シェアリング機能 (MHFS) を使用するため，各ホストが正常稼働していることを他ホストに通知 (ハートビート機能) し，また，共用ディスク上のファイル制御のためホスト間通信を行う必要がある．HLC はこのホスト間通信に使用される．

2.1.3 UTC (Universal Time Coordination)

XTPA を構成するホストに NTT より受信した時刻を供給する装置である．

共用データベースのリカバリ処理ではリカバリの開始点を時刻で指示するが，各ホ



UTC : Universal Time Coordination
DCP : Distributed Communications Processor
SCF : System Control Facility
HLC : Host Lan Controller

RLP : Record Lock Processor
MHFSF-II : Multi Host File Sharing Facility II
XTC : eXtended Transaction Capacity
〈XTC-TIP/UDS : 大規模オンライン・トランザクション処理システム〉

図 1 XTPA システム全体構成図

Fig. 1 XTPA system overview

ストの時刻に狂いが生じていると更新履歴の抜けが発生する可能性があり，データベースのリカバリを保証できない．複数ホストを単一システムであるかのようにデータベース制御を行うためには，全ホストが同一時刻であることが必須である．

2.2 XTPA を構成するソフトウェア

2.2.1 XTC-EXEC

従来から提供されているマルチホスト・ファイル・シェアリング機能に加え，2200/1100 シリーズにおけるリアルタイムシステム環境である TIP（Transaction Interface Processor）環境を複数ホスト間で共用するための以下に示すマルチホスト TIP 機能が提供される．

① 共用 TIP ファイル制御

② TIP のもとでの TPS（Transaction Processing Segment；業務処理プログラム＝TIP プログラム）のスケジュールについて，共用プログラム・ファイルからどのホストでも TIP プログラムのスケジュールを可能とする．

③ 統合リカバリの単位であるアプリケーション・グループをホスト間で共用できるように拡張

④ RLP に対するロック制御

⑤ RLP に対するホスト間通信制御

2.2.2 XTC-UDS と統合回復ユーティリティ（IRU）

従来の UDS（Universal Data System）機能に加え，以下の機能が提供されるが，使用者インタフェースには変更点はない．

① RLP を利用したデータベース・レコードのロック管理

② 主記憶領域に読み込んだページバンクの各ホストでの有効/無効管理

③ 制御テーブルが更新された場合，他ホストに更新情報を通知し，ホスト間で同期をとる機能

④ ホスト障害あるいはアプリケーション・グループ障害時に更新中であったデータベースを回復用ファイル（リテンション・ファイル，オーディット・トレイル）を用いてリカバリを行うミディアム・リカバリ機能が提供される．障害中に他ホストから実行できるため，データベースの迅速なリカバリが可能である．

また，障害発生時に更新中であったレコードは他ホストからの参照ができないよう，RLP 内にロックが保持されているが，ショート・リカバリ，ミディアム・リカバリ処理の最後で，この RLP のロックが解かれる．

2.2.3 XIS（eXtended Information System）

AIS1100 II に比べ，大幅に機能拡張されており，TRITON で XTPA の諸機能を利用するには，XIS（本特集号別稿“銀行システムと XIS”参照）の使用者インタフェースを意識すれば良いようになっている．

TRITON で使用している XIS の XTPA 対応諸機能は以下のとおりである．

① 新プログラミング環境（NPE）対応

② SIB（System Information Base）による実行環境の一括管理と立ち上げ/終了 JCL，各種パラメタ，テーブル生成

③ 障害監視，イベント制御（IOFとの連動機能を含め）およびリカバリ・アクションを記述する運用言語とアクション・ライブラリ

④ 共用データベース・アクセス機能

⑤ 外部システムかXTPAを構成するどのホストに接続されているかを意識せずにメッセージ送受を可能とする通信の透過機能

⑥ ホストごとに存在するオーディット・トレイルのログ・マージ機能

⑦ 複数ホストでのセンタ・カット・スケジュール制御

⑧ 勘定系と情報系，あるいは他系とのホスト内/ホスト間のトランザクション転送による分散トランザクション機能

⑨ 複数ホストのオンライン環境を一元的に管理する統合コンソール

2.2.4 IOF (Integrated Operating Facility)

IOFは分散化された複数ホストの統合運用管理について諸機能を提供してきたが，TRITONでは新たにXTPA対応の以下に示す機能を使用している．

① XTPAを構成するホスト間のジョブ・ネットワークの制御，運行管理

② XTPAを構成する複数ホスト・システムを単一の制御系から監視/制御するためのIOFマスタの仮想化（いずれのホストもマスタ・ホストになることが可能）

③ 統合マス・ストレージ管理

④ 共用データベースのバックアップのためのダンプJCLの生成とリール管理

⑤ ホストごとに存在するオーディット・トレイルのアーカイブ制御とアーカイブ・リール管理

⑥ バックアップされたデータベースとオーディット・トレイルおよびアーカイブ・テープを用いたデータベース・リカバリ処理

3. TRITONシステム構成

TRITONシステムは複数ホスト(2〜4台)から構成される．勘定系を始めとした各業務はそのシステム基盤，必要なCPUリソースにより負荷の平準化を図って各ホストに配置される．ここでは基本となっている4ホスト構成を前提としたTRITONシステムの構成の特徴を以下に紹介する．

① 各業務は業務の独立という観点からAPG（APplication Group)* に分ける．

② 勘定系はコンカレントAPGとし，ノーダウン・システム化を図る．

③ 情報系もコンカレントAPGとし，ノーダウン・システム化を図る．ただしDIP（Dynamic Database Interface Package）での検索（MAPPER）はノーダウンとはしない．

④ 国際勘定系（IBS: International Business System）はシステム基盤がAIS2200であり，ローカルAPGとし，ホット・スタンバイによりリカバリを行う．

---

* APG：リカバリの単位であり，1〜8までの番号を持つ．ローカルとコンカレントという二つの型がある．ローカルAPGはいずれかのホストにのみ存在し，コンカレントAPGは複数ホストにまたがって存在する．ノーダウン・システムはコンカレントAPGでなければならない．また，いずれのAPGにも属さない（便宜上APG#0と表現する）業務が可能である．このAPG#0は統合回復の対象にはならない．

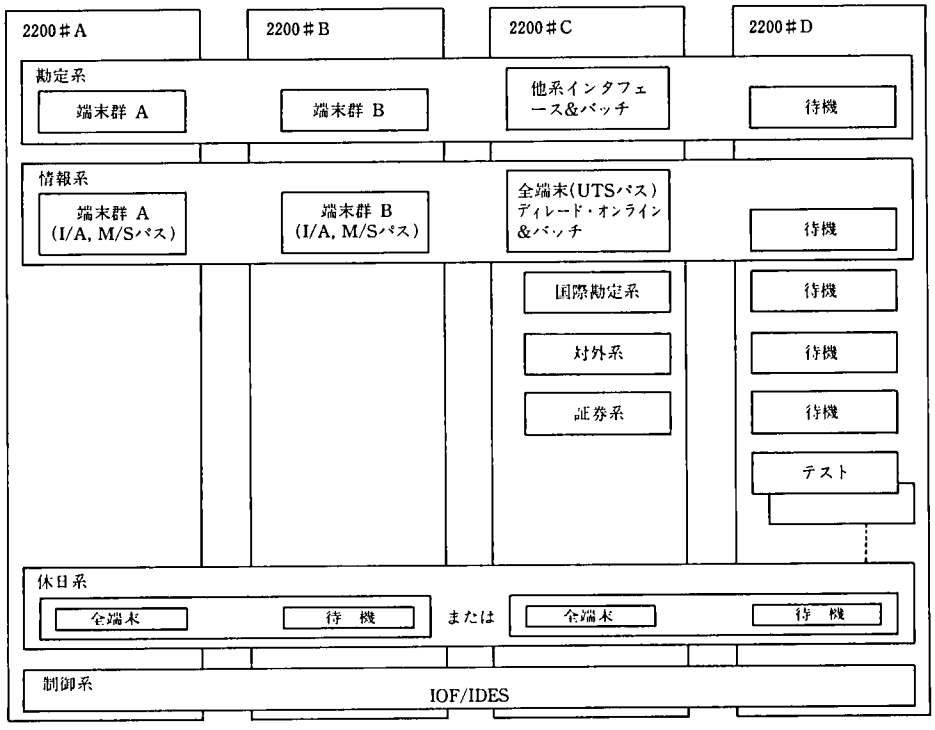


図 2　TRITON システム構成（例）
Fig. 2　TRITON system overview

⑤　対外系（UNITE: UNIsys Total Electronic banking system）はシステム基盤が BOSS-11 であり，APG#0 とし，ホット・スタンバイによりリカバリを行う．

⑥　ホストは 2 台を勘定系/情報系のオンライン機，1 台を勘定系/情報系バッチおよび他系機とし，もう 1 台を待機/テスト機としている．

図 2 に構成例を示す．

4. ノーダウン・システムの基本的な考え方

TRITON におけるノーダウン・システムとは，営業店からみてセンター側システムが停止しない（営業店で全端末が使用不可とならない）ことをいう．現実的にはホスト障害が発生した場合，そのホストで処理中であった端末については応答が返らない．しかし，他の端末については正常に取引が可能であり全端末が使用不可となることはない．応答が返らなかった端末についても再入力（通常，このような場合には最終メッセージ照会を行う）により取引が続行できる．

ホット・スタンバイ・システムとの大きな違いはホット・スタンバイ・システムでは程度の差こそあれ営業店において，一定時間全端末が使用できなくなってしまうところにある．

このノーダウン・システムの実現にあたっては（複数の）ホストの考え方，端末群

（の分割）の考え方が従来と大きく異なっている．

4.1 ホストの考え方

XTPA の特徴を生かすためには，各業務の実行は物理的なホストに固定せず，どのホストでも実行可能でなければならない．そこで，TRITON では物理ホスト，論理ホストという概念を設けている．

物理ホストとは，その名の通り物理的なホストを識別するもので，TRITON では便宜的に#A，#B，#C，#D と表記する．

一方，論理ホストとは稼働する業務によって決まるホストを識別するものであり，#1，#2，#3，#4 等と表記する．代表的な論理ホストを表 1 に示す．

物理ホストと論理ホストの関係は相対的なものであり，物理ホストの障害や運用形態の変更等により，その関係は変わる．各ホスト障害時の論理ホストの推移例を図 3 に示す．

表 1　論理ホストの割り当て

Table 1　Definition of logical host

論理ホスト	内　　容
ホスト # 1	勘定系および情報系（A 群端末）が稼働するホスト
ホスト # 2	勘定系および情報系（B 群端末）が稼働するホスト
ホスト # 3	対外系，図際系，MAPPER が稼働するホスト
ホスト # 4	待機ホスト（論理ホスト # 1～# 3 の待機）および開発用ホスト
ホスト # 7	休日系が稼働する本番ホスト
ホスト # 8	休日系の待機ホスト

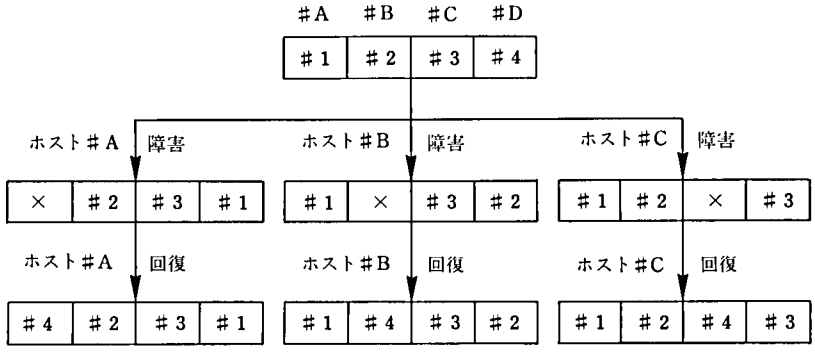


図 3　論理ホストの推移例

Fig. 3　Transition of logical host

4.2 端末群の考え方

一つの営業店の端末を A 群と B 群の 2 群に分けている．たとえば，一つの営業店に端末が 10 台あれば，ある基準に従って A 群を 5 台，B 群を 5 台というように分ける．

そして，これらの A 群端末がつながるホストを論理ホスト#1 とし，B 群端末がつながるホストを論理ホスト#2 とする．

これにより A 群端末がつながっているホストで障害が発生しても，B 群端末がつながっているホストは正常に稼働しているので業務上の全面停止とはならない．

一方，A 群端末についても後述のデュアル・セッション機能により，再入力した時

点では待機ホストにトランザクションがわたり，業務が続行できる．

なお，TRITON では表 2 のような端末群に分けている．

表 2　端末群の種類

Table 2　Definition of terminal groups

端末区分	端末群	接続ホスト	セッション区分
勘定系端末	A 群	論理ホスト＃1	デュアル
	B 群	論理ホスト＃2	デュアル
情報系端末	A 群	論理ホスト＃1	シングル
	B 群	論理ホスト＃2	シングル
対外系	−	論理ホスト＃3	シングル

4.3　営業店での動き

ある営業店においてホスト障害が発生した場合の動きについて具体例（端末仕様等により多少動きは異なる）を述べる(図 4)．

便宜上 8 台の端末が存在し，1/3/5/7 号機は A 群端末，2/4/6/8 号機は B 群端末とし，各々未使用/画面入力中/送信キー押下済み(ホスト TPS 処理中)/送信キー押下済み（前記以外）の状態でホスト＃A に障害が発生したとする．

① B 群端末の 2/4/6/8 号機は何ら影響を受けない．

② 1/3 号機も何ら影響を受けない．ただし，接続先ホストの切り換えの間に送信を行った場合，3 号機はタイムアウト（応答が返らない）になる．

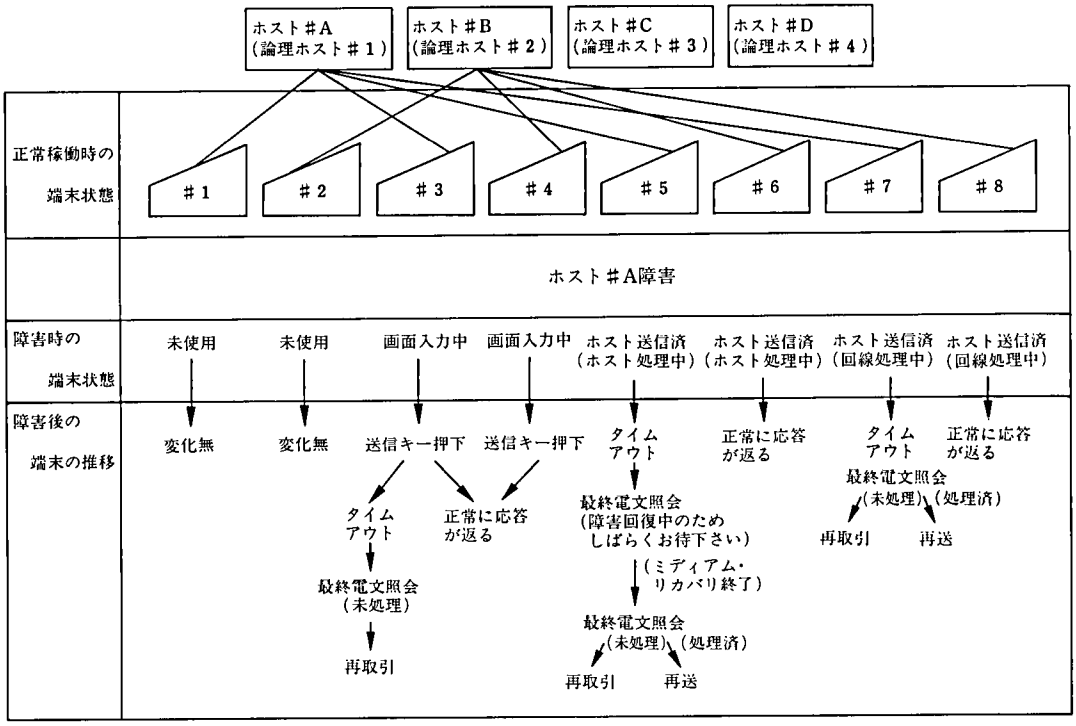


図 4　営業店でのノーダウンの動き

Fig. 4　Terminal action in no-down system

③ 5号機はタイムアウトになる．ホストでのリカバリ処理の間は当該端末は使用不可（入力に対し'障害回復中のため暫くお待ち下さい'の応答が返る）となる．

④ 7号機もタイムアウトとなるが，当該端末は使用可能状況である．最終メッセージ照会（最後の取引がセンターで成立したかどうかの確認）を行い，再入力または再送を行う．

端末での1取引の時間（画面入力開始から出力完了まで）は，一般的には回線，ホスト処理も含め1分程度である．全端末がフル稼働している状況で問題となる③の端末の確率は，ホストでの処理を平均1秒と仮定すると，

1/2（A群，B群）×1秒/60秒→ 1%

程度となる．

5. ノーダウン・システムの実現方法

XTPAをベースにノーダウン・システムを実現しているが，主な機能はデータコミュニケーションとデータベースの機能である．また，支援機能として障害発生時の監視（自動検知）および対応の自動化機能がある．

5.1 データコミュニケーションの機能

ノーダウンを実現するにあたり，次の二点の機能が必要である．

① ホスト障害時に，端末とのパスが切断することなく取引を続行する機能

② 端末ごとに接続ホストを管理し，接続ホストを柔軟に，かつ動的に切り替える機能

TRITONでは，これらの機能をデュアル・セッション機能といい，この機能を運用・管理するシステムをセッション管理と称している．

セッション管理は，広義にはCMS/TELCON（ネットワーク管理システム）とXISおよび運用プログラムを含めた機能を指し，狭義には運用プログラムの機能のみを指す．

プロダクト別の機能では，まず，CMS/TELCONがデュアル・セッション機能やアウトバウンド・オープン機能を提供する．次に，XISはデュアル・セッションを制御するための使用者インタフェースの提供とセッション情報の管理を行う．最後に，運用プログラムはネットワーク運用の管理者として，ホストの運用形態に応じたセッション設定・解放のコマンド発行やXISの管理するセッション情報の更新等を行う．

図5に各機能を階層的に示す．

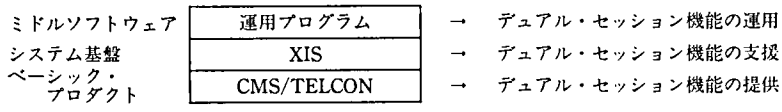


図5 データコミュニケーション機能の階層

Fig. 5 Layer of data communication

5.1.1 デュアル・セッションの考え方

デュアル・セッションとは一つのパスに二つのセッションを対応づけたものである．

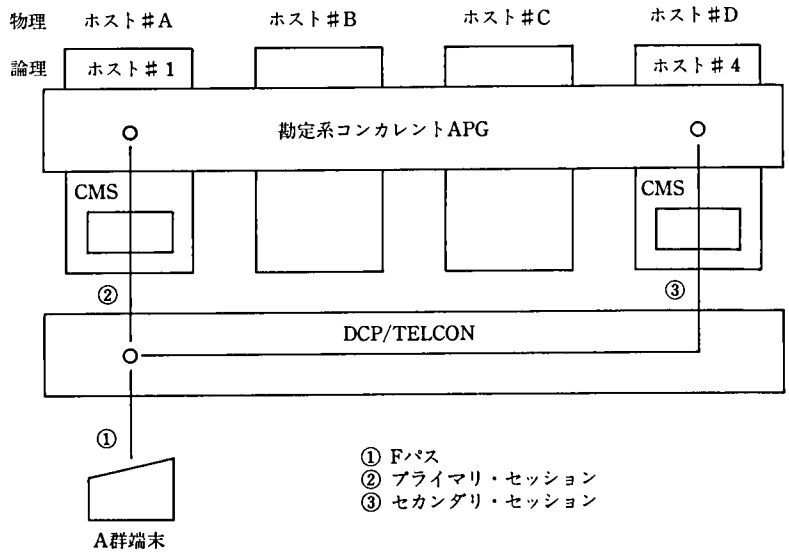


図 6　デュアル・セッションの概念

Fig. 6　Concept of dual session

図6にデュアル・セッションの概念を示す.

① Fパス

端末からTELCONまでの理論的なパスをFパス（機能パス）と呼ぶ.

② プライマリ・セッションとセカンダリ・セッション

デュアル・セッションでは一つのFパスに対して二つのセッションが対応づけられ，第1のセッションをプライマリ・セッションと呼び，第2のセッションをセカンダリ・セッションと呼ぶ.

このセッションの区分は絶対的なものであり，プライマリとセカンダリの関係が変化することはない.

一つのFパスに対して二つのセッションが対応していても，実際にデータの疎通が行われるのはどちらかのセッションである．セッションはモードを持っており，データを疎通するモードをオンライン・モードといい，そうでない方をバックアップ・モードという．このモードは相対的なもので，オンライン・モードとバックアップ・モードは排他関係にある.

5.1.2　デュアル・セッションの切り換え

TELCONは，CMSとの間のプロトコル・レベルの監視によりホスト障害を検知し，ホスト障害処理を行う．すなわち，障害の発生したホストにつながっていたセッションをクローズするとともに，そのセッションのモードがオンラインの場合には，バックアップに反転させる．さらに，もう一方のセッションのモードをバックアップからオンラインに反転させる．これによりデータ疎通のためのセッションが切り替わり，結果的には端末の接続ホストが切り替わることになる.

上記の例で，ホスト#Aで障害が発生すると，ホスト#Aにつながっていたプライマリ・セッションがクローズし，そのモードがバックアップになる.

一方，ホスト#Dにつながっているセカンダリ・セッションのモードがオンラインとなり，以降のデータ疎通はこのセカンダリ・セッションを通して行うことになる．

5.2 データベースの機能

データベースの機能はXTPAそのもので実現されており，アプリケーションにおける考慮点もなく，ここではとくに述べない．

5.3 監視とリカバリ

ホスト障害時に使用不可端末をできるだけ早く復旧させるには，監視とリカバリの自動化が必須である．リカバリの自動化については，各論理ホストごとにリカバリ手順が異なり，また同一論理ホストにとっても時間帯によりリカバリ手順が異なる等のため判断を含めた手順が必要である．この判断および手順を容易に自動化すべく'アクタ'機能を準備した．

5.3.1 ア　ク　タ

アクタは，システム運用，とくに障害リカバリに関わるアクションを運用言語で記述したものであり，アクション・ライブラリに登録することによって実行可能となる．

運用言語機能はXISの機能であり，アクタを記述するための文法と実行のための次の機能を提供している．

① 運用言語で記述された原始プログラムの翻訳
② アクション・ライブラリの構築
③ アクタの登録管理
④ アクタの実行制御
⑤ イベント制御からのアクタ起動
⑥ ラン・ストリームからのアクタ起動
⑦ 標準アクタの提供
　・ホスト障害リカバリ・アクタ（XTCHOST）
⑧ 組み込みアクタの提供
　・ホスト間で参照/更新が可能なアクション共用データの制御を行うアクタ（XACTSHRDATA）
　・ミディアム・リカバリ処理の起動可否の判定を行うアクタ（XTCAPG）

アクタで扱える変数は，そのアクタ内のみで設定/参照可能な変数と，ホスト間で参照/更新が可能な共用データとがある．共用データはアクタがホスト間で情報の受け渡しを行うことのできる唯一の手段であり，障害ケースごとにリカバリ処理を作成するのではなく，システム状況の変化（立上げ/終了，障害/復旧）に応じて共用データを設定することにより，事象に対し単一なアクタを準備すれば良いことになる．

図7にホスト障害リカバリのアクタ例を示す(XTCHOST)．

5.3.2 監視とリカバリ

ホスト障害の監視は，各ホストで実行されるXISのモニタ機能(XISの常駐ラン)により行われる．XISのモニタは共用ディスク上に配置されたタイムスタンプ・テーブルを一定間隔で更新/参照/監視する．監視は，特定ホストのモニタが他ホストを監視するのではなく，すべてのホストが他ホストの稼働を監視する相互監視方式を採用している．

```
ACTOR XTCHOST
MESSAGE  STAT：ホスト状態,      “イベント制御からパスされるステータス”
         FHST：障害物理ホスト,  “イベント制御からパスされる障害ホスト識別名”
         MHST：自物理ホスト     “イベント制御からパスされる自ホストのホスト識別名”
TEMPORARY  リカバリ APG：CHR
 [
 “標準イベント XTCHOST から起動され                    ”
 “障害ホストが自ホストのリカバリ対象ホストであれば,    ”
 “稼働 APG に対しミディアム・リカバリ（XnCMR）を起動する”
  （組込みアクタ XTCAPG を呼び出して判定）

IF STAT='DOWN' THEN
 [  XTCAPG <—FHST：障害物理ホスト  MHST：自物理ホスト  MAPG：1—> リカバリ APG
 “標準アクタ XTCAPG からリカバリラン起動の有無を判定する.”
IF リカバリ APG='YES' THEN
 [ST JCLFILE & 'X 1 CMR/' &障害物理ホスト]
  XTCAPG <—PHST：障害物理ホスト  MHST：自物理ホスト  MAPG：3—> リカバリ APG
IF リカバリ APG='YES' THEN
 [ST JCLFILE & 'X 3 CMR/' &障害物理ホスト]
 ]
```

図7　ホスト障害リカバリのアクタ（例）

Fig.7　ACTOR example for recovery in host failure

あるホストが障害となり，そのホストのタイムスタンプが更新されない状態が一定時間経過すると（16～20秒），正常稼働しているホストの XIS モニタが検知し，ホスト障害イベントを発生させる．XIS のイベント制御はアクションとしてイベント表に指定されたアクタ（ホスト障害リカバリ・アクタ）を起動する．起動されたアクタは，自ホストが障害ホストのリカバリ実行ホスト（すなわち，障害ホストのミディアム・リカバリ実行ホスト）かどうかを判断し，リカバリ実行ホストであれば，障害ホストで稼働していたコンカレント・アプリケーション・グループのミディアム・リカバリ処理を起動する．リカバリ実行ホスト以外で起動されたアクタは何も実行せずに終了する．リカバリ実行ホストの定義はあらかじめ XIS のユティリティを使用してどのホストが障害になったら，どのホストでリカバリするかを登録しておく（休日運転，縮退運転を考慮する）．

ミディアム・リカバリを実行する JCL は XIS が生成するが，この中に使用者のリカバリ処理を組み込むことが可能になっている．

しかし，ホスト障害のリカバリ処理としては，これだけでは不完全である．

TRITON では，コンカレント・アプリケーション・グループ下で稼働する勘定系，情報系の他に対外系，証券系，国際系のアプリケーションが稼働しており，これらはローカル・アプリケーション・グループ下で稼働しているもの，あるいは統合リカバリの範囲外のものである．

これらが稼働しているホストが障害になった場合は，従来のホット・スタンバイ・リカバリと同等なリカバリが必要であるため，ホスト障害リカバリのアクタから使用者のアクタを起動する仕組みが提供されており，使用者はこのアクタにホット・スタンバイ・リカバリ処理を組み込む．図8に監視とリカバリの流れを示す．

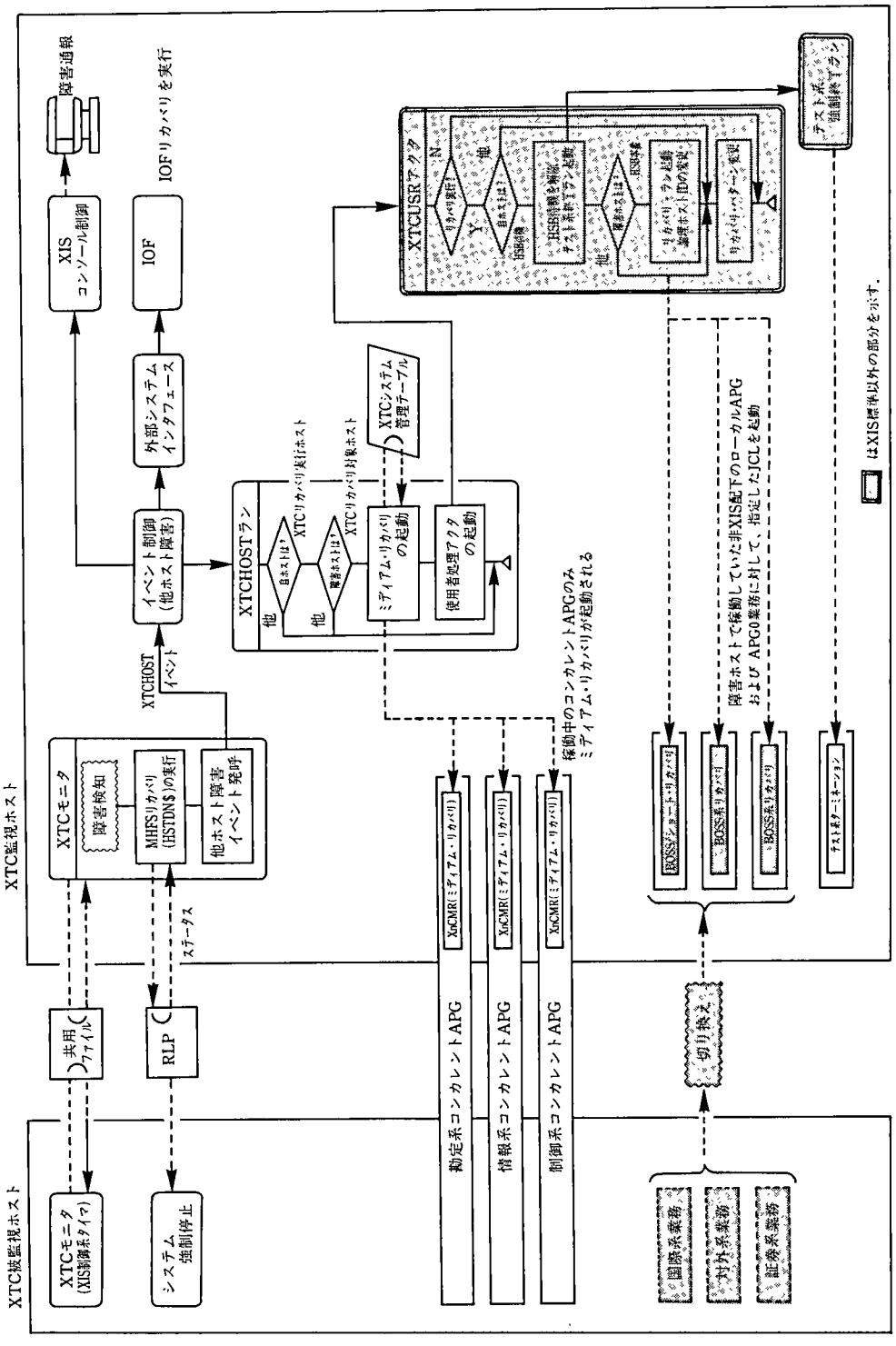


図 8 障害監視および障害リカバリの流れ

Fig. 8 System monitoring and flow of recovery in system failure

6. XTPA 構成における TRITON での考慮点

TRITON では，XTPA を採用したことにより，ノーダウン・システムとして高い信頼性を確保できた．他にも次に述べるように大きな効果をあげている．

① 処理能力：元加処理* 等，特殊処理については全ホストを活用し，高い処理能力を実現している．

② 可用性　：サンデー・バンキング時，2 ホストで運用し残り 2 ホストについてはハードウェアの保守を行う等，柔軟な運用を実現している．

③ 拡張性　：XTPA は 4 ホスト，各ホストは 8 CPU までの計 32 CPU まで拡張可能である．また，CPU の追加が必要となった場合，# A と # D のみ追加するといった小刻みな拡張が可能である．

しかし，XTPA により複数ホスト構成になったため，一部アプリケーションでの考慮が必要になった．ここではその点について述べる．

6.1 アプリケーション処理での考慮点

XTPA にからんでのアプリケーション処理での考慮点は大きく二つある．一つは為替処理のメッセージ・スイッチングのように，入力された端末ではなく別の端末（自ホストに接続されていない場合がある）に出力する場合の考慮である．

もう一つは，前述のホスト障害時，ホスト TPS が処理中であったため，当該端末を使用不可にするための考慮である．

6.1.1 メッセージ・スイッチング処理

為替処理等においては入力と異なる端末に出力を行うため，入力のあったホストとは別のホストに接続されている端末に出力する場合がある．XIS の機能（基本トランザクション転送）で，コンカレント APG 下では，出力する端末がどのホストに接続されているか意識する必要はない．しかし，その場合にはホスト間の転送が発生する．為替処理においては大量の出力を行うため，効率上好ましくない．

したがって，為替処理については出力部分を独立させ（処理では出力イメージをディスクにライトするまで），A 群ホスト，B 群ホスト個別に常駐ランを設け，各々のホストに接続されている端末を意識して出力を行っている．

6.1.2 ホスト障害時処理

ホスト障害時，ホスト内で TPS 処理中で，データベース・レコードのリードロックを行っていた場合，ミディアム・リカバリが終了するまで，そのロックおよびホスト障害状況は RLP で管理されている．

通常 TPS では端末管理テーブルを最初にリードロックしており，そのリクエストに対し，RLP ではデッドロック・ステータス（通常のデッドロックとは異なったステータス）を返す．TPS では，この端末は使用不可状態と判断し，端末に対し'障害回復中のため，しばらくお待ちください'とのメッセージを出力する．図 9 に仕掛けを示す．

6.2 システムの立ち上げ/終了

4 ホストということで，システムの立ち上げ/終了を自動化するにあたり，二つの問

---

* 元加処理：年 2 回利息を盛る処理であり，全口座が対象になるため，非常に高速かつ大量処理が必要とされる．

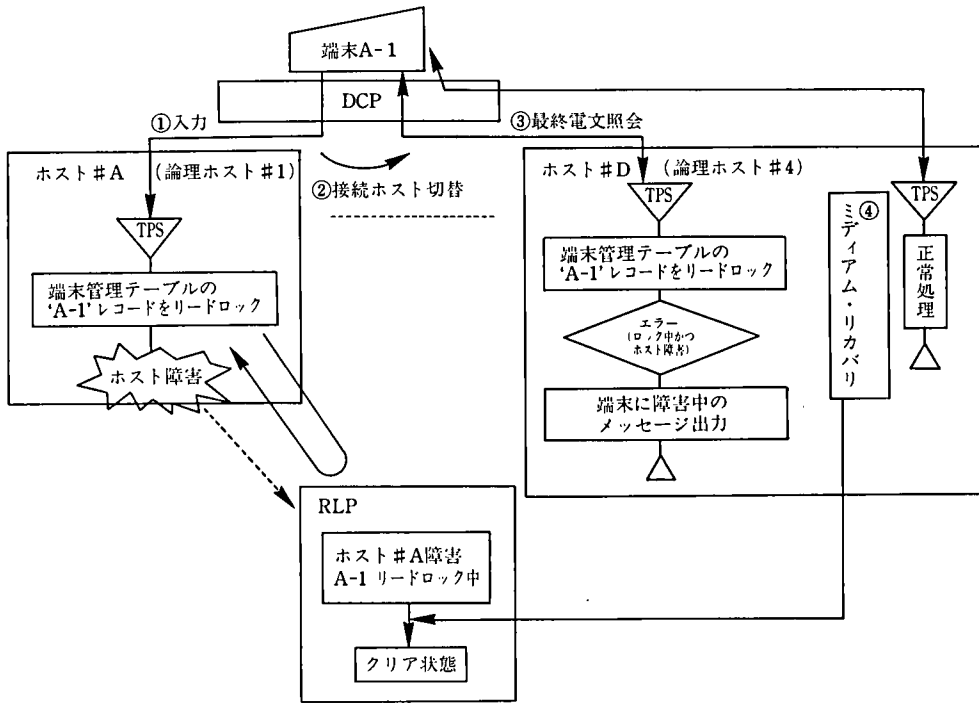


図 9　ホスト障害時の処理

Fig. 9　Process of host failure

題があった．一つはホスト間で順序を保つ必要がある場合等どうするかであり，もう一つは立ち上げ時ホスト障害が発生した場合，どのようにホスト構成を組み替えるかである．

6.2.1　立ち上げの自動化

システムの立ち上げは，XTPA を構成するシステムの電源投入からアプリケーションの一部開始までを自動化（無人化）する必要がある．システムの電源投入による立上げは，SFCP (System Facility Control Processor) のカレンダ・タイマを設定することによって従来同様行うことができる．電源投入に続くブート処理については SCF (System Control Facility) の自動応答機能により自動化を図る．しかし，MHFS ではターミネーション時のファイル・シェアの状況を把握しており，立ち上がってくるホストの順番により出力されるコンソール・メッセージが異なる．このためブートするホストの順番を固定することによって対応している．

ブート以降のシステムおよびアプリケーションの開始処理は，XIS が生成する開始処理 JCL を使用することにより行っている．XIS の開始処理 JCL は，ホスト/アプリケーション・グループ共通の初期化処理，ホストごとのアプリケーション・グループの初期化処理，特定アプリケーション・グループのホスト共通立ち上げ処理，特定アプリケーション・グループのホスト固有立ち上げ処理というようにモジュールを分けて提供されている．

この各々に，処理形態に応じて使用者タスクを組み込むことが可能であり，使用者

は必要であれば実行ホストの論理ホスト識別名を判定し，セットアップを行うことができる．このことにより，ホストごと，あるいは論理ホストごとにセットアップJCLを準備する必要はない．また，ホスト間でタスクの実行を同期をとって行うために，従来のUOSS(Unattended Operation Support Software；無人運転支援プログラム)の機能を拡張し，共用ディスクに配置されたテーブルを介し，使用者がホスト間の同期制御を行う機能を提供している．

6.2.2 立ち上げ時のホスト構成変更

TRITONシステム開始プログラムは，ホスト間の同期とりを行い，立ち上がったホストに論理ホスト識別名を割り当てる．物理ホストと論理ホストの関連は，あらかじめ運用形態を定義するテーブルに登録する．このテーブルには平日，休日およびホスト縮退時の立ち上げを考慮し，すべてのケースを登録する．

当日どの構成で立ち上げるかは前日の終了時に予約できる．立ち上げ時に予定したホストが立ち上がらない時は，縮退立ち上げ時の構成を選択し，論理ホスト識別名を割り当てる．運用を複雑化させないため，なるべく前日の物理ホストと論理ホストの関係を一致させるよう割り当てる．

このことにより，障害ホストが存在した時であっても無人での立ち上げを実現している．表3に稼働ホスト数に応じた論理ホストの順番例を示す．

表3　運用形態管理テーブル
Table 3　Administration type table

運用形態	稼働ホスト数	割当てる論理ホスト識別名
平　日	4	1，2，3，4
〃	3	1，2，3
〃	2	1，2
〃	1	1

7. おわりに

以上TRITONシステムをXTPAの観点(とくにノーダウンを中心として)述べた．

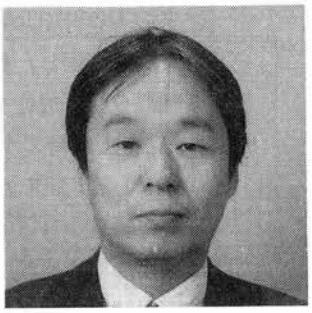
TRITONは平成5年5月6日に百五銀行，紀陽銀行において無事カットオーバーを迎え，以降大過なく稼働している．とくにXTPA（ノーダウン）については全店テスト時の結果（ホスト障害を意識的に発生させたにもかかわらず営業店では気がつかなかった）等から非常に高い評価を受けている．しかし，XTPAは緒についたばかりであり，とくに運用面で立ち上げ/終了，ミディアム・リカバリ等システム運用の簡略化，および時間短縮という課題を残している．これらについては，今後提供されるXPC (eXtended Processing Complex；拡張データ処理装置）により，4ホスト同時ブート可による立ち上げ手順の簡略および時間の短縮/ホスト障害時不要ロックの即解除によるミディアム・リカバリ時間の短縮などが解決されている．さらに，XPCの新たな機能であるVSM*，ASM**によりTPS処理時間の大幅な短縮，ディスク・コント

* VSM (Virtual Storage Manager)：入出力処理のパフォーマンスを高める機能で，グローバル・キャッシュ機能，レジデント（常駐化）ファイル機能，およびプリフェッチ（先読み機能）を提供

** AM (Audit Manager)：オーディット処理の時間を著しく短縮する機能

ロール・ユニットの削減，オーディト処理の高速化等が実現される．

執筆者紹介　沢　田　　啓（Kei Sawada）
1972年東北大学工学部通信工学科卒業．同年日本ユニシス(株)入社．BOSS-11等バンキング・システムのシステム基盤開発および地方銀行のSEサービスに従事．現在金融システム企画開発本部TRITONシステム部部長．



開発工程とIDES

The Process of Developing a System and IDES

奥野信幸

要約 TRITONをささえる基盤ソフトウェアの一つとして「IDES—Integrated Development Environment support System」がある．IDESは，リポジトリを核とし，開発工程を一貫して支援するCASEツールであり，ドキュメントとプログラムの整合性維持，ドキュメント作成・プログラム作成・テストの省力化，管理の機械化を実現した．

TRITONは，大阪，東京，津，和歌山を拠点とする4か所分散の大規模プロジェクトでありながら，個々の開発工程でIDESを採用・適用したことにより，均質化したソフトウェアの開発が実現でき，開発手順の標準化が図られ，単一開発環境と同程度の開発生産性を保持することができた．

今後保守・導入が主要作業となるTRITONにとっては，IDESは必要不可欠なツールであり，さらなる発展・強化が期待される．

Abstract IDES—Integrated Development Environment support System—is one of the basic software items that have turned out very contributive to the successful development of the TRITON system. The IDES, designed to handle repositories as a CASE tool to support all of the system's development process, has helped a great deal to maintain compatibility between documents and programs, to save the manpower required for document-making, programming and testing, and to automate systems management.

The TRITON project has been of a large scale, whose development work has been done at four different locations-Osaka, Tokyo, Tsu and Wakayama. The adoption and application of the IDES at each level of the whole development process has made it possible to attain high homogeneity in quality throughout the software, to standardize the development procedures and to retain the same degree of productivity as in a single-site development environment.

For the TRITON, of which the maintenance and installation will mostly be taken care of from now on, the IDES tool is really indispensable and expected to grow further through continued enhancement efforts.

1. はじめに

長年のシステム開発で膨れ上がったプログラム資産は，都市銀行や大手地方銀行では，1000万ステップを越える規模となった．さらに近年のシステムの巨大化・複雑化の傾向はシステムの保守の難しさに拍車をかけている．バックログの増大，膨張したソフトウェアの信頼性の低下，慢性的なSE不足といった問題が発生し，加えて経済成長の低下で，コスト削減の波は，情報処理部門への重点投資を開発保守の効率化に向かわせている．その中で，生産性の向上・品質の向上を目的とするさまざまな方法論が出現し，特定の方法論に基づいて，各開発工程で用いられる技法をツールにより支援するCASE（Computer Aided Software Engineering）がクローズアップされてきている．

このような背景のなか，当社開発のCASEツールの一つとして，IDES(Integrated Development Environment support System) がある．IDESは，リポジトリを核としてシステムの開発から保守までのライフサイクル（SDLC；System Development Life Cycleの略)支援を目的としたツールであり，TRITONは，このIDESをベースに開発環境を構築した．TRITONがIDESを適用した狙いは，次の5点である．

① ドキュメントとソースの整合性の維持による品質向上
② 成果物の均質化・標準化の保証
③ 大阪・東京・津・和歌山の4拠点を基点とする大規模分散開発を可能とするマネジメント管理および高生産性の維持・確保
④ ドキュメントの電子ファイル化，機械出力による導入コスト・保守コストの削減
⑤ 垂直分散とマイクロ系ハードウェアの活用による開発環境投資の削減

本稿では，TRITONの開発工程の中でIDESをどのように適用・運用していったかについて具体的な事例を基に整理し，反省を踏まえて今後IDESが抱える課題・展望を含めて記述する．

2. 開発工程とIDESの適用

2.1 開発工程とツールの体系

IDESのツールは，ツールの性格上，個々の開発工程を対象に支援するツールと全工程を通して支援するツールに分けることができる（図1)．

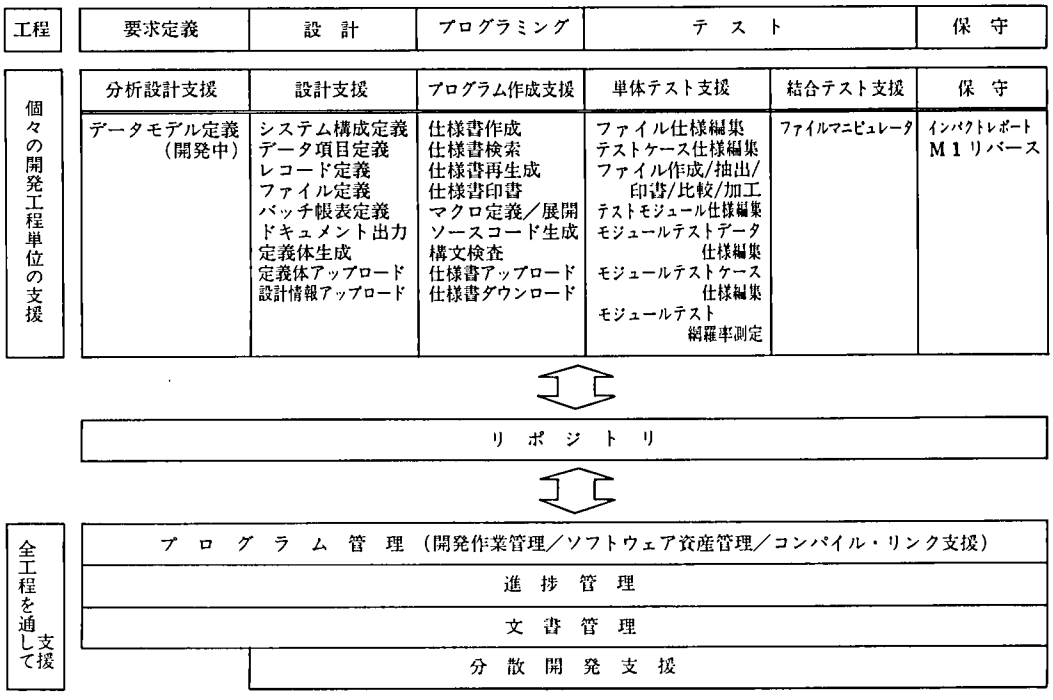


図1 開発工程とIDESの体系

Fig. 1 System development life cycle and IDES

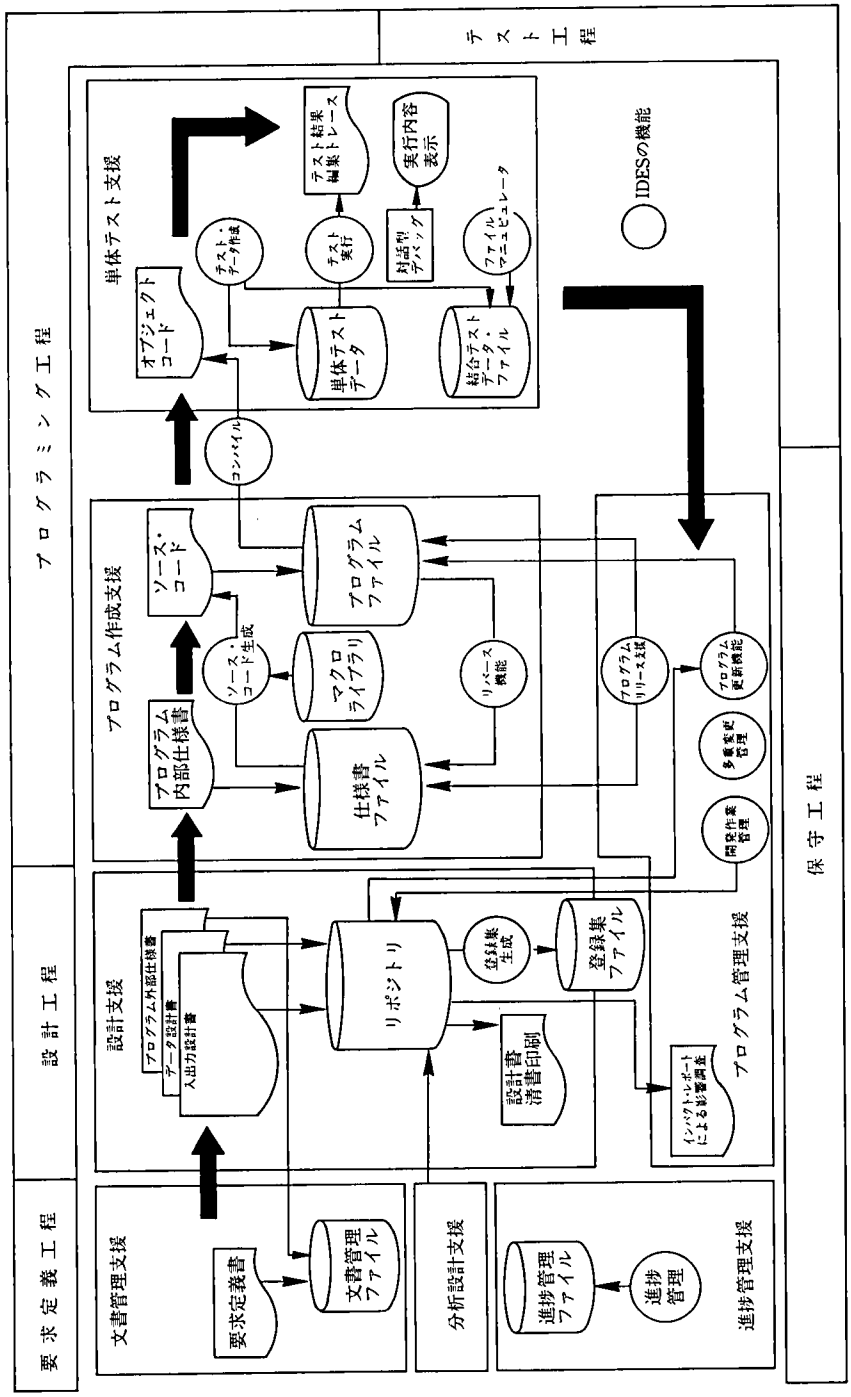


図 2　IDES による TRITON の開発の流れ

Fig. 2　Flow of system development for TRITON by IDES

分析設計支援ツールはTRITON開発時点では，計画中であったため使用していない．

個々のツールについては，2.3節以降で開発工程別に記述する．開発作業の流れとIDESとの関係を入出力の情報とともに表したものが図2である．図からも解るように，リポジトリ（データ辞書）を介して成果物を引き継いで行うフォワード・エンジニアリングとソースから仕様書やドキュメント（M1リバース）へのリバース・エンジニアリングの考え方がツールとして取り込まれている．

2.2 IDESの特徴

IDESの特徴は，以下の通りである．

1) リポジトリ（データ辞書）を中心とした開発・保守環境……開発工程で発生した設計・製造データ（データ項目，ファイル，プログラム等）をその都度一括してリポジトリに登録し，体系的に保存することで一貫性と完全さを保証し，必要に応じてデータの収集・参照・加工を可能とした．

2) 分散開発環境の実現……ハブ，サーバ，ワークステーションの3階層からなる垂直分散開発を実現し，設計・製造工程をサーバ（U 6000シリーズ；以下U 6000），ワークステーション（J 3100）に分担させることにより，ハブ（シリーズ2200；以下U 2200）の負荷を軽減した．

3) 開発技法とツールの融合……リポジトリによるデータ定義の一元化，木構造によるプログラム設計，疑似モジュールによるモジュール単体テストの簡易化等の開発技法を取り入れ，成果物の標準化・均質化と保守性の高いシステムを実現した[1]．

4) 整合性の維持……電子ファイル化された仕様書からソースコードを自動生成するため，従来の設計文書とプログラムの乖離といったことがなくなり，整合性の維持を可能とした．

2.3 各工程でのIDES適用範囲

2.3.1 要求定義～設計工程

要求定義～設計工程を支援するツールとしては，「設計支援ツール」と「文書管理ツール」がある．

TRITONの要求定義から基本設計工程では，要求定義書，基本設計書といったワープロ文書の成果物管理として文書管理ツールを使用した．文書管理ツールについては，2.3.4項で記述する．

データ設計，ファイル設計を支援するものとしては，「設計支援ツール」がある．設計担当者が記入した項目定義書のワークシートを入力情報として，画面より各種項目（データ項目名，レコード名，コード名等）と属性（型，桁数，日本語名等）をU 6000のリポジトリ（Midframe Data Dictionary；以後MDDと呼ぶ）に登録するものである．

それぞれの項目は，関連づけて登録されており，この情報から登録集を生成する．また，項目を変更することで関連する項目も自動的に変更し，登録集も自動生成する．たとえば，データ項目の桁数を変更した場合，それを使っているすべてのレコードを検索し自動的に変更し，登録集を再作成する．

一連の機能は，主に U 6000 と J 3100 とで処理される．また，登録した情報や登録集は，定期的に U 2200 (ハブ) にアップロードする機能があり，とくにリポジトリについては，ホストのリポジトリ (Host Data Dictionary；以後 HDD と呼ぶ) を MDD と同期をとって更新している．MDD，HDD の情報から項目一覧表，項目定義書，レコードレイアウトやインパクトレポート等の各種ドキュメントを出力する (図 3)．

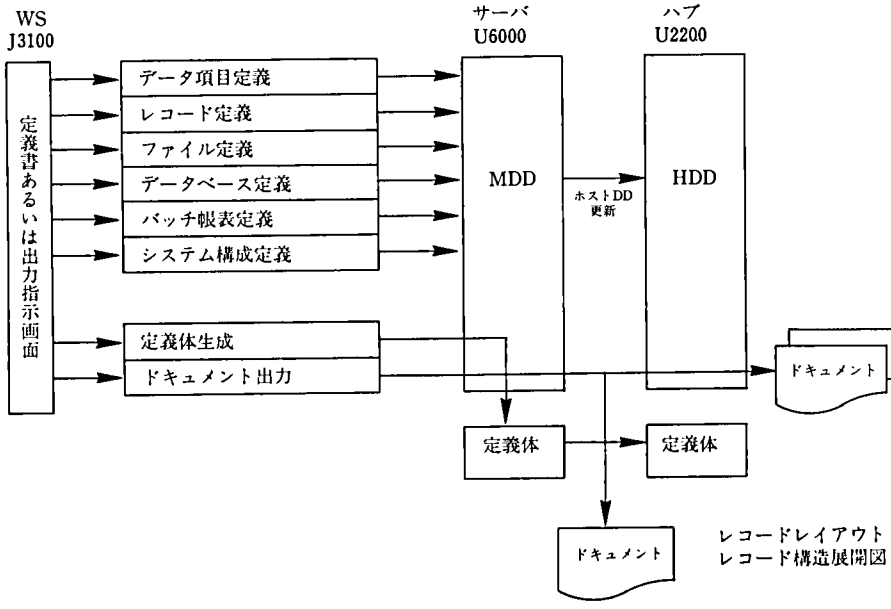


図 3　設計支援ツール

Fig. 3　Design support tool

設計支援ツールを使用することにより，各項目名は，リポジトリで一元管理されるため，システムで唯一のものとして定義しなければならないが，TRITON の場合，基本項目だけでも 10 万項目以上あるため，次のような作業を実施した．

① 「データ項目名統一基準」を定め，(株)百五銀行殿 (以下　百五銀行と略記)，(株)紀陽銀行殿 (以下　紀陽銀行と略記) の両銀行で TRITON 以前のシステムで使用した登録集をもとにデータ項目名を洗い出し，頻繁に使用していた項目名について同じ目的で使用しているものを一つの項目名に統一した．

② 同義であるが，型・桁数等が違うデータ項目やレコードのネーミングについては，本来の意味を失わないように修飾語あるいは補助語を利用して別名で登録した．

③ どうしても同一の項目名を使用したい場合には，接頭語 (漢字 5 文字+"－" ハイフン) つきで登録を許した (ただしレコード項目のみ)．TRITON では，定義体生成で登録集を生成するとき，01 レベルの物理レコード名(漢字 5 文字) を各項目の接頭語として無条件に付加している．これは，ソースコード上の各レコード，データ項目がどの 01 レベルに属しているかを一目でわかるようにするためである．そのため，本来，MDD，HDD 上は接頭語なしで項目名は登録

しているが，上記のような接頭語つきで項目名を登録する場合は，登録集生成時にその接頭語を 01 レベルのレコードに置き換えるようにツールが対応している．

また，基本設計の作業を進めやすくするため，HDD プリミティブ(COBOL プログラムインタフェース）を使用してプリプリント帳表作成ツールを独自ツールとして開発した．プリプリント帳表とは，HDD の項目情報からレコード構成項目だけを左側に印書し設計担当者が右側に各種コメントを記述できるようにした簡易帳表で，各業務処理でのトランザクションやデータベースの項目チェック等に使用した．用途別にフォームオーバレイを変更して 5 種類の帳表を用意した．

以上の機能を活用することで，正当性を機械的にチェックでき，品質の高いレコード・データ項目を作成することができた．また，ドキュメントの機械出力や登録集の自動生成によって作業負荷の軽減が図られた．

2.3.2 プログラミング工程

1) プログラム作成支援ツール……「プログラム作成支援ツール」は，詳細設計/プログラミング工程を支援し，画面入力からモジュール内部仕様書を作成し，その内部仕様書からソースコードを自動作成する．

また，仕様書作成の専用エディターの中には木構造図エディターが用意され，モジュール構造を木構造図で表現し，セクション単位に処理詳細を定義できる(図 4)．

仕様書の雛型（スケルトン）とソースコード部品（マクロ）機能を有していて，

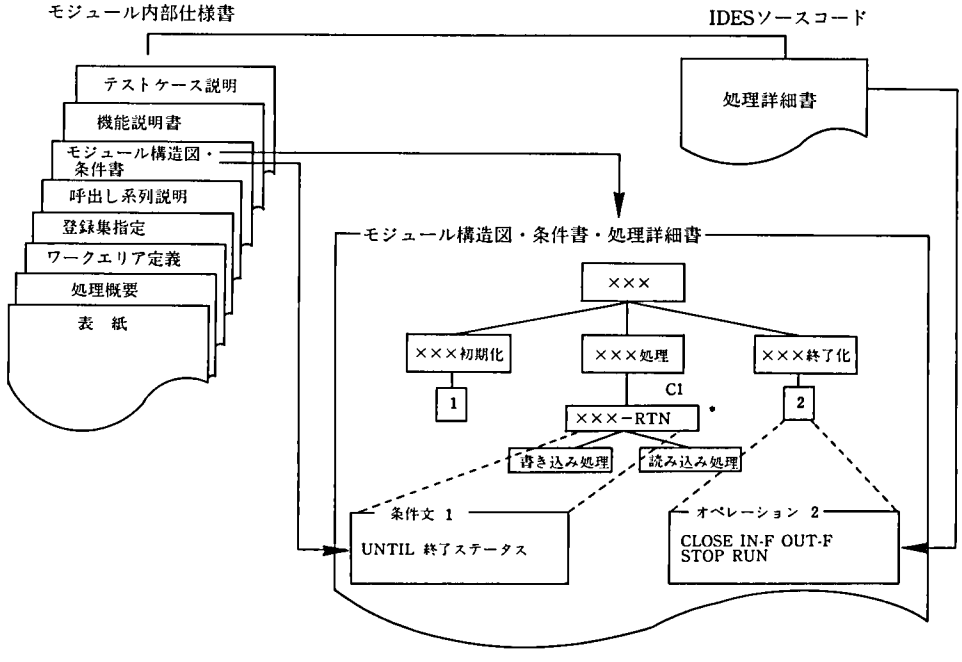


図 4 プログラム作成支援ツールの出力

Fig. 4 Output of programming support tool

標準化された高品質なソースコードを作成する．ソースコード部品は，多くのプログラムで共通に使用する処理をマクロとして登録し，仕様書の処理詳細で引用すれば，ソースコードの自動生成時に COBOL のソースコードに展開する機能である．

これらの機能は，J 3100 と U 6000 側の機能で，作成した内部仕様書やマクロは U 6000 内で保有している．また，その他完成した仕様書，ソースコードを U 2200 へアップロードする機能や U 2200 から U 6000 へダウンロードする機能，ソースコードよりリバース機能で仕様書を再生成する機能もある(図 5)．

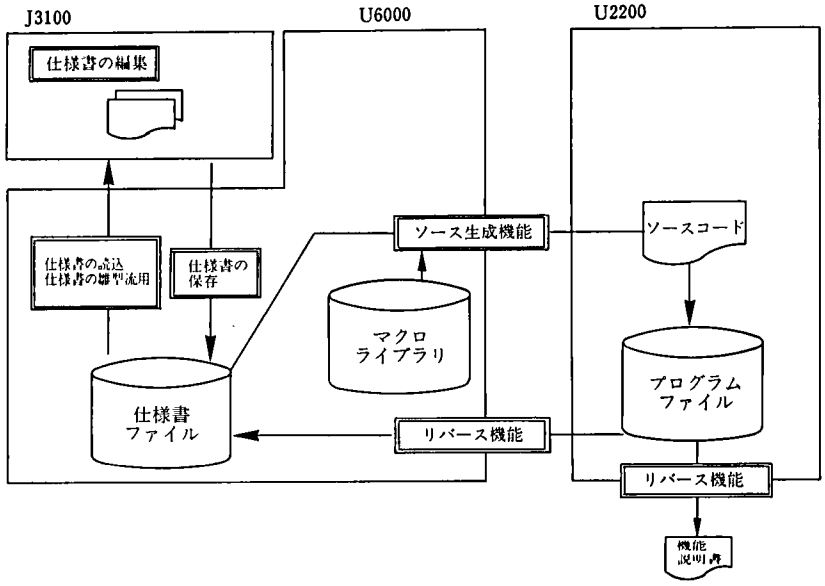


図 5　プログラム作成支援ツールの機能

Fig. 5　Functions of programming support tool

TRITON では，プログラム作成支援ツールを使用するにあたり，次のような規約を詳細設計に入る前に事前に取り決めた．

① ツールをベースとして「TRITON モジュール記述要領」，「TRITON コーディング規約」を作成し，モジュール仕様書作業の流れや木構造内の細かな規約を定め，誰もが理解・作成できるようにした．

② 「TRITON モジュール記述要領」の中で，詳細設計者とプログラマが作成する部分を分離し，各々の役割を明確にした．

(例)　リアルモジュールの場合，詳細設計者は表紙，木構造，機能説明，呼出し系列説明，テストケース指示書を作成する．それを受けてプログラマは，条件書，処理詳細書，ワークエリア定義，登録集指定を作成し，一つの仕様書をつくり上げる．

③ バッチプログラムについては，ツールで作成する木構造の基本パターン（分配，集計，更新，照合，整列，併合）を定めて雛型を用意し，それを流用することによって作成負荷の軽減を図った．

また，次工程の単体テスト工程の負荷軽減策として，単体テスト用のマクロを作成した．単体テスト用マクロとは，XIS(eXtended Information System)*のテーブルリードやファイルリード等で引き数のレコード名は異なるがサブルーチンの入り口名が同じものについて，引き数の種類分入り口名を変えたマクロのことである．これは，入り口名が同じサブルーチンのスタブモジュール(疑似のサブモジュール)については，引き数が1種類しか指定できないという単体テスト支援ツールの制限事項の対応策である．

以上のことより，仕様書とソースプログラムとの整合性が保証され，マクロ命令によるコード化作業の省力化が図られた．

2) 設計支援ツール（バッチ帳表定義）……バッチプログラムは，大部分が帳表作成のプログラムで，レポートライタ機能を使用してプログラミングされている．レポートライタのRD句（レポート定義句）作成を支援するツールとして設計支援ツールのバッチ帳表定義がある．基本設計工程で作成したバッチプログラム処理説明書(帳表出力項目定義書等)のワープロ成果物と帳表設計書（スペーシングチャート）を基に画面に帳表イメージを入力する．情報は，MDDに登録されて，この情報よりRD句の登録集を自動生成する．このツールは，視覚的に帳表をとらえながら入力できる利点があり，報告書機能をあまり知らない人にも容易に修得できるツールである．

2.3.3 テスト工程

テスト工程は，単体テスト工程と結合テスト工程に大きく分かれる．IDESは，単体テストを支援するツールとして「単体テスト支援ツール」を用意している．結合テストについては実行環境の基盤ソフトウェア（XIS等）が提供しているテスト支援ツールが中心となり，IDESは，ファイルマニピュレータを提供している．

1) 単体テスト支援ツール……単体テスト支援ツールは，U 2200でのみ機能するツールで，モジュールテスト系とファイル系の二つの機能がある．モジュールテスト系は疑似モジュールを生成することで，モジュール単体でのテストを支援し，ファイル系はバッチプログラムで入出力に使われるファイルの作成，印書，比較，加工といったファイル操作を行う．

① モジュールテスト系

オンラインの取引処理のように，複数のモジュールから構成されているものをテストするには，その上位のモジュール，下位のモジュールができていなければテストができない．しかし，未作成モジュールがあっても，テスト環境(上位モジュールからの入力条件環境，下位モジュールからの返戻環境指定）を外部的に設定することにより，疑似の上位モジュール（これをドライバモジュールと呼ぶ）と下位モジュール（これをスタブモジュールと呼ぶ）を自動作成すれば，目的あるいは被テストモジュールのテストが可能となり，モジュール作成（テスト）順序に関する制約を排除することができる．また，外部的に環境設定が可能なため，広範なテストケースを指定することができる（図6)．

---

* XISは，オンライン/機能とデータベース機能を有機的に結合し，開発から運用までのシステム・ライフサイクル全般を支援する新世代の統合オンライン支援システムである．本特集号別稿“銀行システムとXIS”参照．

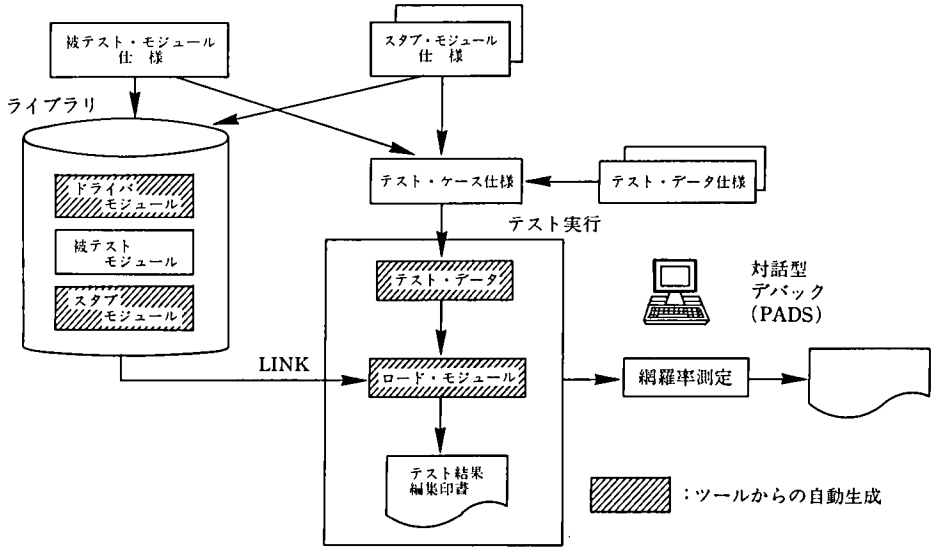


図 6　単体テスト支援ツール　モジュールテスト系

Fig. 6　Unit test support tool

TRITON では，以下の点を考慮して，ツールを使用した．

- ツールをベースにして，「TRITON 単体テスト実施要領」を作成し，単体テストの進め方や，規約や操作例を示して，テスト実行者の作業を明確にした．
- 共通モジュールや XIS プリミティブ等の下位モジュールで，複数のテストモジュールから CALL されるものについて，標準スタブモジュールを用意し，モジュール仕様やテストデータ仕様作成の負荷軽減をはかった．
- 通常のスタブモジュールは被テストモジュールに制御を返すが，ロジカル・エラー処理等の CALL したサブルーチンからリターンしないテストケースのために，被テストモジュールへ戻らないスタブモジュールを作成した．
- EXTERNAL セット・オウンコードを作成した．TRITON では UCS COBOL (以下 UCOB) の EXTERNAL 領域に各種共通テーブルや共通コードをもっていて，本来は XIS の ICP (Initial Control Program；初期制御プログラム) 等でセットする．モジュールテストの場合，ドライバーの入力データとして与えなければならないが，数千項目もありテスト使用者が個々にセットすると大変な手間と工数がかかる．そこで，ドライバモジュールから被テストモジュールを CALL する直前に EXTERNAL 領域に値をセットするモジュールを CALL するオウンコードを作成した．
- データ準備作業の効率化のため，標準テストデータを準備した．標準データの選定のために，入出力を中心とした制御に影響を与えるパターン，テスト担当者が共通的に必要とするパターン，テーブルや共通情報のようにシステム全体として準備しておくべきデータについて業務別の複数のパターンを用意した．これらの選定および作成作業は，各開発プロジェクトあるいはチーム（システム単位）で実施した．

・テスト実行時，PADS（Programming Advanced Debugging System；会話型デバック支援プログラム）を自動的に起動することで会話形式でのデバッグを可能とした．

② ファイル系

バッチプログラムの単体テストで，テストデータを作成するツールとして「単体テスト支援ツール（ファイル系）」がある．このツールにより，システムテーブルやFCSS（File Control Super Structure），DMS（Data Management System）等にデータロードできるようなSDF（System Data File）形式のファイルを作成することができる．

その他，既存のファイルから必要なレコードだけを抜き出してテスト用のファイルを作成するファイル抽出機能や，テストで使用する入力/出力ファイルのレコード内容を印書する機能，二つのファイル内容を比較し変更結果を印書するファイル比較機能，データファイルの容量が大きすぎたり，不用なレコードがある場合に，必要なレコードを抽出/変更したり，新たにレコード追加してファイルを作成するファイル加工機能がある．

ただし，単体テスト支援ツール（ファイル系）は，データ件数が大量の場合には，登録件数が多くなり作業負荷が大きくなるため，レコード属性（レコード番号，レコードサイズ等）とデータ内容を任意のパラメタで直接指定することでSDFファイルが作成できる汎用プログラムをTRITONで別途用意し，必要に応じて使い分けた．

2) ファイルマニュピレータ……「ファイルマニピュレータ」は，リアル系の結合テストで使用するデータ環境構築を支援するツールである．データベース，FCSSファイル，構造化FCSSファイル，システムテーブル等のオンラインデータが対象であり，各データのレコード形式を定義した登録集情報とデータ構造情報（スキーマ情報，ファイル番号，テーブル番号等）を情報ファイル（PDP（Procedure Definition Processor）情報ファイル，構造情報ファイル）に登録することで，各データの検索・追加・更新・削除を会話型処理にて容易に行うことができる．そのほか，データを編集・印書するユーティリティが用意されていて，テスト時にユーザが定義したデータ情報（業務トランザクションや入出力トランザクション等）の編集印書ができる．

TRITONではこのユーティリティを利用して，XISのトレース情報ファイルやATD（Audit Trail Disk）情報を編集印書するユーティリティを作成し，TPS（Transaction Processing Segment；トランザクション処理プログラム）のデバッグの際に活用した．また，入出力トランザクションと入力電文定義の登録集をPDP情報ファイルに登録し，そのファイル情報より入出力編集テーブルを作成する入出力編集コンパイラを独自に開発した（図7）．

2.3.4 保守工程および全工程にまたがる支援

保守工程および全工程を通して管理を支援するツールとしては，「プログラム管理ツール」と，「文書管理ツール」および「進捗管理ツール」がある．

また，4か所に分散開発した大規模開発を支えるツールとして「分散開発支援ツー

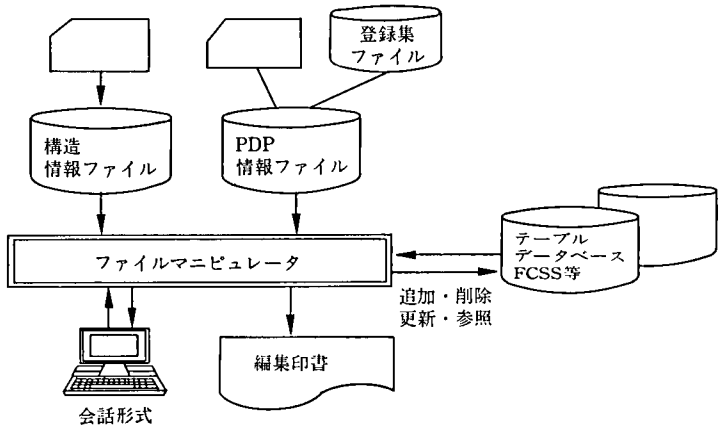


図 7　ファイルマニピュレータ
Fig. 7　File manipulator

ル」がある．「分散開発支援ツール」については次章で記述するため，ここでは他の三つのツールについて記述する．

1)　プログラム管理ツール……「プログラム管理ツール」は，他のツールと違い，百五銀行・紀陽銀行両行の従来のプログラム管理方法・ツールを参考に要求仕様をまとめて開発したものであるため，金融機関のきめ細かな管理機能を持っている．

機能は，開発作業管理，ソフトウェア資産管理，コンパイル・リンク支援の三つがあり，すべて U 2200 側で機能するツールである．

①　開発作業管理

プログラムのソースコード，仕様書，登録集，パラメタ，JCL 等の各種成果物とその状態（開発中，終了等）を開発作業（たとえば「新商品開発」といった開発のくくり）の単位で管理する．また，同一プログラムに対する複数開発作業を制御する多重変更管理を持つ．

②　ソフトウェア資産管理

開発作業に対して，修正コードの累積，修正履歴管理を行い，対象となる修正を最新ファイルに反映する．対象成果物が登録集やプログラムの場合，それを使用するプログラムの自動コンパイル/リンクを行い，プログラム間の整合性を維持する．また，HDD に各対象成果物の関連（たとえばプログラム間の CALL 関連やプログラムと登録集との COPY 関連)づけ処理を行う．これら一連の処理をメインライン処理と呼ぶ．

③　コンパイル・リンク支援

設計支援やプログラム作成支援ツールで作成したソースコードを U 2200 上の作業ファイルでコンパイル/リンクし，コンパイルリストのエラー部分を画面より直接修正してリコンパイルできるように開発者のデバッグ作業を支援する機能である．

メインライン処理は結合テスト工程から使用し，登録集の変更によって影響を

受けるプログラムの自動コンパイル/リンクを週1回実施し，整合性をとるとともにHDDにそれぞれの成果物の関連づけを行った．その他の機能については，カットオーバの約5カ月前から適用し，本格運用していった．TRITONカットオーバ以降，百五銀行と紀陽銀行とでは，プログラム管理の運用/管理対象に違いがあるが，基本的な作業の流れは同じで，図8のようになっている．

従来からのプログラム管理運用の背景から，銀行によって各作業の入力者が担当者であったりライブラリ管理者である場合があり，運用に多少の違いがある．ツールは，各機能単位にセキュリティレベルが設定でき，そのレベルの高低関係でライブラリ管理者用の画面，開発者用の画面をユーザで選択できるように設計されている．

TRITONは，対象システムの範囲を明確にするため，業務単位に19のシステムを定め，システムごとに保守グループを割り振っている．保守グループとは，プログラム管理上のくくりで，この単位で各ファイル(SYM，登録集，仕様書等)が設定される．この19システムの保守グループの他に，JCLやパラメタ等を管理する保守グループがそれぞれの銀行で複数個存在している．メインライン処理の

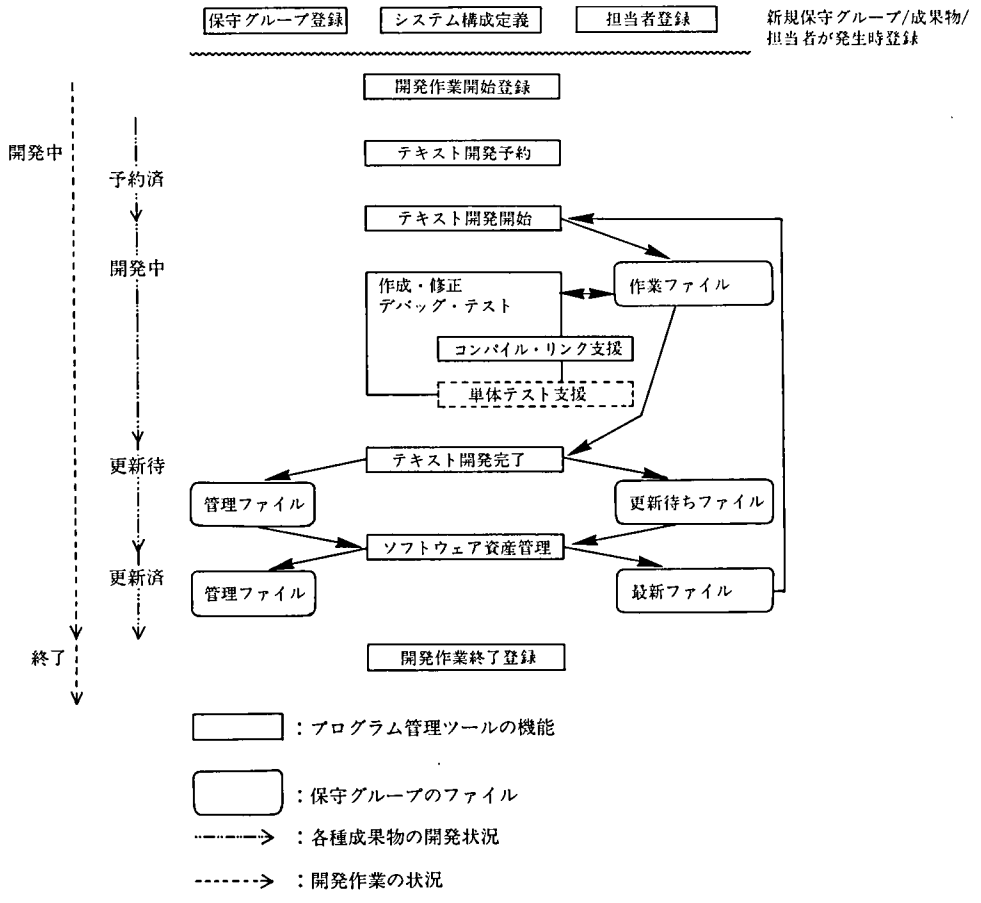


図8　プログラム管理ツールの流れ

Fig. 8　Flow of program management support tool

ようなライブラリ更新作業は，この保守グループ単位に行う．

管理帳表については，IDESで標準帳表を数種類用意しているが，ユーザごとに設定する帳表や出力する項目に関するニーズが多様であるため，その対応策としてHDDに登録されたプログラム管理の情報をMAPPER情報(4 GLの一つであるMAPPERのデータベース＝MAPPER REPORT)として抽出するツールが用意されている．ユーザは，このMAPPER情報を加工することで独自の帳表を作成できる．

2) 文書管理ツール……「文書管理ツール」は，TRITONで作成したワープロの共通成果物をFDDで管理するのではなく，U 6000で一元管理し，任意に検索・参照できるツールである．管理対象は一太郎*，花子*で作成した文書に限られている．また，セキュリティレベルが使用者，文書のそれぞれに指定でき，セキュリティレベルの大小関係によって検索・参照できる文書が規定される．

TRITONでは，要件定義での成果物（システム現状分析，課題検討書等）や基本設計書（業務処理使用説明書，共通モジュール外部仕様書や概説書等）等のワープロ文書はU 6000に登録し，拠点間の文書交換の媒体はテープで行っている．

その他，登録された文書を一括して大量印書するため，いったん文書をU 6000からホストに転送し，ホストプリンタより直接印書できる機能も用意した．そのとき，業務処理使用説明書のような業務処理モジュール単位に作られている基本設計書については，プログラム作成支援ツールで作成したモジュール仕様書とマージして一括印書できる機能もあり，リスト管理を容易にしている．

3) 進捗管理ツール……「進捗管理ツール」は，ソフトウェアの開発・保守が納期通りに見積った予算内で完了し納品できるように管理するツールで，MAPPERで作成されている．TRITONでは，IDESの進捗管理ツールを基本設計工程でのみ使用し，詳細設計以降はTRITONで独自に作成したものを使用した．これは，IDESでは，モジュールごとに発注から製造，検証，納品まで作業単位別の予定・実績管理を行うことが困難であったためである．ただし，開発にあたっては，開発工数を削減するため，IDES進捗管理ツールのD/Bの有効活用を図り，それをベースに改良を加えた．

詳細設計～単体テスト工程での特徴としては，発注作業で，各モジュールごとに発注予定日・発注日を管理し，製造・検証作業では，プログラム内部仕様書の作成検証とプログラミング・単体テストの実施・検証の二つの作業に分けて，それぞれのモジュールごとの予定・実績を管理したことである．さらに，納品作業のために開発部署発送日・管理部到着日等の管理項目を設け，成果物の現物管理を行った．とくに分散開発では，管理部門と開発部門が離れるため，成果物がどの開発場所からいつ発送され，管理部門にいつ届けられたかを明確にしておくことが必須となる．

結合テスト工程の特徴は，進捗管理だけでなく，障害管理も機能として追加したことである．進捗管理は，テストの開始日，終了日，テストケース数の予定と実績を管理し，障害管理は，AP障害記録表をもとに障害件数，懸案数，障害関連

* 一太郎，花子は(株)ジャストシステムの登録商標である．

モジュール等を管理している．いずれも，リアルタイム・システムの場合業務処理モジュール単位，バッチの場合 JOB 単位に管理している．

これら進捗管理のエントリーは，管理部門で一括して行った．また，各種帳表は管理部門を通して各開発グループに配布された．

4) ドキュメント出力……ここではとくに保守工程で使用するドキュメント・ツール（インパクトレポートと M 1 リバース）について記述する．

① インパクトレポート

「インパクトレポート」は，HDD から項目等の変更に対する影響度を調査出力するツールである．リポジトリは，実体（クラス）の階層やクラス間の関連を表現できる拡張 E—R モデルの形態をとっている（図 9）．

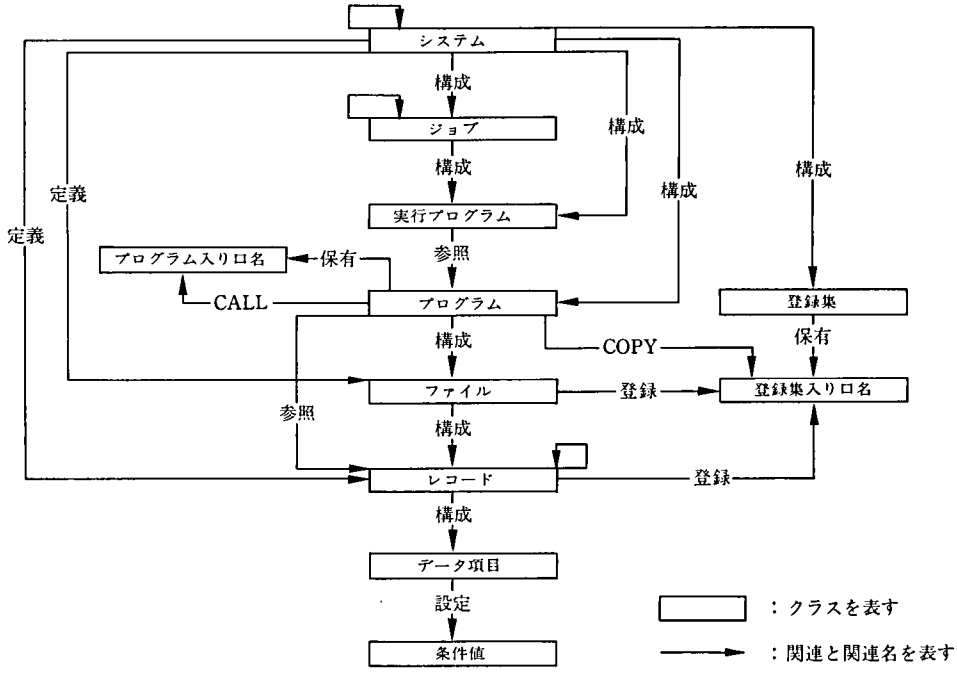


図 9 IDES のリポジトリ

Fig. 9 Repository of IDES

これらの情報は，設計支援ツールやプログラム管理ツールのメインライン機能で登録される．「インパクトレポート」は対象となるクラスから上位のクラスに対し関連をたどり，直接的および間接的に影響するものを調査する．たとえば，あるデータ項目の桁数を変更したい場合，それを使っているレコード（集団，論理，物理）やその登録集名，あるいはその登録集を使用しているモジュール名を HDD より関連をたどってドキュメント編集して出力する．TRITON では，10 万項目以上のデータ項目やレコード等の設計情報と，400 万ステップ以上に及ぶプログラムや登録集のプログラム管理情報を登録し，開発管理情報等は，開発作業が発生するごとに登録が行われ，膨大な量になっている．保守工程において，変更・修正量の調査ツールとして，インパクトレポートは，有

効活用されている.

② M1リバース

「M1リバース」は，TRITONから機能要求して作られたドキュメンテーションツールでは，U2200上で起動するツールである.

M1とは，TRITON用語でモジュール仕様書の機能説明書のことを意味する.

機能説明書は詳細設計者が担当する部分で，TRITON開発時ワープロで作成した文書であり，プログラマがこの機能説明書をもとにコーディングレベルに落とせるような記述内容になっている.「M1リバース」は，この機能説明書をソースコードから作り出し，ホストプリンタより出力する機能である. U2200上のUCOBソースコードを対象とし，生成されるドキュメントはCOBOLの命令語をパターン化して日本語に翻訳，編集したものである.

保守工程で，プログラムの修正変更が発生した場合，ワープロの機能説明書を保守する必要はなくなった. 修正後のソースコードより機能説明書が作り出されるので，モジュール仕様書と同様，ソースコードと整合性のとれたドキュメントを維持管理することができる.

3. ハードウェアおよびソフトウェアの構成

IDESは，3階層（ハブ，サーバ，ワークステーション）からなる分散開発環境を提供している.

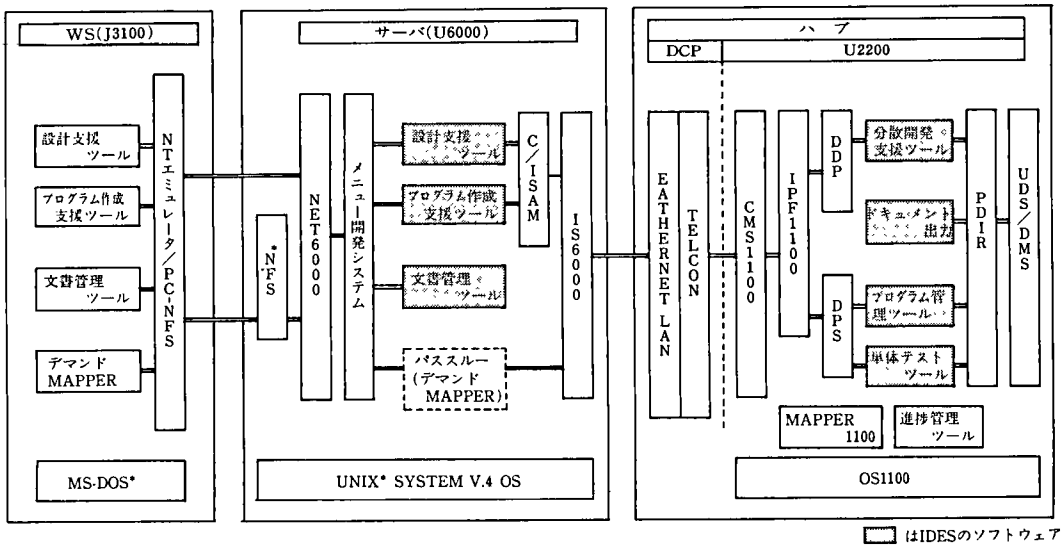
関連するハードウェアは表1の通りである.

表1 IDESのハードウェア一覧
Table 1 List of IDES hardware

	関連ハードウェア
ハブ	2200/1100シリーズ DCP(Distributed Communication Processor) イーサネット*LAN イーサネットLANライン・モジュール トランシーバ
サーバ	U6000/5x，6x イーサネットコントローラ トランシーバキット
WS （ワークステーション）	J3100/GT-041A，GT-SX Dynabook* LANボード マウス トランシーバキット

*イーサネットは米国Xerox社の，Dynabookは(株)東芝の登録商標である.

マイクロ系ハードウェアに関しては，約5年間の開発期間内に次々に新機種が発表，リリースされており，TRITONのU6000で，当初U6000/50からスタートし，51，55，65と導入してきた. J3100も同様で，最近ではDynabookが主流となっている. また，U2200については，詳細設計から単体テストフェーズで，2200/400（大阪・東



UDS-DMS：UDS(汎用データシステム)のアーキテクチャに適合したネットワーク型データベース管理システム
DDP(Distributed Data Processing)：プログラム間/ファイル・ジョブ転送プログラム
DPS(Display Processing System)：画面・帳票定義/メッセージ端末支援プログラム
IPF(Interactive Processing Facility)：プログラム開発環境支援システム
CMS-TELCON：ネットワーク管理システム
＊MS-DOSは米国Microsoft社の，NFSは米国Sun Microsystems社の登録商標である．
また，UNIXオペレーティングシステムはUNIX System Laboratories,Inc. が開発し，ライセンスしている．

図 10　TRITON 適用時の IDES ソフトウェア構成図
Fig. 10　IDES software by TRITON

京拠点)，1100/90（津・和歌山拠点）を使用して開発を行った．結合テスト以降のフェーズでは，津・和歌山拠点で 2200/900 を使用してテストを行った．

TRITON で使用した IDES のソフトウェアは，図 10 の通りである．また，TRITON 開発のネットワークについては，次章に記述する．

4. IDES を利用した大規模分散開発

4.1 垂直分散と水平分散

3 章のハードウェア構成に示したように，IDES はハブ(U 2200)，サーバ(U 6000)，ワークステーション（J 3100；以降 WS と呼ぶ）と 3 階層に垂直分散した環境を提供し，主にハブの作業負荷軽減と WS による操作性の向上を狙っている．

ハブ　：多量のデータの一元管理およびすべてのテスト実行環境を持つ．

サーバ：開発作業の中間結果の格納場所であり，LAN 上のエントリー機としての WS とハブとのデータのやり取りを中継し，WS からハブへのデマンドセッション（会話型処理の要求）の中継を行う．

WS　：個人環境としてスタンドアローン，端末いづれの機能も持ちえる．

開発工程によって開発環境も変化し，詳細設計以降最大 4 拠点での分散開発運用となり，それぞれの拠点がネットワークで結ばれ，同期を取りながら開発を進めた．各工程におけるハブ，サーバ，WS の各構成要素の役割は表 2 のように変っていった．各

表2 開発工程における各構成要素の役割
Table 2 Role of the components in system development life cycle

	要件定義	基本設計	詳細～プログラミング	テスト
ハブ (U 2200)	————	HDD 登録・修正・印書 ライブラリ保守 参照レポート生成 プロジェクト管理	同左 コンパイル&リンク 単体テスト実行 テストデータ生成・編集 DD 整合性維持	同左 結合テスト実行 プログラム管理
サーバ (U 6000)	文書管理	同左 HDD とのインタフェース 定義体生成 MDD 登録・修正・印書 参照レポート維持	同左 仕様書←→ソースコード 仕様書保管 マクロ作成・登録	同左
WS (J 3100)	文書入力・修正・印書 UNIX 端末 デマンド端末	同左 設計支援の入力機能 設計内容の表示・修正・印書	同左 仕様書作成	同左

工程での変遷を以下に示す.

1) 基本設計段階……IDES の利用は，データ仕様の登録・維持といった設計支援が主体であり，大阪拠点に集中し，リポジトリ（MDD, HDD とも）も1か所で集中管理した.

2) 詳細設計～プログラミング段階……プログラム設計，コード化，単体テストの段階であり，津，和歌山，東京，大阪の4拠点に分散し，それぞれがハブ，サーバ，WS によって構成されていた.リポジトリも1か所集中管理から4か所分散管理となり，4か所で個別にデータ仕様の変更等を行い，毎日の一定時間に変更ログをやり取りして拠点間での整合性をとった．また，登録集や共通サブルーチン等も同様に整合性をとるため拠点間でファイル転送等を行い，同じデバッグ環境にした．これらの整合性維持のツールについては，4.2 節で記述する.

3) テスト段階……結合テスト段階以降では結合テスト途中で共同開発から個別開発へ変わり，津，和歌山の2拠点での開発が主体となった．ハードウェア構成も XTC (eXtended Transaction Control)*環境の 2200/900 の4ホスト構成となり，開発環境から実行環境でのテスト作業が中心となった．IDES は4ホストのどのホストからも使用できるようになっている．この時点でプログラムが第一次凍結され，これ以降各行のカスタマイズが加えられ，リポジトリや登録集等も各行での独自管理となった.

開発形態としては，弊社社員や協力会社が拠点内に集中して開発する形態と，主に協力会社が独自に LAN を敷設して拠点と回線を結ぶサテライトでの開発の2通りがあり，必要に応じて使い分けた.

TRITON では，詳細設計からプログラミング工程ではプログラム開発をいくつか

* XTC は，TIP (Transaction Interface Package)/UDS (Universal Data System) とともに，大規模オンライン・トランザクション処理および並列・無停止連続運転の要求に応える拡張トランザクション処理アーキテクチャ XTPA を実現するための機能である.

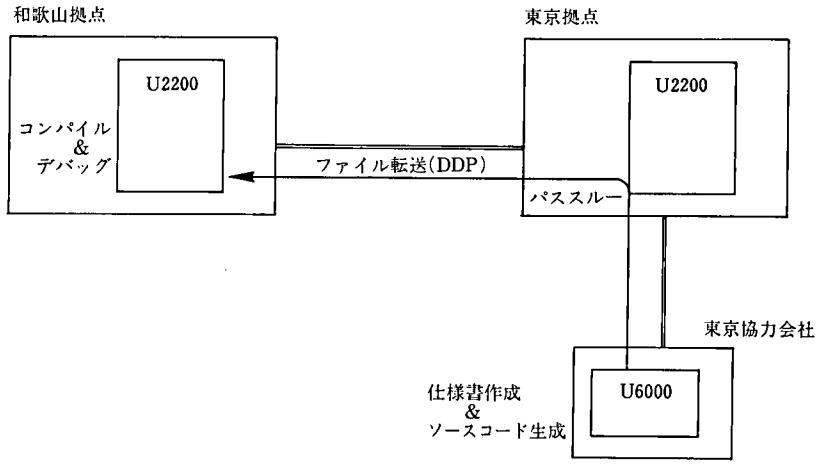


図 11　分散開発時のプログラム開発例

Fig. 11　Sample of program development in decentralized environment

のサブシステム単位に分け，それぞれを拠点単位で分担し，その拠点の責任のもとに開発が進められた．その場合モジュール仕様書の作成作業は，開発者がいる拠点のJ 3100, U 6000の下で実行し，コンパイル，テストデバッグ作業は，責任拠点のU 2200の下で実行することでホストの負荷分散を図った．たとえば図11のように東京拠点のサテライトである協力会社が，和歌山拠点責任のプログラムを請け負った場合，自社のU 6000でモジュール仕様書を作成し，そのモジュール仕様書からソースコードを生成して責任拠点である和歌山のU 2200に転送する．転送後は，和歌山のU 2200にアクセスしてコンパイル・テストを実施する運用をとった．

なお，TRITON開発時のネットワークを図12に示す．

4.2　整合性の維持管理

4拠点（津，和歌山，大阪，東京）を共通の開発環境として維持していくため，各種の分散運用ツールを用意し整合性をとった．とくに，登録集，共通ルーチンとリポジトリについては，4拠点で同期がとれるように毎日拠点間でやりとりを行う運用をとった．開発者にとっては，自拠点内の登録集，共通ルーチン，リポジトリのみを作成・変更・修正するだけでよく，ツールが自動的に他拠点に反映してくれた．リポジトリの同期取りツールは，IDES標準ツールであるが，登録集や共通ルーチンの同期取りツールについてはTRITONで作成したものである．各ツールの機能概要を記述する．

1)　登録集の拠点間同期取り運用……登録集の変更は，原則として一週間に一回更新する運用をとっている．これは，登録集変更分を毎日反映したのでは，単体テスト中のモジュールやスタブ，ドライバモジュール等のリコンパイルが必要となり，テストの生産性を低下させると考えたためである．

運用方法を次に示す．

① 各拠点の設計者は作成・変更した登録集（MDDより定義体生成し，アップロードする）を，自拠点のU 2200の登録集用転送ファイルに蓄える．

② 毎日定刻（16：00）になると全拠点で，転送用ファイルに蓄えられた登録集

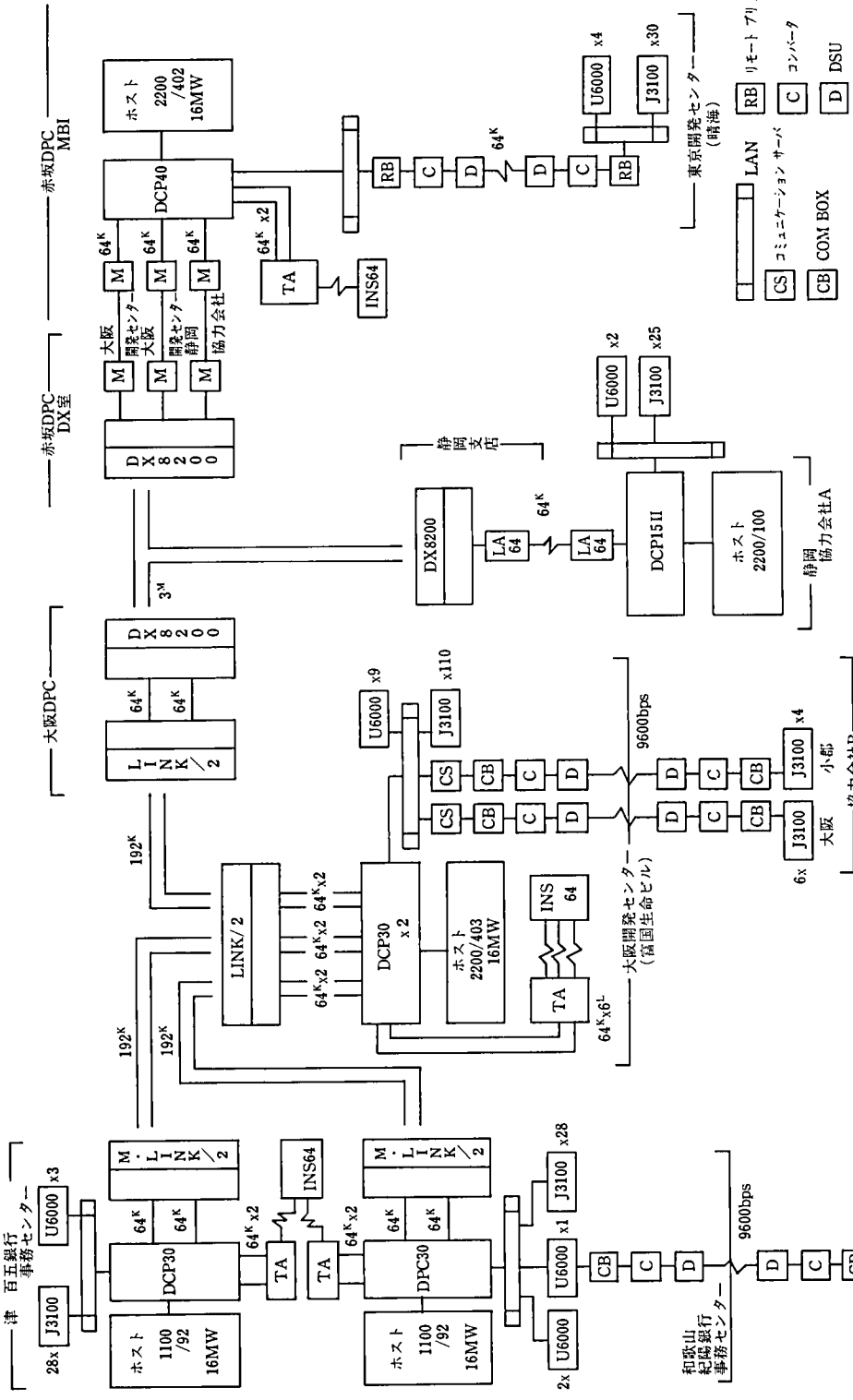


図 12 TRITON ネットワーク詳細設計～単体テスト工程 (平成 2 年 6 月～平成 3 年 6 月)

Fig. 12 Network in TRITON (between 1990.6~1991.3)

を他拠点すべてに向け相互にファイル転送を行う．

③ 自拠点分と他拠点から送られた登録集をマージし，新規作成分と変更分に分けて中間ファイルに保存する．

④ 翌朝のセットアップ処理（ホスト使用禁止時間帯）で新規作成分を中間ファイルからコンパイル用最新登録集ファイルにコピーする．

⑤ 変更分の登録集については，1週間分を蓄積して，翌週の月曜日のセットアップ処理で中間ファイルからコンパイル用最新登録集ファイルにコピーする．

①～⑤の処理はそれぞれの拠点で個別に行われ，翌朝の段階で4拠点ともコンパイル用最新登録集ファイルは同一じものになり，同一のコンパイル環境となる．

2) 共通ルーチンの拠点間同期取り運用……共通ルーチンは，常に登録集と同期をとって変更する必要があり，同一のタイミングで更新される．運用方法を次に示す．

① 各拠点の開発者は作成・変更した共通ルーチンのソースおよびオブジェクトを，自拠点の共通ルーチン用転送ファイルに蓄える．

② 毎日定刻（16：00）になると全拠点で，転送用ファイルに蓄えられた共通ルーチンを他拠点すべてに向け相互にファイル転送を行う．

③ 自拠点分と他拠点から送られた共通ルーチンをマージし，中間ファイルに保存する．

④ 翌朝のセットアップ処理（ホスト使用禁止時間帯）で中間ファイルからテストリンク用ファイルにコピーする．

⑤ 週1回登録集の変更分が更新されるが，それにあわせてテストリンク用に入っている共通ルーチンを中間ファイルにある変更分登録集を使用してリコンパイルする．これを翌朝の月曜日のセットアップ時に登録集と同じタイミングでテストリンク用ファイルにコピーする．

①～⑤の処理はそれぞれの拠点で個別に行われ，翌朝の段階で4拠点とも最新登録集でコンパイルされた同一の共通ルーチンでのリンク環境となる．

3) リポジトリの拠点間同期取り運用（分散開発支援ツール）……4拠点のHDD，MDDの両方を同一の内容にするもので，複数拠点のうち大阪拠点を本部拠点，他を分散拠点とし，本部拠点でリポジトリを集中管理し，他拠点へログを配布する方法で同期をとった．

① 大阪以外の各拠点より当日分の自拠点のMDD更新ログを大阪に転送する（18：00）．

② 大阪では各拠点から送られた更新ログで大阪のMDDを更新し，全拠点分をマージした更新ログを作成する．このとき重複データ等があればエラーリストを出力する．

③ 大阪でマージした更新ログを他拠点にそれぞれファイル転送する．

④ 大阪以外の各拠点では，大阪から送られた更新ログで自拠点のMDDを更新する．このとき更新済みである自拠点分のログは更新しないように考慮されている．

⑤ ①～④は，すべてU 6000内で行われる処理であるが，つぎに各拠点でマージ

されたMDDの更新ログをU 2200にアップロードし自拠点のHDDを更新する．

これらの作業中は，設計支援ツールの使用を禁止し，MDD，HDDとも静的な状態にして作業しなければならない．これら一連の作業は毎日実施され，全拠点のMDD，HDDがすべて同じものとなる．

4.3 開発環境利用のための標準化（ネーミング）

IDESの個々のツールを使用するにあたって，各種ネーミング等の標準化を図り，各

表3 開発者に関するネーミング例(一部)
Table 3 Sample of naming rule for developers

対象	規則
ログイン名	U 6000を利用する時に指定する各開発者個人に割り当てられたID yxxxx y ：n →弊社関連開発者 h →百五銀行関連開発者 k →紀陽銀行関連開発者 xxxx：各社毎連番
単体テスト用個人別モジュール仕様ライブラリ	単体テストでスタブモジュールの仕様を標準モジュール仕様をもとに作成した結果を入れておく開発者個人に割り当てたファイル． DBG * Myxxxx yxxxx：ログイン名(大文字)
モジュール単体テスト・データ番号帯	単体テスト・スタブモジール仕様に対応した開発者個人別テスト・データ番号帯 yyyxx yyy ：ログイン名下4桁より変換する n 60 zz → 6 zz n 61 zz → 7 zz n 62 zz → 8 zz n 63 zz → 9 zz n 64 zz → 5 zz n 65 zz → 3 zz k 40 zz → 4 zz h 20 zz → 2 zz xx ：01～99
個人別シンビオント・ファイル名	単体テスト等で開発者がオルタネート・シンビオント・ファイルを必要とするとき利用 DBG * Syxxxx yxxxx：ログイン名(大文字)
個人別プログラム・ファイル	正規のプログラム・ファイル以外に開発者がテンポラリに各種パラメタ，プログラムを保存しておくためのファイル DBG * Pyxxxx yxxxx：ログイン名(大文字)
デマンド・ユーザID	開発者がデマンド(会話型処理)を利用するときのID zxxxx z ： 送り先コード 1 →百五銀行 2 →紀陽銀行 3 →ユニシス富国 4 →ユニシス晴海 A →協力会社A B →協力会社B C →協力会社C ⋮ M →協力会社M N →協力会社N xxxx：ログイン名の下4桁の数字

開発者が均等に使用できるようにした．U 6000 を使用するためのログイン名は，各開発者単位に割当てる必要があった．U 2200 のデマンドユーザ ID や個人用ファイルのネーミングは，このログイン名をベースとして名付けている（表 3）．このルールは共同開発時点での規則で，全拠点の U 6000，U 2200 とも同じ環境にして，開発者がどの拠点に移動しようとも，その拠点の U 6000，U 2200 を使用できるようにした．また，逆に共同開発が終了した以降（結合テストフェーズ 2 以降）では，百五/紀陽銀行とも独自のネーミングルールに切り換えて自行のメンバー以外は使用できないようにした．

5. 今後の課題

今回の TRITON 開発で，感じたいくつかの反省点と課題について記述する．

1) IDES は TRITON がファーストユーザでもあり，その開発は TRITON の要求にそって TRITON 自身の開発とほぼ並行に行われた．したがって，事前に十分な評価ができないまま開発で使用し，かつ TRITON での適用方法も検討していかなければならない状況であった．また，TRITON 開発はすべてが新規開発であり，IDES に対しても製造面の機能要求を中心に挙げていたため，今後の保守・カスタマイズを考えた時のツールが不足している．この点については，幾つかの改善点を提案している．

2) U 6000，J 3100 といったマイクロプロダクトの活用により，データ設計やプログラム製造についての U 2200 の負荷分散は図られたが，デバッグ環境や単体テスト等は，U 2200 のみの機能であり負荷も大きく，これらの機能分散が今後必要である．

3) 実行環境の基盤ソフトウェア（XIS や IOF*等）にもリポジトリが存在しているが，IDES のリポジトリとは，まったく独立した形態でいくつかの情報については二重入力になっている．今後統合化を図る必要がある．

4) 今後 TRITON を導入するユーザに IDES を適用する場合の移行ツール（現行資産をリポジトリやモジュール仕様書といった IDES の成果物に変換するツール）が必要である．これは，現行資産に対するリバース・エンジニアリング・ツールとしての重要性を持つ．

5) 部品（マクロ）の整備やライブラリの標準化を図る必要がある．

弊社では現在上流工程（システム化計画，要求分析）を支援するツールを計画中で，このツールが完成すると開発工程全体を一貫して支援する統合 CASE ツールとなりえる．

また，高性能・高速のマイクロ系ハードウェアが比較的低価格で次から次へとリリースされるため，IDES 環境のパフォーマンスの格段の向上が予想される．

6. おわりに

今回の TRITON 開発での IDES に対するユーザからの評価は，下記の通りである．

---

* IOF (Integrated Operating Facility) は，コンピュータ運用の分野に対し，すでに提供されている各種運用支援機能の成果を活かすとともに，単一コンピュータ・システムにとどまらず，コンピュータ・ネットワーク全体系にわたる運用環境の整備を目指したソフトウェアである．

・1万人月に及ぶ大規模分散開発であり多数の協力会社が関わっていたが，IDESによって均質したソフトウェアを開発できた．
・IDESというツールがあったため，IDESベースに手順をきっちり定義でき，その結果，開発手順の標準化が図れた．
・4か所分散開発であったにもかかわらず，単一開発環境と同程度の開発生産性を保持できた．

当初IDES適用の狙いとして挙げた点について，一応の評価が得られたことがわかる．

ただし，今後の課題にもあげたように導入や保守が今後の主要作業となるTRITONにとっては，生産性の向上や低コストでの導入が必須要件であり，さらなる発展・強化されたIDESが必要不可欠なツールであると考える．

本稿は，最初にも記述したように，TRITON開発におけるIDESの適用・運用過程を具体的に整理したもので，今後のIDESの適用拡大とIDES自身の進化の検討材料として少しでも役に立てれば幸いである．

参考文献
［1］ 統合開発環境システム　IDES概説書，日本ユニシス，1993.
［2］ 板倉　教，“システム開発環境IDESのリポジトリ”，UNISYS技報，Vol. 12, No. 1, 1992.5.
［3］ 儀間哲雄，“IDESのリバース機能”，UNISYS技報，Vol. 13, No. 2, 1993.8.

執筆者紹介　奥　野　信　幸（Nobuyuki Okuno）
昭和37年生．61年大阪市立大学経済学部卒業．同年日本ユニシス(株)入社．都市銀行のSEサービスを経て，平成元年よりTRITON開発でIDES適用に従事し，現在TRITON企画推進部第2課に所属．

銀行システムと XIS

A Banking System and XIS

宮　村　　洋

要　約　TRITON のシステム基盤は，XTPA (eXtended Transaction Processing Architecture；拡張トランザクション処理アーキテクチャ) に基づき XIS (eXtended Information System；統合オンラインシステム) をベースに構築されている．また，その機能は従来以上にメーカ標準製品として整備されている．これによりユーザは，開発，保守の負担が軽減され，アプリケーション開発に専念できた．

本稿では，TRITON のシステム基盤として，次の特徴ある機能を紹介する．

1) 端末仕様とアプリケーションの独立
2) ファイル容量オーバ対応と新しいファイル・インタフェース
3) 自動化運用を含めたセンタカット・システムの機能
4) アプリケーションの考慮を不要とした研修システム環境の仕組み
5) 多段階ディレードバッチ，多重ラン処理機能を使用したバッチ運用

Abstract In compliance with the Extended Transaction Processing Architecture (XPTA), the infrastructure of the TRITON system has been constructed on the basis of the XIS. The system, so designed as to provide far more sophisticated functionalities as a standard product available from Nihon Unisys, Ltd., enables its users to concentrate more on applications due to its great reduction in user loads of software development and maintenance.

This paper mentions the typical features that are as follows, which serve as the infrastructure of the TRITON system:

1) Independence of applications from terminal configurations
2) File volume management and new file interfaces
3) Functionality of a center-cut system including unattended systems operation
4) Structure of the training system, which no longer requires a need to consider applications programs
5) Batch-processing operation in which multi-stage delayed batch processing and multi-run batch processing are involved.

1. はじめに

銀行システムは規模の拡大とともに，オンライン処理形態の多様化，24時間稼働，ノーダウン・システムの実現等に対応可能なシステム構造を要求している．

この狙いを実現する上で，システムの基盤ソフトウェアの果たす役割も大きい．

今回の TRITON 共同開発ユーザ (BOSS-11 (Banking Oriented Support System under OS 1100；リアルタイム・システム構築支援プログラム) 使用) である(株)百五銀行殿・(株)紀陽銀行殿は，第一次，第二次勘定系オンライン・システムの開発を経て，システム基盤を作り上げてきた．しかし，TRITON の開発にあたり，弊社では，システム基盤をメーカ標準製品として整備し，ユーザの開発負担の軽減を図った．こ

れにより，ユーザはセンタカット・システム，ディレード・システム，研修システム，電文編集機能，ファイル・インタフェース，開発テスト支援等でアプリケーション開発に専念することができた．また，本番以降の保守面での負担軽減も図られた．

本稿ではTRITONのシステム基盤について，XIS (eXtended Information System)をベースとしてどのように構築したかを紹介する．なお，以下に記述するシステム基盤の大半は，銀行システムだけでなく，他業態のシステムにも役に立つものと思われる．

2. システム基盤としての機能要件

2.1 システム基盤の位置付けとXIS

システム基盤は，図1に示すように，アプリケーション・ソフトウェアとメーカ提供のベーシック・ソフトウェアとの中間に位置付けられる．弊社は，過去この分野に対して，BOSS-11（昭和52年），AIS 2200 (Advanced Information System for UNISYS series 2200；昭和59年）を提供し，システム基盤に対するメーカ標準機能の提供範囲を拡大してきた．

今回TRITONが使用するXISは，XTPA (eXtended Transaction Processing Architecture；拡張トランザクション処理アーキテクチャ)に基づき，TIP (Transaction Interface Package) のもとで多様なアプリケーションを構築する，統一的なトランザクション実行環境を提供している．

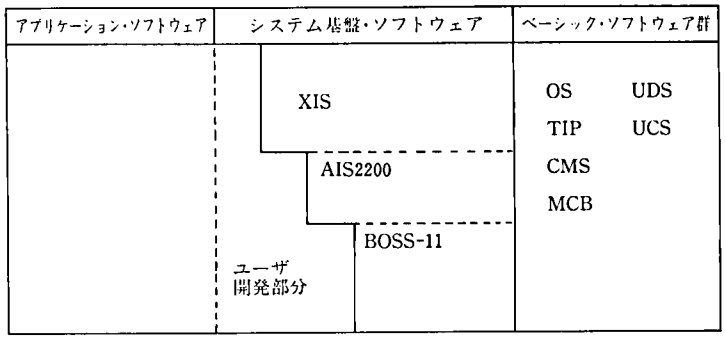


図1　システム基盤の位置付け

Fig. 1　Position of infrastructure

2.2 アプリケーションからのニーズと基盤への取り込み

アプリケーションから見たシステム基盤へのニーズに対して，XISとして実現した部分とTRITONとして実現した部分を次にまとめる．

XISの機能として実現したものを以下に記述する．

1)　ファイル・インタフェースへの要求機能

ユーザがFCSSファイル（File Control Super Structure；TIPが制御するリアル専用ファイル）を使用する上で次の問題があった．

・ファイル容量オーバ発生時，ファイル拡張とデータのリカバリを行うため，システムが長時間停止する．

・効率を考えてファイル分割を行う際に，アプリケーションの変更を伴う．
・店番等のキーを使用したランダムアクセスを可能とするため，使用者がファイル・インタフェースを開発する必要がある．

XIS は，上記問題を標準ファイル・インタフェースとして取り込んでいる．

2) 共通サブシステムへの機能要件

センタカット・システムや研修システムは，従来は，ユーザが共通機能として開発してきた部分が大きい．今回の XIS では，センタカットの自動化運用や研修システムを構築する上での仕組みを提供し，アプリケーション側の開発負荷を軽減した．

3) バッチ運用

XIS は，業後バッチの時間短縮やディレード・バッチ運用の多様化に対して，
・バッチの並行処理を行うための多重ラン機能
・ディレード・バッチ・ランを多段階で実行できる機能
を提供し，バッチ運用を容易にしている．

次に，XIS の機能を使用し，TRITON として実現したものを以下に記述する．

1) 端末仕様とアプリケーションの独立

ホストの更改サイクルより，一般的に端末の更改サイクルは短い．TRITON では，端末に依存する部分とアプリケーション部分を切り分け，端末更改時の変更箇所を明確にし，修正を容易にした．

2) 開発テスト・ツールの整備

XIS は，デバッグの生産性向上を図るために次のツールを提供している．
・オンライン端末を使用しないで，TPS (Transaction Processing Segment；トランザクション (業務) 処理プログラム) のデバッグができるデマンド (会話型) 端末シミュレータ
・テーブル，データベースの参照や更新時のトレース採取機能
・ラッシュ・テストツール

TRITON では，上記 XIS の機能を使用し，テストデータ作成や結果検証ツール等の周辺環境を整備し，使いやすいものにしている．

3. システム基盤の概要

3.1 電 文 編 集

TRITON における電文編集は，端末からの入力電文をアプリケーションが処理する入力トランザクションに変換する入力電文編集と，通帳や伝票等の媒体単位の出力トランザクションを出力電文に変換する仕掛けからなる．

TRITON の開発は共同開発形態をとったが，両行の端末は現行の電文仕様のままであり，アプリケーションの共通化を図る上で，端末の差異を吸収する必要があった．そこで，端末の差異部分をアプリケーションからはずすことによりターミナル・インディペンデント化を図った．具体的には，端末の違いを編集パラメタと編集ルーチンに吸収した点である．さらに，個々の電文により異なる編集パラメタをテーブル上に持ち，編集ルーチンと独立させることにより，変更や追加時の保守を容易にしている．

これにより，開発段階での編集パラメタの正当性確認は，アプリケーションと切り離してテストできるため，開発テスト時の生産性向上にもつながった．

1) ターミナル・インディペンデント……TRITONでは編集を二段階で行うことにより，アプリケーションは端末の違いを意識する必要がほとんどない．具体的には，入力に関しては入力編集を編集パラメタおよびICP（Initial Control Program；業務処理開始プログラム）にて吸収している．また，出力に関しては出力編集を編集パラメタと共通モジュール群にて吸収している．とくに出力編集は，編集データの発生時点で，都度媒体単位の出力トランザクションを蓄積し，メッセージ送信時に一括して編集を行う方式である．電文編集順序はアプリケーションの出力トランザクション蓄積順序ではなく，蓄積時に指定する編集優先度により決まる．これにより，アプリケーションは出力トランザクションを作成する順序に関係なく端末電文仕様に応じた編集が可能となった．入出力編集の流れを図2（a，b）に示す．

2) 編集パラメタの擬似言語化……また，編集パラメタは日本語記述とCOBOL言語相当の条件選択（IF文），繰り返し（DO文），データ補完機能が使用でき，作成や保守が容易にできる．なお，編集パラメタは編集コンパイラにて構文チェックが行われ，編集テーブルにロードされる．

3.2 新しいファイル・インタフェースと容量管理

3.2.1 テーブルとFCSSファイルのインタフェース統合化

テーブルとFCSSファイルの相違は，メモリにあるか，ディスクにあるかだけであり，使用方法に変わりはない．そこで，XISは，テーブルとFCSSファイルのインタフェースをテーブル・インタフェースとして統合化．XISリポジトリ*に配置媒体を定義するだけで使い分けができるようにした．これにより，アプリケーションはプログラミング上での使い分けが不要となった．

3.2.2 論理テーブルと物理テーブル

更新頻度の高いテーブルや大容量のテーブルは，効率面からファイル分割を行い，ディスク配置の分散を図る必要がある．しかし，従来は，ファイル分割によりアプリケーションの変更を伴った．XISでは，使用者インタフェース上は論理テーブル番号として一意であり，リポジトリ定義にて実媒体である物理ファイルを分割するだけでよい．これにより，使用者が意識することなくファイルのパック分散が図れる．

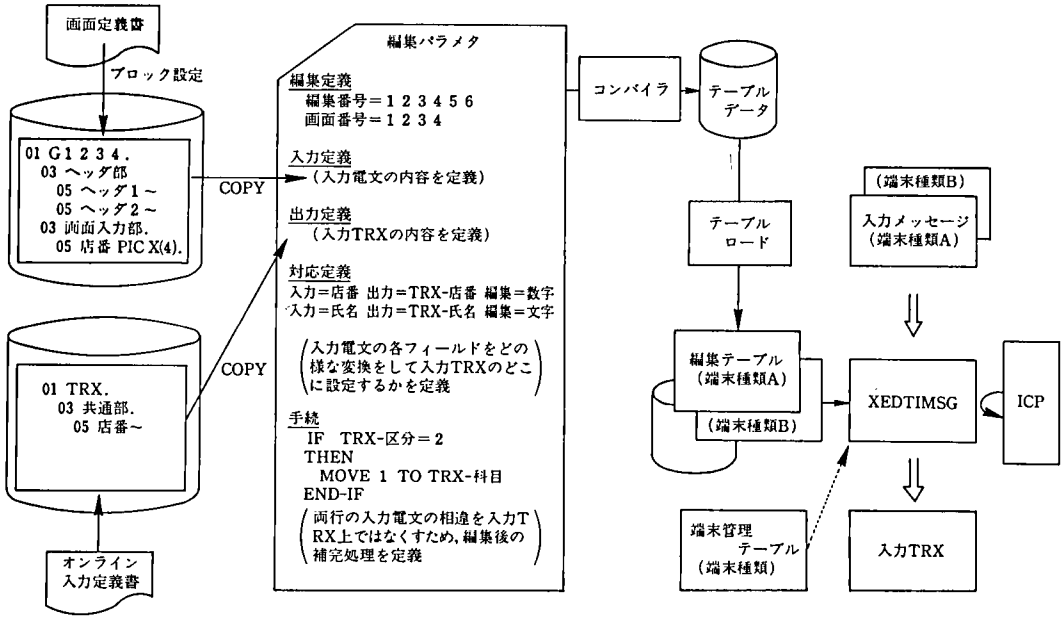
また，XISはオンライン中でのファイル容量オーバに対し，物理ファイルを別途割当てる動的拡張ユーティリティも準備している．なお，使用者は論理ファイルとしてアクセスするため，プログラムの変更は不要である．

3.2.3 構造化FCSSファイル・インタフェース

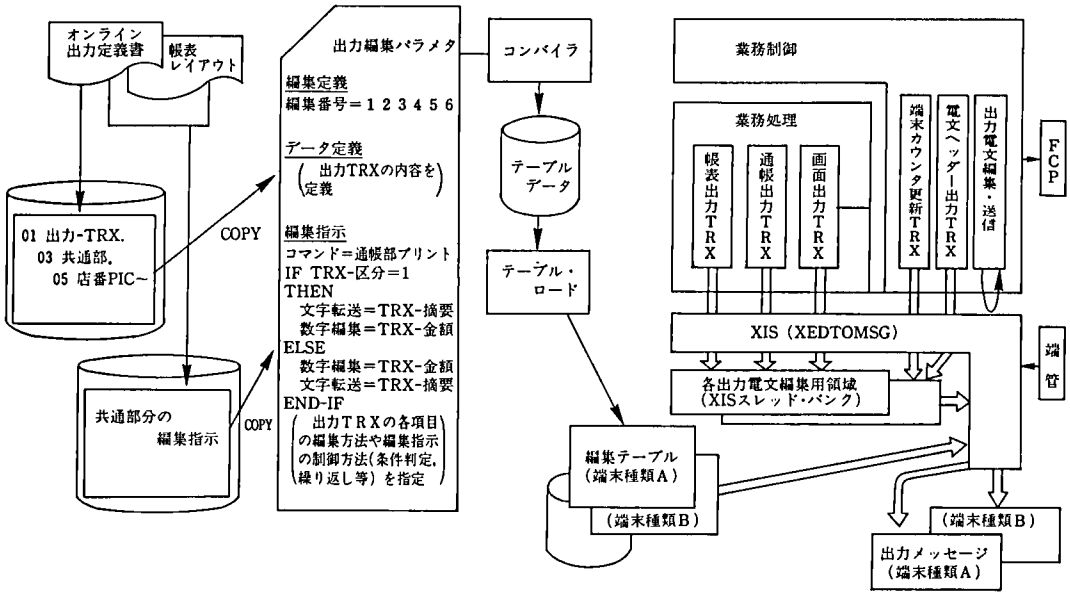
銀行システムにおけるジャーナル・ファイルは，処理効率面からFCSSファイルを使用している．その代表的なファイルは為替のジャーナル・ファイルである．しかし，為替の処理は店や端末をキーとしたランダム・アクセスとなるため，アプリケーション側でファイルのレコード・アドレス管理が必要であった．また，店毎にデータ量の違う点や日によってデータ量が違うことより，オーバフロー対策が必要である．

---

* リポジトリ：システムの構成要素を管理するファイルであり，SIB（System Information Base）と称する．



(a) 入 力 編 集



(b) 出 力 編 集

図 2 入出力編集の流れ

Fig. 2 Flow of editing input-output messages

これに対して，TRITONではXISの構造化FCSSインタフェースを使用することにより，アプリケーションからアドレス管理やオーバフロー管理の考慮を不要とした．

XISは使用者インタフェースとして，ファイル番号，キーおよびレコード通番を指定することにより，読み込み，書き込みができるインタフェースを持っている．なお，採番形態としてレコード通番を指定するランダム通番方式と通番をXISに任せる順通番方式とがある．また，ファイルは各キーごとにあらかじめ確保するレコード領域と溢れた場合に使用する共通のオーバフロー領域からなる．ファイル構造図を図3に示す．

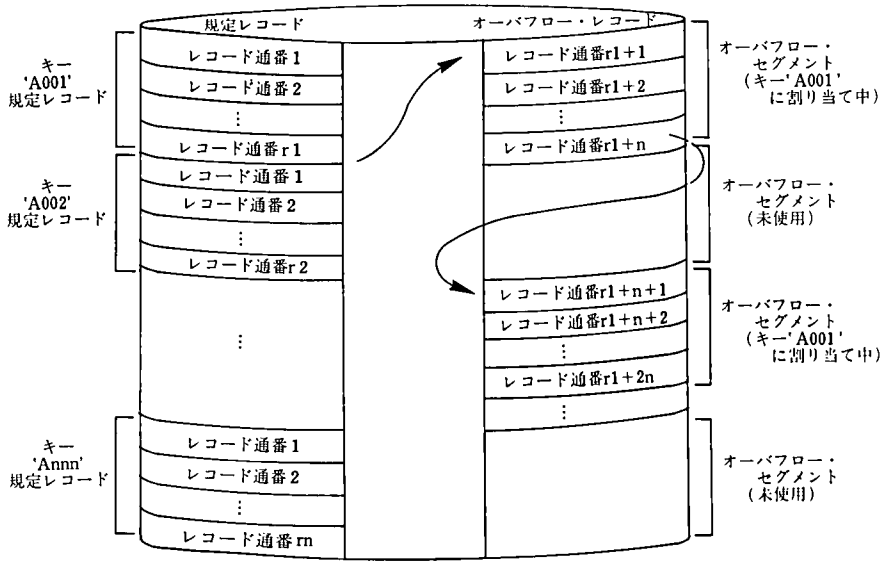


図3　構造化FCSSファイル・インタフェース

Fig. 3　Structured FCSS file interface

3.2.4　ファイル容量管理

新しいファイル・インタフェースにより，オンライン中でのファイル容量オーバは，システム障害とはならない．しかし，オーバフロー領域を使用し続けることはシステムの処理効率を妨げることになる．

従来は，ファイル属性ごとにユーティリティや使用者プログラムを流し，オペレータがチェックしていた．TRITONでは，XISのファイル情報入手インタフェースを使用し，オーバフローポインタの使用を日次ベースで一元管理している．また，情報はシステムのターミネーション時に採取し，MAPPERへの取り込みを行い，オーバフロー状況の監視をするとともに二次加工にて統計をとっている．この情報をもとに，危険な状況になったファイルは，拡張作業やファイルガーベージを行っている．

容量管理の情報として，

構造化FCSSファイル　：キーごとのオーバフロー使用状況
データベース　：エリアごとのオーバフロー・ページの使用状況
センタカット入力ファイル：ファイルごとのセグメント使用状況
トランザクション　：ファイルごとの書き込みブロック数

が使用される．

3.3 共通サブシステム

3.3.1 センタカット・システム

端末入力データ量の伸びに比較して，センタカット・データは今後ますます増えることが予想される．TRITONでは大量のセンタカット・データを取り扱うにあたり，従来はユーザが開発していたデータ管理や自動化運用部分についてもXISに機能を吸収することにより開発の負担を軽減した．

1) データ管理……センタカット処理対象データは，データが処理される振替日とデータの種類（口座振替データであれば電気料や電話料等の単位）ごとにファイルへの蓄積およびセンタカット実行がなされる．また，データは蓄積からセンタカット実行および結果の返却に至るまでの期日管理が必要である．さらに，日によってデータ量も違い，ファイル容量のオーバフロー管理も必要である．

XISではセンタカット処理データの蓄積およびスケジュールの管理を容易にするため，ファイルID，データID(TRITONではデータの振替日，データ種別として扱う）単位に制御可能なセンタカット入力ファイルとファイル・インタフェースを提供している．なお，ファイル容量オーバ時にすでに蓄積済みデータを生かして新たに拡張するユーティリティも用意している．

期日管理についてはTRITON側で，センタカット対象データすべてについて，蓄積から返却に至る状態を一元管理する統合管理ファイルを用意している．

2) 自動化運用……センタカットの自動化運用として，XISの自動走行制御ランは以下の機能を持っている．

① センタカット処理対象のデータは，早朝，業中，業後等の実行時間帯が決められている．XISでは，セットアップ時，使用者が予定テーブルへ各データ種別単位に登録することにより，該当時間帯にスケジューラを自動起動する機能を持っている．なお，自動走行制御ランはスケジューラをデータ種別単位に複数本同時並行処理させるが，別途コマンドにて，各時間帯に応じた並行本数の設定およびオンライン中での変更を可能としている．

② オンライン処理に影響を与えず，かつ大量データを効率よくスケジュールするための流量制御機能を持っている．これは，システムでの正常負荷（インプット・キュー）を事前に定義し，過負荷状況を検知するとスケジュール件数を減算し，正常となった場合に加算する仕組みである．また，複数ホストでのセンタカット走行時，この流量制御はホストごとに働く．

③ スケジューラを並行処理する際に，各データ種別ごとに処理順序を保証する必要がある．XISでは事前にプライオリティ付けを行うことにより，データの処理順序を保証している．

また，センタカット入力ファイル上のデータは，事前の正当性チェック等から店番および口座番号順に並んでいる場合が多い．従来はセンタカットTPS間のデータベースへのロック競合を防ぐため，別途データの並べ替え(ランダマイズ)を行っていた．XISの標準スケジューラは，実行時に入力ファイル上のデータをランダム・リードする機能を持っているため，事前作業としてのランダマイズが

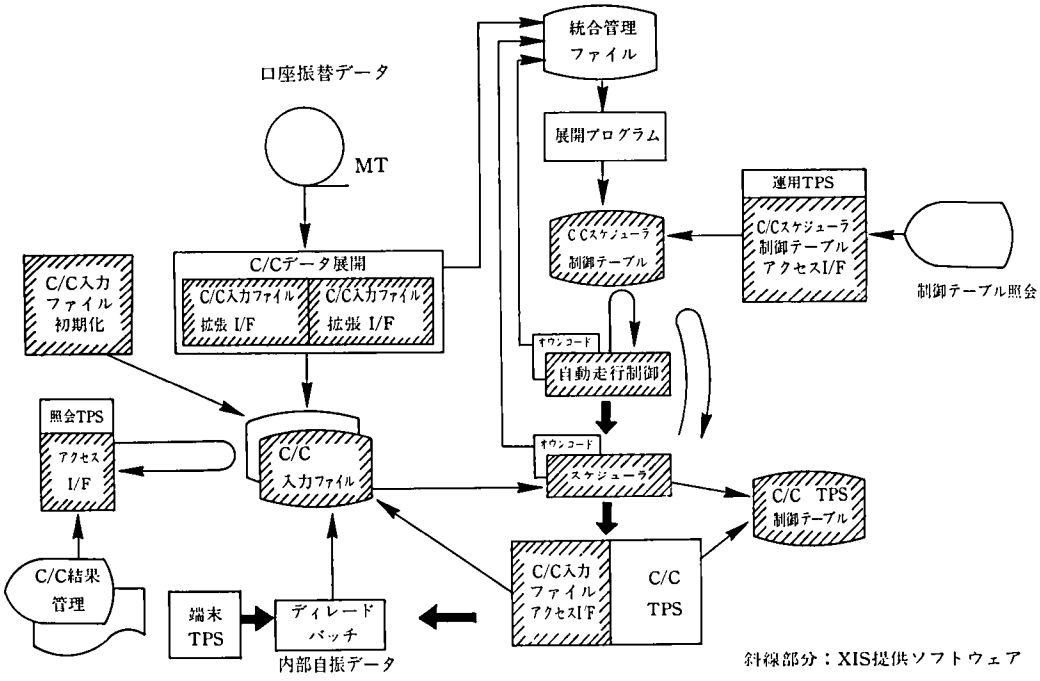


図 4　センタカットの仕組み

Fig. 4　Structure of center-cut

いらない．これにより，センタカット実行までの時間短縮を図ることができる．センタカット全体像を図4に示す．

3.3.2 研修システム

TRITONにおける研修システムは，営業店で行う営業店練習とセンターでの集合研修からなる．システム的には本番環境下での資源のうちテーブルやデータベースの参照・更新について，プログラムを変更することなく本番と研修での使い分けができる仕組みを作ることである．TRITONでは研修システムの実現をXISのテストモード機能を使用し，テーブルやデータベースのアクセスについて，アプリケーションの考慮を不要とした．XISのテストモード機能は以下の通りである．

1) 仮実変換機能……XISは，プログラムのコーディング中に現れるテーブル番号やトランザクションコード等に関して仮値を設定し，ファイル・インタフェース使用時に実際のテーブル番号やトランザクションコードに変換する機能を持っている．なお仮実変換は，本番とテストモード（最大99まで）に対して行われる．TRITONでは，次の項目の仮実変換を研修環境として使用している．

　データベース　　：サブスキーマ名
　テーブル　　：テーブル番号
　ICP　　：トランザクションコード
　業務トランザクション：トランザクション・ファイル番号

2) テストモードの設定……テストモードは，ICP，業務制御プログラム，端末に対して，コマンドおよびプリミティブから設定できる．

TRITON では営業店練習時は，営業店端末から研修登録取引を行うことにより，該当端末にテストモードを設定する．集合研修は，研修専用端末であり，セットアップ JCL にてテストモードを設定する．

3) 本番と研修のファイル共用……XIS では本番と研修とでデータベースやテーブルを共用できる．なお，共用する場合は，研修システム側からの更新はエラーとする保護機能を持っている．

3.4 バ ッ チ

3.4.1 多 重 ラ ン

銀行システムにおけるバッチは，オンライン処理結果である業務トランザクションやバッチ・マスタを入力とする処理形態が多い．同一データを個々のバッチランがそれぞれ参照すると入出力の競合となり効率が悪い．TRITON では上記問題を解決するため，XIS の多重ラン機能を使用してバッチ処理時間の短縮を図った．多重ラン機能とは，同一データを使用する複数バッチを多重ラングループとして定義し，ファイルを読み込むオーナ・ランと，共用バッファ（主記憶領域）を経由してデータの受け渡しをする複数のメンバ・ランとを一体化して稼働させる仕組みである．XIS の多重ラン構造を図 5 に示す．

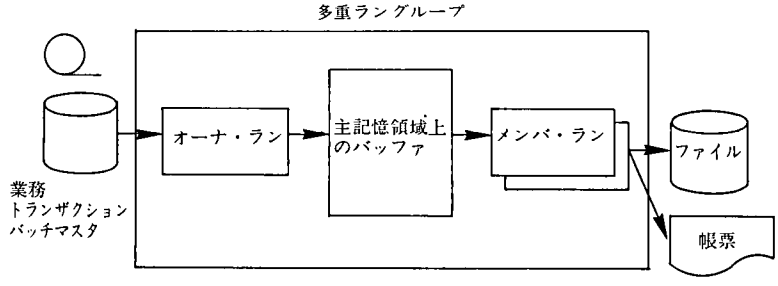


図 5 多重ラン

Fig. 5 Multiple jobs

3.4.2 ディレード・バッチ

自動機の時間延長によりオンライン業務時間帯は長くなってきている．また，営業店への還元資料もタイムリに還元する必要がある．このために，日次バッチの効率化が必要であり，これを具現化する機能として多段階ディレード・バッチを開発した．さらに，日によってトランザクション量が違い，トランザクション・ファイルのオーバフローに対する考慮も必要である．

1) ディレードの多段階化……TRITON では日次バッチで扱う大量の業務トランザクションをディレード・バッチランにて，

① 営業店への配信データとして急ぐトランザクション

② 業後バッチ用トランザクション

③ 情報系用トランザクション

として振り分けや加工を行い，即時性への対応と目的別（預金系，融資系）にデータを分けることにより，後続バッチの処理時間の短縮を図っている．TRITON

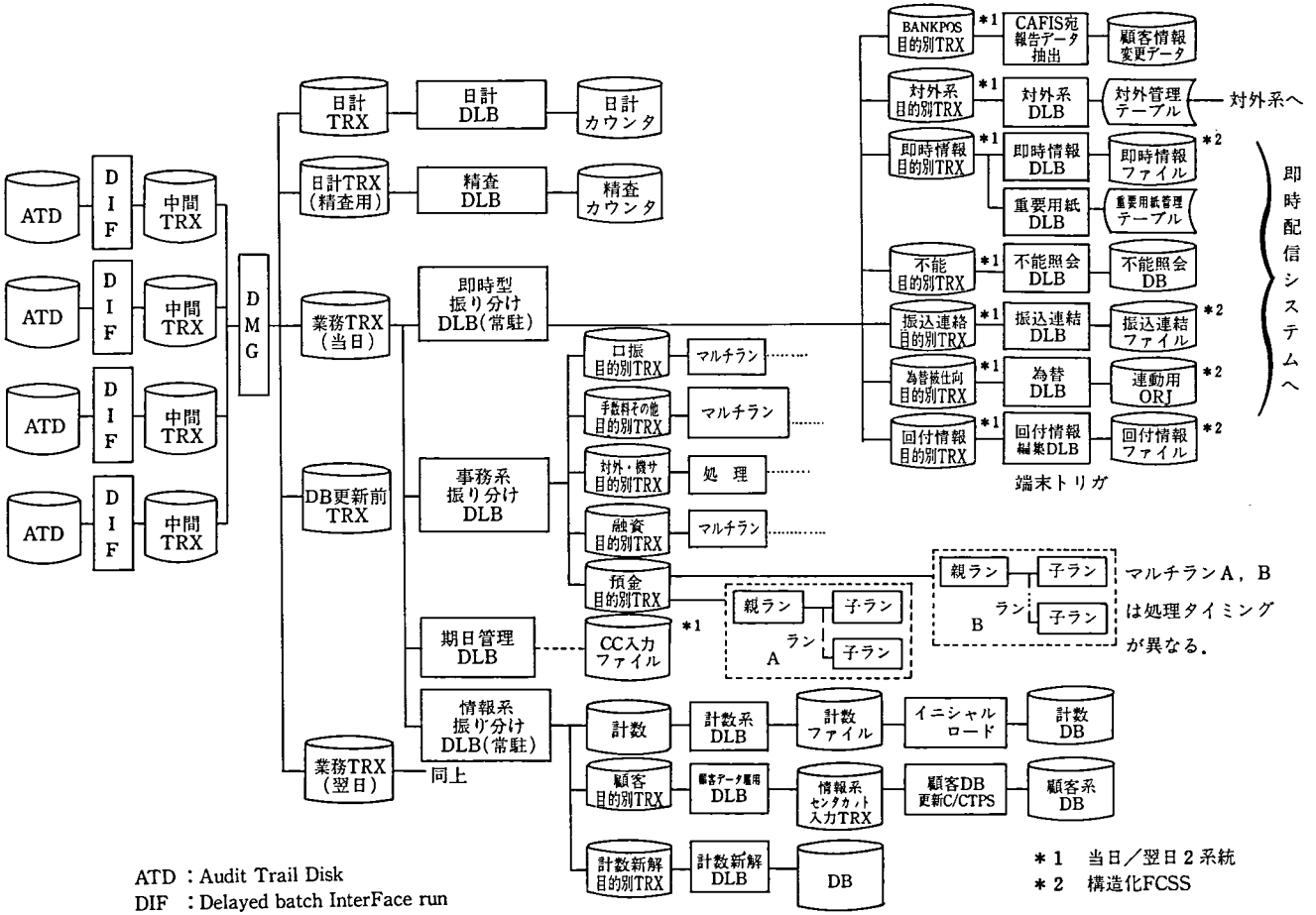


図 6 ディレード・バッチ体系

Fig. 6 System procedure of delayed batch

は，第一段階での振り分けディレード・バッチランと振り分け後の後続ディレード・バッチランからなる三段階（図6のディレード・バッチ体系参照）の形態をとっている．

2) トランザクション・ファイルのファイル容量オーバ対応……流動性元加処理日等のトランザクション量の多い日は，XISのファイル拡張ユーティリティを使用し，予備ディスクにファイルを割り当てることができる．これにより，予め最大容量を確保する必要がなくなった．また，オンライン中にファイル容量オーバの危険が発生した場合にも，この拡張ユーティリティを使用することにより動的拡張が可能である．

TRITONでは，多重ランと多段階ディレード・バッチランを使用することにより，図6の形態をとる日次バッチ・システムを作った．これにより，営業店への還元資料の即時配信，情報系へのデータ蓄積，業後バッチの時間短縮が行えた．

3.5 開発テスト支援

3.5.1 デマンド端末シミュレータ

従来は，開発テスト環境下におけるTPSのテストは，オンライン端末の使用が必須であった．しかし，開発環境に接続されているオンライン端末には限りがあり，また実行途中でのPADS 1100* による解析ができない制約もあった．

XISは，デマンド端末を使用して，使用者プログラムの変更を伴わずにTPSのテストが行える機能を持っている．これにより，オンライン端末台数の制約を受けずにTPSのテストが可能となった．また，デマンド端末シミュレータは，TPS形態でのテストだけでなくバッチ形態でのテストも行えるため，PADS 1100を使用した実行時解析が可能となり，テスト時の生産性向上を図ることができた．

1) テストデータ作成……TRITONは，デマンド端末を使用して会話インタフェースにて，あたかもオンライン端末から入力されたかのように，電文イメージを作成する入力電文ジェネレータを開発し，XISのデマンド端末シミュレータ用ファイルを作成している．このツールにより，テスト用入力電文データの作成が容易になった．

2) テスト実行……上記テストデータを使用し，XISのデマンド端末シミュレータ経由にて被テストプログラムである使用者TPSを起動する．なお，パラメータ指定により使用者プログラムの更新情報をキャンセルする機能もあり，同一データ環境でのテストが複数回実行できる．

3) テスト結果検証……テスト結果として，トレースによる出力トランザクションの検証およびテスト結果編集ユーティリティによる簡易編集ではあるが，出力電文の検証もできるため，実端末を使用した場合とほぼ同等のテストが行える．

3.5.2 トレース機能

XISは，開発プログラムのデバッグ時の生産性向上を図るため，従来の実行時ダンプとは別に，データベース，テーブルの参照や更新を中心としたトレース情報採取機能を持っている．また，トレース情報は印書だけでなく実行結果をディスクにも保存

---

* PADS 1100 (Programmers Advanced Debugging System)：会話型で原始プログラムの項目名を指定しデバッグができる．機能として，プログラムの実行中断・再開・データの表示，変更を持つ．

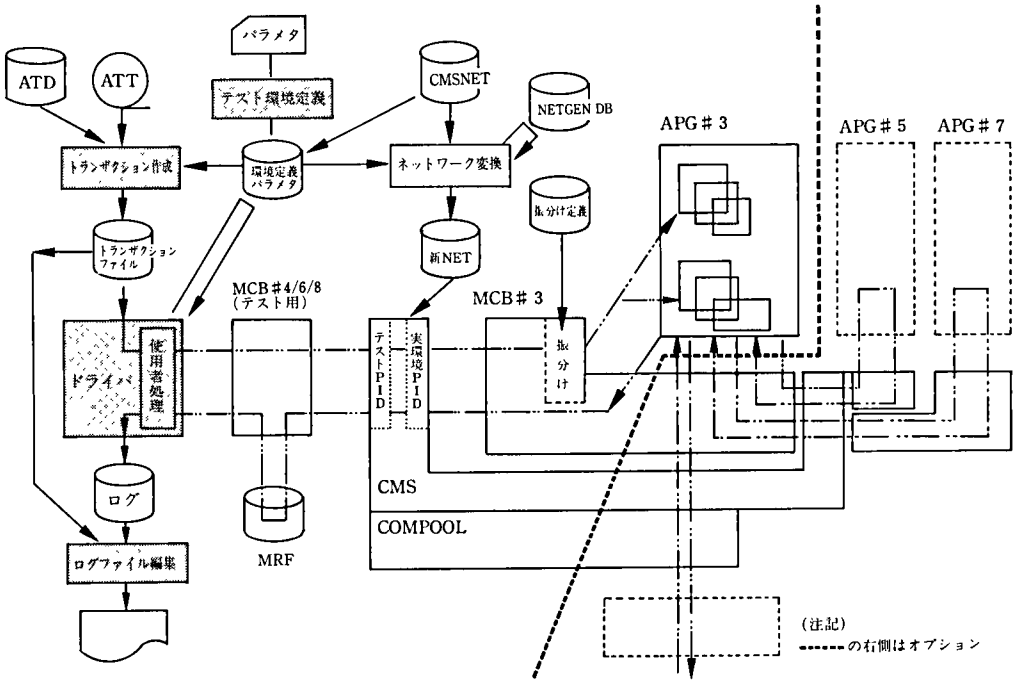
できる．TRITONでは，この保存ファイルを使用して，トレース情報の必要部分のみ印書するツールを開発し，デバッグを効率よく行っている．なお，トレースの設定や解除は，XISのコマンド・インタフェースを使用するため，本番と開発で使用者プログラムの変更は不要である．設定，解除はTPSであれば，ICP，業務制御，端末単位であり，バッチはJCLでの指定となる．

トレース編集の機能は次の通りであり，デバッグ時の生産性向上に役立っている．

1) データベース，テーブルのトレース情報は各レコードの登録集解析情報を使用してデータ定義単位に編集印書できる．
2) トレース情報はいったんトレース・ファイルに出力され，別途必要な情報のみ印書する機能および更新前後で変更項目に印を付加する機能を持っている．
3) モジュール間の渡りを追跡するために，モジュール呼出時にトレースを付加する機能を持っている．

3.5.3 ラッシュ・テストツール

システム全体に関わる環境変更のテストは，多量トランザクションを必要とするが，人手による端末入力では時間的にもコスト的にも不可能である．TRITONでは，実端末を使用しないで大量のトランザクションを発生させ，システム更改時にオンライン・プログラムの結合テストができるツールを開発した．全体像を図7に示す．



ATT：Audit Trail Tape
(ATD：Audit Trail Disk)
APG：APPlication Group
CMS：Communication Management System
MCB：Message Control Bank
MRF：Message Retention File
COMPOOL：COMMon data Pool

図7　ラッシュ・テストツール
Fig. 7　Rush test tool

1) テストデータの作成……TRITON は，ある日のオンライン・データを ATD* より抽出し，ラッシュ・テスト用の入力電文を作成するユーティリティを開発した．

2) テスト実行……XIS は，上記テストデータを使用して被テストシステムとのメッセージ送受信を行い，端末の入出力をシミュレートするツールを持っている．なお，ドライバは実行時パラメータにより一定時間内のトランザクション発生件数を指定できる．また問い合せ応答型取引について入力順序の保証を行っている．

3) テスト結果検証……テスト結果検証として，実行後のデータベースやテーブルの更新結果の正当性を確認する必要がある．TRITON では，新旧マッチング・ユーティリティを開発し，旧システムと新システムでのデータベースおよびテーブルの更新結果をマッチングすることにより検証している．

4. おわりに

本稿では，バンキング・システムにおけるシステム基盤を記述しているが，XTPA，他系インタフェースについては，各々「XTPA に基づくノーダウン・システム」，「アプリケーション連動の仕組み」として別稿で詳細が記述されているため，ここでは省略した．

XIS は，ここに掲げたシステム基盤に対して，従来のメーカ標準ソフトウェアの範囲から一歩ユーザよりに踏み込んで作られている．とくに，構造化 FCSS ファイル・インタフェース機能は，為替システムの構築に寄与したとの評価を受けている．また，ファイルの動的拡張機能は，ファイル容量オーバによるシステム停止を回避できている．さらに，センタカット，ディレード・バッチは，銀行システムでの運用を含めた考慮があり，ユーザの開発負担を少なくしている．

しかし，ターミナル・インディペンデント化については従来より進んではいるものの，編集パラメータの仮想化機能のより一層の推進等，今後の課題として残している．今後 TRITON の使用ユーザが増えていく中でシステム基盤として確立していくべき項目である．

最後に，TRITON のシステム基盤構築にあたり協力していただいた弊社システム企画開発二部の方々に感謝の意を表する．

---

* ATD (Audit Trail Disk)：データベースの更新イメージ，トランザクション等を保存するファイルであり，リカバリ時の入力となる．ここでは，入力電文イメージの抽出に使用する．

執筆者紹介 宮　村　　洋 (Hiroshi Miyamura)
昭和 28 年生．51 年名古屋大学工学部応用物理学科卒業．同年日本ユニシス(株)入社．地方銀行の SE サービスに従事．現在，金融システム企画開発本部 TRITON システム部 2 課に所属．

IOF/WORKによるバッチ運用

IOF/WORK-based Batch-processing Operation

後　博

要　約　バッチシステムの運行制御パッケージは，省力化・自動化を目的として提供されてきた．しかしながら，今日の運行制御パッケージは従来と考え方が変わりつつある．TRITONシステムのバッチ運行は，それを運営する運用部門にとって，非常に難物であると推測された．それは，TRITONシステムが大規模であること，共同開発であること，運用部門が開発に参画していないこと，バッチプログラムがすべて新規であり実績がないこと，さらには，XTPA（疎結合分散処理）によるバッチ処理の分散等，運用部門にとっては不安材料が山積し，しかも運行制御を司るパッケージ(IOF/WORK)が開発途上にあったことに起因している．

運用部門のこのような不安は，本番前後の4か月の原動力に置換され，IOF/WORKに対する積極的な改善支援の実施とともに，バッチ運行ネットワークの連続改善を実施して，今日の安定稼働に至っている．

現時点では，運用部門の担当者は，「IOF/WORKが提供されなかった場合，いまだにデイリー運用が安定していなかったであろう」とIOF/WORKを高く評価している．IOF/WORKがこのように評価される理由はどこにあるのであろうか．運行制御の真の目的あるいは機能が変わりつつあるのであろうか．運行制御パッケージのあり方について利用者ならびに主管部門の方に問いたい．

本稿では，IOF/WORKの概要，TRITONバッチ運用の特徴，TRITONバッチ運用の概要，および現在抱えている問題点と今後の課題について記述している．

Abstract　Workflow management packages for batch-processing systems have so far been used primarily to save manpower and to trigger automation moves. Today, however, the way is apparently changing of viewing their existence. The batch-processing-type operation in the TRITON environment was considered to be a great headache to those who would have to take care of the system's operation, The reasons for that include (1) the TRITON system being a very large-scale application system, (2) it being a joint development project, (3) no people from computer systems operation departments having pitched in for the development, (4) their having had almost no experience with batch-processing programs, and (5) TRITON's batch application being for loosely-coupled (XTPA-based) distributed processing, all combined to offer pleny of problems that worried systems operation people. In addition, their anxiety multiplied when they found that the workflow management package dedicated for the system would be IOF/WORK, which was still then in the making.

The replacement of such anxiety with a positive, continued four-month struggle, before and after the production run, to improve the IOF/WORK package and to realign the batch-processing networking has successfully led to today's stable operation of the system. Now, the systems operation staffs at user sites set a high value on IOF/WORK, even saying that their present day-to-day operation would not be made possible yet if it were not for the package. What on earth makes them welcome IOF/WORK with open arms like this? Are the real objectives and required functions of workflow management just changing?

The author wishes both computer systems users and operation staffs to stop and think what such packages should be like.

This paper is intended to present a profile of IOF/WORK along with an overview and the unique characteristics of a TRITON batch application system, besides mentioning what the associated current issues are about and what related problems are still left unsolved.

1. はじめに

TRITONのバッチ運行については，システムが大規模であるため自動運行が必須となるが，従来の運用システム（COSTAR；Computer Operating Management System for Total Automation & Reliance）では，TRITONの特徴の一つであるXTPAによるバッチ処理の分散（4ホスト）を制御できなかった．そこで，平成4年1月に開発途上の統合運用システム（IOF；Integrated Operating Facility）のサブシステムであるIOF/WORKを採用することに決定し，(株)百五銀行殿(以下百五銀行と略記)および(株)紀陽銀行殿(以下紀陽銀行と略記)の合意を得た．平成4年3月より両行のバッチ運用責任者と弊社で，TRITONバッチの運用方法，IOF/WORKの機能調査，追加機能の検討，ネーミング規約の設定，およびTRITONとしての運用支援機能の検討等を協議し，同年11月に最終運用が決着した．

本稿では，2章で運行システムの中核となるIOF/WORKの概要について紹介した上で，3章でバッチ運用の検討過程で発生した種々の課題に対してTRITONでいかなる工夫・考慮を図ったかを解説し，4章ではその結果として実現されたTRITONにおける実運用について記述する．

2. IOF/WORKの概要

2.1 IOF/WORKの管理項目

IOF/WORKは，バッチ業務処理を制御するためにワーク/ジョブ/プロシジャの3

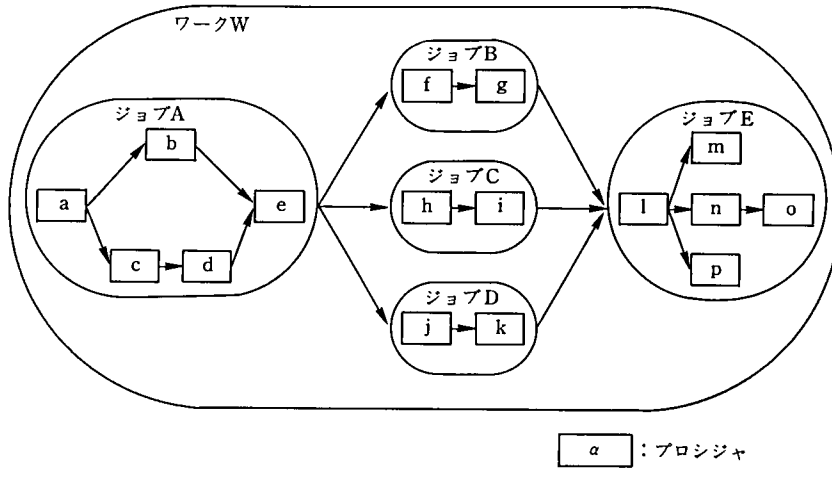


図1 IOF/WORKの管理項目

Fig. 1 Management item of IOF/WORK

階層の管理項目を用意している．

ワークは，たとえば勘定系システムの月次処理というような，業務処理内で意味のあるひとまとまりの処理単位の管理項目であり，スケジュールの作成の単位として使用する．また，ワークはジョブの集合体であり，複数の処理日と複数の実行ホストにまたがることが可能である．ジョブは，特定の処理日/特定のホストで走行する管理項目であり，プロシジャの集合体である．プロシジャは，一つまたは複数のプログラムのかたまりで IOF/WORK の最小の管理項目であり，リランはプロシジャ単位で可能である．ランは，ジョブの構成要素であり，また，プロシジャの集合体でもある．

ネットワークは，ジョブ間ネットワークとジョブ内のラン間ネットワークの設定が可能である．

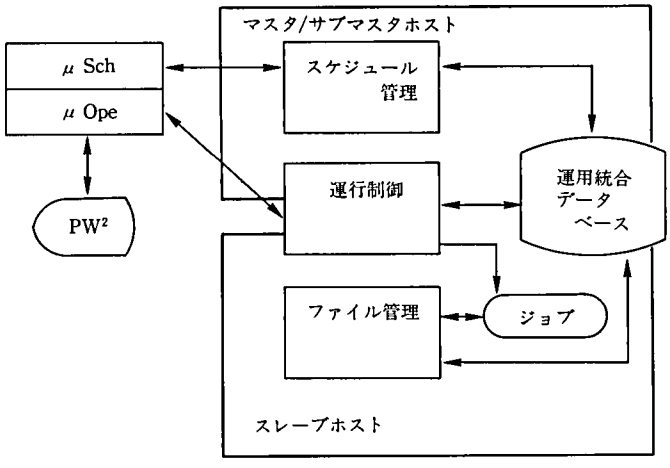
ワーク/ジョブ/プロシジャの関係を図1に示す．

2.2 IOF/WORK の機能

IOF/WORK は，分散ホスト環境下におけるバッチ業務の統合管理を行うソフトウェアであり(図2)，バッチ業務のスケジュール作成と運行制御に関し，以下の機能を提供している．

1) スケジュールの作成・維持

① すべてのホストの業務スケジュールを単一の制御点（ワークステーション）から集中作成・維持・管理することが可能である．



μ Sch　：スケジュール検証画面群の総称で，スケジュール情報の検索や補正および確定等スケジュール関連作業を実施することが可能である．
μ Ope　：運行制御コンソール画面群の総称で，ジョブランの進捗状況の監視やリラン等当日スケジュール業務のオペレーションに関する作業を実施することが可能である．
マスタホスト：集中スケジュール管理が可能なホストであり，各ホストのスケジュールを集中作成し，各運行制御ホストに配布する．
サブマスタホスト：統合運行制御のジョブ制御が可能なホストであり，ジョブの起動制御と進捗管理を実施する．
スレーブホスト：統合運行制御のラン起動制御が可能なホストであり，スケジュールランの起動制御と進捗管理を実施する．

図 2　IOF/WORK のシステム構成

Fig. 2　System configuration of IOF/WORK

・複数ホストのスケジュールの一括作成

② 運用管理部門が直接スケジュールの作成を可能にするための簡易，かつ安全なインタフェースを用意している．

・スケジュールの作成・検索・変更に対するグラフィカルなヒューマン・インタフェースの提供

・スケジュール関連作業の並行処理が可能

2) スケジュール・ジョブの自動起動

① すべてのホストのスケジュール・ジョブの進捗状況を単一の制御点（ワークステーション）から集中監視・制御することが可能である．

・グラフィカルなインタフェースによるジョブ走行状況の表示・制御

② XTPA を構成する各ホスト・システムの統合制御が可能である．

・ホストをまたがったジョブ・ネットワークの起動制御

③ 24 時間，365 日連続運転への対応がされている．

・バッチ連続運転下における中断のない起動制御の実現

④ バラエティに富んだスケジュールが可能である．

・スケジュール関数によるスケジュール条件の指定

・スケジュール条件をパターンとして IOF／WORK に登録することが可能

3) ジョブ使用テープ・ファイルの管理

① ファイルのライフサイクルの管理が可能である．

・メンバシップ（世代管理）ファイルの管理

② ファイルの自動割当てと媒体管理が可能である

・プール方式* による媒体（テープ・リール）割当制御

3. TRITON バッチ運用の基本的な考え方

バッチ運用の検討過程で以下のような種々の課題が発生した．

・バッチは，特定ホストで稼働させるのか．複数ホストで稼働させるのか．

・各ホストで稼働させると，磁気テープの要求メッセージ等の監視のために，各ホストのシステムコンソールを監視しなければならないが，これを簡素化できないか．

・ディレードバッチ，センタカット等，リアルとの接点となる部分について手動による操作が一部残っているが，これを自動化できないか．

・同一ファイルを複数ランが参照するような処理があるが，これらを自動化できないか．

・IOF/WORK の運用を容易にするために支援プログラムが必要ではないか．

本章では，これら課題に TRITON としてどのような対応を行ったかを記述する．

3.1 ホスト使用形態

TRITON の基本ホスト構成では，

・ホスト#A，#B は，大量単純処理の勘定系オンライン，センタカット処理

---

* プール方式：新世代（出力）作成時，リールをプールから新規に割当て，最古世代で使用中であったリールはプールに解放する方式．

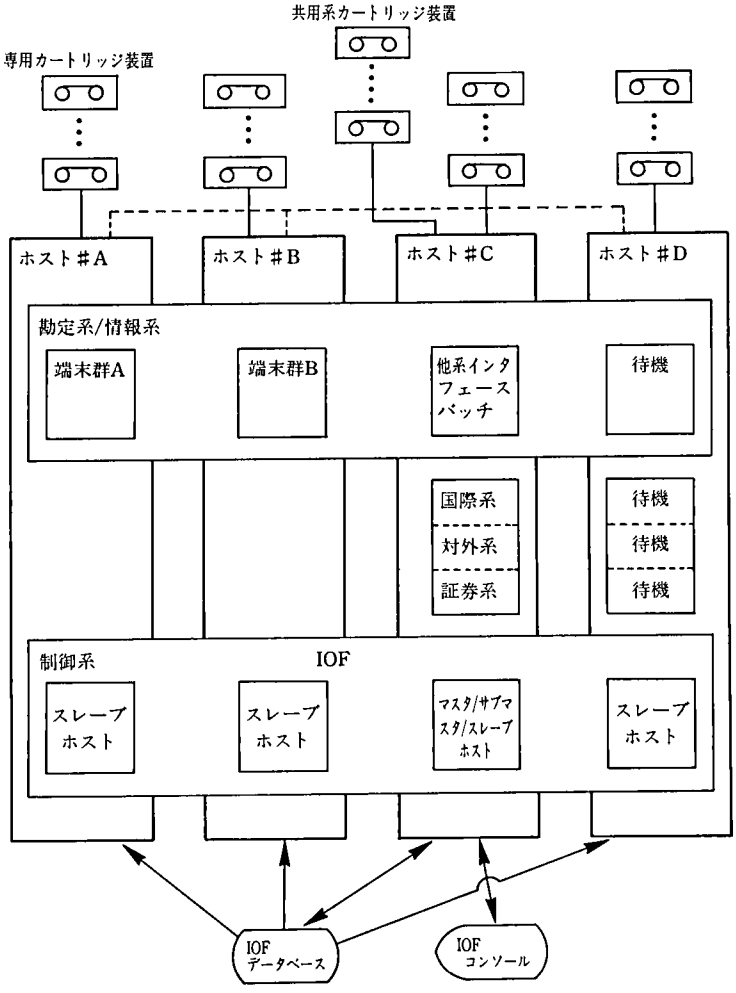


図 3　ホスト構成例（平日）

Fig. 3　Host configuration model (weekday)

・ホスト#C は，小量複雑処理の勘定系バッチ，情報系，他系処理

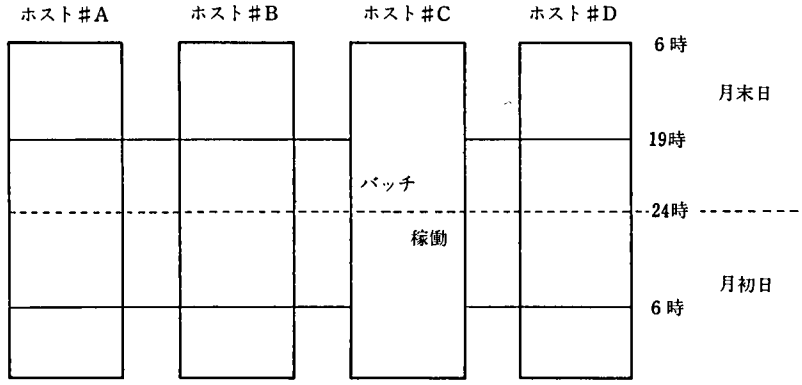
・ホスト#D は，待機およびテスト

というホスト使用形態としている．これに伴い，テープ装置も各ホスト（ATD(Audit Trail Disc) のアーカイブ，およびバッチ用）ごとに配置し，バッチ専用のテープ装置をホスト#C に接続している．また，IOF/WORK のマスタホストとサブマスタホストもホスト#C に設定している．

図 3 に基本的なホスト構成例を，また図 4 に月末/月初日の時間帯別ホスト稼働例を示す．

3.2　磁気テープオペレーション

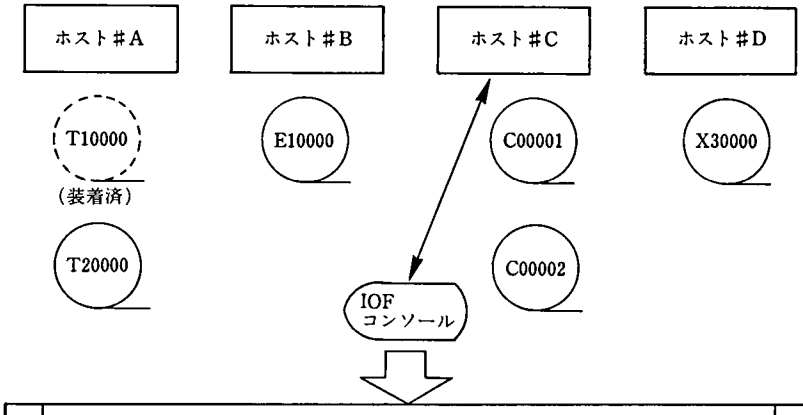
各ホストでバッチを稼働させると，磁気テープの掛け替え操作のために操作員が各ホストのシステムコンソールを監視しなければならず，操作負荷増となってしまう．TRITON では IOF の集中監視機能を使用し，次のように 1 台の IOF コンソールで監



ホスト#Cによるバッチ稼働が基本であるが，
・業中のホスト#A，#Bの余力の有効利用として抽出バッチを実行
・月次バッチの処理時間短縮のため，夜間の全ホスト使用
等，現実的には各ホストでバッチを稼働させている．

図 4　月末/月初日のホスト稼働例

Fig. 4　Host work model between the end of month and the start of month



イベント表示画面

履歴要求　履歴表示　履歴ファイル削除　運用終了

イベント番号	ホスト名	時刻	イベントメッセージ	対応
0004	C号機	09：05	LOAD [C00001]　ファイル名/ラン名	未対応
0020	B号機	09：10	LOAD [E10000]　ファイル名/ラン名	未対応
0021	C号機	09：11	ラン [ABCDE] 異常終了	対応中
0025	A号機	09：15	LOAD [T20000]　ファイル名/ラン名	未対応
0026	D号機	09：16	LOAD [X30000]　ファイル名/ラン名	未対応
0030	C号機	09：16	LOAD [C00002]　ファイル名/ラン名	未対応

図 5　磁気テープのオペレーション

Fig. 5　Operation of magnetic tape

視が可能なようにした.

・全ホストの磁気テープに対する要求メッセージ（ロードのメッセージ）を1台のIOFコンソールに出力し，磁気テープの装着が完了すると，要求メッセージを画面から自動消去する.

図5に詳細を示す.

3.3 自動化の推進

従来からバッチ運用の自動化は行われているが，ディレードバッチ，センタカット等，リアルとの接点となる部分については手動による操作が一部残っていた.

TRITONではIOF/WORKのメッセージ条件の機能を使用し，これらの部分に対し更なる自動化を行った.

メッセージ条件とは，各ジョブ単位に指定可能なジョブの起動条件の一つで，登録されているメッセージが発行されるまでIOF/WORKによって該ジョブの起動が待たされる．メッセージ発行はバッチユーティリティおよびプログラム内から可能である．また，各ジョブに複数個のメッセージ条件が設定可能で，ホスト間にまたがった制御も可能である.

(例)メッセージ待ちジョブの実行ホスト——ホスト#A

メッセージ発行ジョブの実行ホスト——ホスト#B

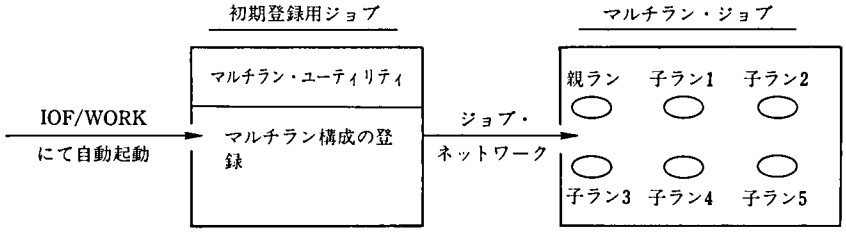


図6　マルチランの初期走行

Fig. 6　First run of multi-run

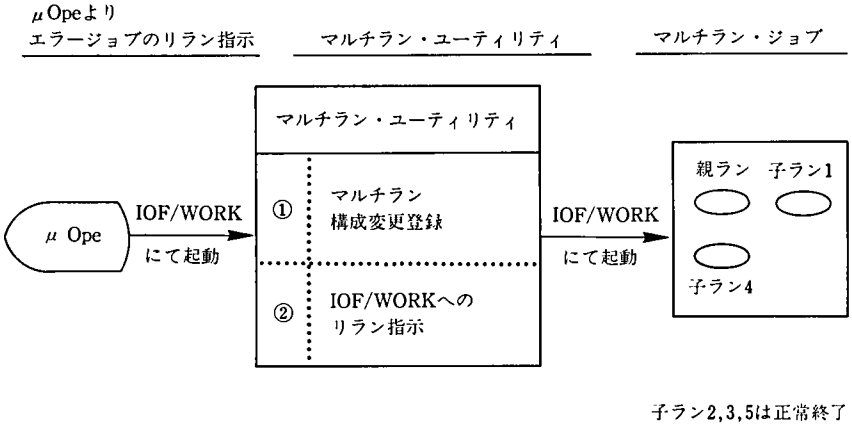


図7　マルチランのリラン走行

Fig. 7　Rerun of multi-run

3.4 マルチラン

TRITONでは，同一ファイルを複数ランが同時に読み込むことによる各ランの処理効率の低下とI/Oの負荷を防ぐため，マルチランを採用した．マルチランとは，「一個のラン（親ラン）がファイルを読込み，メモリ上にそのデータを保持し，その他のラン（子ラン）はメモリ経由でデータを入手する」機能である．

マルチランの場合，親ランと子ラン（複数）をセットで起動する機能が，またリラン時は親ランとある子ランのみをセットで起動するという機能が必要である．この機能をTRITONおよびIOF/WORKに取入れ，マルチランの自動走行を実現した．

マルチランの初期走行とリラン走行の方法を図6，図7に示す．

3.5 リ カ バ リ

バッチはホスト#Cで稼働しているが，ホスト障害の場合はホストを切り換える必要があり，通常はホスト#Dに切り換えて運用を続ける．しかし，早急にバッチジョブを処理させるために，より稼働率の低いホストで実行したい等の理由により，特定のジョブの稼働ホストの変更が必要な場合がある．

IOF/WORKでは，基本機能としてジョブの実行ホストを変更できる機能がある．この機能を使用し，ホスト障害の場合には障害ホストで実行予定であったジョブを一括して代替ホストに自動変更し，また特定ジョブの稼働ホスト変更はIOFコンソールからのオペレーションで実施している．

3.6 ファイルのバックアップ

ファイルのバックアップ機能（バックアップ・テープの管理とファイル修復機能）は，IOFのサブシステムであるCOSTAR/MSSが提供している．そこで，TRITONではバックアップジョブとバックアップテープの管理を自動管理の対象とするため，次の運用を行っている．

データベースのバックアップ・ランはIOF/WORKにジョブとして登録・自動化し，バックアップテープの出庫とバックアップテープの管理はCOSTAR/MSSで行う．

3.7 TRITONの運用支援プログラム

TRITONでは，運用をより容易にするため，表1のプログラムを作成した．マスタ登録ツールは，登録パラメタの種類と登録パラメタ量の削減を目的に作成した．入力画面の集約化（IOF/WORKの登録パラメタ種類の約半数）は，TRITONディフォルトの設定によるパラメタの自動生成等により該目的を実現した．

表1 TRITONの運用支援プログラム一覧
Table 1 TRITON operation support program list

標準プログラム	処 理 概 要
①マスタ登録ツール	IOF/WORK登録パラメタの簡易入力ツール ・画面入力→MAPPER D/B登録→登録パラメタ作成
②ジョブ間ネットワーク図作成	作成済スケジュールファイルを入力として指定された日付のジョブ間ネットワーク図を作成
③ラン間ネットワーク図	作成済スケジュールファイルを入力として指定された日付のラン間ネットワーク図を作成
④マルチラン用ユーティリティ	初期走行およびリラン時のマルチラン登録パラメタを自動生成する

ただし，マスタ登録ツールについては，将来 IOF 提供の画面ツールを使用予定である．

4. TRITON におけるバッチ運用

前述の基本的な考え方に基づき実際にどのような運用を行っているかについて述べる．

4.1 一日の流れ

朝のイニシャルセットアップからバッチ業務の処理およびターミネーションまでの流れは図 8 の通りである．

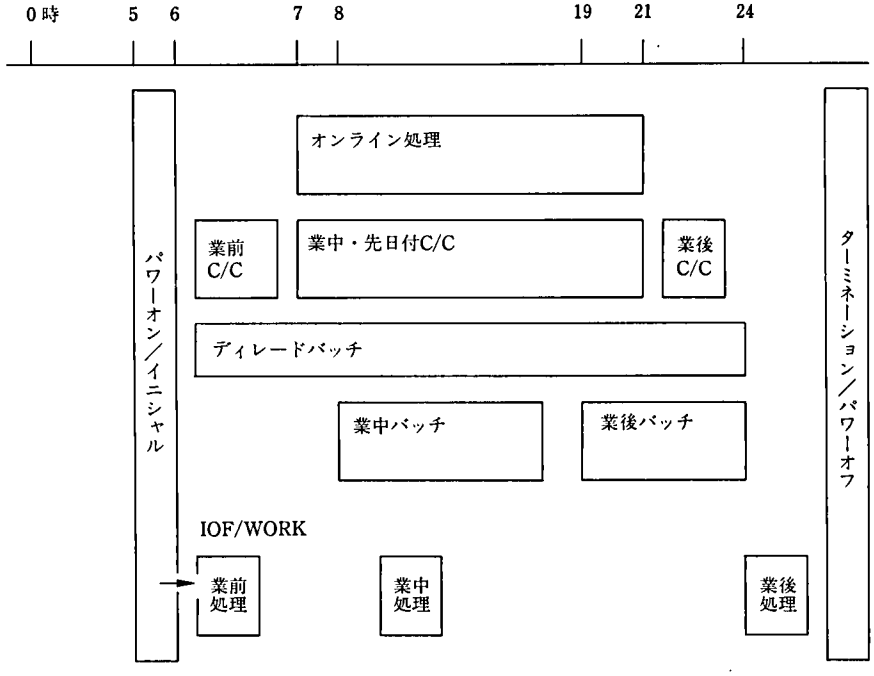


図 8　一日の流れ

Fig. 8　Flow of a day

4.1.1 IOF/WORK の処理

業前処理，業中処理および業後処理は表 2 の通りである．運行制御の開始処理（業前処理）は，自動セットアップ処理に組み込まれている．業中処理は IOF/WORK の機能制限により手動実行してきたが，現在は制限が解除されている．そこで今後両行とも業中処理は IOF/WORK にジョブとして登録し自動化する予定である．

4.1.2 ディレード・バッチランとの連動

ディレード・バッチランの管理方法は次の通りである．

① リアルの開始時より稼働するランはセットアップ処理より自動起動する．

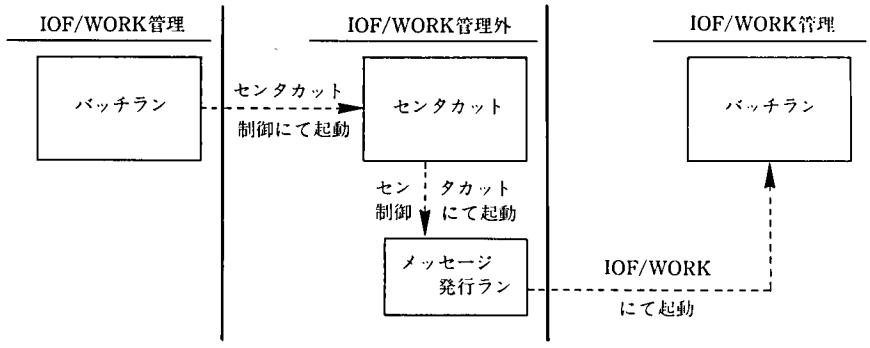
② リアル中に起動されるランは IOF/WORK 管理とする．

トランザクション・ファイルを入力する後続バッチランは，メッセージ条件の機能を使用し，ランの自動起動を行う．

表2　業前処理，業中処理および業後処理

Table 2　Set up and production and termination process

	処　理	説　　明
業前処理	運行制御の開始処理	当日分のジョブを起動可能状態にする．
	ワークステーションの立ち上げ	・端末の立ち上げ操作 ・μOpe 画面より起動処理開始の指示を行う．
業中処理	翌日準備処理	確定スケジュールファイルから翌稼働日用のスケジュールを抽出し，翌日スケジュールファイルを作成する．
	消込みと実績保存	正常終了したジョブを当日スケジュールファイルから削除し，再処理に備えて運行実績を実績ファイルに保存する．
	期限切れメンバの削除	保存期限が過ぎたテープファイルの削除を行う．
業後処理	運行制御終了	起動処理終了の指示をワークステーションより行う．
	日の切り替え	IOF/WORK で管理されている当日日付を1日進めるとともに，スケジュールファイルを翌日用に切り替える．



百五銀行は，ディレードバッチと同様に後続ランキック用ダミーランで後続バッチランを起動している．

図 9　センタカット・システムとの連動方法

Fig. 9　Connection method with CENTER-CUT system

① ディレード・バッチラン終了後，後続バッチランが走行する場合
→ディレード・バッチランの最後にメッセージ条件発行用の IOF のユーティリティを挿入する．

② ディレード・バッチラン稼働中に，後続バッチランが走行する場合
→ディレード・バッチランのプログラム内で，メッセージ条件発行用のコンソールメッセージを呼び出す（紀陽銀行の場合）．
→後続ランキック用ダミーランで後続ランを起動する（百五銀行の場合）．

4.1.3　センタカットとの連動

センタカット関連のバッチランを自動運行の対象とするため，次の対応を行っている．

センタカット・スケジューラとバッチとの連動は次の通りである．

① バッチ　→　センタカットのインタフェース

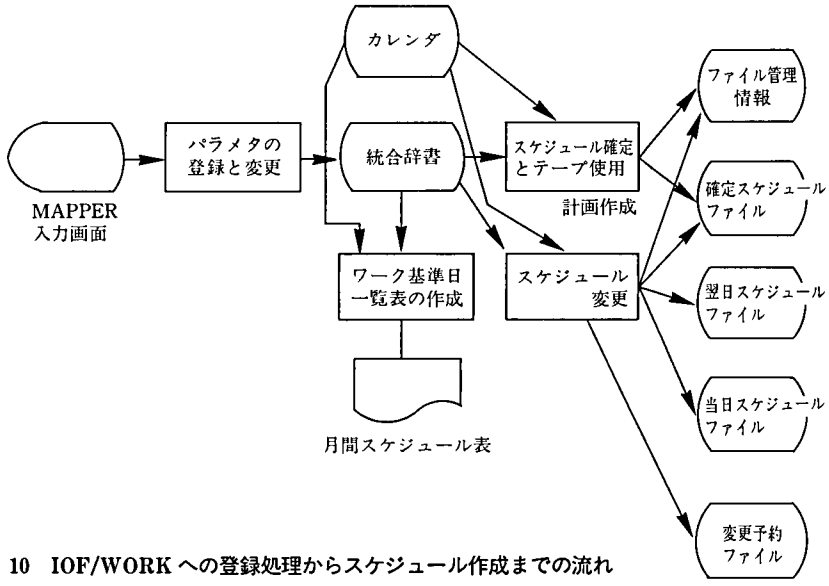


図 10　IOF/WORK への登録処理からスケジュール作成までの流れ

Fig. 10　Flow between IOF/WORK entry process and schedule make

・センタカット制御にて自動起動される．

② センタカット　→　バッチのインタフェース

・バッチランはメッセージ条件で起動待ちとする．

・後続バッチを起動するセンタカットは，センタカット統合管理ファイルにメッセージ発行ラン（IOF/WORK 管理外）を登録する．

上記関連を図 9 に示す

4.2　バッチ業務の登録/変更からスケジュール作成/検証の流れ

新規業務の追加や業務の変更による「IOF/WORK への登録処理からスケジュール作成までの流れ」は図 10 の通りである．

各処理の概要と処理サイクルは表 3 の通りであり，すべての処理は業中に実行可能である．現在，スケジュール確定とテープ使用計画作成の処理は，随時のパラメタ変更に対する確認および各々の処理の正当性確認のため自動化されていないが，パラメタ変更処理の減少および運用システム全体の安定化により今後両行とも該処理も IOF/WORK にジョブとして登録し自動化する予定である．

5. お わ り に

IOF/WORK は，ワークステーションから複数ホストのランの状況監視，およびリラン指示が可能であり，またプログラム単位のリランができる等，進捗管理，運行制御が優れていることで，客先からは運用負荷が非常に軽減されたと評価されている．

本番開始以降，大きなトラブルもなく安定稼働が続いているが，パフォーマンス改善(システムに対する負荷削減および処理時間の短縮)，稼働実績を反映したネットワークの自動再構築(最適化)，XTPA 環境下での各種ファシリティを考慮した運行制御(4 ホストの自動負荷分散)，業後処理の自動化，ジョブの遅延監視機能の採用による監

表3　各処理の概要と処理サイクル
Table 3　Outline of the process and process cycle

処　理	説　　明	処理サイクル (TRITON の運用)
パラメタの登録と変更	MAPPER 入力画面より，パラメタの登録・修正・削除を行うことにより IOF/WORK のデータベースに反映される．	随時 (必要な都度)
ワーク基準日一覧表の作成	指定期間にスケジュール対象のワークジョブとその個々の基準日を示すワーク基準日リストが作成される．月間スケジュール表として使用できる．	随時 (必要な都度)
スケジュール確定とテープ使用計画作成	指定された日(複数日が可能)のジョブ，ジョブ間の前後関係，および各ジョブで使用されるテープを決定する．	毎営業日ごとに翌々営業日分のスケジュールを作成
スケジュール変更	バッチ業務の走行予定を変更することでスケジュール確定日前後とも可能である． ワークに対する機能 ・ワークの追加/取消/スケジュール日の移動 ジョブに対する機能 ・ジョブの追加/削除 ・ジョブの起動条件の変更 　・ジョブの実行日付 　・起動可能時刻の設定/取消 　・実行予定ホスト 　・自動起動停止の設定/取消 ・メッセージ条件の追加/削除 ・ジョブネットワークの追加/削除 ・処理済みジョブの強制削除→ジョブの実績とファイル作成実績が削除される．	随時 (必要な都度)

視強化等々，改善・改良しなければならない課題は多岐にわたっている．

また，バッチ処理時間短縮の観点からは，ジョブやランの構成やネットワーク構造等にも手をつけなければならないと思われる．

バッチ運用は一朝一夕にでき上がるものではなく，その改善は客先運用部門ならびに我々コンピュータメーカが協力して地道に取り組んでいくべきテーマであると考える．

本稿の執筆にあたり，百五銀行殿，紀陽銀行殿には原稿の査読をお願いし，多くの指摘を頂いた．ここで厚く感謝の意を表したい．

執筆者紹介　後　　博 (Hiroshi Ushiro)
昭和46年静岡大学理学部数学科卒業．同年日本ユニシス(株)入社．金融系ユーザの SE サービス業務を担当．現在，TRITON システム部に所属．

アプリケーション連動の仕組み

Applications Linkage

黒　川　　茂

要　約　TRITONは新しいシステム・インフラストラクチャである，XTPA(eXtended Transaction Processing Architecture；拡張トランザクション処理アーキテクチャ）をベースとした複数ホスト構成による疎結合分散処理システムで，ソフトウェア基盤としてXIS(eXtended Information System)をベースに構築している．金融機関の業務は勘定系を中心に，情報系・国際系・対外系・証券系等多種多様なアプリケーションから成り立ち，そのアプリケーション間での様々なデータの授受（以降アプリケーション連動と呼ぶ）が必要とされる．TRITONでは，従来のアプリケーション連動の考慮点に加え，複数ホスト構成による疎結合分散処理システムである点，各アプリケーションのソフトウェア基盤が異なる点等，新たな考慮が必要とされた．TRITONでは，XISの系間インタフェース機能をベースに，使用者にホストの違い，ソフトウェア基盤の違いを意識させないアプリケーション連動の仕組みを実現した．

Abstract　The TRITON system, based on the new system infrastructure: the eXtended Transaction Processing Architecture (XTPA), is a loosely-coupled distributed computing system consisting of multiple hosts, with XIS adopted for its basic software. Financial institutions have varying computer applications, dedicated for accounting (as their primary application), specific information processing, foreign exchange, and electronics inter-bank transactions as well as securities, resulting in the required exchange of date between applications (hereafter referred to as applications linkage). In addition to improving traditional applications linkage, the creators of the TRITON system had to contend with new challenges because the system is designed to be a loosely-coupled distributed system on a multi-host basis, and because software bases differ from application to application. With the full use of XIS's facility, the TRITON system has paved the way for the form of applications linkage that keeps users from being aware of the difference of hosts and software bases.

1. はじめに

銀行システムにおけるアプリケーション連動には，①系内連動（定期預金の利息を普通預金に入金するといった同一系内の連動処理），②系間連動(勘定系の窓議処理において，情報系のデータをアクセスするといった連動処理）の2種類の連動処理が存在する．TRITONでは，前者をHVTIP (High Volume Transaction Interface Package) によるモジュール連動にて実現し，後者をXIS (eXtended Information System)を中心とした系間インタフェースで実現している．本稿では後者の系間インタフェースについてTRITONでどのように実現したかを述べる．

2. アプリケーション分割

銀行システムは勘定系を中心に，情報系・国際系・対外系・証券系等，多種多様な

アプリケーションから成り立ち，そのアプリケーション間でさまざまなデータの授受を行っている．一般的にそれらのアプリケーションは，重要性・運用形態の相違等からアプリケーション分割されている．勘定系パッケージである TRITON も以下の必要性からアプリケーションを分割した．

2.1 アプリケーション分割の必要性

1) アプリケーションの運用形態からの必要性……勘定系は大量単純処理，情報系は小量複雑処理，対外系は外部センタとの接続等，アプリケーションにより処理形態が異なっている．また国際系は IBS (International Business System；国際系統合パッケージ)，対外系は UNITE (Unisys Total Electronic banking system；対外系統合パッケージ)，証券系は有価証券総合パッケージとそれぞれユニシス 2200・1100 シリーズで稼働するパッケージが用意されている．処理形態・運用時間等の運用形態が異なるアプリケーションを分割することにより，アプリケーションの開始・終了を独立して行うことができる等，弾力性のある運用を図ることができる．

2) 障害対応からの必要性……銀行にとって，勘定系処理の障害回避はいまや社会的責任ともいえる．したがって XTPA によるノーダウンを図るとともに，他のアプリケーションの障害による影響を受けないシステム構成とすべきである．

3) ソフトウェア基盤の相違と有効利用からの必要性……勘定系・情報系は XIS，国際系は AIS (Advanced Information System：統合オンライン支援システム)，対外系・証券系は BOSS-11 (Banking Oriented Support System under OS 1100；リアルタイム・システム構築支援プログラム) とソフトウェア基盤が異なる．また国際系・対外系・証券系はパッケージ化されている．ユニシス 2200・1100 シリーズでは，従来のソフトウェアがそのまま稼働できることもあり，資産の有効利用という観点でアプリケーションを分割している．

2.2 アプリケーションと APG

TRITON では，上記のアプリケーションの分割必要性の観点から，各アプリケーションを OS のリカバリ単位で，開始・終了等の運用単位でもある APG (Application Group；OS が管理するリカバリの単位) に分割し，表 1 のように対応させている (本特集号別稿“XTPA に基づくノーダウン・システム”参照)．

表 1 アプリケーションと APG
Table 1 Application and APG

アプリケーション	ソフトウェア基盤	APG 番号	APG 種類
勘定系	XIS	APG 3	コンカレント APG
情報系	XIS	APG 5	コンカレント APG
国際系	AIS 1100 II	APG 7	ローカル APG
対外系	BOSS-11	APG 0	—
証券系	BOSS-11	APG 0	—

コンカレント APG：複数ホストでデータベースを共用しながら処理を行う APG 形態
ローカル APG：単一ホストで構築される APG 形態

勘定系は大量トランザクション処理・ノーダウン化という観点からコンカレントAPG化する．情報系の今後の拡大はめざましく，近い将来勘定系のようにデータ量が増加すること，また勘定系と情報系の負荷のピークがずれていることもあり，負荷分散という観点からコンカレントAPG化している．また国際系はソフトウェア基盤がAIS 1100 IIであることからローカルAPGとなる．

3. アプリケーション連動

3.1 アプリケーション連動に対するニーズと対応

プログラムの開発・テストを行う場合，他のアプリケーションにデータを引き渡す際，他のアプリケーションのホスト/APGが異なる，相手システムのソフトウェア基盤，手順等を意識することは，相手アプリケーションごとにインタフェースを変更する等の考慮が必要となり，開発・テストの生産性を損なうことになりかねない．使用者から見たアプリケーション連動に対するニーズとTRITONの対応を以下に記述する．

1) アプリケーション（系）を意識しないインタフェース……使用者プログラムからすると，他のアプリケーションとデータの受渡しを行う場合，他のアプリケーションが存在するホスト・APGを意識することなしに，同一アプリケーション内でのデータを受渡しを行う場合と同様なインタフェースが望まれる．TRITONではアプリケーション・インタフェースとして，後述する会話インタフェース，電文送受信インタフェースを作成している．

2) プログラム開発・テストの容易性……他のアプリケーションとのインタフェースを業務処理プログラムから独立させ，専用の共通モジュールを作成することにより，開発・テストの容易性を図っている．

3) 障害対応の容易性……APGを分割するということは，アプリケーションの独立性という観点では優れているが，アプリケーション連動にとっては，障害対応を複雑にする要因ともなる．TRITONでは，障害対応の容易性という観点からアプリケーション連動の原則を以下のように設定した．

① コンカレントAPG間のアプリケーション連動は同一ホスト内の連動とする．

② コンカレントAPGとローカルAPGのアプリケーション連動も同一ホスト内を基本とするが，端末取引で端末接続ホストとローカルAPGのホストが異なる場合は，ホストが異なる連動を行う．

③ 障害箇所の局所化という観点から，アプリケーション連動の閉塞機能を設ける．

この原則により，他ホストの障害の影響・障害管理の負荷が軽減される．

3.2 アプリケーション連動のソフトウェア

TRITONではXISの系間インタフェース機能を中心に，各種のアプリケーション連動を実現している．以下にアプリケーション連動のソフトウェアとその特徴を示す．

1) ACLES/DTP(Advanced Communication control eLEements and Support system/Distributed Transaction Processing)

従来の問い合わせ応答型，交換型といったトランザクション形態では，プログラム間で複数回データを送受信する処理の場合，使用者がデータの引き継ぎを考慮する必要があった．ACLES/DTP は，こうした使用者の負担を軽減するため，同一の使用者プログラム間で複数回相互にデータ転送を行える仕組みを提供する．これにより使用者プログラムは，自データバンクを継続して使用することが可能となる．

2) ACLES/ASCOT (Advanced Communication control eLEments and Support system/Ais System COnnecTion)

AIS システムの分散処理インタスェースである ASCOT 手順とインタフェースをとり，使用者プログラムは相手のプログラムがどこのホスト/APG に存在するかを一切意識することなく，処理を行うことができる．しかし，ACLES/DTP とは異なり，プログラム間で複数回データを送受信する場合，使用者がデータの引き継ぎを考慮する必要がある．

3) ACLES/ICCP (Advanced Communication control eLEments and Support system/Inter Computer Communication Program)

BOSS-11 のもとで分散処理を支援する ICCP 手順とインタフェースをとる．相手プログラムの存在位置・使用者データの引き継ぎ等は ACLES/ASCOT と同様である．また，相手手順が ASCOT 手順/ICCP 手順の違いを意識する必要なく処理を行うことができる．

4) ディレードバッチ

XIS のディレードバッチ機能は，従来の AIS 1100 II・BOSS-11 のディレードバッチ機能に加え，以下の機能が追加されており，使用者はホスト・APG の違いを意識することなく処理を行うことができる．

- ホスト/APG またがり機能
- 多段階トランザクション機能 (3.3.5 項参照)

5) TRITON 会話インタフェース

勘定系の処理の中で情報系・国際系等，複数のアプリケーションと連動する処理が存在する．情報系と国際系はソフトウェア基盤の相違から，インタフェースが異なっており，使用者プログラムにとっては好ましくない．そこで両インタフェースを同一とするような仕組みを TRITON で開発した．これにより，使用者プログラムは相手アプリケーションのソフトウェア基盤を意識せず，同一インタフェースで処理することができるばかりでなく，将来ソフトウェア基盤の変更があっても，使用者プログラムを変更する必要がなくなる．

6) ファイルシェア

MHFSF 1100-II* により，複数のホストより，同一の磁気ディスク装置上のファイルを共用でき，アプリケーション間のデータ受渡しがホストを意識することなく容易に行える．

---

* MHFSF 1100-II (Multi Host File Share Facility 1100-II)：複数の 1100 および 2200 シリーズシステムから同一の磁気ディスク装置上のファイルを共用する機能．

3.3 アプリケーション連動の仕組み

3.3.1 アプリケーション連動パターン

アプリケーション連動を行うソフトウェアを選択する場合，データ量・即時性等の特性を考慮して，使用するソフトウェアを選択する必要がある．表2，表3でTRITONの連動パターンとデータ量・即時性から見た特徴についてまとめる．

表2　データ量から見たTRITONアプリケーション連動パターン
Table 2　Data volume of TRITON application interface

			被連動				
			勘定系	情報系	国際系	対外系	証券系
連動	勘定系	リアル	—	小	小	中	—
		バッチ		大	小	大	小
	情報系	リアル	小	—	—	—	—
		バッチ	小		小	—	—
	国際系	リアル	小	—	—	—	—
		バッチ	小	中		—	—
	対外系	リアル	中	—	—	—	—
		バッチ	小	—	—		—
	証券系		小	小	—	—	—

表3　バッチデータの即時性から見たTRITONアプリケーション連動パターン
Table 3　Immediate transfer of batch data of TRITON application interface

		被連動				
		勘定系	情報系	国際系	対外系	証券系
連動	勘定系	—	要	不要	要	不要
	情報系	不要	—	不要	—	—
	国際系	不要	不要	—	—	—
	対外系	不要	不要	—	—	—
	証券系	不要	不要	—	—	—

表4　TRITONアプリケーション連動パターンとソフトウェア
Table 4　Software of TRITON application interface

			被連動				
			勘定系	情報系	国際系	対外系	証券系
連動	勘定系	リアル	—	①⑤	②⑤	③	—
		バッチ		④⑥	⑥	④⑥	⑥
	情報系	リアル	①⑤	—	—	—	—
		バッチ	⑥		⑥	—	—
	国際系	リアル	②⑤	—	—	—	—
		バッチ	⑥	⑥		—	—
	対外系	リアル	③	—	—	—	—
		バッチ	⑥	—	—		—
	証券系		⑥	⑥	—	—	—

これらの特徴にもとづいて，TRITON では各アプリケーション連動を表4に示すソフトウェアをベースに実現している（表中の数値は3.2節のソフトウェアの番号を示す）．

以下に各連動パターンの仕組みと特徴について説明する．

3.3.2 勘定系/情報系連動

本項では，勘定系と情報系の連動の仕組み，および特徴について説明する（図1）．

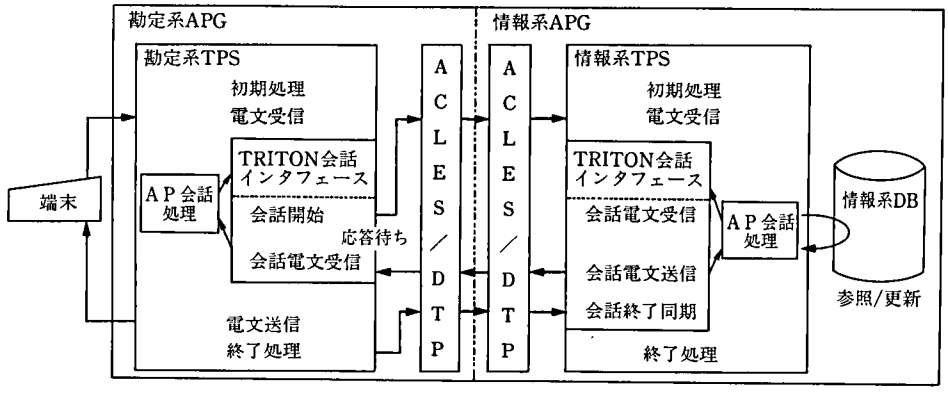


図 1 勘定系/情報系連動の仕組み

Fig. 1 Structure of accounting system and information system interface

1) 同一ホスト内でのデータ授受……勘定系と情報系のインタフェースは勘定系・情報系ともコンカレント APG であることから同一ホスト内でのデータ授受を前提とする．他ホストの情報系 APG と連動を行うことも可能であるが，この場合，勘定系と同一ホストの情報系 APG 障害時にもデータ授受が可能となる長所はあるが，他ホストの障害管理を行う必要性があり，障害対応の簡素化という観点から，同一ホスト内を前提とした．

2) TRITON 会話インタフェースの利用……勘定系と情報系のデータ授受を行う業務処理は，3.2項の5)で記述した，TRITON の会話インタフェースを使用して行う．同インタフェースは XIS の ACLES/DTP とのインタフェースのため，ACLES/DTP の特徴であるデータの引き継ぎを考慮することなくデータ授受を行うことができる．

3.3.3 勘定系/国際系連動

国際系は国際系専用端末を持たず，勘定系端末を使用する．したがって国際系の取引は勘定系経由となる．本項では，まず国際系取引について説明し(図2)，次に勘定系→国際系連動について(図3)，その後国際系→勘定系連動について(図4)説明する．

1) 国際系取引

① パススルー TPS の開発による整合性の保証

TRITON では，国際系の取引を勘定系経由で実現するため，ASCOT 手順による接続を中継する勘定系依頼 TPS（Transaction Processing Segment；トランザクション処理プログラム）・回答 TPS を作成している（これをパススル

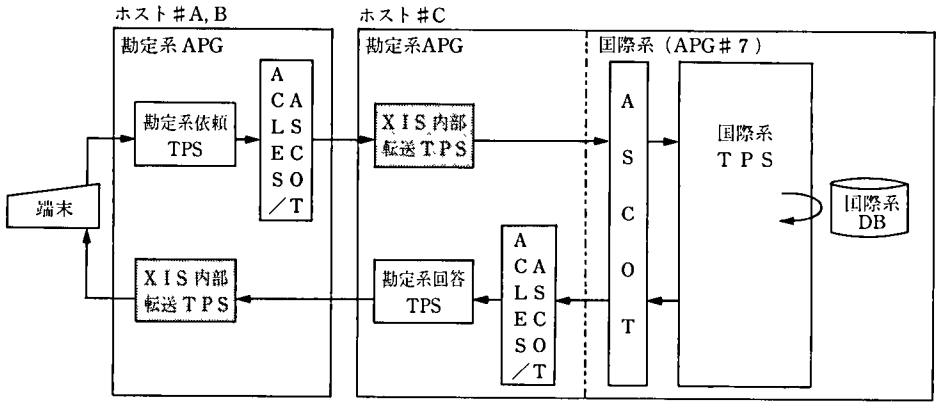


図 2　勘定系→国際系パススルー連動の仕組み

Fig. 2　Structure of accounting system and foreign exchange system path-through interface

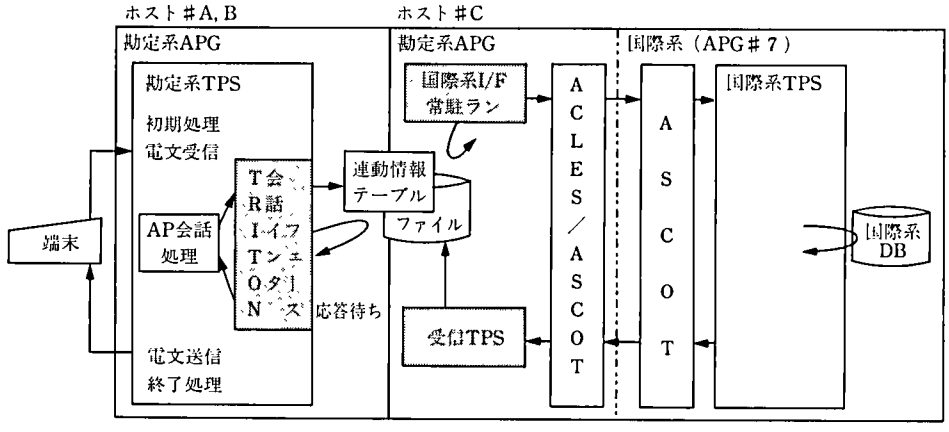


図 3　勘定系/国際系連動の仕組み

Fig. 3　Structure of accounting system and foreign exchange system interface

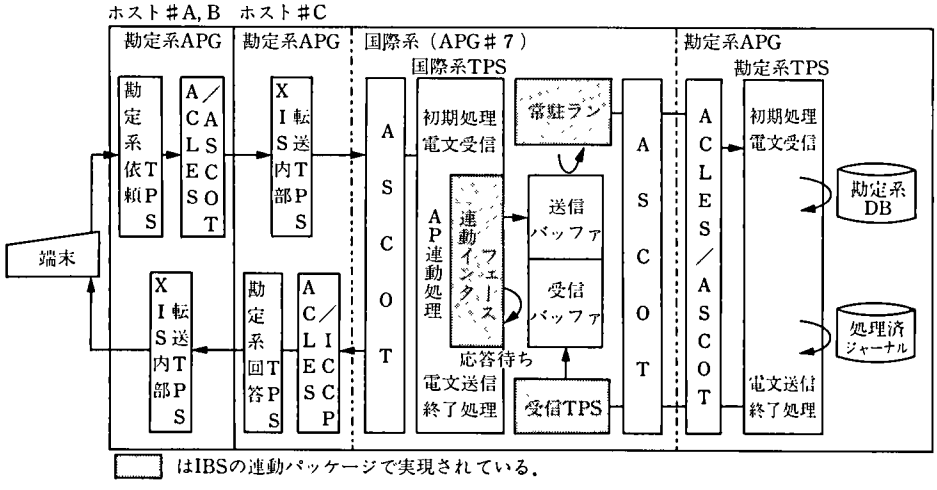


図 4　国際系/勘定系連動の仕組み

Fig. 4　Structure of foreign exchange system and accounting system interface

ーと呼ぶ)．パススルー TPS は電文の加工を一切行わず，国際系へ電文を送出している．これにより国際系は端末から入力されたと同様な処理を行うことができ，国際系へのインパクトを極力減少させている．

② ホスト間転送によるデータ授受

取引は勘定系の複数のオンラインホストから，国際系のローカル APG の存在する一つのホストへの転送となり，ホスト間の転送が発生する．ホスト間の転送は，XIS の基本トランザクション転送機能* により実現されているため，使用者はホストの違いを意識することなく処理することが可能である．

2) 勘定系→国際系連動

① TRITON 会話インタフェースの利用

勘定系と国際系のデータ授受を行う業務処理は，3.2 節の 5)で記述した，TRITON の会話インタフェースを使用して行う．同インタフェースは XIS の ACLES/ASCOT とのインタフェースをとる．ACLES/ASCOT はデータの引き継ぎを考慮する必要性があるが，同インタフェースでデータの引き継ぎを行うことにより，業務処理はデータの引き継ぎを考慮することなくデータ授受を行うことができる．

② 疑似 ACLES/DTP

TRITON の会話インタフェースは国際系とのデータ授受の際，直接 ACLES/ASCOT とインタフェースをとらず，ACLES/DTP と同一インタフェースのサブモジュールをコールし，そのサブモジュールが ACLES/ASCOT とインタフェースをとる方式を採用している．これにより，基盤ソフトウェアの変更があっても会話インタフェースの修正を極小化できる．

③ 複数アプリケーションとの連動（同一インタフェース（情報系・国際系））

勘定系の融資業務の一部である稟議取引は，情報系および国際系とのデータ授受を必要とする複数アプリケーション連動処理である．TRITON 会話インタフェースは情報系・国際系という相手を指定するだけで同一のインタフェースでデータ授受を行うことが可能である．

3) 国際系→勘定系連動

① 同一ホストによるデータ授受

国際系ローカル APG から勘定系コンカレント APG への連動となるため，同一ホスト内の連動としている．

② パススルーと連動の複合

国際系から勘定系への連動は，国際系が端末取引の場合，3.3.3 項の 1)パススルー形態との複合となる．すなわち，勘定系→国際系→勘定系→国際系→勘定系となる．

③ IBS 連動パッケージの利用

国際系からの連動処理は，IBS の連動処理パッケージを利用する．同パッケージは TRITON 会話インタフェースと同様，データの引き継ぎを考慮する必要がない．

---

* XIS 基本トランザクション転送機能：使用者が電文送受信を行う場合に指定するインタフェースで，電文送受信等を行う機能．

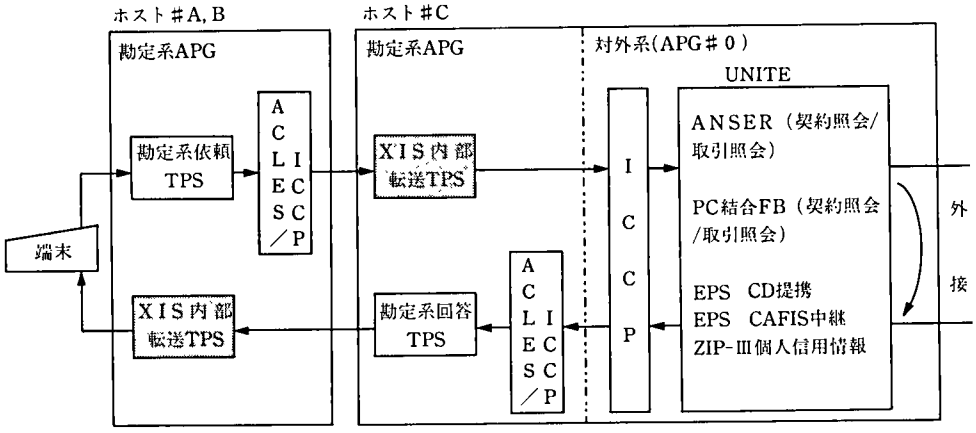


図 5　勘定系/対外系連動の仕組み

Fig. 5　Structure of accounting system and electrical banking system interface

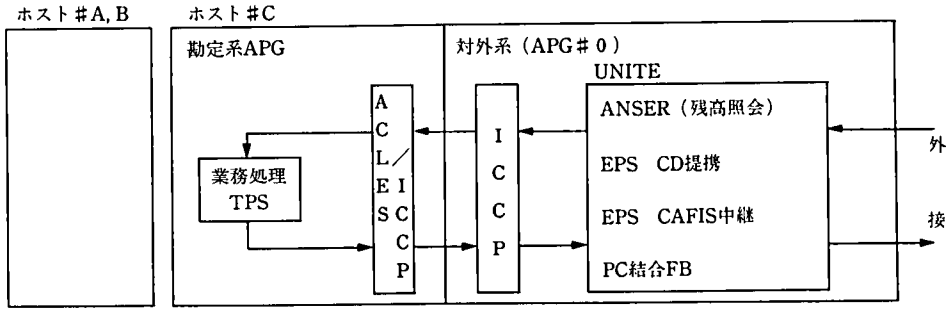


図 6　対外系/勘定系連動の仕組み

Fig. 6　Structure of electrical banking system and accounting system interface

3.3.4　勘定系/対外系連動

本項では，勘定系と対外系の連動の仕組み，および特徴について説明する(図 5, 6)．

1)　勘定系→対外系連動

①　ICCP 手順との接続

対外系は ICCP 手順との接続となるため，ACLES/ICCP を使用してデータ授受を行う．

②　ホスト間転送によるデータ授受

取引は勘定系の複数のオンラインホストから，対外系の BOSS-11 システムが存在する一つのホストへの転送となり，ホスト間の転送が発生する．ホスト間の転送は，国際系の連動と同様，XIS の基本トランザクション転送機能により実現されているため，使用者はホストの違いを意識することなく処理することが可能である．

③　UNITE 利用による外部センタ接続

外部センタとの接続は UNITE が行っているため，勘定系の対外系業務処理は UNITE とのインタフェースをとり，外部センタ接続を意識する必要がない．

2) 対外系→勘定系連動

対外系から勘定系への連動は，BOSS-11システムから勘定系コンカレントAPGへの連動となるため，同一ホスト内の連動としている(同一ホストによるデータ授受)．

3.3.5 ディレードバッチ

本稿では，ディレードバッチによる連動の仕組み，および特徴を説明する (図7)．

① 他アプリケーションとのデータ授受

勘定系のオンライン処理で作成された業務トランザクションは，XISのディレードバッチ処理機能とTRITONのディレードバッチ処理機能により，他のアプリケーションへのデータ授受を可能としている．主な受渡しデータは以下のとおりである．

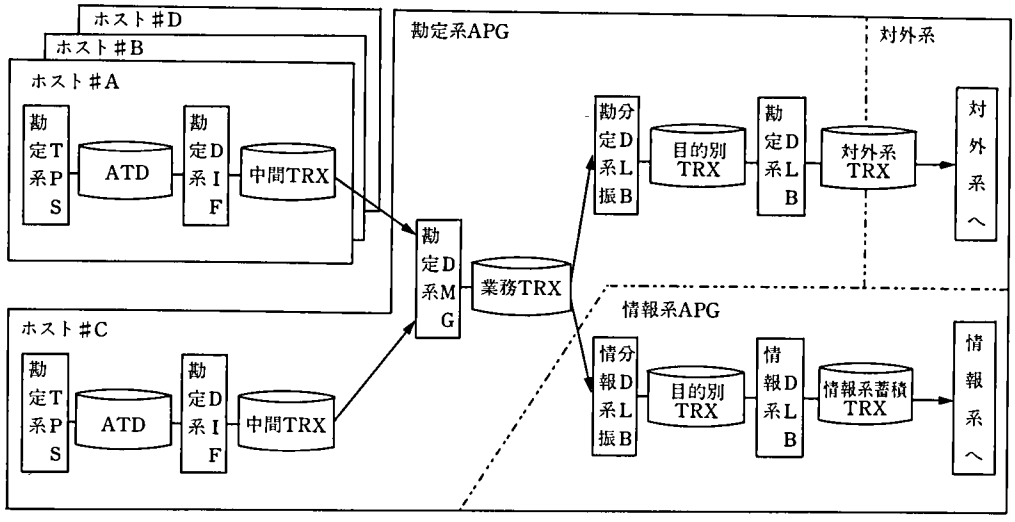
・対外系へのFB* 結合/ANSER** 用データ

・情報系への基礎データベース更新データ

② ホスト間データ授受

業務トランザクションはXISのディレードバッチ処理機能により，他ホストで稼働するディレードバッチ処理にデータを受け渡すことが可能である．これによりディレードバッチ処理を勘定系オンラインホストとは異なるホストで稼働でき，処理の負荷分散を図ることができる．

③ 多段階トランザクション処理機能の利用



ATD (Audit Trail Disk)：データベースの更新イメージ，トランザクション等を記憶し，リカバリ時の入力となるとともにディレードバッチの元データともなる．
DIF (Delayed batch InterFace run)：ATDよりトランザクションデータを抽出するラン
DMG (Delayed batch MerGe run)：各ホストで発生したトランザクションを処理順番にマージするラン

図7 ディレードバッチによる連動の仕組み

Fig. 7 Structure of delayed batch interface

* FB (Firm Banking)：対企業・個人に対する入出金データ・残高データのデータ伝送

** ANSER (Automatic answer Network System for Electrical Request)：音声照会通知システム

XIS のディレードバッチ処理機能により，多段階のトランザクション処理機能が可能となり，バッチおよび他のアプリケーションの目的に合わせたトランザクション（目的別トランザクションと呼ぶ）を作成している．この目的別トランザクションの作成により，他のアプリケーションが必要とするデータの追加・変更の場合は，目的別トランザクションを追加・変更するだけで実現でき，バッチ処理の弾力性を図ることができる．

3.3.6 ファイルシェア

共用磁気ディスク装置の利用によるファイルシェアは，ホスト処理負荷分散という観点で，各アプリケーション処理ホストを分散している(図8)．他アプリケーションとのデータ授受については共用磁気ディスク装置にファイル作成することにより，テープオペレーション等の負荷を軽減する．ファイルの構築・削除・アクセスについては，MHFSF 1100-IIのファイルシェア機能により，各ホスト間の整合性が保証されている．

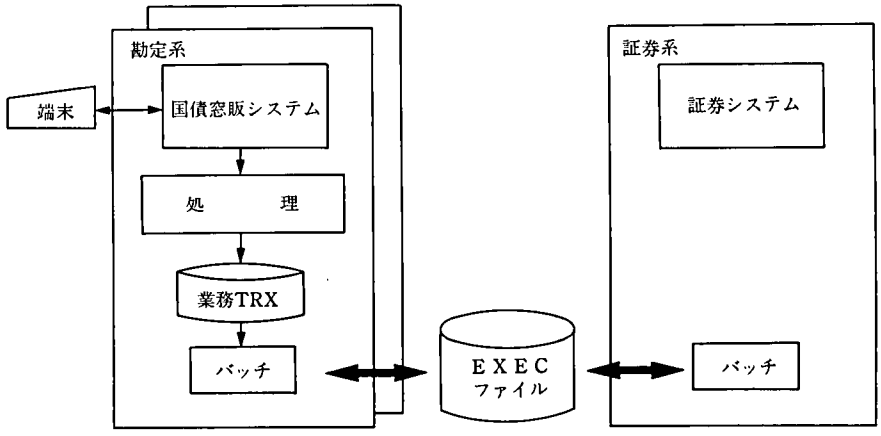


図 8　ファイルシェアによる連動の仕組み
Fig. 8　Structure of file share interface

4. おわりに

資産の有効利用という観点から既存のソフトウェアを使用することにしたが，ソフトウェア基盤の違いにより，アプリケーション連動の仕組みが異なり，これにともなってインタフェースの相違・障害対応等が異なることとなる．これを回避するためにTRITONでは，標準インタフェースの採用・開発を行った．たとえば，勘定系/情報系連動はACLES/DTPを使用したことにより，データの引き継ぎという面で使用者の負荷が軽減され，TRITON会話インタフェースにより，情報系と国際系のインタフェースは統合され，使用者は他のアプリケーションのソフトウェア基盤を意識しなくてもよくなった．今後ますますアプリケーションが多様化する中で，データの一元管理という意味でも，アプリケーション連動はより多様化・複雑化することが予想される．アプリケーションの特性により，アプリケーション連動の仕組みを選択する必要はあるが，ソフトウェア基盤を統一することは，インタフェース・障害対応等が統一

され，使用者にとって大きな長所となる．早期の統一が必要とされる．

執筆者紹介 黒　川　　茂（Shigeru Kurokawa）
昭和49年東京教育大学理学部数学科卒業．同年日本ユニシス(株)に入社．金融担当SE，TRITONシステム部を経て，現在，百五システムサービス部に所属．

24時間365日稼働

A Non-stop Banking System
—— For a 365 24-hour Days Computer Operation

中北晴久

要　約　金融機関のオンライン・システムでは，金融機関相互の競合により自動機の時間延長サービスが拡大されている．時間延長の拡大はいずれ24時間稼働に結びつくことになると予想され，TRITONでは，24時間365日稼働における共通機能の開発を大きな目標の一つとした．

その目標を実現するには，

第一にすべての処理がオンラインと並行して処理できること，

第二にシステムの静止状態がない中で，静的ファイルを作ること，

が求められていた．この要求を満たすために，TRITONでは以下の機能を開発することになった．

1)　階層構造を持つディレード・バッチの採用

2)　先日付/入金待ちの開発

3)　業中ダンプから確定残高を持つ静的ファイル作成ツールの開発

限られた開発期間・環境の中で，いくつかの課題は残されたものの，アプリケーション共通機能は実現することができた．ここでは，その共通機能と実際の対応についてまとめている．

Abstract　Competition among the financial institutions has triggered up demands for extending the on-line service time of automatic teller machines and cash dispensers. The expanded operation time of those terminals heralds a need for a 24-hour-per-day computer operation in the banking business in the long run. One of the TRITON development objectives was to provide applications-common functionalities for non-stop computer operation that lasts 24-hour-per-day and 365-day-per-year.

To reach this goal, it was mandatory to implement the following requirements:

1)　Processing of all applications concurrently with on-line real-time processing

2)　Creation of static files while the system keeps on running.

For those requirements, the TRITON system forced its developers to newly make available and incorporate the following capabilities:

1)　Delayed batch processing based on hierarchical structure

2)　Predated transactions processing and deposit-waiting transactions processing

3)　A support tool for creating static files with closing balances out of dumped logical on-line databanks.

With all the limited development period and environment, the system was successfully embedded with those functions common to applications. This paper provides their profiles and how they work on the actual scene.

1. はじめに

昭和61年に始まった金融機関の自動機時間延長サービスは，諸経費の増加とその回収という問題を含みながら，金融機関相互の競合により徐々に拡大した．平成2年に西日本の地銀・信金で開始されたサンデーバンキングは，年間のオンライン稼働日数を50日以上も増加させることになり，金融機関に対する負担をさらに増大させた．このサンデーバンキングは平成3年1月，都銀の参入によりほぼ全金融機関に広がり，一部の金融機関では祭日にまで稼働を広げている．また，金融機関のサービス改善に対する各種世論調査結果にも「平日窓口終了時間延長…44.8%」「土・日・祭日の自動機機能拡大…42.0%」[2] 等が高率のニーズとして挙げられている．

このような社会環境の変化は，金融機関のコンピュータシステムに対してオンラインの長時間稼働を要求することになり，その結果，多くの金融機関ではオンライン終了後の業後処理，月次処理をはじめとするバッチ分野に多大な影響が出ており，一部では翌日のオンライン開始時間までの余裕時間をとれない金融機関もある．したがって，オンライン時間延長・休日稼働に対するコンピュータシステムへの抜本的対策が求められている．

TRITONでは，次世代の金融機関システムを目指し，24時間365日稼働を想定した基盤（表1）を構築した．

24時間365日稼働を実現するには，システム基盤，アプリケーション共通機能の両分野でのシステム的考慮が必要となるが，本稿ではアプリケーション共通機能を中心に記述する．

表1　24時間365日稼働のためのシステム的考慮点
Table 1　Systems consideration for the non-stop real-time system

分　野	要　　件	
システム基盤	無停止システム	
	オンライン中のハードウェア保守	
	オンライン中のファイル保守	ファイル再編成
		不要ファイルの削除
	ソフトウェアのバージョンアップ	インフラ
		アプリケーション
	オンライン中のネットワーク変更	
	オンライン中のXISリポジトリ変更	
	短時間でのセットアップ/ターミネーション	
アプリケーション共通機能	オンライン終了を意識しないシステム	
	静的ファイルを要求しないシステム	

XIS（eXtended Information System）：統合オンライン支援システム

2. 金融機関コンピュータシステムの現状と課題

2.1 オンラインとバッチの運用時間帯の現状と課題

現在の金融機関コンピュータシステムは1日の運用時間帯から見ると，図1に示すとおり，大きくオンライン時間帯とバッチ時間帯に分かれている．

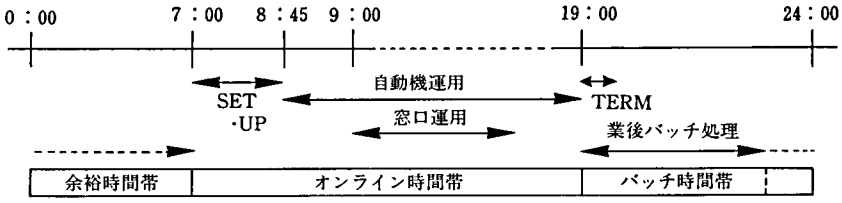


図 1　現行運用時間帯

Fig. 1　Timetable under existing systems

オンライン時間帯は，営業店の窓口端末や自動機からの入力を即時処理し，顧客への直接サービスを行っている．自動機時間延長サービスの拡大は，そのままオンライン時間帯の延長となり，バッチ処理可能時間帯の短縮となる(余裕時間帯の減少)．一方，オンライン終了後のバッチ時間帯では，オンライン時間帯で処理されたデータを業務トランザクションとして入力する日次バッチ処理や翌日分のセンタカット（以下C/Cと呼ぶ）処理等を行っている．データ量の増加に加え，顧客サービス向上による口座振替データ量の拡大により，バッチ時間帯も増加の一途をたどっている．

以上の状況より，ピーク日のバッチ処理終了時間が翌日のオンライン開始時間に迫り，オンライン時間帯とバッチ時間帯に分かれた運用をしていては，これ以上の時間延長に耐えられない金融機関が出てきている．このため，「オンライン終了を意識しないシステム」が求められている．

すなわち，オンライン・システムは24時間365日稼働を続け，バッチシステムはオンライン稼働に影響されず処理が行えることを意味している．

2.2　静的ファイルを前提としたバッチシステムの課題

静的ファイルとは，オンライン・トランザクション処理やC/C処理によるオンライン・データベース（以下オンラインDBと呼ぶ)/システムテーブルへの更新が発生しない状態で，かつその内容が計数管理情報（総勘定元帳）の内容に合っている時点のファイル状態を示す．

従来のバッチ処理は，バッチ処理可能時間帯に実行される処理のうち，「静的ファイルを前提とした処理」が多くある．言い替えれば，「オンラインが停止していることを前提とした処理」を中心にバッチシステムが構築されている．このため，24時間365日稼働した場合には，以下に述べるようなバッチシステム運用上の課題が内在している．

2.2.1　ファイル保存

オンライン終了の契機がなくなることになり，ファイル障害対応のための静的ファイル保存が不能となる．

2.2.2　日次バッチ

当日勘定に計上すべきトランザクション(以下TRXと呼ぶ)と，翌日勘定に計上すべきTRXが混在することとなり，当日/翌日に関する考慮が必要となる．

また，オンラインDBへの直接更新や直接参照が行われているケースも多々あり，抜本的な対応が必要となる．

2.2.3 勘定計上/締上げ

大多数のシステムにおいて，勘定計上は精査ファイルへ当日分のみを計上する方法をとっているが，当日/翌日に対する考慮が必要になる．

2.2.4 センタカット

従来のC/C処理には，オンラインDBの静的状態を前提にした2回戦/3回戦処理がある．また，口座振替処理はバッチ時間帯に翌日分を翌日日付で処理している．オンラインDBの静的状態とデータ量の増加に伴なう処理時間帯の確保が必要となるが，24時間365日運用では確保不能である．

2.2.5 月次バッチ

月次バッチシステムは，各月末日時点の静的オンラインDBを入力としている．したがって，稼働中のオンラインDBより月末時点の静的ファイルを作り出す対応が必要となる．

2.2.6 元加処理

従来の元加処理は，利息決算日時点の静的ファイルの確保を前提として処理されている．24時間365日稼働では，従来どおりの考え方による静的ファイルを前提にした元加処理方式ではシステム上無理がある．

2.2.7 金利変更

従来の金利変更は，静的な金利管理テーブル（ファイル）を中心に実施されてきた．しかし，24時間365日稼働では，静的な状態を作り上げることが不可能であり，抜本的な対策が必要となる．

以上の課題から24時間365日稼働を実現するには，「静的ファイルを要求しないシステム」が求められる．

これは，オンラインDB（システムテーブル含む）の静的状態を要求しないTRX処理システムを構築することである．

3. 24時間365日稼働実現の考え方

3.1 システムの目標

24時間365日稼働という環境の下で，営業店取引・自動機取引・C/C取引といったさまざまな処理を行いつつ，顧客の利益あるいは銀行の利益を極力保証することを目指し，システム目標を以下の項目とした．

- 24時間365日稼働に柔軟に対応できるシステム体系の構築
- オンラインおよびバッチ処理プログラムへの24時間365日稼働対応共通機能の組込み

3.2 基本的な考え方

24時間365日稼働可能なシステム構築にあたり，基本的な考え方を以下の通りとした．

1) 日次は，トランザクション入力を主体としたバッチ処理とする．
2) オンライン終了を意識しないシステムとして，次に示す処理を考える．
   - ディレード・バッチ処理
3) 業後センタカット・トランザクションの前日業中処理による処理時間の確保の

ために，次の処理を考える．

・先日付処理

4) 静的な状態が不要なシステムのために，次に示す機能を用意する．

・ファイルバックアップと任意時点のファイル作成

・オンライン中のファイル再編成処理

・特殊処理（元加処理）

5) バッチ処理のオンライン機能への組込みのために，次の機能を用意する．

・金利変更

6) バッチ処理のオンライン化として，次の機能を用意する．

・端末照会機能

・配信機能

表2　24時間365日稼働の形態

Table 2　Application forms in the non-stop real-time system

形態	営業日		土休日	
	運用時間	元帳	運用時間	元帳
A	7時～23時	本元帳	7時～23時	ミニ元帳
B	0時～24時	同上	同上	同上
C	7時～23時	同上	7時～23時	本元帳
D	0時～24時	同上	0時～24時	同上

完全24時間への移行ステップとしては

① A→B→D

② A→C→D

のいずれかの形態で推移することとなる．

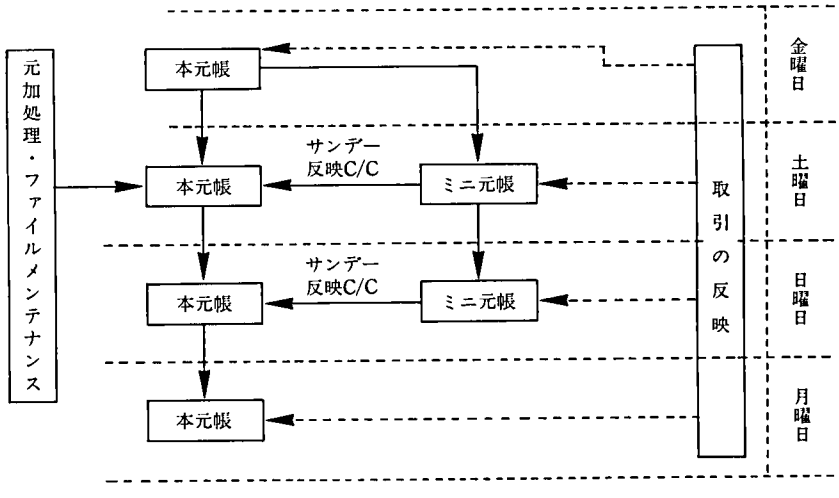


図2　ミニ元帳の運用形態

Fig. 2　Application form of the miniature ledger

3.3 24時間365日稼働システムの運用

完全24時間稼働を実現するまでに表2に示す4形態を想定する．

形態A，形態Bはミニ元帳方式を継続することによって，本元帳は土休日にオンライン処理からはずされ，元加処理，ファイルのメンテナンス処理に使用することができる．運用形態概略としては図2のとおりである．

4. TRITONで実現された機能

4.1 ディレード・バッチ

4.1.1 ディレード・バッチ処理の目的

TRITONのディレード・バッチ（以下DLBと呼ぶ）は，勘定系や他系で発生した様々なTRXをオンラインの終了を待つことなく，逐次，バッチ処理に展開すると共に，先日付センタカット処理にも対応し，以下の目的を実現する．

- オンライン処理とバッチ処理の有機的結合
  - 各イベント終了*による後続バッチの自動起動
- 業後バッチ処理の一部業中化
  - オンライン中に，細分化された各種バッチ目的別ファイルを作成
- カウンタ照会の即時化
  - ディレード処理でのカウンタ更新による端末応答時間の短縮
- システム負荷の平均化
  - オンライン取引の間隔利用
    - →オンライン取引時間帯のコンピュータ余裕リソース
- 即時配信システムとの連動
  - 即時性を必要とする各種情報の提供
- 勘定日日付変更の容易性確保
  - 当日TRXと翌日TRXの分離処理の実現

4.1.2 ディレード・バッチ処理の機能

TRITONではDLBを多段階に展開する方法を採用している．

この仕組みを使うことにより，用途別に作られている後続DLB，後続バッチラン（ディレード・バッチラン）を図3のように階層的に組み合わせ，オンラインとの並行処理を可能にするとともに，バッチ処理の入力データを目的別に分割作成し処理の効率化を計っている．

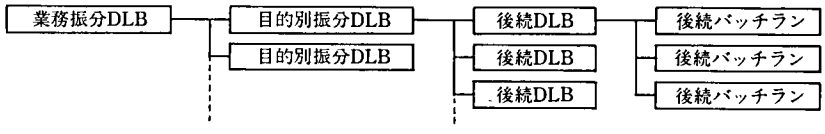


図3 多段階ディレード概要

Fig. 3 Multiple delayed-batch system

* イベント終了とは，一日のオンライン稼働の中で，処理内容ごとに取引許容時間帯を設定し，その時刻を経過した時点で特殊TRXを出力する．DLBは特殊TRX入力時に後続のバッチランを起動する．

DLBは取扱うデータにより即時性を必要とするものと，あるタイミングごとに処理すれば良いものとがあり，TRITONでは以下の3種類に分割している．

- 精査日計DLB……TRXが発生した時点または設定した最大許容時間の監視により起動させる（日計DLB，精査DLB）．
- 即時型DLB……設定した最大許容時間の監視と，ホスト内にて実行されたステップ数の監視により起動させる（即時DLB）．
- 遅延型DLB……固定タイミングまたは一日の運用で，業務の区切り時に作成成される「制御TRX」の入力を引金として起動する（事務系DLB，情報系DLB，期日管理DLB）．

これらの各種DLBは，各業務の性質により選択して使用される．

また，DLBランは以下の制御機能を有している．

- ディレード・バッチラン強制起動
- ディレード・インタフェースラン終了
- ディレード・バッチラン強制終了
- マージ対象ホストの切り離し

4.1.3 ディレード・バッチ処理の構成

TRITONは4ホストを基本構成としており，各ホストごとに出力されるTRXを集約する必要があった．

さらに，オンラインとの並行バッチ処理をより効率的に行うため，多段階DLBを採用し，以下の構成とした．

- ディレード処理インタフェース/ディレードマージ
- 振分けDLB（使用者プログラム）
- 後続DLB（使用者プログラム）
- 後続バッチラン（使用者プログラム）

振分けDLB，後続DLB，後続バッチランはすべてディレード・バッチランであり，ディレード・バッチラン制御を合わせて持つことができる．

① ディレード処理インタフェースラン（DIF）/ディレード・マージラン（DMG）によるTRXの集約（ホスト分散されたTRXの集約）

オンライン・システムは各ホストごとに処理結果を出力TRXとしてオーディット・トレイル・ディスク（ATD）へ出力する．

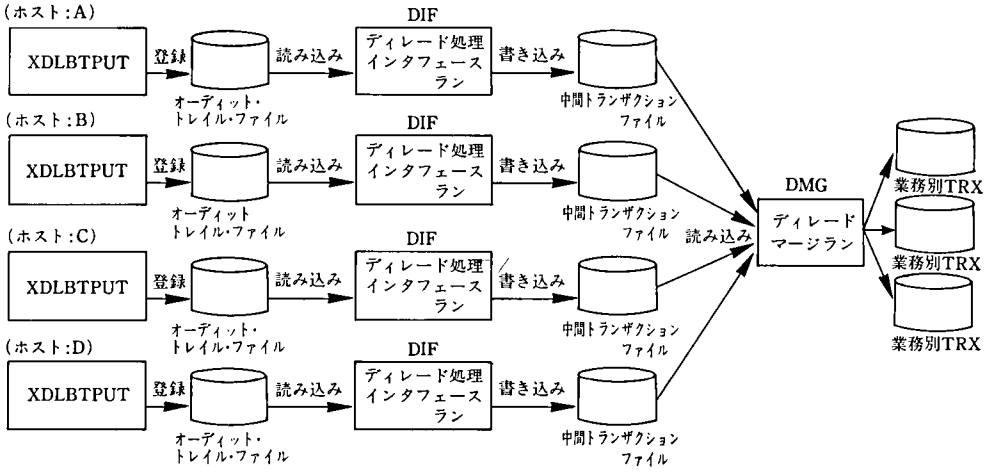
4ホストから出力されたTRXは，図4で示すように，ディレード・バッチ・インタフェース（DIF・DMG）によりデータタイプ別に中間TRXファイルとして作成され，マージ処理を経て，集約された業務別TRXファイルが作成される．

② 振分けDLBによる目的別トランザクションの作成

振分けDLB（使用者プログラム）は，業務別TRXファイルを入力として，目的別TRX（バッチランおよび後続DLBの入力データ）を作成する．

③ 後続DLBによるアプリケーション処理

後続DLB（使用者プログラム）は，目的別TRXファイルから逐次TRXデータを読み込み，任意に処理することができ，また後段の各種ファイルにも書



DIF：ATDを読み込み，中間TRXファイルを出力する．
DMG：各ホストから出力された中間TRXファイルを読み込み，各TRXデータタイプごとに振り分ける．
《コンカレント・アプリケーション・グループ環境下》

図 4　DIF/DMG による TRX 集約

Fig. 4　Transaction gathering by DIF and DMG

き込みをすると同時に，後続するバッチランの制御を行う．

④　後続バッチランの処理

後続バッチラン（使用者プログラム）は，振分け DLB や後続 DLB からの自動起動により目的別 TRX ファイルや各種ファイルを入力として，最終出力情報として各種帳表やデータファイルの作成を行う．

⑤　全体構成とトランザクションの流れ

TRITON では DLB の出力を，階層構造を利用し，細分化した目的別 TRX として作成している．目的別 TRX の項目は，帳表類やデータファイルを出力するために必要な項目を洗い出して作られ，業務終了を契機に，バッチ帳表作成等に使われる．

即時 DLB が作成した目的別 TRX は，さらに後続 DLB により即時還元データとして作成される．

図 5 は TRITON のディレード・バッチ全体図である．

4.2　先日付処理/入金待ち

オンライン 24 時間稼働体制を支える重要な仕掛の一つとして，先日付処理システムを構築した．

この先日付処理システムによって，2.2 節の課題のうち，「2.2.2 項の日次バッチ」「2.2.4 項のセンタカット」「2.2.6 項の元加処理」の対応を行っている．

4.2.1　先日付処理のねらい

TRITON における先日付処理システムは，以下の事項を主なねらいとしている．

1)　業中 C/C の拡大（センタ業後作業の短縮）

・従来，オンライン終了後に行われていた C/C は，確定残高が前提となっていた

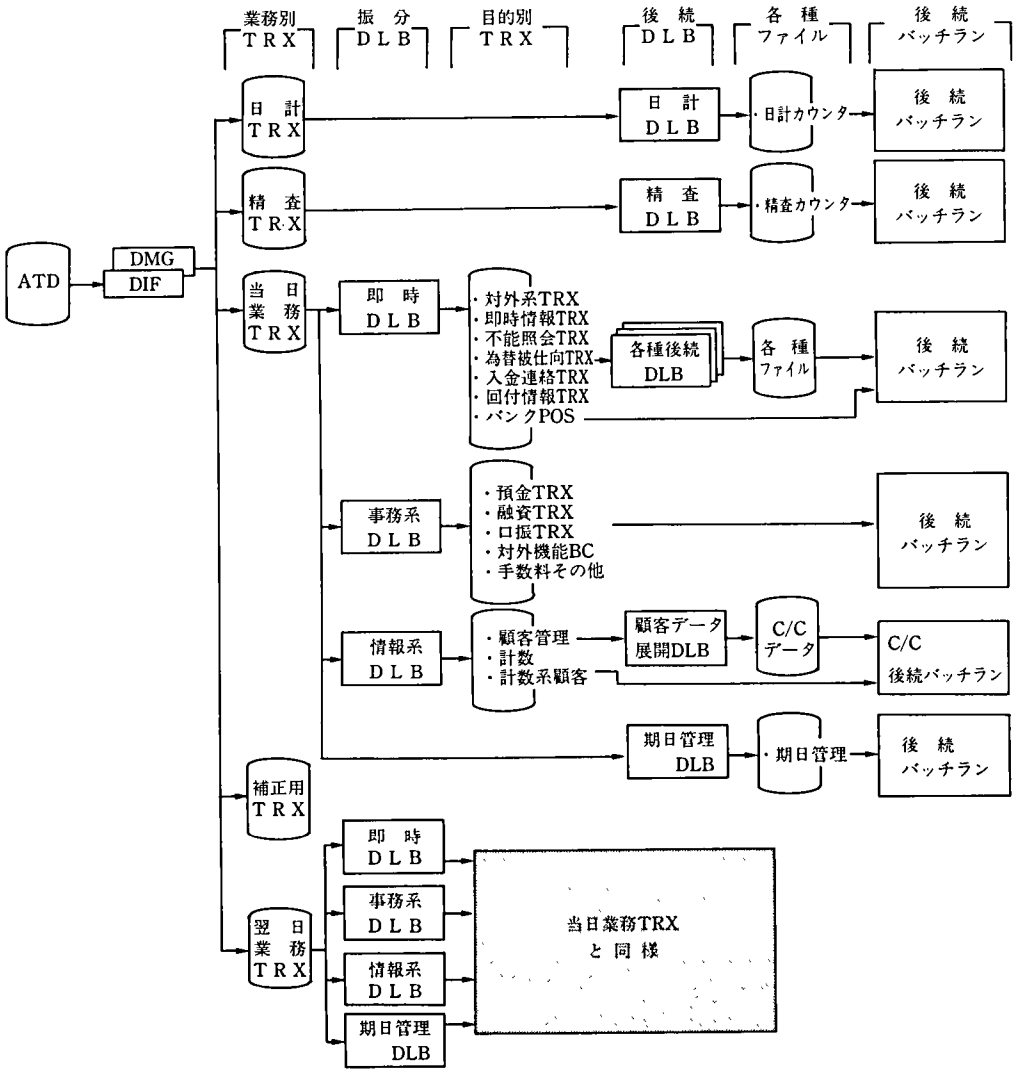


図 5　ディレード・バッチ全体図

Fig. 5　Delayed-batch system

が，先日付処理機能をもった集中C/Cの実施により，確定残高ベースで行ったのと同様の結果を得ることができる．

・C/Cデータ間相互，あるいは営業店端末取引・自動機取引との処理の優先順位（引落し順位）をきめ細かく指定できるようにした．

このため，従来は運用上一部前夜C/C処理にて行っていた処理も，業中C/C処理化できるようになった．

2)　自動化機器運用時間帯の延長対応（含む休日稼働）

・今後ますます，自動機の深夜稼働・休日稼働が拡大され，業後バッチ作業やC/Cの処理時間確保が不可能となる．

このための対応として，オンラインと並行して当日以降のC/C処理が必要となった．

3) C/C 処理量の平準化

・銀行業務の中で，口座振替を含む C/C 処理は最も繁閑差が激しく，そのピーク日ピーク時対策として先日付処理を位置付けた．

・また，入金待ちシステムの採用により，従来実施している C/C 2 回戦および再振替作業を廃止した．

4.2.2 先日付処理の機能

先日付処理は，一般的には何日か先の取引を事前に処理しておき，その後当該先日付取引の期日（勘定日）の到来により取引を確定させる処理である．

TRITON での先日付処理は，システムニーズとシステム効果およびシステム負荷を検討し，翌営業日までの処理を行うこととし，取引形態としては C/C のみに限定した．

毎日の C/C データ量の繁閑差は，外部口座振替データの量により左右される．外部自振の繁忙日でも，業中より C/C を実行すれば十分対応が可能と判断した．

4.2.3 先日付処理の方式

C/C の先日付処理は，その C/C 実行から確定の間に，優先取引（端末・自動機）が発生する事を前提で行うものである．

これに対するメンテナンス作業・確定日以降に実取引化する作業負荷（システム負荷・開発負荷）と，先日付採用による効果とを比較検討して，その方法を決定する必要があった．

TRITON では，全 C/C 取引を，C/C データ量・メンテナンスの発生頻度・各科目のファイル構造等の観点より，流動性取引と固定性・融資取引に大別し，各々の特徴により図 6 の通りまとめ，先日付処理を二つの方式で行うこととした．

	流動性取引	固定性・融資取引
C/Cデータ	・大量データ ・日による繁閑差が激しい ・単純入出金取引	・流動性に比べて小量データ ・絶対量が少なく，繁閑差は問題無し ・複合取引で更新方法も複雑
メンテナンス発生頻度	・自動機取引を含め取引度合いが多く，発生頻度は高い	・ほとんどが月間1回の取引しか発生しないので，発生頻度は低い
ファイル構造	・一般取引で使用するファイルはマスタと取引明細で単純	・連動取引がほとんどであり，構造も流動性に比べて複雑
採用先日付方式	予約方式	実更新方式

図 6 予約方式と実更新方式

Fig. 6 Reserved update system and real-time update system

上記 2 方式の処理方法については次に詳述するが，両方式の特徴と各採用理由は以下の通りである．

1) 予約方式

先日付で実行された C/C 情報を，先日付用の各種スレーブにて制御し，マスタ更新は行わない．

特徴としては，

① 先日付データが確定する迄に取引が発生しても，メンテナンス処理で対応，

② C/Cの実行順序を意識させない，

③ メンテナンス処理が容易である，

④ 入金待ち処理により自動再引き落しが可能，

等が挙げられる．

予約方式は，メンテナンス負荷が軽微であることが条件であり，大量データ処理に適している．TRITONでは単純取引である「流動性」に対して採用した．

2) 実更新方式

先日付C/C実行時点で直接マスタ等を更新する方式である．

特徴としては，

① 確定日以降の作業が不要であり，処理が単純である，

② システム開発負荷が軽微である，

③ メンテナンス処理の機能を軽減（流動性取引に限定）する，

等が挙げられる．

実更新方式は，先日付処理から確定日時まで当該口座に取引が発生しないことを前提とし，C/C実行は営業店オペレーション終了後に行う．

この方式は「固定性・融資」に対して採用した．

4.2.4 先日付処理の予約方式

流動性預金マスタおよび雑勘定口座マスタを対象とし，取引を実施するがマスタを更新せず，取引の内容を保有する先日付明細スレーブと，先日付取引の確定日時単位にその時点の残高を管理する先日付残高スレーブとを，取引結果として作成する方式である．

予約方式は以下の3段階の処理で構成される．

- 予約処理：C/C実行により先日付明細スレーブと先日付残高スレーブを作成し口座マスタにリンクさせる処理
- メンテナンス処理：優先取引発生により先日付明細スレーブを成立から不成立へ，不能理由解消により不成立から成立への反転処理
- 解放処理：確定日時経過により先日付取引の内容を口座マスタおよび取引明細ファイルへ反映させる処理（同時に当該先日付明細スレーブと先日付残高スレーブを削除する）

1) 予約処理

予約方式の処理概要を図7に示す．

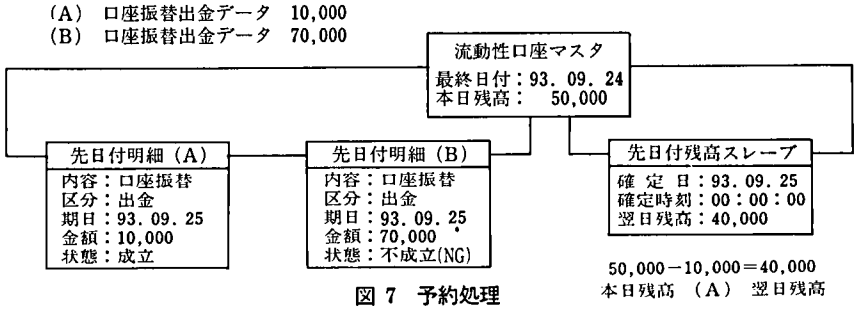


図7 予約処理

Fig. 7 Reserved update system

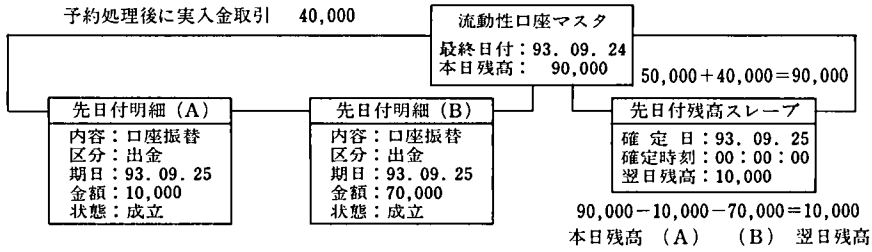
本日口座残高が50,000円の普通預金口座に対して，ⒶⒷ2個の口座振替C/Cが実行された．

Ⓐ 口座支払可能残高が50,000円であり，予約支払は成立として先日付明細を作成する．

Ⓑ 予約支払後の翌日残高は40,000円であり，予約支払は不成立として先日付明細を作成する．

2) メンテナンス処理

予約処理で予約スレーブが作成された後，実入金取引によりメンテナンス処理が行われると図8，表3の通りファイル更新が行われる．



4万円の入金処理が行われ，本日残高は9万円となる．入金処理後にメンテナンス処理が起動する(表3)．

図 8 メンテナンス処理

Fig. 8 Maintenance of the balance

表3 メンテナンス処理の推移

Table 3 Change in the balance caused by maintenances

	本日残高	メンテ前状態	入出金額	メンテ後状態	翌日残高
開始時	50,000	----------	----------	----------	50,000
入金取引	90,000	----------	＋40,000	----------	90,000
(A)	(更新せず)	成 立	－10,000	成 立	80,000
(B)	(更新せず)	不成立	－70,000	成 立	10,000

3) 解放処理

確定日後の解放処理により図9の通り取引明細が作成される．

Ⓐ 実入金取引に先だって先日付明細の解放処理を行う．

・先日付残高スレーブの本日残高への反映

・先日付明細スレーブの取引明細への反映

・先日付明細スレーブの削除

Ⓑ 本日取引分の処理を行う．

4.2.5 先日付処理の実更新方式

固定性預金マスタおよび融資マスタを対象とし，将来の取引であるが実取引と同じように取引の時点で口座マスタを更新し取引明細も作成する方式である．

また，連動取引の場合は連動相手の流動性預金側は予約方式とし，先日付明細スレーブを作成する．

特定時点（月末等）での確定口座状態を再生するために，実更新時のビフォアルッ

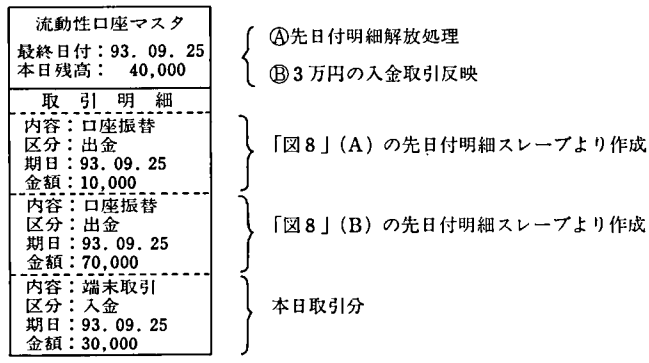


図 9　解放処理のマスタ更新内容

Fig. 9　Reflection of predated transactions in the master records

クを採取して対応する．

この方式では，メンテナンスの考慮を行わないため，先日付 C/C の実行を固定性預金・融資系の営業店オペレーション終了後に実行する．

図 10 は以下の前提で実更新処理が行われた場合の概略である．

4.2.6　入金待ちの機能

先日付 C/C 処理の「予約方式」で作成される先日付明細スレーブは，C/C 実行時の処理結果を保有し，残高不足等で不成立となった取引でも作成される．

その取引が確定日を過ぎても不成立であった場合，入金待ち期限までの間は入金待ち状態となる．

当該口座への入金取引により，メンテナンス処理が自動起動され，入金待ちスレーブの再引落し取引が実行される．

入金待ちの期間は，各処理の不能管理・延滞管理・再処理方法等により決定される．

《各処理の入金待ち期間》

- 口座振替……確定日当日
- 定期積金掛込……確定日より 7 暦上日
- 証書貸付返済……エンドレス　　等

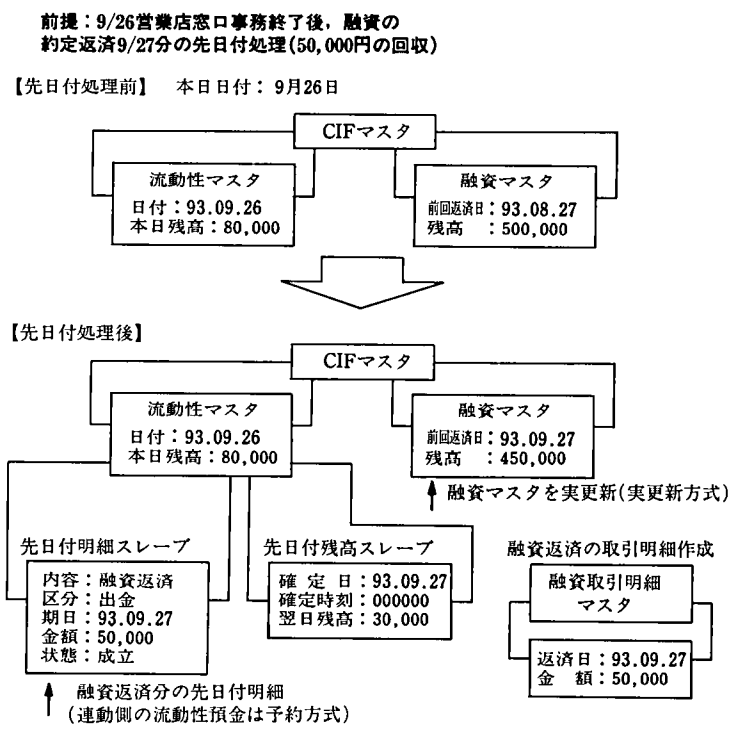
4.2.7　C/C の実行順序と引落し優先順位

C/C の繁忙日対策として，前日のオンライン稼働中に先日付処理により C/C を実行する．

TRITON の先日付処理においては，その取引の成立・不成立の確定を，実際の処理作業時点に従属させる事なく，作業実施日時と異なる日時（確定日時）において確定させることにより，C/C 実行順序に関係なく処理優先順序を決定させることを可能とした．

これは，先日付明細のキー項目に「優先順位コード」を保有し，C/C が後で実行されたデータで，優先順位の高いものはメンテナンス処理により，反転処理をさせることで実現させた．

先日付明細に保有しているキー情報は図 11 のとおりである．



■実更新処理における入金待ち処理

固定性預金および融資に関する先日付処理はすべて実更新方式であるが，残高不足等の不能時は実更新は行わず，流動性預金側で入金待ち状態を発生させる．

流動性預金の該当口座に端末，ATM，為替振込，センタカット処理等により資金が入金されると，自動的に固定性預金および融資側の更新処理を実行する．

■先日付処理における反転処理

固定性預金および融資においては実更新方式を採用するため，実取引と同様に取引処理時点での口座マスタが更新される．

仮に，融資返済において，先日付C/C時点で流動性預金に引落し残高が確保されていたとすると，当然，実更新処理が行われる．

しかし，自動機の支払を「0：00～24：00」まで可能とするため，顧客が実更新型先日付C/C終了後に該当口座より引出しを行い，口座の残高不足が生じた場合は，更新済の固定性預金・融資に対して更新情報の反転処理を自動的に行う（流動性預金についてはメンテナンス処理を行う）．

図 10 実更新方式の処理概略

Fig. 10 Real-time update system

したがってC/Cについては，きめ細かな優先順位規定が必要とされる．

銀行業務においては，内部自振データ（固定性・融資等）を外部自振データより優先処理するのが通例であり，C/C実行順序と優先順位を区別する必要が発生する．

図12は，優先順位による反転処理の例である．

4.2.8 先日付処理に伴い必要となった機能

以上，TRITONで実施した先日付処理についてその機能を述べたが，付随して必要となった機能について，以下に述べる．

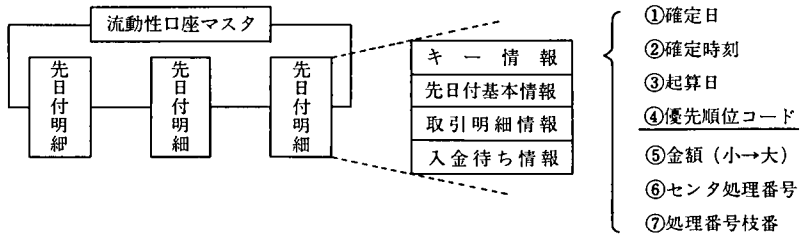


図 11　先日付明細の構成

Fig. 11　Composition of predated transactions

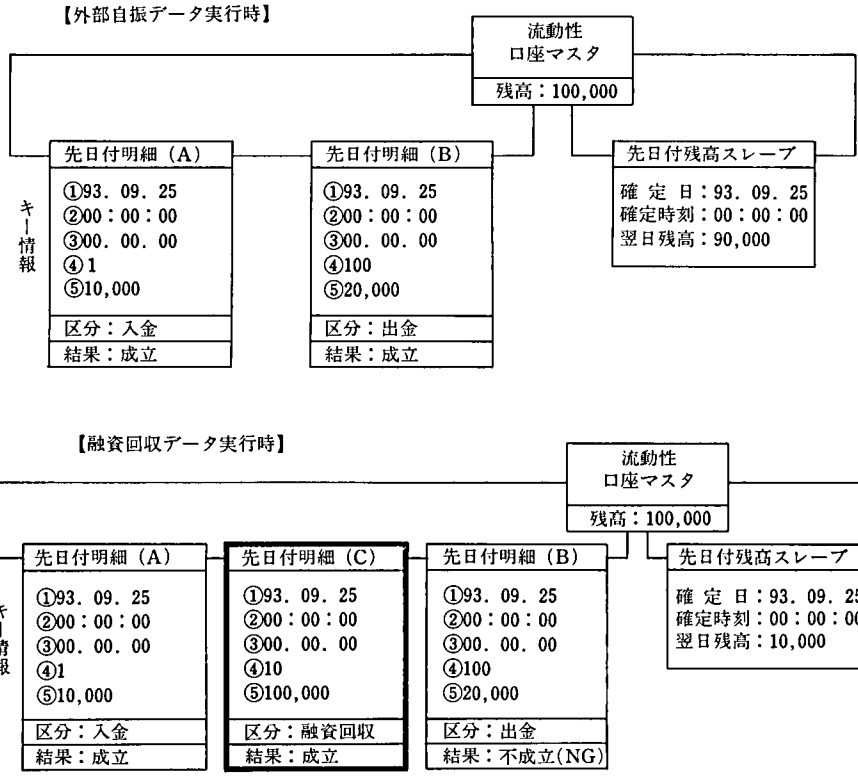


図 12　優先順位による反転処理

Fig. 12　Reverse process occurred according to priority

① TRX ファイルの 2 系統化（当日分，翌日分）

先日付機能を持つことにより，1 日の処理中に「当日分」と「先日付分」の取引が混在することになる．

TRITON では，日次バッチの入力となる業務 TRX・目的別 TRX を 2 種類保有し，当日分と先日付分に分離して管理する．

日付変更処理時に先日付分のファイルが当日分に切替登録され，当日分ファイルは初期化され，先日付分として再登録される．

当日分と翌日分のファイルはサイクリックに使用される．

図 13 は TRX の切替使用例である．

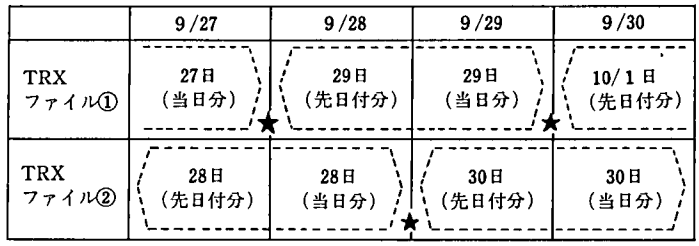


図 13 TRX ファイルの切替

Fig. 13 Switch in use of transaction files

9月28日に当日分として処理されるのは「TRX ファイル②」で，内容は9月27日に処理された先日付分と，9月28日に処理された当日分が含まれている．

9月28日に先日付分として使用されるのは「TRX ファイル①」である．

各 TRX ファイルは「★」のタイミングで初期化される．

② オンライン元帳参照時の確定残高算出処理の仕組み

詳細については，5.2節の“月次処理”にて記述する．

③ 保存ファイル作成時の確定残高算出処理の仕組み

詳細については，5.2節の“月次処理”にて記述する．

4.3 静的ファイルの作成

オンライン・ファイルとしては静的状態にないが，バッチ側の処理によって静的ファイルを作成している．

例として，CMF の静的ファイルの一つである CMF バッチマスタを作成する場合について記述する．

4.3.1 CMF バッチマスタの作成

前述のごとく，完全 24 時間稼働へは段階的に移行する事となるが，静的状態が保証されている段階と，静的状態が確保できない段階での対応とに運用が二分される．

《第1段階》 静的状態が保証されている段階（7：00～23：00 運用時）

従来通り高速データベース・リードルーチンによってダンプテープを作成し，実更新方式の先日付 C/C 更新分の補正を行って作成する．

図 14 は，第一段階の処理概略図である．

《第2段階》

① PALDUM による作成（完全 24 時間対応時）

稼働中のオンライン DB より静的ファイルを作成するには，PALDUM でダンプテープを採取し，その後，確定時点までの追いつき処理と実更新方式の先日付 C/C 更新分補正を行って，作成する．

したがって，オンライン稼働状況が静的状態を確保できなくなった場合には，PALDUM を使用して CMF 静的ファイルを作成する．

図 15 は，第二段階の PALDUM 使用時の概略図である．

② 片レグ切り離しによる方法（完全 24 時間対応時）

オンライン・ファイルが二重化されていることが前提であるが，静的データベー

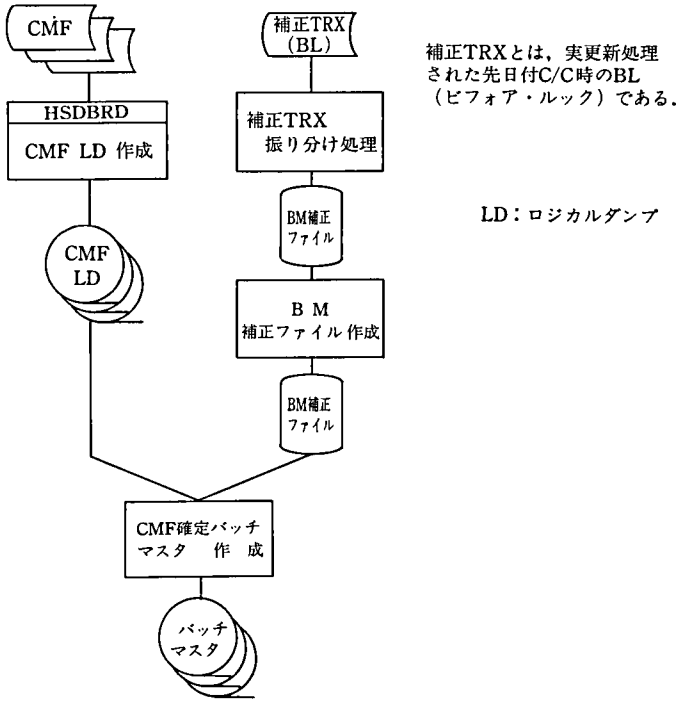


図 14　高速データベース・ルーチン使用時

Fig. 14　Making of static files using high-speed-databank-read program

スが必要な時間帯に片レグを切り離し，静的ファイルとして，高速データベース・リード・ルーチンにより，LD を作成する．その間，オンラインは片レグで続行される．

切り離された片レグは，LD を作成後，ダイナミック・レグコピーでコピー後，オンラインに戻される．

(ダイナミック・レグコピー)

オンライン中に，データベースのコピーを行うが，図 16 に示すレグ 1 を READ/WRITE 可，レグ 2 を WRITE ONLY にすることにより，コピー中に発生した取引によるデータベースの更新は両レグに正しく反映される．

したがって，コピー終了時点では，両レグの整合性は保証される．

図 16 は，第二段階のダイナミック・レグコピーによる概略図である．

4.3.2　CMF 静的ファイル作成時の考慮点

先日付 C/C を採用しているため，CMF 静的ファイルの作成にあたっては，次に示す先日付の確定処理が必要となる．

・流動性預金に対する先日付処理（予約方式）分の確定

・固定性預金・融資に対する先日付処理（実更新方式）分の確定

詳細については，5.2 節の“月次処理”にて記述する．

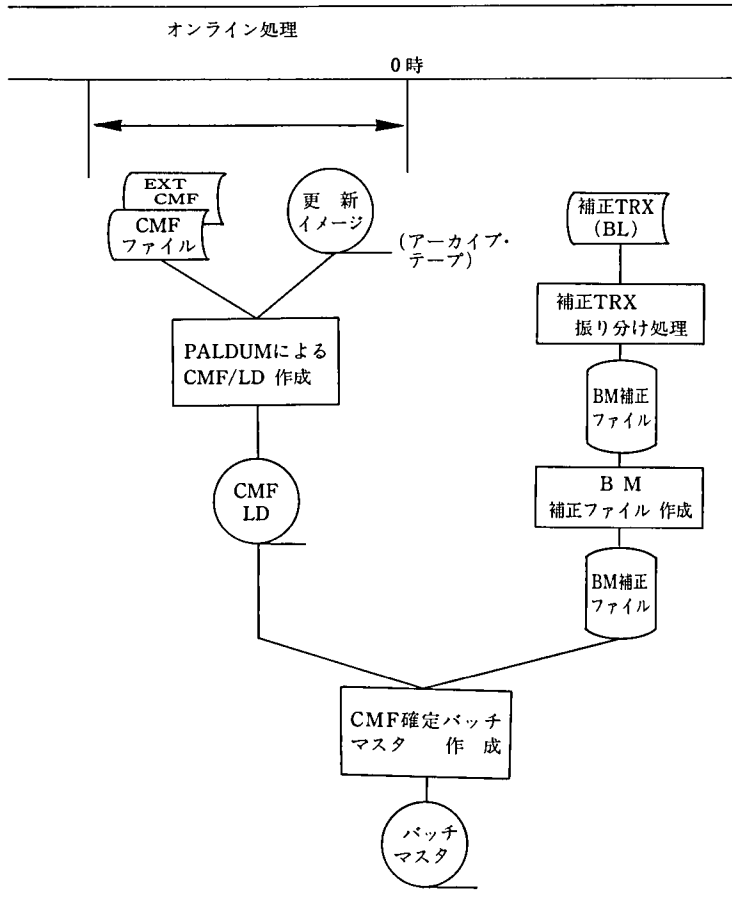


図 15 PALDUM 使用時

Fig. 15 Making of static files using PALDUM

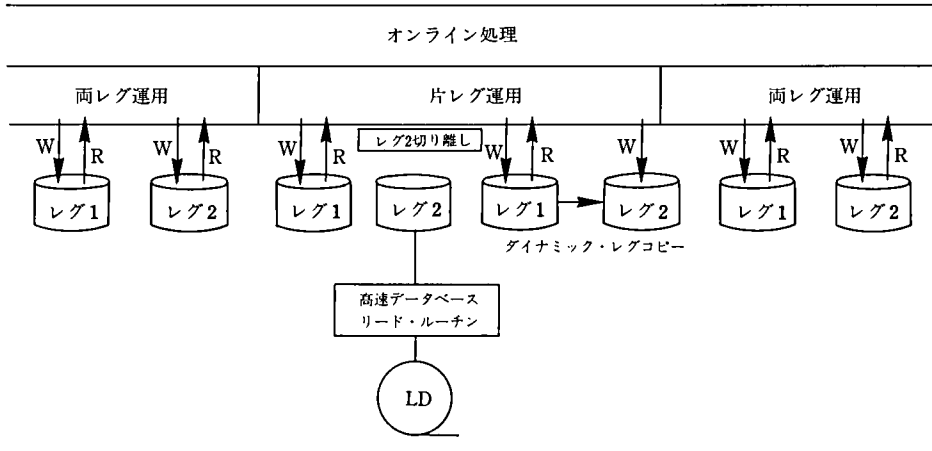


図 16 片レグ切り離しによる方法

Fig. 16 Making of static files by cutting off one component of databanks

5. TRITONにおける運用

TRITONのアプリケーションは，24時間365日対応という観点より，前述してきたアプリケーション基盤のもとに作成されている．

本章では，個々のアプリケーションが固有に対処している機能(表4)について記述する．

表4　処理サイクルごとの対応
Table 4　Functions used by programs

処理サイクル	処　理	処 理 内 容	アプリケーション共通機能
日次処理	ファイル保存	障害対応のためのファイル保存	静的ファイル作成
	日次バッチ	ディレード処理	先日付・ディレード
	センタカット処理	優先順位を意識しない先日付処理	先日付処理
月次処理	月次バッチ	先日付C/Cに対する対応 ・確定処理	静的ファイル作成
随時処理	元加処理	先日付C/Cによる元加処理	先日付処理
	金利変更	金利変更処理のオンライン対応	オンライン機能追加

5.1 日 次 処 理

5.1.1 ファイル保存

日次処理で保存されたファイルは，業務処理で使用されることはなく，あくまで障害対応のための処理である．

TRITONでは，オンライン処理中にオンライン処理に影響を与えずにPD (Physical Dump) を作成し，そのPDとアーカイブテープをそのまま障害対応として保存する方法を指向している．

現行のTRITON運用では，完全24時間稼働ではないため，オンライン終了後に静的状態でPDを採り，障害時対応用として保存するファイルを採っている．

5.1.2 日次バッチ

日次バッチへの考慮は，以下の項目が挙げられる．

1) 日次バッチ処理は，常にオンライン処理が稼働している環境の中で実行される事を前提としているため，以下の二点を原則とした．
・バッチ処理でのオンラインファイル更新は行わない．
・オンラインファイルの更新が必要な場合はセンタカット化する．

2) TRITONでは，日次バッチの入力となる業務TRX・目的別TRXを，2種類保有し，当日分と先日付分に分離して管理しており，アプリケーションではTRX内の「先日付フラグ」等によりどちらのファイルへ格納するかを判定する．

日次バッチが入力するファイルは，前日実行された当日指定の先日付分TRXと当日分TRXが格納されているので，意識することなく入力処理する．

3) 日次バッチは，オンライン終了を意識せずに各業務単位の終了を判断して，処理を開始する．

日次バッチ処理は一部の例外を除き，ワークフロー管理下で自動スケジュールするための情報を登録し，管理されている．

仕掛としては，業務単位の基準終了時刻に「先行ダミー処理」がワークフローにより自動起動し，システム制御テーブル上に保有している業務単位の終了フラグの該当業務ビットがセットされているかを一定間隔でチェックする．終了フラグがセットされている場合は「先行ダミー処理」自身は終了し，以降ネットワークに従って日次バッチ処理を自動走行させる．

処理の概略は図17，処理例は図18のとおりである．

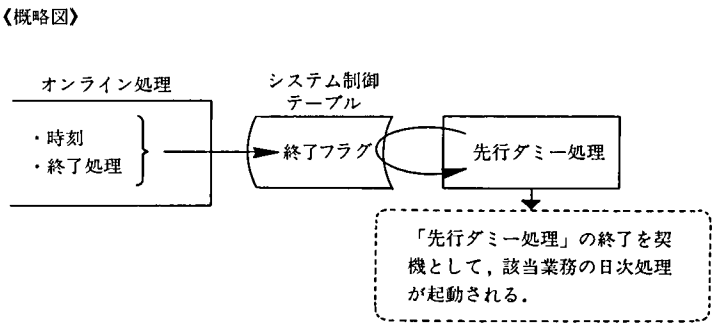


図 17　先行ダミー処理

Fig. 17　A dummy program preceding the daily routine

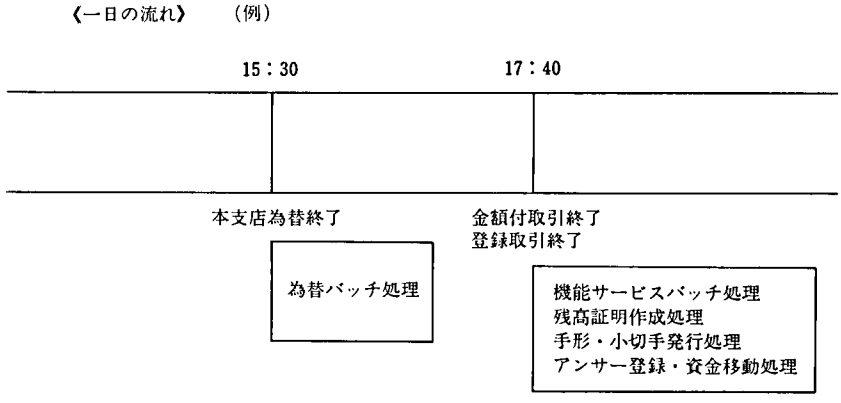


図 18　バッチ処理の起動例

Fig. 18　Example of starting a batch program

5.1.3　センタカット

業後に実行していた翌日分C/Cは，先日付機能によりオンライン中の処理とした．

先日付データは，オンライン処理の中で優先順位を付けて実行される．したがって予約型の先日付C/Cの実行は，優先順位に関係なく実行することができる．

先日付センタカットの中でも実更新型（固定性，融資）の処理は，C/Cから確定日までに固定性，融資の取引がないことを前提として作られている．

したがって実更新型のC/Cは

・金額付取引終了

・登録取引終了

の後に実行されなければならない．

5.2 月次処理

月次バッチの入力となるバッチマスタ（以下BMと呼ぶ）の作成にあたっては，先日付センタカットの採用により残高確定処理が必要であり，そのためにTRITONではCMF内の先日付/入金待ちデータについて以下の残高確定処理を行い，BM内容を確定している．

5.2.1 流動性預金に対する先日付処理（予約方式）分の確定

先日付処理の確定とは解放処理を意味しており，先日付残高スレーブがリンクされている口座が，該当先日付取引の期日（勘定日）到来後に動きがあった場合，実更新を行う処理を言う．

仮に，先日付残高スレーブがリンクされたままの口座が月末日まで何の動きもなく，解放処理が行われなかった場合，その内容は口座マスタに反映されない．

このため月末確定を行う際，当該先日付残高スレーブについては，BM作成処理にて解放処理を行う．

ただし，この解放処理はBM確定用であり，リアルCMF内の先日付残高スレーブを解放するものではない．

また，翌月分の先日付処理がすでに実行されている場合は，その翌月分の先日付残高スレーブの内容は，口座マスタには反映しない．

CMF確定BM作成の際，すべての先日付明細スレーブは削除処理のみ行う．

また，CMF確定BMは流動性口座マスタ内取引明細を保有しない．

5.2.2 固定性預金および融資に対する先日付処理（実更新方式）分の確定

固定性預金および融資は，実更新方式にて処理が行われる．

オンラインにおける実更新は，実取引と同様に取引処理時点での口座マスタの更新を行う．

たとえば，融資の自動返済のような連動取引を先日付処理する場合，流動性マスタ側は予約取引処理を行い，先日付スレーブを保有する一方，融資マスタ側は実更新処理が行われる．

CMF確定BMを作成するためには，この実更新された口座マスタに対して，BM補正ファイルを使用して補正（戻し）処理を行う．

なお，ここで使用されるBM補正ファイルとは，オンラインDBの実更新前レコード（ビフォア・ルック・レコード；BLレコード）イメージを保有する補正TRX（BLレコード）より，CMFファイルに関する最古のBLレコードを抽出・作成したファイルである．

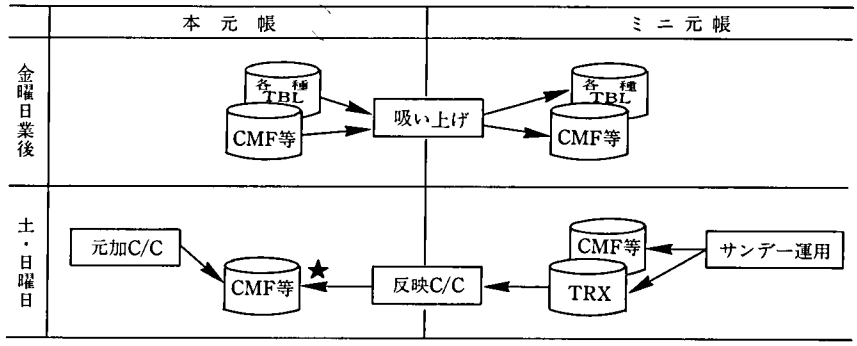
5.3 随時処理

5.3.1 元加処理

流動性預金に対する元加処理については，先日付C/C処理にてサンデーバンキング中の土曜日に本元帳に対して実行する．

処理時点の残高を決算日（翌日曜日）の最終残高と仮定して，決算日までの利息計算を行い，確定日が月曜日の先日付明細を作成する．

元加C/C実行後において，サンデーバンキングで取引が発生した口座に対しては，反映C/C取引を契機として，利息元加額の修正処理を行う．



★ 反映C/C時にTPS処理で元加訂正と再元加処理が行われる.

図 19 元加処理概要

Fig. 19 Profit adding procedure

作成された先日付明細スレーブは，利息決算日以降の最初の取引時に解放処理により，取引明細に反映させる.

元加処理の概略は図 19 のとおりである.

24 時間 365 日稼働時に，ミニ元帳方式を使用しない場合は，本元帳でサンデーバンキング中に，元加 C/C 処理を行う.

5.3.2 金 利 変 更

金利変更処理は，オンライン機能として TPS(Transaction Processing Segment；トランザクション処理プログラム）に組み込まれている.

金利変更日以降，最初の取引を契機として金利変更処理を行った後にその取引を行う.

流動性預金において，取引が発生しない口座については，何らかの強制処理が必要であるが，年 2 回の元加処理で必ず追いつくので，とくに強制処理は行わない.

このため，利息計算期間の金利変更回数分の金利履歴を保有している.

固定性預金は，次回継続時より新金利適用となるため，特別な考慮は不要である.

また，融資の固定金利口座については，一括金利変更は基本的に発生しない．個別対応としては端末取引による「金利変更処理」で対応する.

融資の変動金利口座で取引が発生しない口座は延滞している口座である．延滞中は金利変更対象外と規定しているので，更新処理は不要であり，強制処理の必要はない.

6. 今後の課題

TRITON において，24 時間 365 日稼働に対するアプリケーションの主要機能は実現することができた.

しかしながら，24 時間 365 日稼働に対しては，解決すべき課題が残っている.

6.1 アプリケーション共通機能における課題

・セットアップ/ターミネーションの時間短縮（限りなく 0 へ）

午前 0 時の日付切替時，日付の更新，業務トランザクション・ファイルの切替，プログラムの入替等の処理時間短縮を図る必要がある.

6.2 システム基盤における課題

1) ソフトウェアのバージョンアップ

オンライン業務処理プログラムは基本的に入れ替え可能であるが，OS・UDS・MCB等のインフラソフトウェアの入れ替えに対する対応が必要である．

2) ネットワークの変更

オンライン中の入れ替えに対する対応

3) ハードウェアメンテナンス

メジャーコンポーネント（CPU，IOP，メモリ，ディスク）の切り離し不可能な場合の対応

7. おわりに

金融機関のオンライン稼働時間延長サービスは，24時間365日を目指して拡大してきた．

TRITONは，24時間365日稼働の基盤整備が終了し，制限付きではあるが，その運用時間と運用形態を変更するだけで対応できる仕組みが整えられた．

しかし，開発の目的にある「顧客の利益」あるいは「銀行の利益」を極力保証するとすれば，完全24時間稼働とするのか，7時～23時の運用なのかは議論の余地があろう．

最後に本稿をまとめるにあたり，技術的アドバイスを頂いたTRITON関連部の方々，それに，最初から最後まで協力頂いた黒田亘氏に深く感謝の意を表したい．

参考文献
[1] 村田豊彦，“24時間運転”，ユニシス技報第16号，1988.2.
[2] ニッキン10月15日「貯蓄と消費に関する世論調査」（貯蓄広報中央委員会）1993.

執筆者紹介 中北晴久（Haruhisa Nakakita）
昭和24年生，47年中央大学理工学部物理学科卒業．同年，日本ユニシス(株)入社．金融機関のオンライン・システム開発に従事．現在，金融システム企画開発本部TRITON適用推進部第2課課長．

銀行における集計レス・伝票レス

An On-line Banking System Free from the Need to Check Closing Balances and from Sorted Journals

関口賢造

要　約　銀行業務の中でも最もシステム化が遅れている分野は，営業店での精査に係わる各種事務処理である．

TRITONでは，当初よりこの事務処理の合理化を重点課題の一つとして検討し，開発されたシステムが集計レス・伝票レスである．

集計レス・伝票レスを実現するためには，勘定の発生するあらゆる処理を対象とする必要があり，各種新機能の採用が必要となった．

集計レスの基本機能をまとめると以下の通りとなる．

1) 受付単位のバランスチェック
2) 回付処理
3) 現金のオンライン化と係別精査
4) 障害時対応
5) 他系システムとの照合

集計レスの実現は，精査事務に係わる伝票の分類・集計を不要とするが，伝票を仕訳元帳として存続させるならば，仕訳元帳作成のための分類・集計・一括伝票起票等は営業店事務として存続することになり，目的が達せられない．

そこで，センタでは仕訳元帳に替わる日記帳を自動作成し，仕訳元帳を廃止する事により，営業店終了後の事務処理はより一層合理化・短縮された．

このセンタ自動作成の日記帳をシステム伝票と呼び，営業店において従来型の仕訳元帳が廃止された事を伝票レスと呼んでいる．

なお，TRITONの集計レスは開発保守に関し，次の特徴を持つ．

1) 複数取引においても，各科目の取引は個別に完了する方式であり，一括処理方式に比べセンタの仕組みが単純であり，開発負荷が少ない．
2) 日計サブシステムが提供する共通ルーチンは，集計レスに係わる諸機能を吸収するため，各科目の業務処理モジュールはとくに意識する必要はない．
3) 専用回付の採用により，特殊な業務に対しても木目細かな対応が可能であり，新規業務に対する機能追加時の対応も容易である．

Abstract　Least computerized of all the banking applications is a variety of clerical work involved in the checking of closing balances at financial institutions. The development of the TRITON system was focused, from the very beginning, on the streamlining of this field as one of the challenges, and the newly created system is so designed as to remove needs for the manpower checking of closing balances and for sorted journals.

Making those needs feasible required a thorough study of all transactions related to accounting, resulting in the system's adoption of new functional capabilities. In summary, the basic functionalities

include:

1) Balance checking for each transaction at a front office
2) Passing balance checking on to a back office from a front office
3) Incorporation of cash handlings and individual balance checking into the on-line system
4) How to deal with breakdowns of central hosts or terminals
5) TRITON's checking of balances in linkage with other applications systems

The new system serves to get rid of the need to sort and add up the slips required for balance checking, but if those slips need to be kept as journal ledgers, then traditional clerical work such as sorting, adding-up and issuing collective slips continues to exist. Then, the newly developed system falls wide of its initial aims. So, it is the central computer center that plays a role in abolishing journal ledgers by automatically keeping diaries to replcase them, leading to a further improvement and reduction in the work volume at a front office after it is closed. Those diaries are called system slips, and no existence of journal ledgers at a front office is termed a 'journal-less' way of doing.

In addition, TRITON's balance checking-free system provides the following features for applications development and maintenance:

1) The system is so built as to process multiple transactions all on an individual transaction basis; thus making the center's opertions easier and less loaded for development efforts than in a batch-processing environment.
2) The common routines embedded in TRITON's daily accounting subsystem, which has absorbed functions for unrequired balance checking, has enabled users to stay unaware of sub-routine modules for an individual transaction.
3) The adopted special function for the passing of balance checking has enabled users to respond to specific requirements in a more efficient way, and react properly to needs for the addition of new functions for new applications.

1. はじめに

銀行におけるオンライン・システムの歴史は古く，昭和40年の普通預金の単科目オンラインに始まり，現在では銀行で扱うほぼ全商品がオンラインの対象となっている．

一方，得意先担当者を支援するための渉外支援システム，本部・営業店を対象とした情報系システム等も充実してきた．

このような状況のなか，銀行は現金を主たる取扱い商品としているにも係わらず，唯一合理化に取り残された分野が営業店終了後の現金合わせを主とした精査と言われる作業であった．

システム化が最も進んでいると言われる銀行ではあるが，営業時間終了後，発生した伝票を科目別・入払別に分類し，分類単位に伝票を手集計し，コンピュータの集計結果との突合を行い，最後に現金照合を行うのである．

これらの作業を「精査」または「締上」と呼んでいる．

この精査の考え方を抜本的に見直し，取引終了後の仕訳・集計を一切なくし，営業店の事務を合理化したシステムが集計レス・伝票レスのシステムである．

TRITONは共同開発の当初，重要課題のいくつかをタスクフォースとして走らせたが，この「集計レス・伝票レス」もアプリケーション・システムの主要テーマとしてタスクフォースとして発足した．

当時，すでに某銀行の第三次オンラインの目玉として集計レス・伝票レスのシステムが導入されていたが，TRITON では如何に開発負荷が少なく，かつ効果的な集計レス・伝票レスを実現するかを主題に検討が進められた．

2. 集計レス実現のための機能

集計レスを実現するためには，全業務を網羅した形で実施しなければ意味をなさない．

また，集計レスを実現するために，営業店における個々の取引の操作性を低下させることも逆効果であり，避けなければならない．

営業店端末の操作性，窓口事務の利便性もよく，かつ開発負荷の少ない集計レスを実現するためには，営業店端末，ホスト勘定系システムの双方に以下に述べる各種機能を必要とする．

2.1 営業店端末の機能

集計レスを実現するために端末機能は，以下に示す項目の通りである．

なお，機能の詳細は 2.2 節の「勘定系システムに必要な機能」の記述の中で説明されるため，ここでは省略する．

1) 処理モード
   - 一線モード/二線モードの設定
   - 単独現金処理モード/複数処理モードの選択
   - 障害モードの設定
   - 回復モードの設定

2) 端末カウンタ
   - 構造……2.2.1 項の「4)複数処理のセンタの仕組みの①端末カウンタ」参照
   - 待避処理
   - ローカル照会

3) 端末ステータス
   2.2.1 項の「4)複数処理のセンタの仕組みの②端末ステータス」参照

4) 現金を処理する特殊画面
   - 現金受入・放出画面
   - 回金用画面
   - 両替用画面

5) キー追加
   - 回付枠取りキー
   - 待避キー
   - 複数キー

6) 電文内容付加
   センタへの上り電文に端末ローカルカウンタを付加する取引
   - 端末精査
   - 障害回復時端末通知

7) 障害対応

・カウンタ復旧（回復モード）
・ローカル出納処理（障害モード）

2.2 勘定系システムに必要な機能

2.2.1 受付単位のバランスチェック

集計レスの原理は，顧客との取引において受付単位の取引が間違いないことを保証することにより，1日の取引集計はこの受付単位の取引を累積すれば，終了時点では精査のための集計が不要になるとの考え方による．

そのために，受付単位の完結化を図っている．受付単位の完結処理は，入払のバランスチェックを端末ソフトおよびセンタにより実施するが，一線処理モードと二線処理モードにより処理形態が異なる．

1) 一線処理におけるバランスチェック……一線処理での取引は，すべて現金扱いの取引と見なし，処理を行う．処理の方式には以下の2種類がある．

① 単独現金処理

受付単位の処理件数が1件だけの現金取引を単独現金取引と呼ぶ．単独現金取引では，科目の取引と現金の移動が必ず一致する．現金の受入額（または放出額）と科目の取引金額のチェックは端末側で行い，取引額より現金受入額の方が多い時は端末において釣銭処理を行う．

基本的には従来型の一線処理と同様の処理となるが，TRITON では取引科目と現金の連動処理と見なし，センタに保有する端末カウンタの現金有高更新を行なうとともに，日計/精査カウンタの科目および現金用カウンタを更新する．

【例】 現金1万円を持参し，普通預金に8千円入金

端末処理機能		センタ処理機能
① 現金受け入れ処理（AD または手許） 1万円		
② 端末カウンタ更新処理 現金有高カウンタ更新 ・1万円受け入れ		
③ 電文送信（普通預金入金 8千円）	→	① 電文受信
		② 端末カウンタ更新 (現金有高カウンタ) ・8千円受け入れ (取引累積カウンタ) ・8千円入金
		③ 精査・日計カウンタ更新 ・普通預金 入金 8千円 ・現金受け入れ 8千円
④ 電文受信	←	④ 電文送信
⑤ 釣銭処理（AC または手許）2千円		
⑥ 端末カウンタ処理 現金有高カウンタ更新 ・2千円放出 取引累積カウンタ更新 ・8千円入金		

② 複数処理

受付単位の処理件数が複数の時の処理形態を複数処理と言う．複数処理の取引は以下の流れとなる．

- 複数開始宣言（複数キー押下）
- (開始現金受入処理)：顧客より現金を受け入れる時
- 科目取引処理
- 複数終了宣言（複数キー再押下）
- 終了現金受入処理，または終了現金放出処理

なお，一線処理での初期状態は単独現金処理モードである．

【例】 普通預金出金 100 万円・定期新規 80 万円・定積入金 5 万円・現金出金 15 万円を同時に受付

端末処理機能		センタ処理機能
① 複数開始キー押下		
② 電文送信（普通預金出金 100 万円）	→	① 電文受信 ② 端末カウンタ更新 ・バランスチェック・カウンタ初期化 ・バランスチェック・カウンタ更新 出金 100 万加算 100 万 | ・累積カウンタ 出金 100 万加算 ③ 精査・日計カウンタ更新 ・普通預金出金 100 万
③ 電文受信	←	④ 電文送信
④ 端末カウンタ更新 ・バランスチェック・カウンタ 出金 100 万加算 100 万 | ・累積カウンタ 出金 100 万加算		
⑤ 電文送信（定期預金新規 80 万円）	→	⑤ 電文受信 ⑥ 端末カウンタ更新 ・バランスチェック・カウンタ 入金 80 万加算 100 万 | 80 万 ・累積カウンタ 入金 80 万加算 ⑦ 精査・日計カウンタ更新 ・定期預金新規 80 万
⑥ 電文受信	←	⑧ 電文送信
⑦ 端末カウンタ更新 ・バランスチェック・カウンタ 入金 80 万加算 100 万 | 80 万 ・累積カウンタ 入金 80 万加算		
⑧ 電文送信（定期積金入金 5 万円）	→	⑨ 電文受信 ⑩ 端末カウンタ更新 ・バランスチェック・カウンタ 入金 5 万加算 100 万 | 85 万 ・累積カウンタ 入金 5 万加算 ⑪ 精査・日計カウンタ更新 ・定期積金入金 5 万
⑨ 電文受信	←	⑫ 電文送信

⑩ 端末カウンタ更新 ・バランスチェック・カウンタ 入金　5万加算 100万 | 85万 ・累積カウンタ 入金　5万加算	
⑪ 複数終了キー押下	
⑫ 現金放出画面自動表示	
⑬ 電文送信（現金放出15　→ 万円）	
	⑬ 電文受信
	⑭ 端末カウンタ更新 ・バランスチェック・カウンタ 現金放出15万加算 100万 | 100万 ・累積カウンタ 現金放出15万加算
	⑮ 精査・日計カウンタ更新 ・現金放出15万
⑭ 電文受信	← ⑯ 電文送信
⑮ 現金放出（ACまたは手許）	
⑯ 端末カウンタ更新 ・バランスチェック・カウンタ初期化 ・累積カウンタ 現金放出　15万	

2)　二線処理におけるバランスチェック……二線での取引はすべて振替処理とみなし，現金の授受は発生しないものとする．

現金取引は発生しないため，単独現金処理は取り扱えない．

二線処理における複数処理は，基本的には一線における複数処理の考え方と同じであるが，二線モードにおける取引の性格から次の違いがある．

- 二線処理での初期状態は複数処理モードである．
- したがって，複数開始宣言は不要である．
- 複数終了宣言時，バランスチェックを行った時点で入り払いに差額がある時はエラーを表示する．
- バランスチェックがOK（入り払い差額がない）の時は複数処理の初期状態となる．

3)　複数処理のセンタ対応方式……複数処理のセンタ対応方式として次の2方式の考え方があるが，TRITONでは開発負荷を考慮し，各科目取引が都度独自に完結する個別処理方式を採用した．

①　一括処理方式

複数の科目取引が端末の操作時に同時に処理され，端末のバランスが取れていることを確認後センタに電文が送信され，センタは複数の科目取引を一括処理する．

同時に処理可能な件数はシステム上決まっており，一定件数を越えた複数取引は，後述の回付処理を利用し対応する（同時可能件数は6件程度）．

なお，訂正も一括訂正が原則である．

②　個別処理方式

個々の科目取引は発生のつどセンタで処理され，バランスチェックは，複数処理終了宣言時に，端末により行われた後センタに報告される．

両方式の特徴は次の通りである．

項　目	一括処理方式	個別処理方式
開発負荷	センタの構造が複雑になり開発負荷が大きい． 端末の改造度合が大きい．	集計レスのための特別な構造は必要なく，機能の追加で済むため，開発負荷は小さい． 端末の改造は従来の一線処理の改善程度でよい．
操作性(通常取引)	一括処理のため端末処理効率が良い(認証印字は1伝票)．	個々にセンタの処理が発生するため端末処理効率は悪い(認証印字は処理件数分)．
操作性(訂正取引)	一括処理のため全体を取り消す場合は効率が良いが，正しい処理も含めて訂正が発生する無駄がある．	複数処理の中で該当取引だけを訂正すれば良いため小回りが効く．
現金処理方式	バランスを保証された取引しか発生しないため，とくにセンタで意識しなくてよい．	バランスを保証するため，現金の授受も独立した取引としてセンタ処理する必要がある．

4)　複数処理のセンタの仕組み

①　端末カウンタ

端末カウンタの構造は以下の通りである．

(端　末)

バランスチェック
他店券照合　＊
現金有高
取引累積
待避用バランスチェック
管理情報 (複数通番・待避通番等)
障害時現金処理

(センタ)

バランスチェック
他店券照合　＊
現金有高
取引累積
待避用バランスチェック
管理情報 (伝票編綴通番・担当者名等)

＊バランスチェック・カウンタ上の他店券の受け入れ，または放出金額と科目取引における他店券取引との金額一致を確認するためのカウンタ．

②　端末ステータス

端末とホストの複数処理の同期をとるため以下の端末ステータスを準備し，端末からの上り電文に付加する．

種類	意　味
0	複数処理の一件目の取引(待避無し)
1	複数処理の二件目以降の取引(待避無し)
2	複数処理の一件目の取引(待避有り)
3	複数処理の二件目以降の取引(待避有り)

③　複数処理中割り込み

大量の取引途中に緊急な取引が発生した時等，1回だけ割り込み可能とする．端末にて待避キーが押下されると，バランスチェック用カウンタが待避用バランスチェック・カウンタに保存され，別の複数処理または単独現金処理の

実行が可能となる．

割り込みの複数処理，または単独現金処理が終了した時点で，待避キー再押下により割り込みは終了し，仕掛かり中のバランスチェック・カウンタが復元される．

2.2.2 回 付 処 理

集計レスの基本は前述受付単位の完結処理であるが，同時に大量の処理を受付た場合，または複雑な取引を受付た場合等は，一線での完結処理を行なうと窓口が混雑し営業店の運用に支障を来す．

こうした時は，一線は現金処理あるいは流動性預金等の単純取引のみを行い，その他の取引は二線（後方）に依頼し処理を行なう．

このように，一部を後回しにして処理する形態を回付処理と言う．

回付処理には，一般回付と専用回付の2種類が用意されている．

1) 一般回付……テラーの判断により，受付単位の処理の一部を回付する処理を一般回付と呼び，以下に示す内容である．

① 回付枠取り

端末オペレータが，受付単位の処理の一部を回付処理したい時に回付番号を採番し，同時に回付勘定を発生させるための処理である．

回付枠取りの方法には，次の2種類の方法がある．

【例】 普通預金100万円出金と定期預金新規50万円2件を受付け，定期預金を回付する．
① 回付枠取りキーを押下し，普通預金出金100万円のオペレーション実施．
② ・普通預金出金100万円のオペレーション実施．
・回付枠取りキーを押下し，回付入金100万円のオペレーション実施．

② 追加枠取り

回付枠取りで確保した資金だけでは回付する取引の資金として不足する時の枠取り金額の追加処理である．

回付入出金取引で回付番号を入力し，回付枠取りキーを押下することにより追加枠取りの処理が行われる．

③ 回付番号採番帯　　101番～999番（店別）
回付枠取りされる都度採番される．

④ 対象業務　　特定されない．

2) 専用回付……最初に述べた通り，集計レスを実現するためには，営業店のあらゆる取引を網羅して対応しなければならない．業務によっては一線完結処理に向かない業務もある．これらの個々の必要性により業務個別の専用回付が準備されている．

専用回付の特徴は以下のとおりである．

① 回付番号：1～99番までの固有番号を店別に割当てる．
② 画　　面：業務により専用画面が準備されている．
③ 伝　　票：業務により専用伝票が準備されている．
④ 対象業務：専用回付番号ごとに業務が特定される．

また，専用回付の利用目的と代表的な例は以下の通りである．

目的	専用回付名称	回付番号	利用方法
現金管理	出納専用回付	1	出納元方の現金移動を伴なう取引に使用.
	オフテラー専用回付	21～50	オフテラーの現金移動を伴なう取引の時に使用.
複数部署管理	地区センタ専用回付	2	営業店より地区センタにテレ為替の仕向処理を代行依頼する時に使用.
	本部専用回付	10	複数本部間で収支がバランスする処理の時に使用.
障害対応	端末不能専用回付	98	端末の回復不能な障害発生時の仕掛中取引に対応.
	障害回復専用回付	99	センタ障害時の端末ローカルによる出納処理に対応.
特殊業務対応	テレ為替専用回付	3	テレ為替の資金受入と送信処理の時間差のバランスを保証.
	税金等収納専用回付	4	各種税金を受入処理時グロスで仮処理し，業務終了時に種類別に仕訳処理するために使用.
	文書為替専用回付	5	文書為替を受入処理時グロスで仮処理し，業務終了時に種類別に仕訳処理するために使用.
	予約取引専用回付	6	端末による予約処理の実行，センタカット時のバランスを保証するために使用.
	融資専用回付	9	手貸の特殊な書替・現金処理等融資に係わる回付処理が発生した時に使用.

3) 明細記帳処理……回付された取引の後処理を明細記帳処理と言う.

明細記帳処理は複数モードで行い，回付番号を入力することにより，明細記帳処理であると認識される.

ただし，専用回付の明細記帳処理で専用画面が準備されている場合は，回付番号の入力を省略できる取引も存在する.

明細記帳処理は，常に回付勘定（ダミー勘定）との連動取引となる.

回付の種類により以下の特徴がある.

① 一般回付時の特徴：回付枠取り時の反対勘定，および反対勘定の訂正のみ取引可能.

たとえば，普通預金出金時に回付枠取り（回付入金勘定発生）した取引の明細記帳は，入金または入金訂正のみ取引可能.

回付枠取り金額を越えた明細記帳処理はエラーとなる.

② 専用回付時の特徴：入払混在可能.

特定の業務のみが対象.

回付枠取り・追加枠取りは行えない.

他の回付との振替処理が可能である.

4) 回付処理センタの仕組み

① 勘定処理

ダミー勘定として回付勘定を使用する.

回付勘定は端末カウンタ・精査カウンタ・日計カウンタ上は通常の勘定科目と同様，実更新の対象となるが，日計ファイル（総勘定元帳ファイル）上は存在しない.

営業店終了時点で回付勘定の入り払いが一致していない時は，営業店精査は

終了できない.

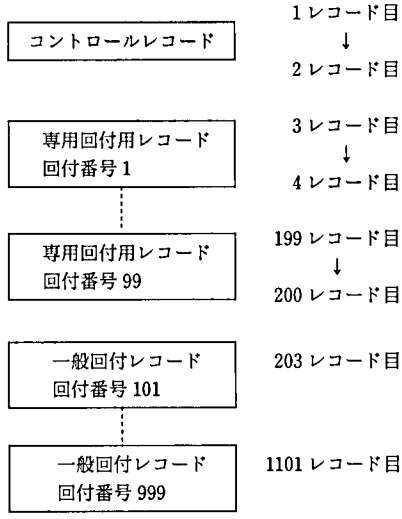
② 回付明細ファイル

回付処理に関するバランスチェックは，回付明細ファイル上で管理する.

回付明細レコードのフォーマットは，一般回付・現金管理用専用回付・地区センタ用専用回付・その他専用回付の4種類に分かれる.

回付明細ファイルの構造と主要項目は以下の通りである.

・各店舗ごとの回付明細ファイルの構造



・各レコードの主要項目

	コントロールレコード	専用回付レコード	一般回付レコード
	・店番 ・基準日 ・NEXT回付番号 ・オフテラー情報 ・オフアート情報 ・一般回付処理済情報	・設定回付番号 ・店番 ・端末機番 ・職員コード ・入出金区分 ・開局表示 ・処理済表示 ・基準日 ・回付名称 ・取引累計情報 (自店分) ・前日繰越現金有高情報 ・開局オペ情報 ・現金有高情報 ・取引累計情報 (地区センタ分) ・回付残高情報	設定回付番号 店番 端末機番 職員コード 入出金区分 開局表示 処理済み表示 基準日 回付名称 当初枠取り情報 追加枠取り情報 明細記帳情報 明細記帳訂正情報 回付残高情報

③ 回付入出金処理

回付入出金勘定の発生する取引は以下の通りである.

・回付枠取りおよび回付追加枠取り（連動）

・明細記帳処理（連動）

・回付入出金（原則単独の入金・または出金）

【例】 普通預金出金 100 万円・定期新規 80 万円・現金出金 20 万円を同時に受付け，定期預金を二線に回付.

端末処理機能		センタ処理機能
一線処理		
① 複数開始キー押下		
② 電文送信（普通預金出金 100 万円）	→	① 電文受信 ② 端末カウンタ更新 ・バランスチェック・カウンタ初期化 ・バランスチェック・カウンタ 出金 100 万加算 100 万 ・累積カウンタ 出金 100 万加算 ③ 精査・日計カウンタ更新 ・普通預金出金 100 万
③ 電文受信 ④ 端末カウンタ更新 ・バランスチェック・カウンタ 出金 100 万加算 100 万 ・累積カウンタ 出金 100 万加算	←	④ 電文送信
⑤ 電文送信（回付入金 80 万円・回付枠取りキー押下）	→	⑤ 電文受信 ⑥ 端末カウンタ更新 ・バランスチェック・カウンタ 入金 80 万加算 100 万 | 80 万 ・累積カウンタ 入金 80 万加算 ⑦ 回付番号採番（101 番） ⑧ 回付明細レコード更新 ・枠取り金額 80 万 80 万 （枠取り） ⑨ 精査・日計カウンタ更新 ・回付入金 80 万
⑥ 電文受信 ⑦ 端末カウンタ更新 ・バランスチェック・カウンタ 入金 80 万加算 100 万 | 80 万 ・累積カウンタ 入金 80 万加算 ⑧ 複数終了キー押下 ⑨ 現金放出画面自動表示	←	⑩ 電文送信
⑩ 電文送信（現金放出 20 万円）	→	⑪ 電文受信 ⑫ 端末カウンタ更新 ・バランスチェック・カウンタ 現金放出 20 万加算 100 万 | 100 万 ・累積カウンタ 現金放出 20 万加算 ⑬ 精査・日計カウンタ更新 ・現金放出 20 万
⑪ 電文受信 ⑫ 現金放出（AC または手許） ⑬ 端末カウンタ更新 ・バランスチェック・カウンタ初期化	←	⑭ 電文送信

・累積カウンタ
現金放出　20万
二線処理
① 電文送信（定期預金新規　→
80万円・回付番号101
入力）

② 電文受信
③ 端末カウンタ更新
・バランスチェック・カウンタ
入金　80万加算
出金　80万加算
80万 | 80万
・累積カウンタ
入金　80万加算
出金　80万加算
④ 複数終了キー押下
⑤ 端末カウンタ更新
・バランスチェック・カウンタ初期化

① 電文受信
② 端末カウンタ更新
・バランスチェック・カウンタ初期化
・バランスチェック・カウンタ
入金　80万加算
出金　80万加算
80万 | 80万
・累積カウンタ
入金　80万加算
出金　80万加算
③ 回付明細レコード更新
・明細記帳金額　80万加算
80万 | 80万
（明細記帳）（枠取り）
④ 回付処理完了フラグセット
⑤ 精査・日計カウンタ更新
・定期預金新規80万
・回付出金　80万
← ⑥ 電文送信

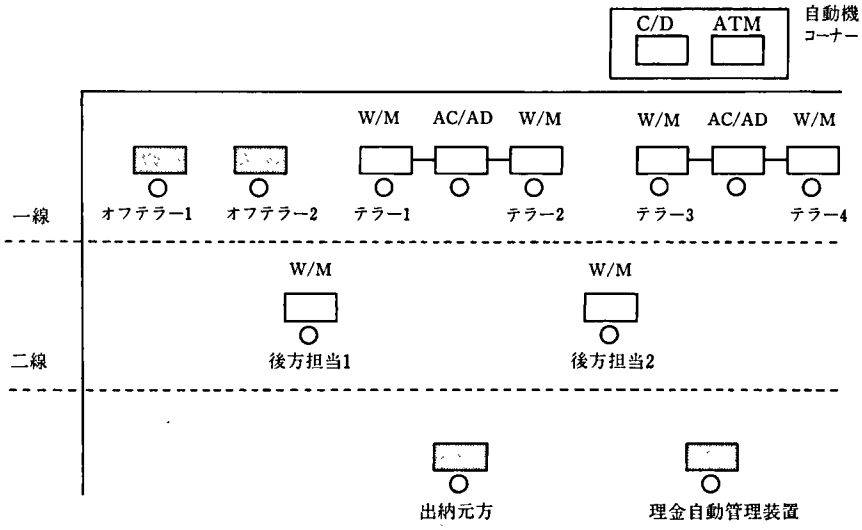
2.2.3 現金のオンライン化

1) 現金管理の単位……営業店レイアウトを図1の通り想定すると，○印の付いている係が現金管理の単位に相当する．
なお，網がかかっているところの係はオンライン端末を持たない係を意味する．

2) 現金管理のセンタの仕組み……オンライン端末を使用する係（オン・テラー/オン元方/AC・AD）の現金管理は，ローカル端末カウンタおよびセンタ端末カウンタで実施する．
オンライン端末を持たない係（オフ・テラー/出納元方/オフアート）の現金管理は回付明細レコードで行う．

3) 現金のオンライン化に伴なう取引の種類
センタで現金処理を行う取引の種類は以下の通りである．
・単独現金取引
・複数開始現金受け入れ
・複数終了現金受け入れ
・複数終了現金放出
・出納専用回付使用の取引
・オフテラー専用回付使用の取引

・オフアート（現金自動管理装置）専用回付使用の取引
・回金
・テラー端末開局（開始時の出納元方からの回金）
・端末再起不能通知
・障害回復時端末通知
・他店券両替



W/M：Window Machine　AC/AD：Auto Cashier / Auto Deposit
自動機：自動機は仮払口座と1対1に対応し，自動機取引は外部勘定扱いとなる．自動機内の現金は自動機管理ファイルで集計レスとは別の仕組みで管理されている．

図 1　営業店レイアウト例
Fig. 1　Sample——Layout of the office

2.2.4 障 害 対 応

1)　端末障害対応……複数処理のバランスチェックおよび現金管理は端末主体で行っているため，複数処理の取引の一部を処理した時点で端末に障害が発生した場合，通常の処理では複数処理のバランス保証，該当テラーおよびAC・ADの精査ができなくなる．

そのため，復旧不能の端末障害が発生した時は，以下の手順で仕掛かり中の残取引を完結させる．

①　端末再起不能通知処理

・センタの端末カウンタから仕掛かり中取引の入り払い差額を算出
・入り払い差額を端末不能専用回付に入金，または出金処理　入り＞払いの時，出金処理 入り＜払いの時，入金処理
・該当端末の現金残高（手許/AC・AD含めて）を出納に回金

②　残った取引を端末不能専用回付を使用して，明細記帳処理実施．

③　端末不能専用回付の入り払い一致の確認．

2)　センタ障害対応……センタの長時間ダウンが発生し，テラーが端末ローカル機能で現金処理を実施した時，センタ回復後の該当取引のバランスチェックを以下の手順で実施する．

①　端末ローカル機能での現金処理を実施した端末の障害回復時，端末通知を実施．

当該電文の現金受け入れ・放出の差額を障害回復専用回付に入金，または出金処理． 現金受け入れ＜現金放出　の時，出金処理 現金受け入れ＞現金放出　の時，入金処理

②　上記取引の記帳処理を障害回復専用回付を使用し実行．

③　障害回復専用回付の入り払い一致の確認．

2.2.5　他系システムとの関連

国内勘定系とは別に，他系システムにて勘定移動を伴なう取引が発生するケースは，両系相互においてバランスが取れている事を保証する必要がある．

以下に国際勘定系（他系システム）を参考例にして取扱い方法を述べる．

1)　外為振替勘定……外為振替勘定（ダミー勘定）を創設し，国内勘定・国際勘定の双方に係わる取引が発生した場合には，国内勘定系，国際勘定系双方に外為振替勘定を計上する．

外為振替勘定が発生する形態は以下の通りである．

①　連動処理

国際勘定系からのパスオフ電文による連動処理が発生した場合に，連動処理勘定の反対勘定を外為振替勘定として計上する．(国際勘定系では連動処理勘定と同側勘定を計上)

連動処理として発生する勘定科目は，流動性預金および仮受・仮払である．

②　外為振替勘定入出金

外為振替勘定を外為振替勘定入出金オペレーションにより行うケースは次の通りである．

・外為取引の相手勘定が国内勘定系（流動性・仮受・仮払）であるが，パスオフによる連動は行われず，伝票により処理される時

【例】　国内勘定系普通預金を出金して外貨預金の新規を行う．

国際勘定系　　・外貨預金新規（振替扱い）
国内勘定系　　・普通預金出金
　　・外為振替勘定入金

・外為取引が現金取引の時の現金受け入れ・放出を行う時

【例】　外貨預金新規を現金で行う時

国際勘定系　　・外貨預金新規（現金扱い）
国内勘定系　　・外為振替勘定入金（単独現金処理）

2)　営業店取引終了時の照合……国内勘定系店別精査オペレーション実施時に国際勘定系に連動し，国際勘定系における外為振替勘定の入り払い差額を入手し，国内勘定系の外為振替勘定入り払い差額とチェックする．

終了時，上記照合を行う事により，国内勘定系・国際勘定系を包含したバラン

ス，および現金の管理を行う．

2.2.6 精査と日計処理

1) 係別精査……それぞれの係に応じた精査（締上）用オペレーションを行い，各係の保有する現金とシステム上で管理している現金を突合し，一致していれば完了となる．

代表的な例は以下の通りである．

① オンテラー：以下の確認をテラー端末精査オペレーションで行う．
端末カウンタが仕掛り状態でない．
当該端末で行った回付枠取りが未処理状態でない．
手許現金有高が正しい．

② オフテラー：オフテラー締切登録オペレーションにより，手許現金有高の正当性を確認する．

③ AC・AD ：AC締上票作成オペレーションにより，現金有高の正当性を確認する．

④ 出納元方 ：出納元方締切登録オペレーションにより，現金有高の正当性を確認する．

2) 営業店精査……営業店精査は従来の勘定精査の意味は無く，営業店で当日必要な事務処理が完了しているか否かを確認するいわゆる事務精査である．

主なチェック項目は以下の通りである．

- 全端末精査の終了確認
- 全回付明細の終了確認（現金締上処理の終了確認を含む）
- 仮受・仮払・別段の当日残高繰越不可口座の残高0の確認
- 当座過振未処理口座無し（または過振容認登録済み）の確認
- 国際勘定系の締上終了確認および外為振替勘定バランスの確認
- 為替取扱時間帯の終了確認

3) 現金有高票自動作成……従来精査終了後手書き作成していた営業店全体の現金有高票をセンタで自動作成する．

4) 精査・日計関連ファイル/テーブルの使用目的

種　類	更新方法	使 用 目 的
端末カウンタ	オンライン	・受付単位のバランスチェック ・現金有高管理 ・伝票編綴通番採番
精査カウンタ	ディレード バッチ	・内部勘定精査表作成 ・為替精査表作成 ・予約精査表作成
日計カウンタ	ディレード バッチ	・日計ファイル作成のための中間ファイル ・ファイル残チェック(預金・貸出金等)
日計ファイル	バッチ 一部オンライン	・総勘定元帳作成 ・ファイル残チェック(経費・手数料)
回付明細ファイル	オンライン	・回付処理のバランスチェク ・出納元方・オフテラー等の現金管理

3. 伝票レス実現のための機能

通常，銀行における伝票は，会計帳簿としての仕訳帳として位置づけられている．

伝票が仕訳元帳として存在する限り，営業店の精査終了後に仕訳元帳作成のための伝票仕訳・一括伝票作成等の作業が残る．

この仕訳元帳としての伝票を無くすことにより，伝票レスが実現できる．

また，集計レスと合わせ，より効果的な営業店事務の効率化が実現できる．

3.1 伝票の位置づけ

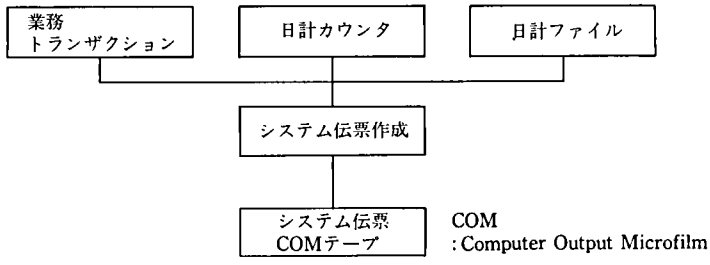
伝票レス実現前後の伝票の位置づけは，次の通りである．

	伝票レス実現前	伝票レス実現後
会計上の位置づけ	仕訳元帳	単なる証憑
保存期限	10年	7年(実務上は10年)

3.2 システム伝票

伝票レスに伴い，従来の伝票による仕訳元帳を廃止したため，仕訳元帳に変わる帳簿としてセンタ作成の日記帳が必要になる．このセンタ作成の日記帳をシステム伝票と呼ぶ．

システム伝票の作成は，以下の通り行われる．



3.3 保管事務の合理化

伝票レスの実現より，営業店における伝票の編綴・検索方法を自由に行うことが可能となった．この編綴検索方法により，営業店の伝票に係わる雑務の負荷が異なる．TRITONでは下記に示す対応により営業店負荷を軽減している．

	伝票レス実現前	伝票レス実現後
編綴方法	科目別・入払別発生順	端末別発生順
伝票通番	科目別・入払別通番	端末別通番(伝票編綴通番)
検索方法	日別科目別に伝票綴りを検索	取引明細照会またはシステム伝票より機番・通番を知り，伝票綴りを検索

3.4 伝票（証憑）の削減

1) 種類の削減……従来科目別・個別に準備されていた伝票を汎用化，統合化することにより伝票の種類を大幅に削減した．

旧システムにおいて約250種類あった帳票の4分の1を削減している．

2) 量の削減……外部勘定取引の拡大，複合伝票の拡大により，営業店で保管が必要となる伝票枚数を削減した．

4. 営業店の組織と運用

集計レス・システムは受付単位の完結処理を基本とした考え方であり，効果的な運用を行うためには，それに見合った組織・権限規程が必要となる．

4.1 係別処理から一線完結処理へ

従来の営業店組織は科目別係の縦割組織が主であったが，科目別係の運用をもとに集計レスを実現すると，係間にまたがる取引を同時に受け付けた場合は回付処理が必要となり，かえって運用効率を落とす危険性がある．

集計レスでの運用は科目別係を廃止し，横割組織として一線完結処理を極力拡大することにより，より効果的な営業店運用が可能となる．

4.2 権限の拡大

テラー権限を拡大し，テラー離席をより少なくすることにより，効率のよい営業店運用が可能となる．

テラー権限の拡大とは以下のことを意味する．

1) テラー取扱可能現金金額の拡大

権限規程上の問題であり，とくにシステム的に対応する課題ではない．

2) 事前役席承認オペレーションの削減

TRITON では従来の役席承認の考え方を役席承認と役席照合とに分類し，役席照合は事後の確認とする事により，事前承認の必要な取引を極小化している．

5. 導 入 効 果

1993 年 5 月に TRITON が稼働し，計数的な分析・評価は今後の課題であるが，営業店事務の改善点，営業店の稼働状況は以下の通りである．

1) 営業店精査事務の改善

	作業項目	導入前	導入後
当日	端末精査	機能なし	○ (随時)
	伝票仕訳	○	不 要
	伝票集計	○	不 要
	営業店精査	精査票と集計結果の照合(中間)	事務処理終了確認
	現金有高帳作成	手書き	センタ作成 手書き補足
翌日	一括伝票作成	○	不 要
	前日勘定補正取引精査	○	○
	営業店精査(前日確定)	○	不 要

2) 営業店事務終了時刻の短縮

① 営業店精査オペレーションの終了が約 10 分早まっている．

② 新システムでは係別現金有高を精査の対象としている事を考慮すると，実質 30 分〜1 時間の合理化効果が出ている．

③ 1993 年 6 月の実績で，90%の店で 5 時に仕事が完了（従来は集計が終わったといっても，照合，事務整理等まだ仕事は残っていた）している．

6. おわりに

TRITONにおける集計レスの仕組みを述べてきたが，最近では各都銀，および一部地銀においても，集計レスが積極的に導入されつつある．

しかしながら，勘定系オンライン構築時に当初から集計レスを設計思想に取り込み開発を行ってきたシステムは，TRITON以外では非常に数少ない．

構築済みのシステムに集計レス機能を追加するケースと当初の設計思想として集計レスに取り組むのとでは，システム構造の合理性，業務面での木目細かさに自ずから違いが出る．

集計レスはTRITONにおける業務面での誇れる機能の一つである．

集計レスに残された今後の課題を以下に挙げる．

1) 出納元方システム，現金自動管理機等，営業店における出納処理合理化のための機器がセンタ端末として接続できないため，二線での代行オペレーション等の無駄が発生する．

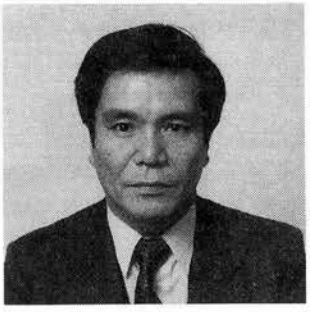
2) 1オペレーションの連動処理は，原則入り払いが一致しているため，バランス・チェックでのオペレーションミス検出は不可能である．現在，連動処理のチェック照合のための仕組みはとくにないが，1オペレーションの連動処理時の確認メッセージ出力等，何等かの対応が必要と思われる．

3) 営業店の組織運用は，一線・二線・元方を明確に分離した運用を想定している．また，オンライン・テラーにはAC・ADが設置されていることが前提の運用となっている．

小規模金融機関では，一店舗に端末が一台しかなく，役割の明確な分離ができないケースや，AC・ADを導入していないケースも多く見受けられており，集計レス導入は十分な検討が必要である．

今後ハードウェアの改善，および実運用の結果を反映したシステムの改善を行うことにより，一層充実したシステムになるだろうと予測している．

執筆者紹介　関口賢造 (Kenzo Sekiguchi)
昭和43年慶応義塾大学経済学部卒業．昭和46年日本ユニシス(株)入社，金融関連フィールドサービスに従事．その後TRITON開発部を経て，現在TRITON適用進進部に所属．



オープン時代の企業情報システムを担う新世代メインフレーム UNISYS 2200/500 シリーズ

日本ユニシスは企業情報システムのオープン化に対応する新しい時代のメインフレームである「OPEN 2200」の第一弾として，UNISYS 2200/500 シリーズを発表した．

ユニシスは，今後の情報システムの課題は，多種多様なシステム群からなるマルチ・システム環境にあり，それらを単一のシステムのように自由に扱っていくことであると考えている．このような世界を，ユニシスは 90 年 11 月に発表した UA (Unisys Architecture) において，「インフォメーション・ネットワーク」，すなわち「統合分散システム」として位置付け，その実現のために早くから国際標準，業界標準の取り込みを地道に推進してきた．

このインフォメーション・ネットワーク実現の基本的な考え方は，オープンシステムと固有システムとの共存・協調であり，シリーズ 2200 上で具体化したのが「OPEN 2200」である．すなわち，「OPEN 2200」は従来のメインフレームの持つ高付加価値機能（インテグリティ・サービス機能）に加えて，国際標準/業界標準を幅広く採用して異なるプラットフォーム上のアプリケーションやデータベースが相互に連携して動作できる相互運用性を可能とすることにより，急速に進む企業情報システムのオープン化に対応できる新世代メインフレームとして開発を進めてきたものである．

共存・協調させる具体的な機能は，メインフレームの特徴であるインテグリティ・サービス（機能）と国際標準/業界標準を採用して体系化されたオープン・サービス（機能）である．

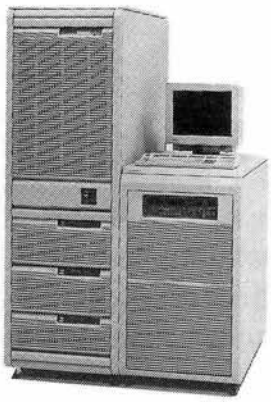
インテグリティ・サービス（機能）

- 大規模トランザクション処理
- 限りないシステムの成長性
- 無停止/連続処理
- システムの無人運転/統合管理運用
- 高度なセキュリティ

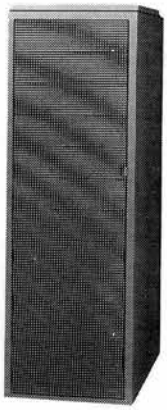
オープン・サービス（機能）

- オープン・デベロップメント
- オープン・分散アプリケーション
- オープン・データ・マネジメント
- オープン・ネットワークキング
- オープン・マネジメント

この「OPEN 2200」の一番手のシステムである UNISYS 2200/500 シリーズは，国際標準の ISO や業界標準の X/Open はもとより，事実上の標準となった技術を積極的に取り込んでいる．また，超大型機 2200/900 シリーズのアーキテクチャ (2200/XPA：拡張処理アーキテクチャ) を最新の CMOS VLSI テクノロジにより中大型機の分野



UNISYS 2200/500 シリーズ



拡張データ処理装置（XPC）

において実現している．

同時に高速な入出力処理とビジネス処理における並列処理/連続処理の実現を具体化する拡張データ処理装置(XPC：eXtended Processing Complex）も発表した．

XPCはまったく新しい考え方のもとで，複数のRISCチッププロセッサの並列処理と大容量メモリにより大規模なトランザクション処理，無停止連続運転等を可能としている．

1. OPEN 2200

「OPEN 2200」はUNIXを中心とした国際標準/業界標準に基づくオープンな機能群とメインフレーム固有の高付加価値機能群とが共存・協調し合う新しいプラットフォームである．ユニシスはこの二つの機能群をそれぞれ「オープン・サービス」,「インテグリティ・サービス」として提供する．

1.1 オープン・サービス

「オープン・サービス」は，UNIXシステム等がもつ相互運用性・移植性・統合運用性を実現するサービス（機能）である．シリーズ2200が提供するオープン・サービス機能は，UNIXやパソコン/ワークステーションとの協調分散処理環境における開発・実行・運用の各段階を支援する機能として，前述した五つのサービスを提供する．

これらのサービスを実現するためのソフトウェア製品は三つの製品体系により提供され，IEEE/POSIX，OSI，X/Open，OSF，OMG，ANSI等の国際標準/業界標準に準拠し，マルチ・ベンダ環境における異機種間の相互接続性，相互運用性，移植性を高める（表1）．

・ASDF（Advanced Solution Development Framework）は92年5月に発表した，有効で使いやすいアプリケーションを構築するために，アプリケーション構築技術自体をオープンシステムを含めた新しい枠組みの中でとらえて，さまざまな開発支援ツール/サービス/方法論を統一し，新しく体系化したもの（開発環境）．

・ACCF（Advanced Cooperative Computing Framework）は93年5月に発表した，分散協調処理の実行環境におけるオープンな相互運用性を提供するためのフレームワーク（実行環境）．

・運用管理体系（計画中）はマルチベンダ/マルチプラットフォームからなるインフォメーション・ネットワークを容易に効率よく統合運用管理するためのオープン時代の統合システム管理体系（運用環境）．

各サービスの概要は次の通りである．

1) オープン・デベロップメント……オープン・デベロップメントはソリューション構築体系ASDFにより，国際標準/業界標準準拠のオープンな開発環境を提供する．

① 2200 POSIX

2200 POSIXは，IEEEで標準OSインタフェースを定義している「POSIX」を採用する．POSIXを採用することにより，標準OS環境がシリーズ2200上に実現でき，POSIXの上で開発あるいは稼働したアプリケーションは，他のUNIXとのアプリケーションのポータビリティ（移植性）を可能とする．当初は「IEEE 1003.1」「10003.2」の規定を提供し，今後は順次拡大予定である．

表1　オープン・サービスの製品体系

オープン・サービス	製品体系	主なプロダクト
オープン・デベロップメント	ASDF（開発）	POSIX, SX 1100, UCS, IDES, MAPPER, LINC
オープン・分散アプリケーション	ACCF（実行）	Open/OLTP(X/Open DTP モデル), DstDDL, DstPRT
オープン・データ・マネジメント		SQL * Star, ユニシス・リポジトリ(ANSI-IRDS), OODB (OMG)
オープン・ネットワーキング		NFS, TCP/IP, MHS, FTAM
オープン・マネジメント	運用管理体系（運用）	CNMS/IM, CNMS/DCA

「POSIX 1003.1」
- ファイル・システム運用管理
- ファイル管理
- 入出力管理
- プロセス管理
- 環境サービス

「POSIX 1003.2」
- シェル機能(コマンド，プログラム・ロード，環境変数)
- ユーティリティ（実行環境，ソフトウェア開発，C言語開発)

② UCS (Universal Compiling System)

UCSは複数の標準言語を一元的に支援する統合コンパイル・システムである．標準準拠のCOBOL，C，FORTRAN等を一元的に支援し，これにより，整合性のある異種言語間連携，統合された効率改善策，異種言語間でのデータ・ファイルの共有，同一のデバッグ方法等が実現可能となる．

③ IDES (Integrated Development Environment Support System)

シリーズ2200をハブ，UNIXをサーバとする分散開発環境を実現し，ターゲット・システム上のアプリケーション・プログラムの開発を支援する「IDES」を提供する．IDESは大規模基幹業務の開発に最適な3GL/CASEであり，リポジトリをベースとする開発環境の構築を支援するとともに，一連の開発工程の作業を支援するツール群によって，システム開発，保守の生産性向上と品質向上を実現する．

④ MAPPER

「MAPPER」は，エンド・ユーザ・コンピューティング（EUC）によるソリューション構築を支援する4GLであり，ユニシスの各種プラットフォームで稼働している．これらのプラットフォーム間におけるMAPPERアプリケーションの移植性は極めて高い．またEUC基盤の強化として，ネットワーク機能，MRI機能の強化，コオペラティブ処理，GUI，DW，等がある．

⑤ LINC

「LINC」は，オンライン・トランザクション処理(OLTP)システム構築の全工程を一貫して支援する基幹業務向けの統合CASEツールである．LINCで構築された基幹データベースをMAPPERデータベースに展開し，EUCの推進が図られる．

2) オープン分散アプリケーション……オープン分散アプリケーションは，分散協調処理体系ACCFにより，国際標準/業界標準に準拠したオープンな分散協調処理の実行環境を提供する（図1)．

① Open/OLTP（X/Open DTPモデル）

オープン分散アプリケーションは，X/Open DTPモデルに基づくトランザクション・マネージャ機能をシリーズ2200上に搭載し，Open/OLTP(分散トランザクション処理）を実現する．Open/OLTPは，分散トランザクション処理には不可欠な二相

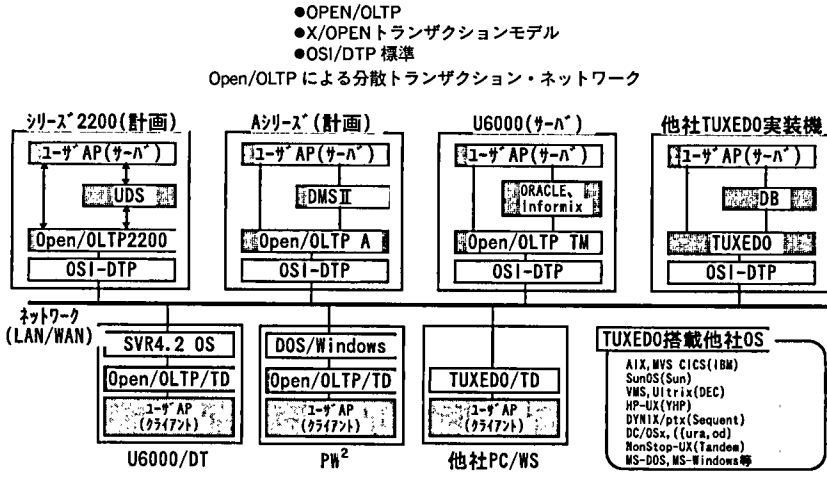


図1 オープン分散アプリケーション（相互運用性）

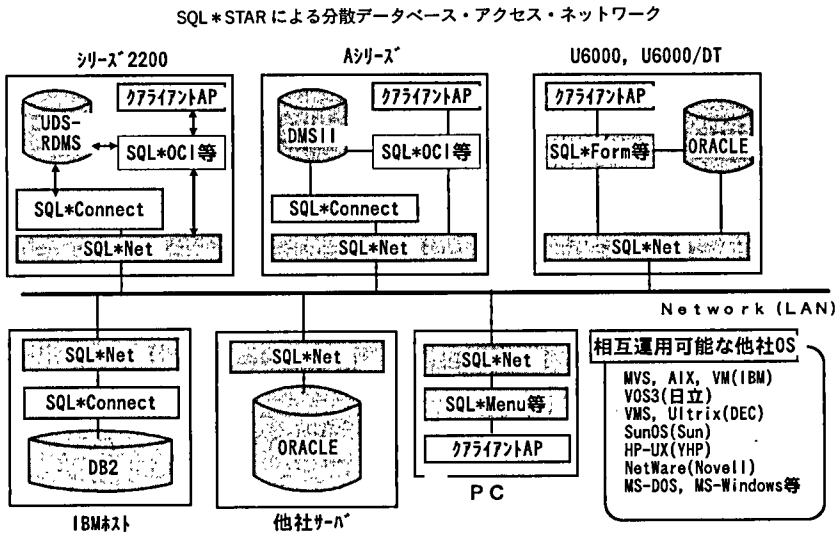


図2　オープン・データ・マネジメント（相互運用性）

コミットメント・プロトコルを実装しており，マルチ・プラットフォーム環境における一貫性のある分散トランザクション処理が可能である．

TUXEDOが実装されているプラットフォームであれば，トランザクション処理の相互運用性が可能である．

② Dst DDL（分散データデリバリ）

ホスト，サーバ，ワークステーションからなるマルチ・プラットフォーム環境において，アプリケーションとの連携が可能な蓄積交換型の分散データ配送環境を提供する．

③ Dst PRT（分散プリント）

様式情報（罫線等の枠情報）と帳票データからなる帳票を分散プリント・サーバに保存・蓄積し，ネットワーク内のどのサーバ，ワークステーションのプリンタからでも自由に出力が可能である．

3) オープン・データ・マネジメント……オープン・データ・マネジメントは，分散協調処理体系ACCFにより，SQLベースのデータアクセス，および標準のリポジトリを取り入れて，異なるプラットフォーム間でのクライアント・サーバ・コンピューティングの実現，分散協調コンピューティング環境を実現する（図2）．

・UDS（Universal Database System）

CODASYL仕様のネットワーク・データベース，ISOのリレーショナル・データベース，階層型データベースを一つのプログラムから同時に使用できる汎用データベース・システム「UDS」を提供してきた．さらにUDSはACCFに基づき業界標準の分散データベース・システム，分散トランザクション・システムと連携し，オープンなデータベース環境を提供する．

その他にも次のプロダクトを提供する．

・IRU（Integrated Recovery Utility）
・SQLベースのデータ・アクセス（SQL＊Net，SQL＊Connect，MRI）
・ユニシス・リポジトリ（UREP ANSI/IRDSベースのリポジトリ）
・分散データベース（RDA）

また，提供する商品は次のような標準に準拠している．

・データ・マネジメント標準
ANSI SQL，SQL ACCESSグループ，OODBTG，ISO/RDA Committee，ANSI/IRDS，POSIX Committee

4) オープン・ネットワーキング……オープン・ネットワーキングは，分散協調処理体系ACCFにより，標準に準拠した製品群により，企業サーバ(インフォメーション・ハブ)，分散サーバ，パソコン/ワークステーションの

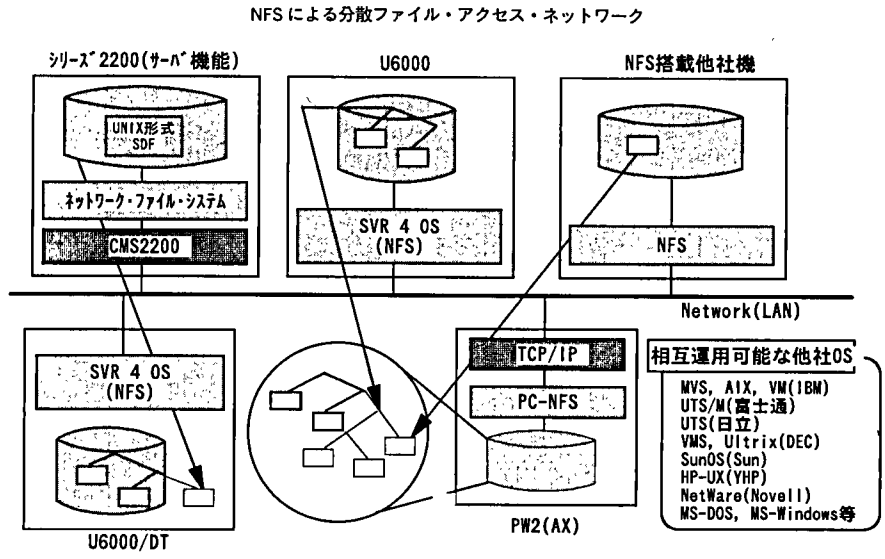


図 3 オープン・ネットワーキング（相互運用性）

各プラットフォーム間の相互接続性/相互運用性を提供する．

分散ファイル・アクセスを可能とするネットワーク・ファイル・システム（NFS）をOPEN 2200上に利用すると，OPEN 2200はファイル・サーバの役割をもった機能が実現できる（図3）．

・分散ファイル・アクセス
：ネットワーク・ファイル・アクセス（NFS）
・データ配送
：MHS，DDP-PPC，Dst DDL

5) オープン・マネジメント

オープン・マネジメントはSNMPおよびOSI管理標準に準拠し，マルチ・ベンダ・ネットワーク管理ができるCNMSを提供する．マルチ・ベンダ環境，分散配置された各プラットフォームやネットワークを一元的に統合して運用，管理（障害，性能，ネットワークの構成）を実現する（図4）．

① CNMS/インタネット・マネージャ（IM）

業界標準のネットワーク管理プロトコルであるSNMPを使用して，TCP/IPベースのLANに接続されている各種装置の一元管理が可能である．

② CNMS/DCA（計画）

OSI管理標準（CMIP）に準拠し，シリーズ2200とDCP（Distributed Communication Processor）通信制御装置で構成されるDCAネットワークの一元管理が可能である．

1.2 インテグリティ・サービス

「インテグリティ・サービス」は，シリーズ2200がもっている高度な付加価値機能群を意味し，2200/XPA（拡張処理アーキテクチャ）によって提供する．この高付加価値機能群は具体的には，次の五つの目的を達成する．

① 大規模トランザクション処理
② 限りないシステムの成長性
③ 無停止/連続運転
④ 無人運転
⑤ 高度なセキュリティ

これにより，大規模トランザクション処理の実

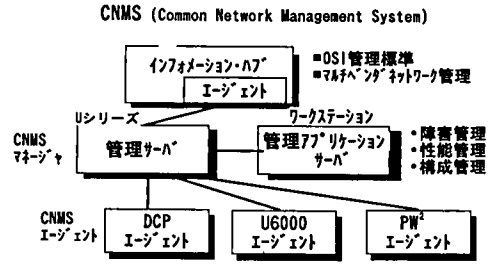


図 4 オープン・マネジメント（統合運用性）

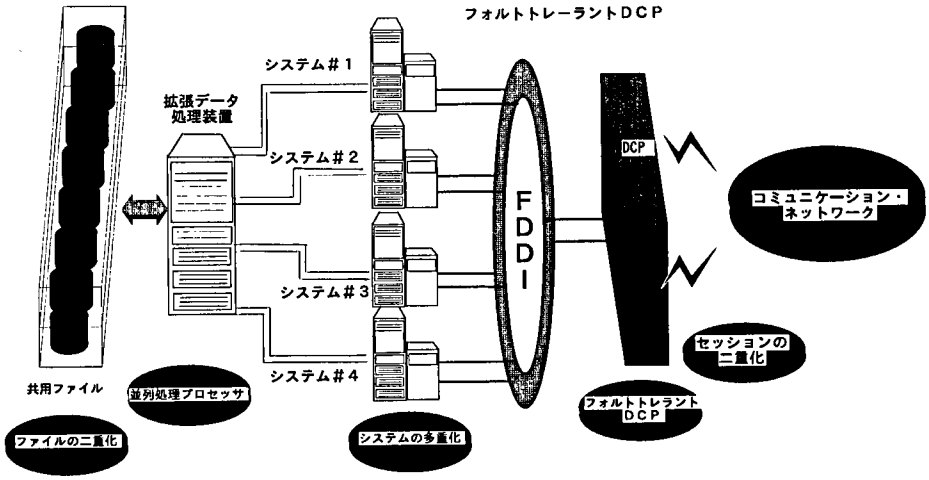
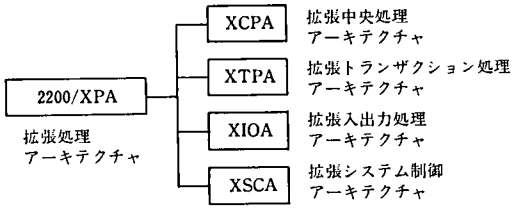


図 5　拡張トランザクション処理アーキテクチャ（XTPA）

現，情報システムへの投資削減，運用コストの低減，基幹業務を始めとするシステムの機密保持，等の効果を得ることができる．

シリーズ 2200/500 は 2200/XPA のもとで，ユーザの膨大なソフトウェア資産を保護しながら，さらにメインフレーム独自の新機能を提供する．

「2200/XPA」は，次の四つのサブ・アーキテクチャから構成される．



① XCPA (eXtended Central Processing Architecture)

XCPA はマルチ・プロセッサ・システムと巨大アドレス空間，大容量記憶装置，高度なセキュリティ等を支援する中央処理装置系のサブ・アーキテクチャである．XCPA のもとに開発された「2200/500 シリーズ」では，560 テラバイト（TB）の巨大なサイバ空間，276 ギガバイト（GB）のプログラム・アドレス空間，そして 1 GB の大容量記憶を実現している．これにより大規模トランザクション処理，大規模データベース処理が可能となり，さらにデータベースや使用頻度の高いファイル，またソート作業領域をサイバ空間に展開し，会話型処理やバッチ処理の処理効率を飛躍的に高めることができる．サイバ空間は，アーキテクチャ上は 72 ペタバイト（PB）まで利用可能である．

② XTPA (eXtended Transaction Processing Architecture)（図 5）

ユニシス独自の実績のある XTPA をさらに充実し，2200/500 シリーズと拡張データ処理装置（XPC）により最大 4 台のシステム群（16 プロセッサ）がシングル・システムとして並列処理を実現し，限りないシステムの拡張性と本格的な無停止/連続運転が可能となる．(アーキテクチャ上接続できるプロセッサの最大数は，128 である．)

③ XIOA (eXtended Input/Output Processing Architecture)（図 6）

XIOA はデータベースを含む入出力処理時間を飛躍的に短縮させる，入出力処理ア

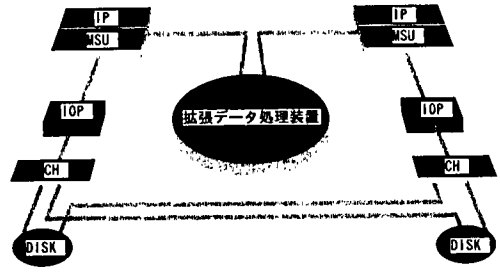


図 6　拡張入出力処理アーキテクチャ（XIOA）

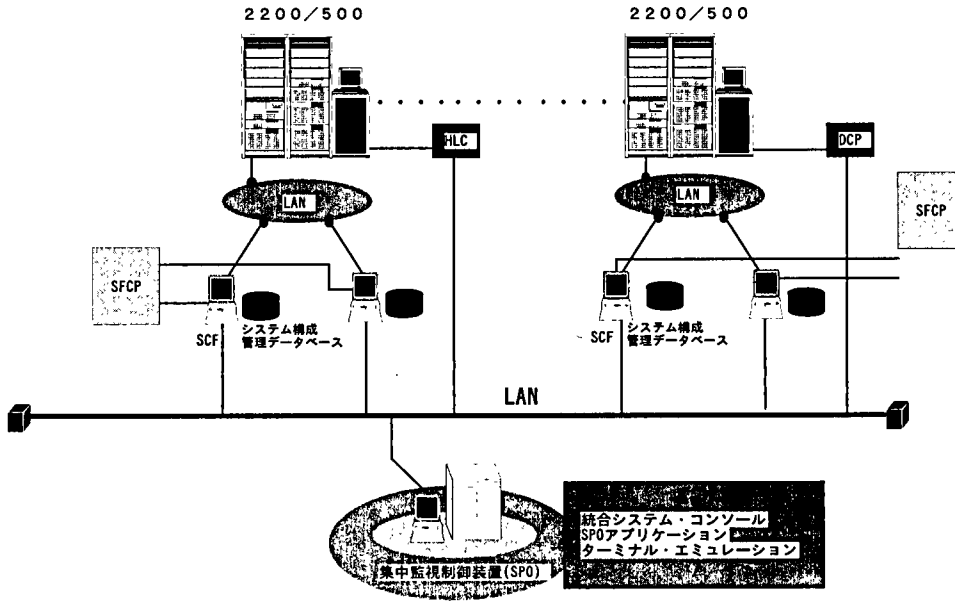


図 7　拡張システム制御アーキテクチャ（XSCA）

表 2　インテグリティ・サービス製品体系

インテグリティ・サービス	2200/XPA の サブ・アーキテクチャ	主なプロダクト
大規模トランザクション処理	XCPA	サイバ・バッチ，XIS，XTC-TIP/UDS，拡張データ処理装置
限りないシステムの成長性	XTPA	マルチ・プロセッサ，XTC-TIP/UDS，拡張データ処理装置
無停止/連続処理	XIOA	XTC-TIP/UDS，UNIT DUPLEX，拡張データ処理装置
無人運転	XSCA	拡張 IOF，SPO，UTC
高度なセキュリティ		OS 2200(B 1 セキュリティ)

ーキテクチャである．中央処理装置の高性能化に対応し，入出力処理能力を向上させ，トランザクション処理，データベース処理，バッチ処理，障害回復等の処理効率向上を実現する．拡張データ処理装置（XPC）は XTPA と XIOA を具体化するプロダクトとして提供される．

④ XSCA（eXtended System Control Architecture）（図 7）

XSCA は，マルチ・ベンダ/マルチ・プラットフォーム環境にあるシリーズ 2200 の無人運転と統合運用の基盤を実現するためのアーキテクチャである．

インテグリティ・サービスは，2200/XPA の基で表 2 のような製品を提供する．

1) 大規模トランザクション処理

高速/大規模トランザクション処理を実現するソフトウェアは次の通りである．

① XTC-TIP/UDS（大規模オンライン・トランザクション処理システム）

「拡張データ処理装置（XPC）」により，ロック制御を高速に行い，汎用的なデータベースを共用して，最大 16 台までのプロセッサの並列処理が可能である．

これにより大規模トランザクション処理，データベース処理，さらにアプリケー

ション開発支援等への強力な対応が図られるのと同時に，無停止連続処理を実現する．

② XIS (eXtended Information Support system)

大規模オンライン・トランザクション処理システムのアプリケーション・システム構築を，開発から運用，保守に至るまでトータルに支援するソフトウェア群である．

③ サイバ・バッチ・システム

大規模なサイバ空間と大容量メモリを活用して，高速バッチ処理を実現するソフトウェア群である．使用環境により異なるが，1/2～1/6倍の処理時間短縮を図ることが可能である．次の三つのプロダクトから構成される．

・EXFILE

処理の中間データ・ファイルをサイバ空間に展開し，入出力処理時間を大幅に短縮．

一時的ファイル，カタログ・ファイルのいずれにも適用可能．

ポストストア・モード，およびストアスルー・モードが可能．

・EXPIPE

サイバ空間上の循環バッファを経由してタスク間のデータ受け渡しを可能とし，バッチ・タスクの入出力の並列処理による処理時間短縮により，バッチ処理全体のターン・アラウンドを短縮．

・ESORT

ソート作業域をサイバ空間に割り付けてソート処理を高速化し，バッチ処理全体のターン・アラウンドを短縮．プロセッサ形式でも，ライブラリでも利用可能．

作業域は最大64 MBまで任意に設定可能．

2) 限りないシステムの成長性

4台までの密結合マルチ・プロセッサと拡張データ処理装置 (XPC) およびXTC-TIP/UDSにより，16台のプロセッサまでの成長性を提供する．

3) 無停止/連続処理

システムにおける完全な無停止連続処理を実現．拡張データ処理装置 (XPC)，XTC-TIP/UDS，UNIT DUPLEX（ディスク・ファイルを二重化し，ディスクの障害によるファイルやデータの破壊，システム・ストップ等を回避する）により，万一，あるホストに障害が生じても切り替え時間なしで他のホストへ切り替わり，処理は寸断することなく継続可能である．

4) 無人運転

マルチ・ベンダ環境，マルチ・プラットフォーム環境のシステムの運用管理とネットワーク管理を自動化し，無人運転化が可能である．

① 拡張IOF (Integrated Operation Facility)

システム監視制御，自動・無人運転，ワークフロー制御，システム・リソース管理等システムの運用管理に求められる機能を提供する．

② UTC (Universal Time Coordination)

NTTの時報を基に各ホストに正確な時刻を供給する．XTPA環境下においては各ホスト間のトランザクション処理のタイム・スタンプが正確になり，システム全体のリカバリを高速，正確に行える．

③ SPO (Single Point Operations)

複数ホストのコンソール・オペレーションを統合して，一貫性のある運用管理機能を提供する．複数ホスト・システムの集中監視，集中操作/制御を可能とし，GUIによるコンソール・オペレーションのシングル・ビュー化とマウスによる簡易な操作を実現し，オペレータ要員数を削減．

5) 高度なセキュリティ

OS 2200自身がデータやプログラムを不正なアクセスから保護するリング/ドメイン数を約65,000個用意し，大規模システムに対応したセキュリティを実現する．またユニシスは米国NCSC (National Computer Security Center) から，ビジネス用では最高レベルのB1セキュリティの認定を受けている．

1.3 OPEN 2200の効果

「OPEN 2200」はオープン・システムの特性である相互運用性，移植性，統合運用性と，メインフレームの特性である大規模トランザクション処

理，システムの拡張性，無停止/連続運転，高度なセキュリティ等を一つのプラットフォーム上で同時に実現する．これによってユーザは次の効果を得ることができる．

① インテグリティ・サービス機能の利用
② オープン・システムの特性（相互運用性，移植性，規模の拡張性）の享受
③ 適材適所コンピューティングによるトータルコストの削減

OPEN 2200 はメインフレーム独自のもつ強力なパワーを利用しながら新しい流れであるオープン・システムの世界を積極的に利用することができるオープン・メインフレームである．

2. UNISYS 2200/500 シリーズ

2200/500 シリーズは，前述した OPEN 2200 の第一弾として，インテグリティ・サービスとオープン・サービスを提供する．また，ユニシスの高い CMOS 技術を中大型機分野に全面採用して，処理の高速化/高信頼性/低価格/運用コストの低減/高コスト・パフォーマンス/省電力を実現している．

拡張処理アーキテクチャ (2200/XPA) により企業の基幹業務に将来にわたって十分対応できるハードウェアであり，ソフトウェアが準備されている．

2200/500 シリーズは現行のユーザのシステム資産を継承することができる．

システム構成は，最小構成のモデル 5111 から最大構成のモデル 5422 までの 5 モデルがあり，システム能力は同一構成の 2200/400 シリーズの約 2.5～3 倍である．拡張データ処理装置 (XPC) により，並列処理を可能としている．

1) UNISYS 2200/500 シリーズの概要

① 飛躍的な高コストパフォーマンスの実現
30 万ゲートの CMOS を採用して，低コスト，高性能を達成

② OPEN 2200 の実現
従来のメインフレームの高付加価値機能（インテグリティ・サービス）とオープン・サービスを同時に提供するメインフレーム

③ 2200/500 シリーズの位置づけ
- 同一構成で 2200/400 の 2.5～3 倍の処理能力
- 最大 IP 数は 4，メモリ容量は 128～1024 MB，チャネル数は最大 64

④ CMOS の採用
新しい 30 万ゲートの高密度・高性能 CMOS 技術を中大型機分野に採用

⑤ 拡張処理アーキテクチャ (2200/XPA) の採用
インテグリティ・サービスを実現し，メインフレームの高付加価値機能を提供

⑥ 運用コストの低減
省電力/省スペースと空冷方式により運用コストを低減

⑦ SFCP の標準装備
SFCP (統合システム監視制御装置) により，自動化/無人化オペレーションを提供

⑧ 拡張データ処理装置 (XPC-後述) の装備
- 入出力処理を新しい技術で高速化
- 16 プロセッサのビジネス処理での並列処理を実現
- ホストシステムの構成をきめ細かく設定できる
- 大規模トランザクション処理システムの構築が可能

2) ハードウェアの特徴

① 2200/900 シリーズと同一の 2200/XPA の採用
- 大規模トランザクション処理，大規模データベース処理を低価格で実現
- 2200/900 シリーズの設計を最先端の高密度，高性能な 30 万ゲート CMOS で実現
- 入出力処理装置 (IOP) とシステム監視制御装置 (SFCP) に同一の設計思想を採用

② 16 M DRAM の採用による大容量メモリの実現
- 16 M DRAM を両面実装し，高密度で小型化・高速化を実現
- PCC (Processing Complex Cabinet) 当たり最大 512 MB を実現

③ 高コストパフォーマンスの実現
- 最新の CMOS 技術の採用により，低価格で高い処理能力を実現
(2200/400 シリーズの 2.5 倍～3 倍の処理能力)
- XPC により，低価格で高い処理能力を

もった並列処理システムを実現
(約13倍の処理能力拡大が可能)
・アプリケーション分割運用を可能とし，柔軟な運用と投資コストの削減が可能

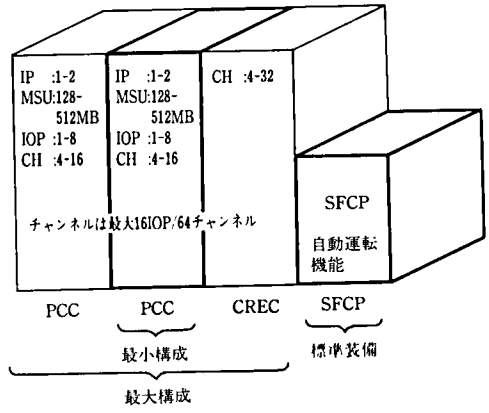
④ 30万ゲートの高密度・高性能CMOSの採用
・18個のCMOSを多層基板に実装し，1ボード・プロセッサを実現
・最大構成で，1.75 m² という省スペース化を実現
・空冷方式で，設備コスト，運用コストを大幅に低減

⑤ 信頼性の向上
・高密度CMOSにより，部品点数の極少化が図られ信頼性が向上

⑥ ハードウェア・アーキテクチャの強化
・命令の実行において，次の命令語の先読みを行うことによって，処理を高速化
・分岐命令の時にも，命令語の先読み機能の効果が低下しないように分岐先予知機能を新たに採用
・演算回路には，ダブルワード・データバスを全面的に採用

⑦ キャッシュ技術の強化
・各プロセッサ当たり64キロバイト(KB)の高速キャッシュ・メモリを実装
・SIF (Storage InterFace) には，1,048 KBの大容量高速キャッシュ・メモリを採用
・2段キャッシュ構造によりキャッシュ・メモリのヒット率の向上や主記憶のアクセス時間の短縮を実現

⑧ 高信頼性電源装置の採用
・中央処理キャビネット (PCC) には，高信頼性電源装置を採用し，冗長電源機構，ホット・リプレース機能を実現
・システムに影響を与えることなく交換が可能



IP : Instruction Processor
MSU : Main Storage Unit
IOP : I/O Processor
PCC : Processor Complex Cabinet
CREC : Channel Rack Expansion Cabinet
SFCP : System Facility Control Processor
CH : Channel

図 8 UNISYS 2200/500 シリーズの構成概要

3) 構成概要
図 8 (UNISYS 2200/500 シリーズの構成概要)参照

4) 構成と仕様
表 3 (UNISYS 2200/500 シリーズの構成と仕様)参照
・チャネル・インタフェースは FIPS 60 標準に準拠した BMC (ブロック多重チャネル)を提供し，さらに SCSI, FDDI も提供する．
・データムーバは XPC 接続専用チャネル

5) 性 能 表
表 4 (UNISYS 2200/500 シリーズの性能)参照

表3 UNISYS2200/500 シリーズの構成と仕様

システム・モデル	中央処理キャビネット	中央処理装置	主記憶容量 (MB)	チャネル数	データムーバ
モデル 5111	1	1	128～ 256	8～32	0～2
モデル 5211	1	2	128～ 256	8～32	0～2
モデル 5222	2	2	256～1024	8～64	0～4
モデル 5322	2	3	256～1024	8～64	0～4
モデル 5422	2	4	256～1024	8～64	0～4

表4 UNISYS2200/500シリーズの性能

中央処理装置	命令数	エクステンドモード	240
	IP数		1～4
キャッシュ・メモリ	容量(KB)	IP	64/IP
		MSU	512/MS
主記憶装置	最小記憶容量(MB)		128
	最大記憶容量(MB)		1024
入出力装置	IOP数		2～16
	チャネル・モジール数/キャビネット		8
	ブロック多重チャネル(BMC)	最大チャネル数	64
		チャネル数/モジュール	4
		転送速度(MB/S)	4.5/3.1
	データムーバ	データムーバ数/キャビネット	2
		転送速度(MB/S)	50

3. 拡張データ処理装置（XPC；eXtended Processing Complex）

UNISYS 2200/500シリーズと同時に発表した拡張データ処理装置（XPC）は，入出力処理の高速化と複数システムの並列処理を実現し，システム全体の処理能力と安定性を大幅に向上させ，無停止連続処理を実現するプロダクトである．拡張データ処理装置は「2200/XPA」の中のサブ・アーキテクチャであるXIOAとXTPAのもとに設計され，これからの新しいメインフレームの新しい利用形態を業界に先駆けて提供する当社独自の製品である．

拡張データ処理装置の目的

拡張データ処理装置はXIOAを実現し，トランザクション，バッチ，デマンド，データベース等のすべての入出力処理を数10ミリ秒から1ミリ秒以下へ大幅に高速化して，プロセッサと入出力のバランスをとりシステム全体の処理能力を高める．グローバルキャッシュ機能，レジデント・ファイル機能，プリフェッチ機能，オーディット処理機能等が提供される．

また，XTPAの実現として，複数のシリーズ2200を疎結合し，ファイルを共有するためのレコードロック機能により，ビジネス処理における並列処理を可能としている．この並列処理によって，大規模なトランザクション処理，データベース処理に対応するとともに，無停止連続処理も同時に実現している．

ビジネス処理の分野において並列処理を実現した最初のプロダクトである．

この画期的なプロダクトの利用により，インテグリティ・サービス（機能）の大幅な強化が図られる．

1) XPCのハードウェア

XPCはモジュール構造をベースとした並列処理アーキテクチャを採用．一つのモジュールには二つのRISCプロセッサからなるXIP（XPC Instruction Processor；演算処理エンジン）とHIP（ホスト・インタフェース・プロセッサ）が各四つ搭載される．最大8モジュールでは128個のRISCプロセッサによる並列処理により各機能が実行される．

ホストとは最大8チャネルによる光ケーブル経由での同時並列データ転送を行う．チャネル当たり50 MB/Sの高速転送を行う．システムの重要なファイル，レコードを扱うため万全な障害対策が図られている．処理はモジュール構造のエンジンにより相互のバックアップがされる．最大3.7 GBあるメモリは同じ容量のミラーメモリを有し，データベース，ファイルの保全は万全である．電源も電源系列，電源装置を完全二重化し，1系列の電源障害にも稼働を継続する．2系列共に障害の時でもデータをデータ待避機構（DSD）に待避させてデータを保証する（図9，10）．

2) 機能概要

XPCが提供する機能と2200/XPAとの関係は表5の通りである．

① バーチャル・ストレージ・マネージャ

(VSM)

- ホストからディスクファイルへのI/Oアクセスにおいて，XPCをキャッシュおよび高速レジデント・ファイル(XRF)として利用し，アクセス時間を短縮する機能
- キャッシュからディスクへのデータ更新は，XPCからキャッシュへのデータ更新とは非同期にポストストア方式により行われるのでI/Oのアクセスの高速化と，更新処理の効率的実行により，システム全体の処理性能を高める．
- シングル・ホストおよびマルチ・ホストにおいても，ローカル・ファイルおよび共用ファイルのアクセス/更新処理を高速化

【VSM機能利用による機能】

- グローバル・キャッシュ：複数のシステムが共有ファイルをアクセスする時に全システムに共通のキャッシュとして利用可能．モディ

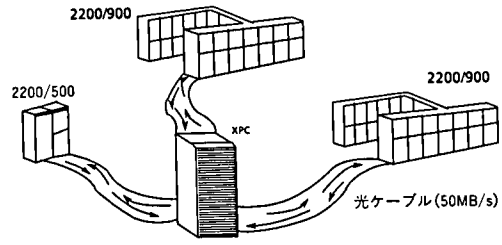
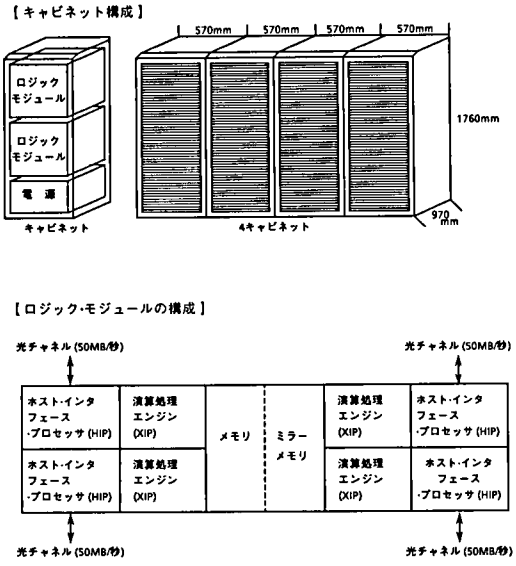


図 9　XPCの接続形態



【モデル別構成】

モデル・タイプ		モデル1	モデル2	モデル3	モデル4	モデル8
キャビネット数		1	1	2	2	4
モジュール数	基本モジュール	1	1	1	1	1
	拡張モジュール	0	1	1	3	7
最大ホスト・インタフェース(HIP)数		4	8	12	16	32
演算処理エンジン(XIP)	基本数	2	4	6	8	16
	最大数	4	8	12	16	32
メモリ容量(MB)	基本	128	256	384	512	1,024
(ミラー構成)	最大	128	640	1,152	1,664	3,712
電源制御機構		1	1	1	1	1

(注) HIP：Host Interface Processor　XIP：XPC Instruction Processor

図 10　XPCのハードウェア構成

表5　XPCが提供する機能と2200/XPAの関係

	サブアーキテクチャ	XPCの機能
XPA (拡張処理 アーキテクチャ)	XIOA (拡張入出力 処理アーキテクチャ)	◎VSM機能(バーチャル・ストレージ・マネージャ) ・グローバル・キャッシュ機能 キャッシュ機能によりデータ入出力処理を高速化 ・プリフェッチ機能 シーケンシャル・ファイルの事前アクセスにより入出力処理を高速化 ・レジデント・ファイル機能 ファイルの常駐化により入出力処理を高速化 ◎AM機能(オーディット・マネージャ)
	XTPA (拡張トランザクション 処理アーキテクチャ)	◎DSM機能(ディストリビューテッド・システム・マネージャ) ・レコードロック機能 ・メッセージ通信機能 ・負荷分散

ファイド・ラウンド・ロビン方式等，高いキャッシュ・ヒット率を実現する高度なキャッシュ機能

・プリフェッチ：次の処理で必要となるデータをXPC上に先読みしてディスク装置とのアクセスを不要にし，バッチ処理を高速化

・レジデント・ファイル(XRF)：アクセス頻度の高いテーブル/ファイルを常駐（レジデント）化して，アクセス/更新処理をXPC上で完了させ，データ入出力処理を高速化

② オーディット・マネージャ（AM）

・オーディット・ログ情報がXPCに書き込まれた時点で終了処理が行われ，またディスクへの書き込みはトランザクション処理とは非同期に行われ，効率的にオーディット処理が実行される．

・リカバリ時のデータ更新をXPC上で実行し，リカバリ処理時間を短縮

③ ディストリビューテッド・システム・マネージャ（DSM）

・マルチ・ホストを疎結合し，大規模なシングルシステム・イメージのトランザクション処理システム（XTPA）を実現する機能

・現行レコード・ロック・プロセッサ(RLP)の後継機として，より優れた機能/性能を提供

・マルチ・ホスト結合でのシングル・システム・イメージ実現のため，同一ブートテープによるシステム初期化，ハートビートによるホスト異常検知/障害ホストの自動リカバリ指示等の優れた機能を提供

【DSM機能利用による機能】

・レコードロック機能：共有ファイルに対する複数ホストからのアクセス要求に対して，データの整合性維持を目的にデータ更新を制御

・メッセージ通信機能：マルチホスト間の制御や同期を取るためのメッセージを通信

・負荷分散/業務分散 ：マルチホスト下で並列処理されるトランザクション処理の負荷を分散

拡張データ処理装置(XPC)の利用形態を図11に示す．

3) 特徴・効果

① 高速な入出力処理を実現しシステム・スループットを飛躍的に向上させる

・システム全体でのキャッシング機能によりデータベースを含む全ファイルの入出力処理を画期的に高速化し，システム全体のスループットを向上

・レジデント・ファイル機能によりアクセス頻度の高いテーブル，ファイルを

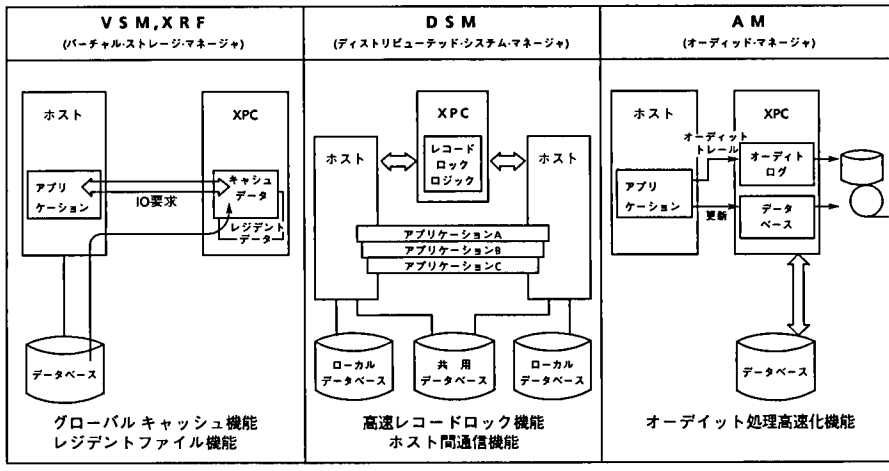


図 11 拡張データ処理装置（XPC）の利用形態

XPCに常駐化し，入出力処理能力を向上
・プリフェッチ機能により次の処理で必要となるデータをXPCに先読みして，処理を高速化
・オーディット処理の自動化/高速化によりトランザクション処理の効率を向上し，さらにオーディット・テープのボリュームを削減
・ホスト/XPC間は最大32本の光チャネルにより接続され，高速パラレルデータ転送により入出力の高速化を実現(最大データ転送能力：1.6ギガバイト (GB)/秒)

② 大規模トランザクション処理能力の提供
・レコードロック機能によって可能となる並列処理により，大規模トランザクション処理システムを手軽に構築

③ フォールトトレラント・システムの構築
・並列処理により無停止連続処理を実現

④ ビジネス処理における並列処理の実現
・ビジネス処理を複数のシステムで並列に処理することがアプリケーションの変更なしに，低価格で実現

⑤ システム間の高速接続機能の実現
・複数システム間の高速データ通信がXPCを介することにより容易に実現

⑥ 利用性の向上，導入の容易性
・アプリケーションの変更なしに導入が可能
・全てのデータモデルが利用可能
・磁気ディスク装置への入出力処理負荷が軽減し，ディスクの効率的使用が可能

⑦ 先進のハードウェア・テクノロジの採用
・10種以上の特許により構成される先進のハードウェア
・128個のRISCエンジンによりXPC自体も高速の並列処理を実現

⑧ フォールトトレラントXPCの実現
・プロセッサ/演算処理エンジンは処理を相互にバックアップ
ミラー・メモリによりデータは完全二重化
入力電源も含めて電源の完全二重化
ホット・メンテナンス・データの完全性保障
・データの不揮発性保障
二つの入力電源が共に停止しても無停電装置/データ待避機構によりデータ不揮発性保証

4. 強力な周辺装置群

2200/500シリーズと共に，新しい周辺装置を提供する．

各機器の主な特徴および性能は次の通りである．

1) 高性能・大容量磁気ディスク装置「N 8490型磁気ディスク装置」
・低価格，小型化
・モード：磁気ディスクモード，キャッシュモード
・最大容量：93.4 GB/台
・HDA容量：2.92 GB
・キャッシュメモリ容量：32 MBまたは64 MB，および同一リダンダンシ・メモリの増設可
・転送速度：4.2 MB/秒

2) 高速拡張半導体ディスク装置(N 7055型拡張半導体ディスク装置」
・最大容量：8 GB
・平均アクセスタイム：0.2ミリ秒
・転送速度：4.2 MB/秒
・データ保全機構：72時間バッテリ・バックアップ，データ退避機構

3) 小型カートリッジ磁気テープ装置「U 40 E型カートリッジ磁気テープ」
・低価格，小型装置
・マガジン方式の自動装置
・記録密度：37.871 BPI
・データ転送速度：3 MB/秒
・テープ走行速度：2 m/秒
・リワインド時間：48秒

4) 日本語カット紙印書装置「V 0480型日本語カット紙印書装置」
・3種類の縮小印書機能，両面印書
・2書体印書
・スタッカの自動切り替えによる無停止連続印書
・印書速度　：60頁/分

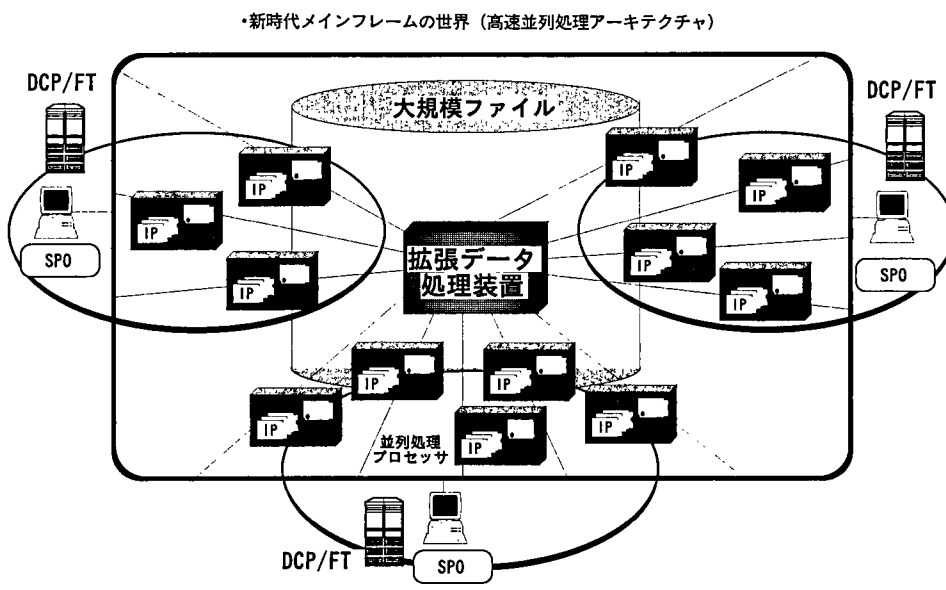


図 12　並列処理コンピュータを可能にした最先端技術

・印字ドット：240 DPI
・用紙サイズ：A 3, A 4, B 4, B 5, レター

5)　高速リモート・チャネル・エクステンダ装置　「N 1945 型光チャネル・エクステンダ装置」
・リモート入出力処理用チャネル
・入出力装置の遠隔電源制御
・光伝送速度：80 メガビット（Mb)/秒
・最大データ伝送速度：4.5 MB/秒
・最大伝送距離：3.0 Km

6)　高速チャネリング装置「JPP 2125 型チャネリング装置」
・遠隔地(数百 Km)入出力装置接続が可能
・ホスト直結による高速データ転送
・複数印書装置接続が可能
・最大接続距離　：1,000 Km
・データ伝送速度：192キロビット（Kb)/秒～1.5 Mb/秒

7)　ホスト LAN コントローラ　「HLC-II 型ホスト LAN コントローラ」
・FDDI　LAN（ISO　FDDI 準拠)，イーサネット LAN（IEEE 802.3 に準拠)
・異機種を含む複数ホスト相互間の大規模高速トランザクション処理を実現
・ユニシスの DCA，国際，業界標準の各プロトコルを提供
・接続チャネル：HBMC（4.5 MB/S)，BMC（3.0 MB/S)

8)　通信制御装置　「DCP/618」「DCP/628」
・フォールトトレラント・ネットワーク・プロセッサ「DCP/600 シリーズ」の最上位機
・障害時のセッションの連続性の保証によるフォールトトレラント機能の提供
・800 TPS の業界最高レベルのトランザクション処理能力
・最大 912 回線をサポート
・国際標準/業界標準に準拠したオープン・ネットワークのサポート
・16 台のプロセッサによる並列処理

5.　新世代メインフレームの世界

これからのユニシスの 2200 シリーズは CMOS 技術と並列処理技術の進展にそって，高性能・低価格・大規模トランザクション処理，無停止連続処理を実現して企業の基幹業務を担っていく．

同時にオープンの世界と協調する機能を持って，コンピュータ利用の最適な選択を可能にする(図 12)．

掲載論文梗概

TRITONのバッチ運行は，TRITONが大規模である，共同開発である，運用部門が開発に参加していない等，運用部門にとり不安材料が山積し，しかも運行制御パッケージIOF/WORKが開発途上にあったことに起因し，運用部門に難物視されていた．後博は，**IOF/WORKによるバッチ運用**の中で，IOF/WORKの概要を紹介した上で，バッチ運用検討過程で発生した課題に対しTRITONでいかなる工夫・考慮を払ったかを解説し，結果として実現されたTRITONにおける実運用について記述している．

金融機関の業務は勘定系を中心に情報系・国際系・対外系・証券系等多種多様なアプリケーションから成り，アプリケーション間で様々なデータ授受が必要とされる．TRITONでは従来のアプリケーション連動の考慮点に加え，複数ホスト構成による疎結合分散処理システムである点，各アプリケーションのソフトウェア基盤が異なる点等，新たな考慮が必要とされた．黒川茂は**アプリケーション連動の仕組み**の中で，XISの系間インタフェース機能をベースにアプリケーション連動をどのように実現したかを述べている．

金融機関のオンライン・システムは，金融機関相互の競合により自動機の時間延長サービスが拡大されている．時間延長の拡大はいずれ24時間稼働に結びつくことになると予想され，TRITONでは，24時間365日稼働における共通機能の開発を大きな目標の一つとした．中北晴久は**24時間365日稼働**の中で，いくつかの課題は残されたものの実現することができた共通機能と実際の対応についてまとめている．

銀行業務の中で最もシステム化が遅れている分野は，営業店での精査に係わる各種事務処理である．TRITONでは，当初よりこの事務処理の合理化を重点課題の一つとして検討し，開発されたシステムが集計レス・伝票レスである．関口賢造は**銀行における集計レス・伝票レス**の中で，その実現のために採用した各種新機能について説明している．

掲載論文梗概

ISSN 0914-9996

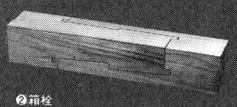
技　報
UNISYS TECHNOLOGY REVIEW
Vol. 13　No. 4　(No. 40)

発　行　日　平成6年2月28日
編集発行人　柳　生　孝　昭
発　行　所　日本ユニシス株式会社
東京都江東区豊洲1-1-1 〒135
TEL(03)5546-4111（大代表）
印　刷　所　三美印刷株式会社

UNISYS

どうやってつながったんだろう。

❶四方かま継ぎ
柱に用いられる継ぎ手。
真横からはどっちの面からも
入りそうにない…。

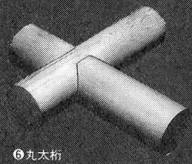


❷箱栓
表を見るとただくっつけただけ。
ところが、実は裏側に細かい細工が。

❸十字めちいれ
十字に切り込みを入れて
組み合わせ、材のずれや
回転を防ぐための継ぎ方。

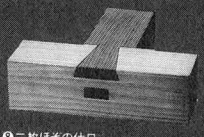
❹宮島継ぎ
斜めの切り口なので接合面が広く、
密着させるには高度な技術が必要。

❺貝の口継ぎ
塔の中心柱などに
使われる。接合面を
大きくし、強度を確保。

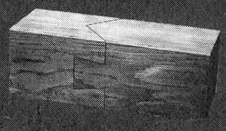


❻丸太桁
まるで2本の丸太がかっちり
組み合わされているようですが、さて。

❼腰かけはま継き
土台などに使われ、
上からの大きな重量に耐えるように
考えられている。



❽二枚ほぞの仕口
上のクサビ型の部分だけでも難しいのに、
さらに下にもうひとつのホゾが。



❾大坂城大手門控え柱の継き手
以前クイズにもなった、不思議な継き手。

「つながるはずがない」を、つなぎます。ユニシスのオープン・システム・テクノロジ。

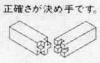
上にご紹介したのは、古くから木造建築で使われてきた「継ぎ手」と呼ばれる接合技術。まるでパズルのような不思議なものもあります。昔も今も、もの作りの基本は、組み合わせの技術。コンピュータ・システムも、異なるメーカーのさまざまなコンピュータをどう組み合わせ、最適にするかを考える、オープン・システムの時代。ユニシスはオープン・システムに早くから本格的に取り組み、そのノウハウの蓄積を活かして高品質のシステムを提供しています。そのひとつが、U6000シリーズをサーバにしたクライアント/サーバ・システム。異なるメーカーのワークステーションやパソコンを結んでデータの共有化を実現するのはもちろんのこと。ユニシスの場合は、そのクライアント/サーバ・システムを、いまお使いのホスト・コンピュータに、メーカーを問わず接続。ホストが持つ既存のソフトやデータをフルに活かすとともに、将来的にはホスト・システムのダウンサイジングをも可能にします。ユニシスが実現する、一歩進んだオープン・システム。その先進性を支えるU6000シリーズが、いまいちだんと性能を向上して新登場。全方位へつながるオープン・システム・テクノロジが、さらに広がりました。

➡こうやってつながりました。

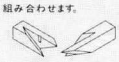
❶斜めの方向から
組み合わせるのが秘密。

❷両側からはさむように
組むのがミソです。

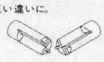
❸凸型と凹型の
正確さが決め手です。



❹互い違いの切り込みを
組み合わせます。

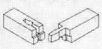


❺1/4ずつの切り込みが
互い違いに。



❻強度を上げるために
ホゾ穴を最小限に。

❼腰を掛けたような状態で
上からの力を受ける。



❽ホゾ穴の斜めの
勾配にそって入れる。

❾斜めにずらしながら入れる。

資料提供：住吉 寅七・松井 源吉 共著 鹿島出版刊「木造の継手と仕口」

ユニシス UNIX サーバ
U6000シリーズ
Pentium™プロセッサ搭載のU6000/300新登場

日本ユニシス株式会社　本社　東京都江東区豊洲 1-1-1 〒135 電話03-5546-4111(大代表)