TECHNOLOGY REVIEW

UNISYS 技報 通巻 42

1994年8月発刊 Vol. 14 No. 2

特集：実行運用支援環境

日本ユニシス

ISSN 0914-9996

金融，電力をはじめとするユーザでは新システムへの大規模更改の時期にあり，新プログラミング環境・拡張トランザクション処理アーキテクチャの発表と相俟って，新たなシステム・インフラストラクチャの提供が強く望まれた．これを契機に，シリーズ2200の開発・実行・運用の各環境を支援するIDES, XIS, IOFが提供された．育野准治は**統合オンライン支援システムXISの概要**の中で，実行環境支援ソフトウェアXISの紹介を行っている．

XISは，XTPAシステムの特徴を十分に活用するためのさまざまな機能を提供している．石川清人は**XTPA環境におけるXISのデータ通信制御**の中で，XISのデータ通信制御機能が提供するXTPA対応機能について説明している．

分散処理システム化の進行に伴い，集中型で構築されている既存ホストシステムを効果的に生かし，分散システムを構築するライトサイジングが求められている．栗原幸代の**XISの分散トランザクション処理**は，まず一般的な分散トランザクション処理の方式と要件について，さらにXISが提供する分散トランザクション処理制御の機能と実現方式について記述している．

システム運用記述言語はJCLが一般的であるが，ヒューマンインタフェースが単純な頃に開発されたまま進歩していない．XOLは，コンピュータ利用形態の多様化に伴い複雑化したシステム運用，特にXTPAシステムの運用記述のために開発された言語である．米津政紀・徳永路晴の**XISのシステム運用記述言語XOL**は，現状の問題点を明確にし，その上でXOLの狙い，機能，実現方式に言及し，XOLに複雑化したシステム運用の記述能力があることを明らかにしている．

XTPAは，最大4台のホストを疎結合し，RLPのもとで複数ホストのプログラムにTIP・UDSのDBを共用させてトランザクション処理量を増大させている．しかし，ホスト台数の増加は，システム運用がさらに複雑化することが予想される．竹内久の**XISのXTPAシステム運用**は，XTPAシステム環境で容易かつ柔軟にシステムを運用していくためのXIS運用機能，ノーダウンシステムの実現方式について言及している．

コンピュータシステムは，単一ホスト集中型処理から複数マシンを使用した分散処理に移行しつつある．北川達朗・宮下真は**3階層分散システムにおける協調分散トランザクション処理**の中で，シリーズ2200とU6000といった異なるプラットホーム間の協調分散トランザクションの実現方式，ゲートウェイシステムの処理構造・機能を述べるとともに，今後の問題を挙げている．

オンライントランザクション処理は分散処理を含め様々な形態をとるようになってきている．このため，OLTPの基盤システムの中核をなすTPモニタ（Aシリーズ上のCOMS）は，多様性・柔軟性・信頼性を持ち，大量トランザクション処理を可能とする高処理能力と負荷分散の考慮がなされている必要がある．森良行・原広仁は**Aシリーズのトランザクション処理の実現方法**の中で，OLTPを支える機能を明らかにし，AシリーズOLTP環境の中核をなすCOMSでのこれらの機能の実装技術を紹介している．

統合運用システム（拡張IOF）はシリーズ2200の運用全般を支援するソフトウェアであり，IOF/BASE-II，IOF/MONI，IOF/WORK，IOF/MASS，IOF/RLSから構成されている．今西秀文は**統合運用システム拡張IOFの概要**の中で，拡張IOF（レベル2R）の基本概念を説明し，合わせて構成サブシステムの概要紹介を行っている．

コンピュータシステムの高度化・分散化に伴いバッチ運用は複雑化・多様化の一途をたどっている．安坂敏秀・友枝研は**IOF/WORKによるジョブスケジューリングとユーザインタフェース**のなかで，バッチ処理運用に求められる新たな要求を分析し，IOF/WORKがそれらの要求に対して実現した機能の説明を行っている．

特集「実行運用支援環境」の発刊によせて

本池洵

本号はアプリケーション・システムの実行運用基盤を提供するミドル・ソフトウェアに関する特集号である.

1964年（昭和39年）にリアルタイム・システムの構築，1968年（昭和43年）にタイムシェアリング・システムの構築等，我国初のシステムの稼働に際し，当社は大きな役割を果たした．その後，これらの技術基盤を基に幾多の顧客システムの構築と技術支援，当社ソフトウェア商品の開発を手掛けてきた．この活動を通じて，我国あるいは産業ごとの特有なニーズを把握し，課題克服のために必要な要素技術に焦点を当て，技術者の育成，技術の継承，新技術の開発，オペレーティング・システムへの機能具備や新規ソフトウェア商品の開発提供を行い現在に至っている．当初より継続的に対応してきたニーズを観るに，様々な分野の課題が提起されてきたものの，とりわけ処理能力向上，無停止稼働，運悪く障害発生した場合の障害復旧の精度向上・時間短縮などが主たる課題であったと言えよう．そして，この課題への対応が実行運用支援環境の整備の始まりであった．ハードウェアとソフトウェアの様々な組み合わせによって，これらの課題に対して取り組みがなされてきたが，当初は個々のユーザ・プログラムでの対応も大きな割合を占めていた．これらの課題への対応を念頭に置いたユーザ・プログラムの開発は，オペレーティング・システムやハードウェアの振る舞いを熟知した上で取り掛からねばならない技術的難度の高い仕事であった．

1970年代後半の，基幹業務処理のオンライン化を背景に，データベース障害に対する復旧の精度向上やデータベースの扱いを容易にするためのサブルーチン群など，従来それぞれのユーザ・プログラムに組み込む必要のあったルーチン群を抜き出し一般化し，さらに入出力電文の保存回復機能や集配信機能などの必要な機能を追加し，実行運用支援環境の整備のために統合化した．これをユーザ・プログラムとオペレーティング・システムの中間に位置付けてミドル・ソフトウェアと呼ぶようになった．ソフトウェア商品として，AIS 1100，XISおよびそれぞれのファミリ・ソフトウェアなどがこれにあたる．この間，オペレーティング・システムにおいても，実行運用支援のための機能である遅延更新機能やミラー・ファイル機能等が具備されてきた．さらに無停止稼働のためのホット・スタンバイ・システムやXTPA (eXtended Transaction Processing Architecture) に基づく疎結合システムの運用を支える各種の機能プログラムが，AIS 1100やXISに統合化されている．

商用のコンピュータが一般化され，多くの企業で利用され始めた時からの課題であった処理能力の向上，無停止稼働，障害復旧の精度向上と時間短縮などの要求に対して，長い年月を経てソフトウェアとハードウェアの技術の進歩とコンピュータの利用者・提供者の絶え間ない挑

戦によって，現在のメインフレーム・システムの丈夫さ（Robustness）が確立されている．これを語るとき，前述の実行運用支援環境の具現化を果たしているソフトウェア群や支える技術を除いては語ることはできない．

一方，更なる処理能力の向上と処理の役割の明確化による分散処理等の要請に応え，XTPAによる疎結合システムやコンピュータ・ネットワーキングによる分散システムが実現してきた．複数のコンピュータ・システムの接続によって構成されている複雑なシステムにおいては，新たにシステム運用の安全性，容易性を図るためのシステム運用管理者やオペレータを支援する省力化や省脳化の要請と，分散処理環境におけるエンドユーザを支援する利便性の向上の要請があり，これらに応えるべく，拡張IOF，A-IOF，集中コンソールーSPO等や分散プリント，分散データデリバリ等を開発提供してきた．さらに，複数メインフレーム・システム，複数のUNIX* サーバと複数のUNIXクライアント/複数のPCクライアント等の接続によって構成された大規模な分散システムを一つのシステムとして捉えた（Single System View）運用を可能とする，分散システム運用管理システムとして各種の管理システムの開発提供を計画している．

以上のような経緯を経て現在に至っているが，コンピュータを効率良く使おう，上手に使おうと言う絶え間無い要求や意欲がある限り，ハードウェアとソフトウェアの新技術開発と相俟って実行運用支援環境も今後共発展し続けるであろう．

今般，実行運用支援環境の特集として，当社が現在手掛けている支援ソフトウェアを中心にまとめたが，発展の経緯を思い，さらに今後の発展を想い御一読お願いしたい．システム構築に携わるシステム技術者の参考になれば幸いである．

（システム技術第一本部　本部長）

*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．

統合オンライン支援システム XIS の概要

An Overview of XIS(eXtended Information System)

育野准治

要　約　金融，電力をはじめとするユーザにおいて，新システムへの大規模更改の時期がきた．現システムを支えるシステム・インフラストラクチャは，AIS 1100 II をはじめ提供以来10年の歳月が経過しており，UNISYS における新プログラミング環境(NPE: New Programming Environment)，拡張トランザクション処理アーキテクチャ(XTPA: eXtended Transaction Processing Architecture)の発表と相俟って，新たなシステム・インフラストラクチャへの期待は膨らみ，その提供が強く望まれた．

日本ユニシスでは，これらを契機に UNISYS シリーズ 2200/1100 における開発環境，実行環境，運用環境を支援する三つのシステム・インフラストラクチャが更改され，IDES, XIS, IOF となった．各環境支援機能の拡充はもちろんのこと，同時開発による環境間連携強化により，ユーザ・システムのライフサイクル全般にわたる統合支援機能がより充実した．

IDES(Integrated Development Environment System)：開発環境支援ソフトウェア
XIS (eXtended Information System)　：実行環境支援ソフトウェア
IOF (Integrated Operating Facility)　：運用環境支援ソフトウェア

本稿では，実行環境支援ソフトウェア XIS の概要について紹介している．

主な機能は次の通りである．

1) 大容量トランザクション処理環境の提供
2) フォールトトレラント・システムの実現
3) NPE 環境，および XTPA 環境を基に，高水準かつ使いやすい使用者アプリケーション・インタフェースの提供
4) 分散処理，自動運転/連続運転等多様かつ高度な処理形態の支援

Abstract The time has come when computer users typically in the financial and electric power industries are urged to renew their systems on a large scale. The systems infrastructure software supporting existing systems includes ten-year-old AIS 1100 II and others. The announcement by Unisys Corporation of NPE (New Programming Environment) and XTPA (eXtended Transaction Processing Architecture) inflated customer expectations for an advent of a new systems infrastructure, and users in Japan strongly wanted their availability.

Then, Nihon Unisys took this opportunity to enhance three systems infrastructure software tools to support development, execution and operation environments respectively for the Unisys Series 2200 and 1100 systems, resulting in the release of IDES (Integrated Development Environment System), XIS (eXtended Information System) and IOF (Integrated Operating Facility). Those efforts to expand support functions for all environments and to enhance inter-environment linkage, which was done at the same time, have given rise to all integrated, enhanced support functionalities which also help extend the life cycle of user systems.

This paper is intended to sketch out XIS, which provides the features that are primarily (1) for high

-volume transaction processing, (2) for the creation of fault-tolerant systems, (3) for NPE- and XTPA-based high-level, user-friendly applications interfaces, and (4) for sophisticated and versatile forms of data processing including distributed processing, automatic/non-stop opration, and so forth.

1. はじめに

コンピュータ・システムは年々大規模化し，分散処理の進展とともに，ユーザおよびシステム利用形態はますます多様化，高度化し，同時にシステムの構造は複雑化してきている．このような環境において進展著しい新技術に対応し，相次ぐアプリケーションの追加開発に長期的に耐え得るシステム基盤の整備確立は必須である．システム・インフラストラクチャの整備は高技術と長期にわたる拡大/保守コストが必要であり，大規模ユーザといえどもメーカ標準製品に依存する傾向が顕著となっている．この分野は業種によらない共通性をもっており，当社でも，1983年以来統合オンライン支援ソフトウェアとしてAIS 1100(Advanced Information System)を提供してきた．

金融，電力をはじめとするユーザにおいて，新システムへの大規模更改の時期がきた．ユーザはホスト，ネットワーク，ワークステーションにまたがった処理形態，システム・ライフサイクル全般に対する統合性，異機種を含めたオープンシステム化，また大容量トランザクション処理，24時間連続運転，フォールトトレラント・システムの実現を志向している．UNISYSの最新技術EM(Extended Mode：拡張アーキテクチャ)とXTPA(eXtended Transaction Processing Architecture：拡張トランザクション処理アーキテクチャ)の出現と相俟って，オンライン・トランザクション処理への更なる支援，新システム・インフラストラクチャの提供が強く望まれた．当社では該要件に応えるべく，AISユーザのEM化を支援するXIS 1Rを提供し(1991年11月)，XTPA対応等大幅に機能拡張したXIS 2Rを1993年4月提供した．

XIS 2Rは，シリーズ2200拡張アーキテクチャと拡張トランザクション処理アーキテクチャに基づく統合オンラインソフトウェアであり，システム基盤として開発・実行・運用を統合したシステム環境を提供することにより以下を実現する．

① 大容量トランザクション処理環境
② フォールトトレラント・システムによる高可用性
③ EM環境，各種支援ツールによる開発・保守・運用の高生産性
④ 分散処理，開放型システム，自動運転/連続運転等，多様かつ高度な処理形態に対応する高機能性

これにより，ユーザは次の利点が得られる．

① システム更改に長期的に耐え得る柔構造システムの実現
② システム・ライフサイクル全般にわたる生産性の向上とコストの低減
③ 社会的責任を遂行する高信頼性システムの実現
④ ハードウェア/ソフトウェアの機能，性能の最大限の活用

本稿では，XIS2R(eXtended Information System 2R)の概要を紹介する．

2. XISの位置付け

XISは，使用者が高度なコンピュータ利用をより少ない労力で実現できるよう，デ

ータベース制御，データコミュニケーション制御，リカバリなどのオンライン中枢機能を統合し，高水準で使いやすい使用者アプリケーション・インタフェースを提供する．使用者プログラムは XIS とインタフェースをとることだけですべての DB/DC 機能を使用することができ，従来使用者独自に開発管理してきたミドル・ソフトウェアから一部業務処理までも XIS 標準機能として使用することができる．

また，システムの開発から運用にいたるシステム・ライフサイクル全般を支援する各種機能を有し，システム構築の各局面での使用者負荷を減らし，使用者がその業務処理に専念できるシステム環境を提供する(図 1)．

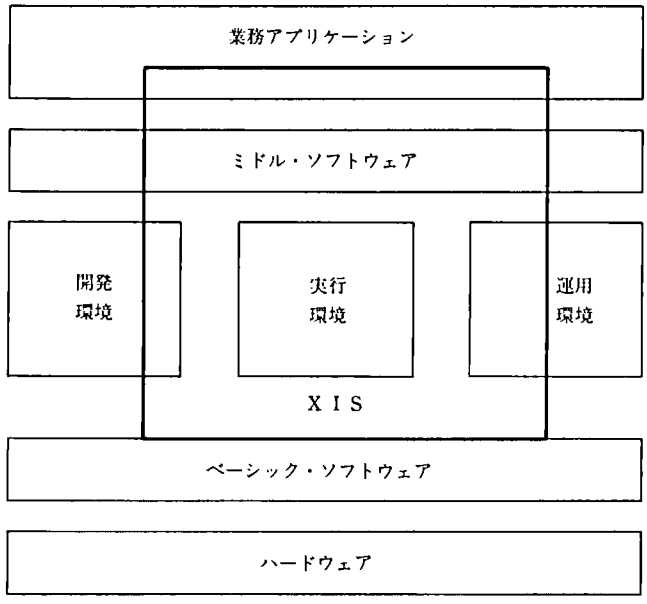


図 1　XIS の位置付け

2.1　XIS と三つの環境

XIS は，実行環境を中心に種々の開発環境支援，運用環境支援機能を提供する．該環境は，統合開発環境(IDES)，統合運用システム(IOF)との連携により，さらに機能拡張される(図 2)．

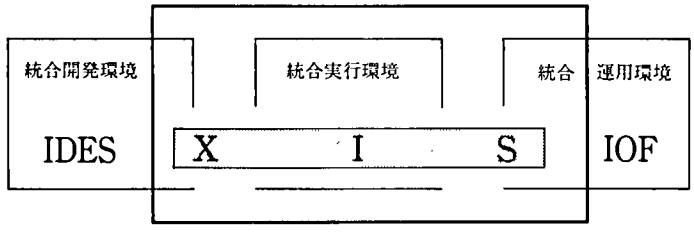


図 2　XIS と三つの環境

2.2　XIS とハードウェア

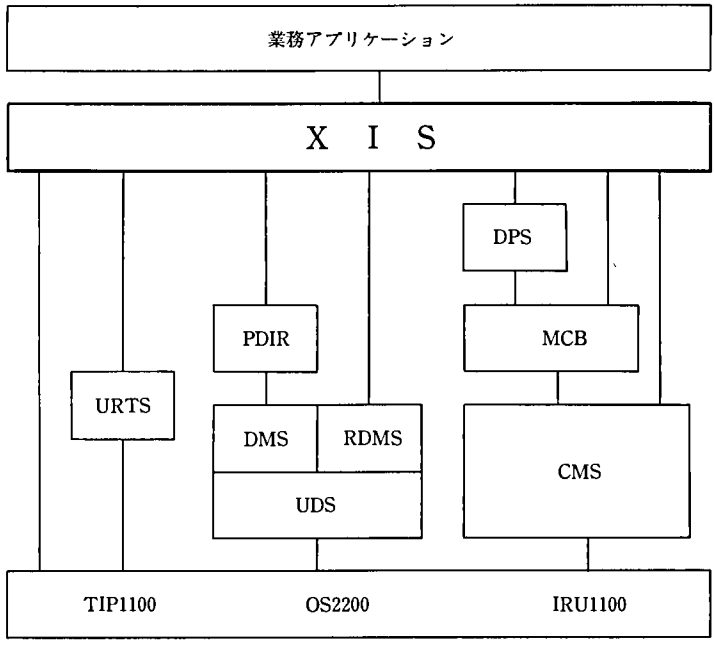
XIS は，2200/400 シリーズ，2200/600 シリーズおよび 2200/900 シリーズなどの

OS 2200 のもとで実行される．データ通信制御装置は DCP (Distributed Communication Processor) ファミリを前提とし，XIS コンソールとしてワークステーション PW²ファミリを使用する．

また，XTPA システムを構築する場合は，RLP (Record Lock Processor)*, HLC (Host Lan Controller)**, UTC (Universal Time Coordination)***を必要とする．

2.3 XIS とベーシック・ソフトウェア

XIS は，OS 2200 および TIP 1100/IRU 1100 および URTS と一体となって動作しており，それらの機能を前提としている．さらに，データ通信制御は CMS 2200/MCB 1100 と，データベース制御は UDS 1100 (DMS 1100, RDMS 1100)/PDIR II と，画面制御は DPS 1100 II と機能を分担している．業務アプリケーションは，これらベーシック・ソフトウェアと直接インタフェースをとる必要はない (図 3)．



TIP (Transaction Interface Package)	：オンライン・トランザクション処理システム
IRU (Integrated Recovery Utility)	：共用ファイルの統合回復プログラム
URTS (Universal Run time System)	：UCS (後述) の実行時ライブラリ
CMS (Communication Management System)	：通信制御システム
MCB (Message Control Bank)	：メッセージ制御プログラム
UDS (Universal Data System)	：複数のデータモデルを統一的に管理する汎用データシステム
DMS (Database Management System)	：UDS のネットワーク型データベース管理システム
RDMS (Relational Database Management System)	：UDS のリレーショナル・データベース管理システム
PDIR (Primitive DMS Interface Routine)	：プリミティブ DMS インタフェース・ルーチン
DPS (Display Processing System)	：画面・帳票定義，メッセージ編集支援プログラム

図 3 XIS とベーシック・ソフトウェア

2.4 XIS とアプリケーション

使用者は，XIS のインタフェースのみを理解すれば業務処理が実装できる．さらに，下記に示すシステム共通処理，運用処理および業務制御処理が XIS 標準機能として使

* RLP は XTPA の中核となるハードウエアであり，主として複数ホスト間のレコードロック制御を行う．

** HLC はマルチホスト・ファイルシェアリング機能を実現するためのホスト間通信機能を提供する．

*** UTC は XTPA を構成する複数ホストに NTT より受信した時刻を供給する装置である．

用できる.

- 入力メッセージの振り分け処理
- 入力/出力メッセージの通番管理
- 出力メッセージの保存/再送処理
- ネットワーク運用処理
- システム監視とプログラム閉塞処理
- 開始/終了/回復処理
- 標準コマンドと標準メッセージ
- 標準 ICP(Initial Control Program)/FCP(Final Control Program)
- 標準エラー処理とエラー文言ファイル
- 標準センタカット・スケジューラ
- ディレード・オンライン処理
- 複数バッチによる同一ファイル読み込み制御機能
- バッチプログラム障害後のリスタート機能

2.5 XIS と NPE(New Programming Environment)

NPE は，プログラムの生産性向上，仮想アドレッシングによるアドレス空間の拡大，使用者プログラムとデータの安全性向上を目的とする新プログラム開発環境であり，EM という新しいハードウェア/ソフトウェア・アーキテクチャにより実現される．

XIS は，EM アーキテクチャの利点をフルに活用し，EM プログラムを支援する．

1) BM(Basic Mode) と EM(Extended Mode)

従来のアーキテクチャは，基本モード(BM)と呼ばれ，拡張モード(EM)に較べてアドレス空間，コンパイル方式，セキュリティ面で種々制約があった．

EM アーキテクチャと BM アーキテクチャは共存可能であり，同一システム上で EM アプリケーションと BM アプリケーションが互いに干渉することなく稼働できる（デュアルモード）．また，一つのアプリケーション・プログラムが EM バンク（ユーザバンクあるいはコモンバンク）と BM コモンバンクを使用して稼働することができる（ミックスモード）．

2) アドレス空間

仮想アドレス（VA : Virtual Address）の採用により，69 ギガ語のアドレス空間が使用可能である．

3) 開発環境

EM では，統合コンパイル・システム UCS(Universal Compiling System)，リンキング・システム*，PADS 1100(Programmer's Advanced Debugging System for series 1100)など新しいソフトウェアが提供され，プログラミングの生産性向上を実現している．

EM では複数のプログラム言語が，同一概念で翻訳(コンパイル)され，異言語間の標準インタフェース，日本語機能が容易に使用できる．

BM では，コンパイル（コンパイル結果をリロケータブル・エレメントと呼ぶ）後，必要リロケータブルを連結編集したアブソリュート・エレメントのみ実行可

* リンキング・システムは，UCS コンパイラで翻訳した拡張モードの使用者プログラムを連結編集するプログラムである．

能である．

EMでは，コンパイル後(コンパイル結果を目的モジュール(OM：Object Module)と呼ぶ)，即実行が可能である．必要に応じて他OMが動的連結されるので，開発中のプログラム・デバッグが容易である．なお，完成したプログラムおよび実行効率が重視されるプログラムは，静的連結をすることもでき，ZOOM(Zero Overhead Object Modules)と呼ばれる．また，PADS 1100の使用により，シンボリック・エレメントのデバッグが可能である．

4) セキュリティ

EMでは，あるバンクから他のバンクにアクセスする場合の保護機能がソフトウェアおよびハードウェア機能を使用し実現される．

使用者プログラムは，EXEC, システム・プロセッサ，XIS等コモンバンクを経由し実行されるが，各々のバンク経由時のセキュリティ環境は，ハードウェアにより設定されるキー/ロックおよびゲート機構により実現され，ソフトウェア・サブシステムという概念が提供された．

サブシステムは一群のOMから構成され，それぞれのOMは互いに密接な関係を持っている．一つのサブシステムはソフトウェアのある機能単位を成し，他のサブシステムとは分離した存在としてアクセス制御される．なお，EMコモンバンクをFGSS(Fixed Gate Sub System)と呼ぶ．

2.6 XISとXTPA

XTPAは，最大四つの独立したシステムを結合して，データベースを密接に共存させることによって，すべてのシステムをプロダクション系として活用しながら相互にバックアップさせ，冗長構成を無駄なく活用した無停止連続運転処理を実現する．

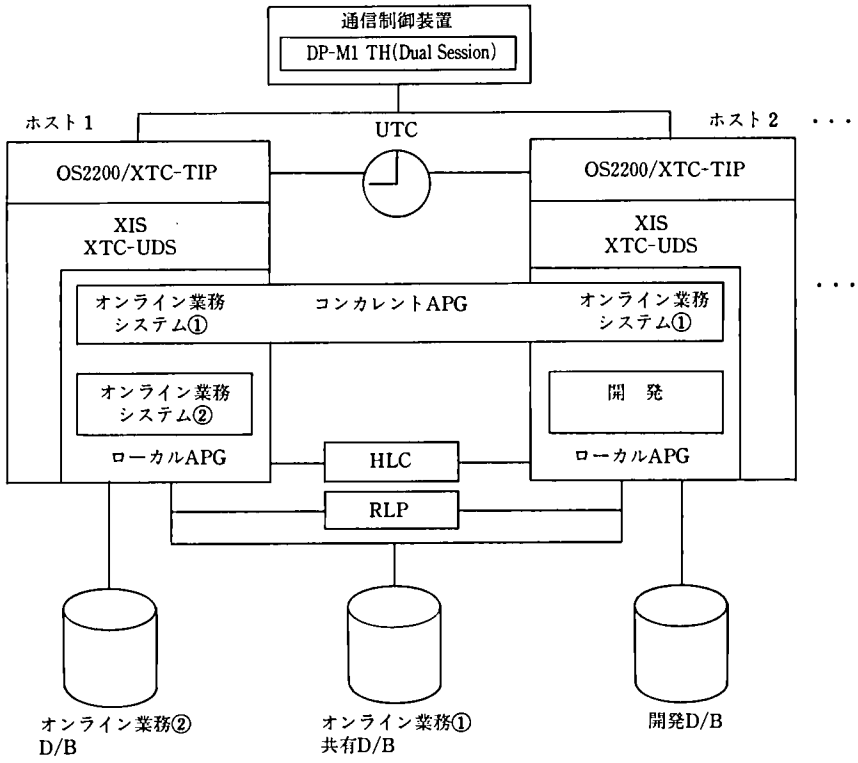
RLPというハードウェアの出現により，レコード単位のロック制御機構によるデータベース・シェアが実現した．複数ホストの疎結合システムによる大量トランザクション処理が可能となり，CPU増設によるシステム増強に加え，ホスト増設によるシステム増強が可能となった．

また，ホスト障害時，XTPAを構成する他ホストからのデータベース回復が可能であり，障害ホストの復旧を待つことなく回復処理が可能となった．しかも，回復処理は，ホスト障害時に仕掛かっていたトランザクション（端末）にのみ関係し，他トランザクション処理は該回復処理と並行して正常処理できる．障害時仕掛かり中であった端末以外は，まったく取引を中断されることがない（ノーダウン・システム）．

図4にXTPAシステムに必要なソフトウェア，ハードウェアを示す．

XTPAは複数ホストでのデータベース・シェアを実現している．しかし，オンライン・システムにおけるホスト負荷分散，ノーダウン・システムを実現するためには，さらにネットワーク・シェアおよび複数ホストを考慮したシステム/ネットワーク/アプリケーション運用制御が必要である．XISではこれら運用を補完し，該アーキテクチャの利点をすべて導き出すべく以下の機能を提供する．

・XTPA環境下のAPG(APplication Group)は，XTPAを構成する複数のホスト・システムにまたがって実行されるように拡張され，従来のAPG（ローカルAPG）と区別しコンカレントAPGと呼ぶ．



・APG(APplication Group)：
　運用/導入/リカバリの単位として OS が管理する概念
・コンカレント APG：
　XTPA のもとで複数ホストにまたがり稼働する APG
・ローカル APG：
　あるホストでのみ稼働する APG
HLC：HYPER channel または IPCC または HLC

① XTPA システムに必要なソフトウェア
・XTC-TIP(eXtended Transaction Capacity-TIP)：
　XTPA 環境下での TIP 機能(注)
・XTC-UDS：
　XTPA 環境下でのデータベース制御機能(注)
・DP-M1 TH(Dual Session)：
　XTPA 環境下でのデータ・コミュニケーション制御機能(DP-M1 TH のみ)
・XIS
(注)XTC-TIP/UDS によって，複数のシステムでデータベースを共有することができ，TIP1100 管理下の FCSS ファイルと UDS 管理下のデータベースのアクセスの過程で発生するデッドロックの検知も可能になる．

② XTPA システムに必要なハードウェア
RLP：レコード単位のロック制御を行う．
UTC：各ホストのディ・クロックの時刻合わせを自動的に行う．
HLC：XTPA 構成ホスト間で XTC 制御情報転送を行う．

図 4　XTPA 概要図

XIS では，ローカル APG とコンカレント APG を支援する．

・XIS は，TCDBF(TIP Common Data Bank File)を利用して XTPA 構成ホストのメモリ同期をとる．

・XTPA 下でのディレード処理を実現するため，XIS は XTPA 構成ホストごとのトランザクション・ログをマージして，CSN(Commit Sequential Number)順にユーザ・ディレードバッチに渡す．

・XTPA システムの開始，終了，回復処理においては，XTPA を構成している複

数ホスト間で各種同期をとる必要がある．XISは，該同期を考慮した標準開始/終了/回復処理を提供するとともに，ユーザ個有処理の同期とりを支援する．

・XISは，XTPA構成ホストまたはAPGの障害を監視し，自動回復処理を行う．

・DP-M1 THのデュアル・セッション機能との連携により，XISはホスト障害時の自動セッション切替え，回復処理を行う．また，外部システムがXTPAを構成するどのホスト・システムに接続されているかを意識することなくメッセージの送受信が行えるようメッセージの自動振り替え転送を行う．

3. XISの概要

3.1 全体構成

XISの全体図を図5に示す．

3.2 機能概要

XISは，シリーズ2200の拡張処理アーキテクチャ(2200/XPA*)と拡張トランザクション処理アーキテクチャ(XTPA)に基づき，従来のAIS 1100 IIをさらに拡張，発展させた統合オンライン支援システムである．従来以上の大規模，大容量トランザクション処理が可能であり，無停止システムが実現できる．

主な機能は，下記のとおりである．

1) システム制御機能……外部システム（端末，ホストなど）から投入される大量かつ多種多様なトランザクションおよびバッチ処理を行うのに必要なプログラム制御，メッセージとデータベースの回復を統一的に扱う統合リカバリなどの機能を提供する．

① XISはTIP/IRとの連携により，メッセージ制御，データベース制御，リカバリ機能などのソフトウェア構成要素を独立してもつことができるAPGという単位を提供する．各APGは，システムの導入から運用にいたる全局面で相互に干渉することはなく，同一ホスト・コンピュータ内に複数のAPG(最大8個)をもつことができる．また，XTPA環境下のAPGは，XTPAを構成する複数のホスト・システムにまたがって実行されるように拡張され，従来のAPG（ローカルAPG）と区別しコンカレントAPGと呼ぶ．

② トランザクション処理中の障害であるプログラム異常終了やシステムダウンなどに対して，障害波及範囲の局所化を図り，回復処理に備えリカバリ用の情報（オーディット・トレイル情報など）を取得する．

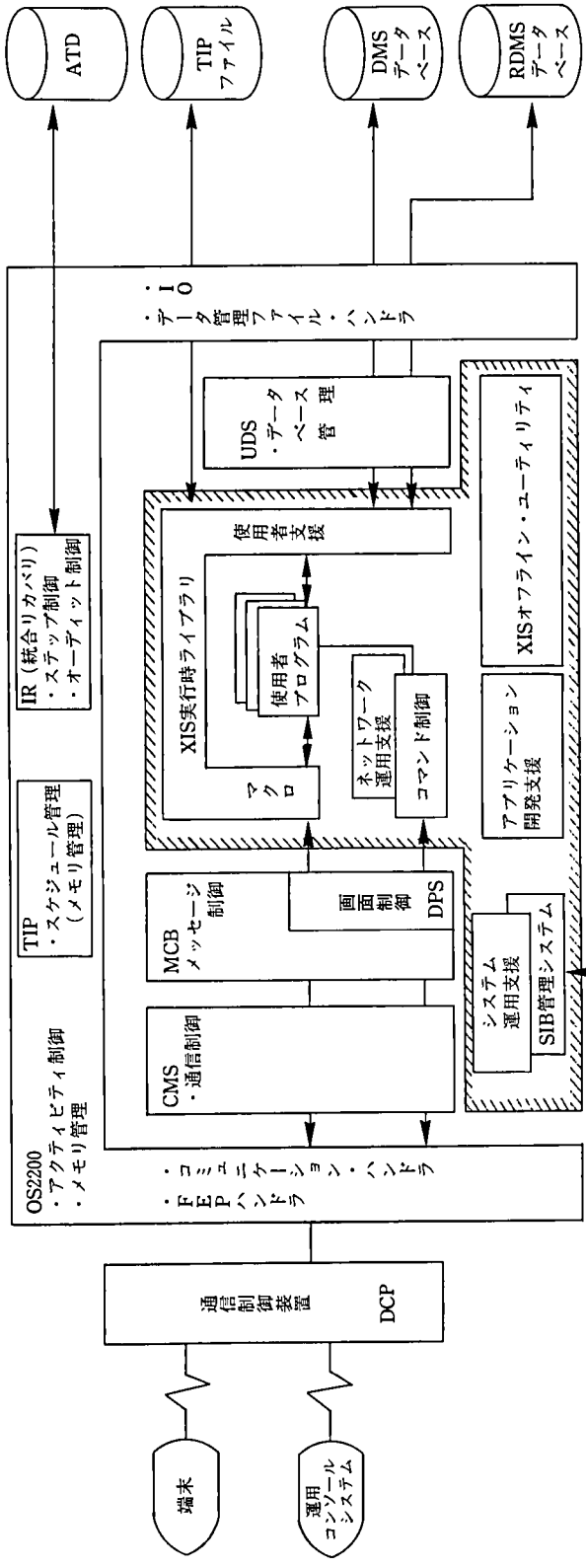
③ 回復ユーティリティIRU 1100との連携により，システム障害あるいはファイル障害からの回復処理を行う．また，コンティンジェンシ・プログラム，システムエラー・プログラムなど標準エラー処理を行う．

④ 同一ホスト・コンピュータ内に複数のXISシステムを構築することができる(マルチXIS機能)．

2) メッセージ制御機能……外部システムとの種々のメッセージのやりとりを支援し，メッセージ回復をはじめとする豊富なメッセージ処理機能を提供する．

---

* 2200/XPAは，中央処理系の能力を大きく向上させると同時に，画期的な入出力処理アーキテクチャによってシステム処理能力を飛躍的に高め，さらにノンストップ大規模処理も可能にしている．拡張データ処理装置XPCは，この2200/XPAを具現化する中核として，並列処理ベースの新しいデータ処理環境を実現している．



DCP : Distributed Communication Processor　　ATD : Audit Trail Disk
FEP : Front End Processor　　SIB : System Information Base
IR : Integrated Recovery

図 5　XIS の概要

① 入力メッセージ中のある項目あるいは端末種別により，処理プログラムを特定し起動する．

② 入力メッセージに対し，多重受信，分割受信および通番管理機能を提供する．

③ 出力メッセージに対し，保存/送信，送達確認，通番管理，回復/再送機能を提供する．また，交換メッセージに対しては，さらに連続出力，選択送信，多重宛先送信，流量制御機能を提供する．

④ 使用者プログラムが外部システムと複数回の対話が行える（会話リンク機能）．

⑤ 使用者プログラムから他の使用者プログラムをメッセージ付きで起動する(パスオフ・メッセージ)．また，システム障害時の該メッセージ回復処理を行う．

⑥ 画面/帳票定義，メッセージ編集を支援する DPS 1100 II とのインタフェースを提供する．

⑦ TELCON/CMS との CENLOG(Critical Event & Notification LOG)，TH (Terminal Handler)ステータス，オペレーショナル・メッセージおよび DCP アップ/ダウン，セッション・オープン/クローズ通知を受信し，CMS 2200 コマンド，TELCON/NMS(Network Management System)コマンド発行等ネットワーク運用を支援する．

DP-M 1 TH のデュアル・セッション機能との連携により，ホスト障害時の自動セッション切替え，回復処理を行う．また，XTPA 環境下で，外部システムが XTPA を構成するどのホスト・システムに接続されているかを意識することなくメッセージの送受信が行える（自動メッセージ転送機能）．

3) データベース制御機能……複雑なデータ構造を効率よく支援し，多様な処理要求に適切に応える．

① UDS 1100 による複数データモデルの統合管理としてネットワーク型データベース（DMS 1100）とリレーショナル型データベース（RDMS 1100）を，共通アーキテクチャのもとに制御し，統一的なリカバリ，ロック管理などを実現する．

② 自動エリアオープン，マルチエリアのエリア決定，エリアロック，マルチスキーマ・インボーク，テストモード機能を含む DMS ベーシック・インタフェースおよびロジカルセット，インバーテッド検索などを可能とした，ハイレベル DMS インタフェースを提供する．

③ 物理媒体や APG 形態を意識させずレコード・アクセスを実現するシステム・テーブル機能を提供する．従来の FCSS(File Control SuperStructure)とシステム・テーブルを統合し，1 テーブルのメモリ/ディスクまたがり配置，複数ディスクパックへのまたがり配置を可能とし，効率，サイジング，I/O 負荷分散等チューニングが使用者プログラムの変更なしに環境（媒体）変更のみで実現可能である．

また，テーブルごとの独立性を保証し，テーブル単位の内容変更が容易であり，テーブル単位の動的追加が可能である．

システム・テーブル使用者は，テーブル番号，レコード番号で直接アクセスでき，ロック機能，サーチリード機能が使用できる．回復対象とすることも可能である．

また，XTPA 環境下では，TCDBF を利用して XTPA 構成ホストのメモリ同期がとられる．

④ オーバフロー管理をもつ蓄積型ファイル制御機能を提供する．

⑤ レポートの行単位のリード/ライト/デリートが可能な MAPPER(Maintaining Prepairing and Producing Executive Report)ファイル・インタフェースを提供する．

4) 使用者プログラム支援機能……使用者プログラムがトランザクションおよびバッチ処理をする上で有用な，各種支援機能を提供する．

① トランザクション処理の初期/終了処理を標準 ICP/FCP として提供する．

② 時間/時刻指定（秒単位）により使用者プログラムをメッセージ付きで起動する．

③ センタ側で発生したトランザクションを，あたかも端末からの入力と同様に処理プログラムに渡すセンタカット処理を支援する．標準センタカット・スケジューラ，スケジューラ起動制御，センタカット TPS(Transaction Processing Segment)自動流量制御，入力ファイル蓄積，種々スケジュール制御(コンカレント/シリアル/キー・シリアル）機能を提供する．

④ トランザクション処理の一部を，オンライン時間帯でのバッチ処理として展開していくことにより，トランザクション処理の効率向上を図るとともに，業後バッチの業中化を可能にする（ディレード処理）．同一 APG 内だけでなく異なる APG 間，XTPA 下でのディレード処理が可能であり，ディレードバッチの多段階化も可能である．

⑤ バッチプログラムの障害回復時のリスタートを可能とする．

⑥ バッチプログラム間におけるメモリ上でのデータ共有により，I/O 負荷の軽減を図る．

⑦ OS インタフェースおよび ATD リードルーチンを提供する．

5) システム運用支援機能……オンラインシステムで業務処理以外に必要となる開始/終了/回復処理，回線網の管理および状況照会/変更のためのオペレータ・インタフェースなどの運用管理を支援する．

① SIB リポジトリにシステム構成情報を一元管理し，各種システム・パラメタ，運用 JCL およびシステム・テーブルを自動生成する．

② IOF の自動立ち上げ/終了機能と連動し，XIS システム開始/終了処理の自動化を実現している．

さらに，XIS 自身のハートビートによる他ホスト監視，XIS システム・パニック監視，ATD 使用監視および IOF のシステム状態変更通知(イベント)取得により，障害検知/回復処理の自動化，アーカイブ（トランザクション・ログのセーブ)制御/IOF 起動をも果たす．新たな XIS 運用言語と言語環境の提供により，よりきめ細かな運用制御が可能となり，これら自動化を支えている．

また，XTPA 環境下では，さらに XTPA 構成ホスト間での同期とりが行われる．

③ 運用コンソール・システム（後述）を介して XIS/IOF 共通制御卓を実現している．

XIS コンソールは，運用コンソール・システムの 1 AP として稼働し，該システムの高度な操作が使用できる．また，多くの標準運用コマンドが提供され，メッセージはコンソール・メッセージと詳細メッセージという形で，オペレータ（運用）に必要な情報とプログラマ（保守）に必要な情報を明確に区別することにより，運用の迅速化および保守の容易性を追求している．

6) 使用者開発支援機能……容易にオンライン・システムの開発/テストができるように支援する．

① プログラム・トレース/編集ユーティリティ

使用者プログラムのトレースデータ（XIS とのインタフェース）を，プログラムの変更や再リンクなしに外部からの指示のみで採取することができる．採取したデータは，直接印書装置に出力できるほか，トレースファイルに書き込み，後で編集印書することができる．該編集印書機能は，IDES のファイル・マニピュレータ，単体テストツールと同一のソフトウェア部品により実現されており，使用者は，単体テスト段階（IDES 支援）と結合テスト段階（XIS 支援）において同一のマンマシン・インタフェース（出力フォーマット）が使用できる．

② テストモード

使用者プログラムの実行時に，外部から設定された実行モードによって，そのプログラムがアクセスするデータベース，システム・テーブルなどを，あらかじめ設定されている別のものに振り替えることができる．使用者は，プロダクション環境の中で並行してプログラムの実行テストが可能であり，複数のモードを利用してテスト担当者ごとにテスト環境が設定できる．

3.3 拡張機能（XIS ファミリ）

XIS の運用支援機能の一部は別プロダクトである SIB 管理システム，運用コンソール・システムにて実現される．ねらいは，システム情報の一元管理と MMI の統合であり，開発/IDES，実行/XIS，運用/IOF という三つの環境/プロダクト間の連携強化による，ユーザシステム・ライフサイクル全般の統合支援をめざす．

また，分散処理支援ソフトウェア群の適用により，多様な分散処理形態，開放型システムが可能であり，これら XIS ファミリの充実・拡張により長期的に耐え得る柔構造システムが実現できる．

3.3.1 SIB 管理システム

SIB 管理システムは，複数プロダクト，複数システム/ホストの情報を保有管理するとともに，SIB リポジトリ（情報収納庫）アクセス・インタフェースを提供する．XIS は実行環境として必要な情報を SIB リポジトリに収納/取得する（図 6）．

3.3.2 運用コンソール・システム

運用コンソール・システムは，OS 2/PM による高度な GUI によりハイレベル・マンマシン・インタフェースを実現している．全体制御/通信制御/ウィンドウ制御等基本機能を提供し，上位に各種 AP 画面が実装できる．XIS コンソール・ウィンドウ，

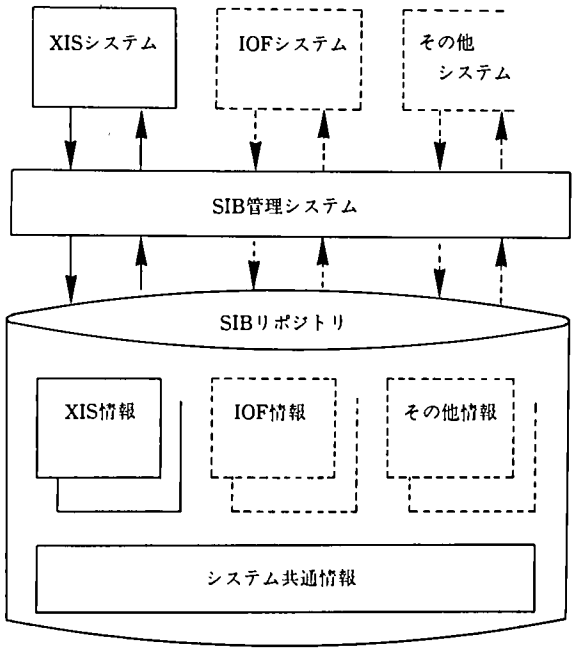


図 6　SIB 管理システム

IOF コンソール・ウィンドウ搭載により，XIS/IOF 共通制御卓（マルチウィンドウ）が実現できる(図 7)．

また，該コンソールは目的別/システム別等任意の台数設置が可能である．

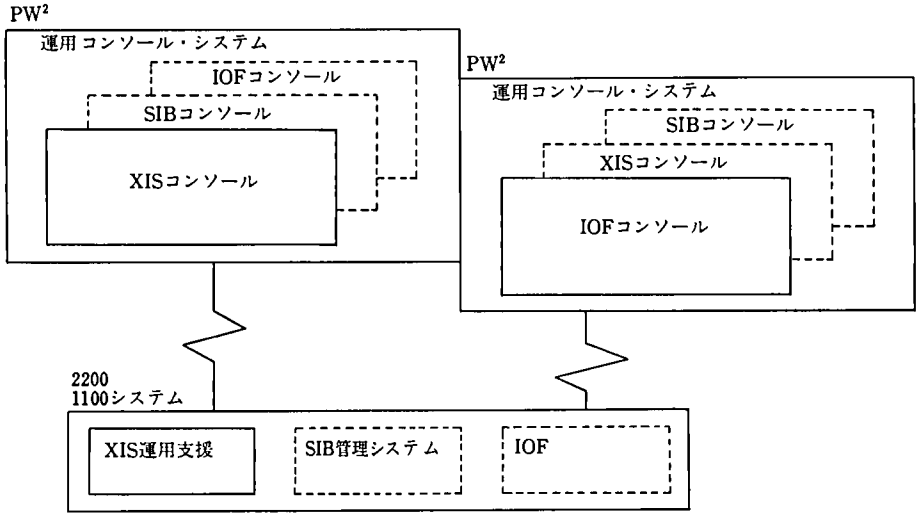


図 7　運用コンソール・システム

3.3.3　分散処理支援ソフトウェア群

1)　ACLES（OSI 通信）

ACLES(Advanced Communication controL Elements and Support system)はトランザクション処理におけるデータ・コミュニケーション機能を体系化し，柔軟性，拡張性のあるデータ・コミュニケーション構造を提供するXISファミリプロダクトである．OSI通信機能を含む基本機能(ACLES/BASE)と機能/プロトコルごとの支援プロダクトから構成される(図8)．

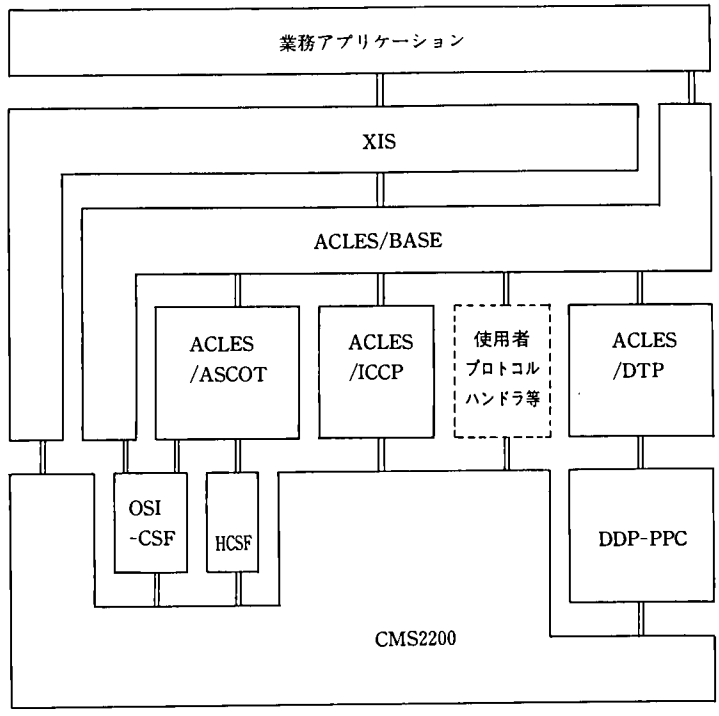


図 8　ACLES ファミリの概要

2)　ACLES/ASCOT, ACLES/ICCP（ホスト間トランザクション転送）

XIS環境下で分散トランザクション転送を支援するプロダクトをACLES/ASCOT（ACLES/Advanced information System COnnecTion：当社ソフトウェア商品)と呼ぶ．ACLES/ASCOTは使用者プログラムが相手システムおよび相手システムとの接続媒体を意識することなくトランザクション転送処理を行うことを可能にする．

また，ACLES/ASCOTのサブシステムとしてBOSS 11/ICCP 1100とのトランザクション転送処理を支援するACLES/ICCP(ACLES/Inter Computer Communication Program：当社ソフトウェア商品）がある．

3)　ACLES/DTP（分散トランザクション処理）

XIS環境下で分散トランザクション処理を支援するプロダクトをACLES/DTP（ACLES/Distributed Transaction Processing：当社ソフトウェア）と呼ぶ．

ACLES/DTPは，使用者プログラム間で複数回相互にデータ転送を行う会話

処理を可能にし，その一連の処理がトランザクションとして完結でき，障害が発生した場合の回復が可能となるよう同期処理を行う．

4) XIS/RDAF（分散データアクセス）

XIS 環境下で分散データアクセスを支援するプロダクトを XIS/RDAF（XIS/Remote Data Access Facility：当社ソフトウェア商品）と呼ぶ．XIS/RDAF は通信媒体で結合された遠隔地にあるホストコンピュータのデータに対し，使用者プログラムから XIS の使用者インタフェースを使用することによりデータの所在を意識することなく，自由なアクセス－参照，検索，更新－が可能となる．

4. XIS の構造

XIS は，プログラムとデータの形で提供される．使用者は，該プログラム/データを使用し，業務システム環境を構築する．システム情報を SIB リポジトリに定義し，システム・テーブルを生成する．

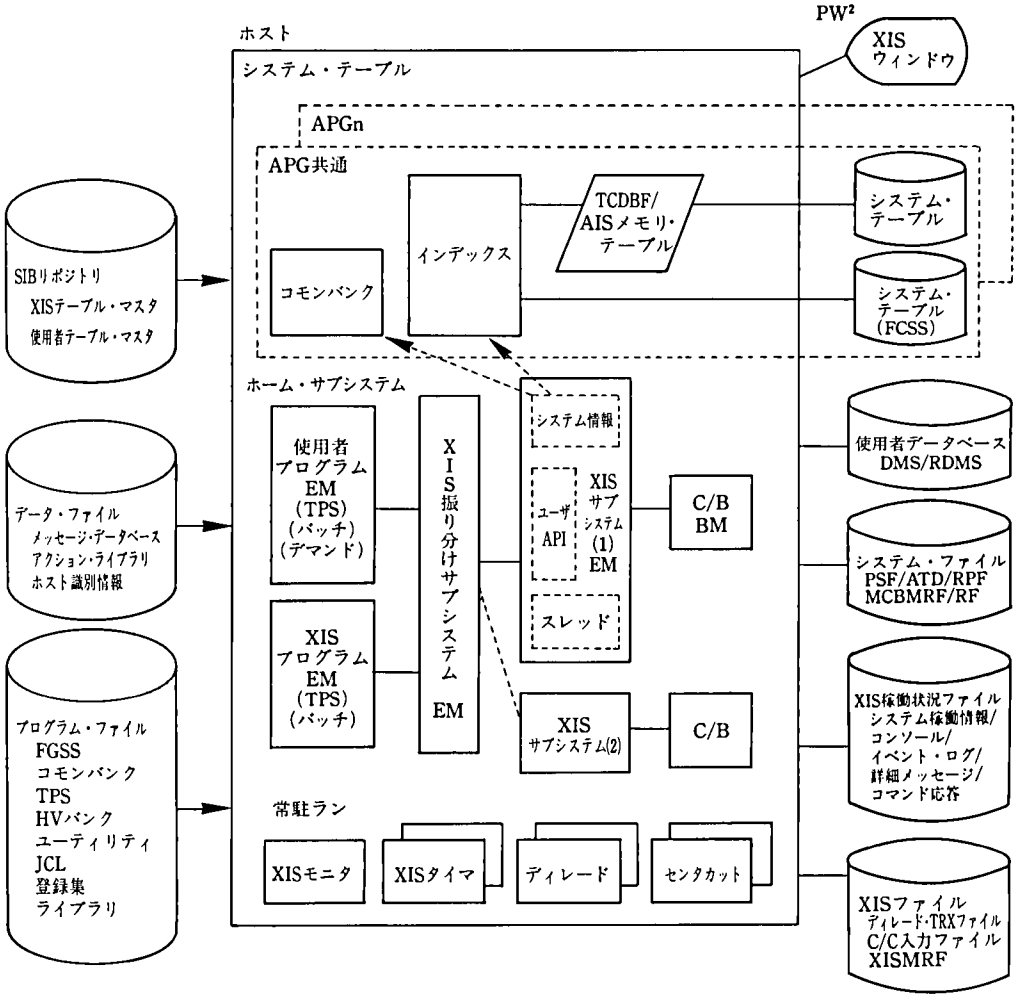


図 9　XIS の構造

使用者プログラムは，ホーム・サブシステムと呼ばれるサブシステムからXIS-FGSSにアタッチする．XISは稼働時，スレッドと呼ばれるメモリ上の作業領域を使用し，上記システム・テーブルから種々情報を取得/更新して機能を果たす．また，モニタ，タイマ，センタカット，ディレードの一部機能は常駐ランとして機能し，ディスク上のXIS稼働状況ファイル，XISファイルを使用する(図9)．

5. XISの現状と今後の課題

5.1 XISの現状

5.1.1 プロダクト・ライン

XISプロダクトは，現在XIS 1 R, XIS 2 Rの2ラインが標準提供されている．
現プロダクトAIS 1100 IIとの相違を表1に示す．

表1 AIS1100 IIとXIS

	AIS	XIS 1 R	XIS 2 R(本稿)
支援ユーザ	BMユーザ	EMユーザ AISアプリケーションのEM化を行うユーザ	EMユーザ XTPAを必要とするユーザ 新たにオンライン・システムを構築するユーザ
機能	—	AIS 1100 II同等機能	XIS 1 Rを大幅に機能拡張(XTPA対応ほか)
プログラム・インタフェース	AISインタフェース	AISインタフェース AISと極僅かの非互換あり	XISインタフェース AIS/XIS 1 Rとの互換性なし
環境	ISD(Integrated System Dictionary)	ISD 新規パラメタ追加	SIB AIS/XIS 1 Rとの互換性なし
支援環境	2200/1100シリーズ	NPE環境対応の2200/1100シリーズ	

5.1.2 ソフトウェア品質

ISO/IEC 9126にて定義されている六つの品質特性に基づきXISを評価すると，下記のとおりである．

1) 機能性（合目的性/正確性/接続性/整合性/セキュリティ）

① 汎用プロダクトであり，使用者インタフェースを部品化することにより，各種機能を実現している．

② 各種基準書(技術設計基準,設計書/仕様書作成基準/各種手続き/名付け規約/コーディング規約等)に則った開発および外部インタフェース(プログラム/コマンド/メッセージ/パラメタ)の標準化により，ドキュメント/ソフトウェアに追跡性，一貫性がある．

③ 標準出荷後，41件の改造要求を受信しているが，多くは実運用にて気がついたメッセージの一部変更および後続ユーザを意識したスケーリング，新端末対応である．

④ XIS-FGSSは機能ごとの複数OMで構成されており，新たな機能追加に際しては新規OM追加あるいは一部OMの変更で対応される,新機能追加により

発生するインタフェース非互換は該機能使用の有無がテーブルにて管理されるので既存プログラムには影響しない.

⑤ APG という運用の単位にプログラム/データベース/端末を割り振ることにより，互いに干渉することのない業務システムが構築できる.

⑥ プログラム，端末，データベース情報は，事前に SIB 管理システムに登録する必要があり，登録されていないものは XIS を使用することができない.

2) 信頼性（成熟性/障害許容性/回復性）

① 平成 6 年 1 月現在, XIS 2 R 使用本番ユーザは 3 社であるが, システム全体に影響するメジャートラブルは発生していない.（マイナー・トラブル 0.8 件/サイト・月）

② XIS 本体部分は FGSS 構成となっているため，他サブシステム（使用者プログラム等ホーム・サブシステム）から該 FGSS 内データバンクへの書き込みは保護される.

③ OS 2200/TIP 1100/IRU 1100 等ベーシック・ソフトウェアとの連携により，データベースの二重化，システム過負荷の監視を実現しており，障害の波及から保護している.

④ 障害になった端末/プログラム(TPS, HV バンク)，データベース(エリア/テーブル）単位の閉塞により，障害箇所の局所化を図ることができる.

⑤ XTPA システム使用時, ホスト障害により業務システム全体障害とならない（ノーダウン・システム）. また，APG 分割によりホスト/AG 単位の障害・回復が可能であり，これら障害検知，回復処理が自動的に行われる.（ホスト/AG: XTPA システムの場合，複数のホストをまたがって APG が構成される. よって，あるホストの同 APG を表すときホスト/AG と表現する.）

3) 使用性（理解性/習得性/操作性）

① 使用者プログラムは XIS とインタフェースをとることだけで，すべての DB/DC 機能を使用することができ，OS 2200, TIP 1100, IRU 1100, CMS 2200, MCB 1100, UDS 1100(DMS 1100, RDMS 1100)，PDIR II，DPS 1100 II 等ベーシック・ソフトウェアと直接インタフェースをとる必要はない.

また，XIS ファミリ使用による，ホスト/APG 間トランザクション転送（対 XIS/AIS/BOSS システム)，分散トランザクション処理，分散データアクセス，OSI 接続処理において，使用者プログラムに分散を意識させないインタフェースが提供される.

② 同一ホストに複数の XIS を搭載できるマルチ XIS 機能を使用することにより，既存システムに全く影響を与えずに新システムを構築することが可能である.

③ ドキュメントは，使用者向けに目的，読者を意識したマニュアルが整備されている.

4) 効率性（実行効率性/資源効率性）

① NPE 環境支援，豊富な機能実装により，従来の基本モード・システムに較べると多くの資源(プロセッサ，メモリ，ディスク)を必要とする. ただし，XIS

本体のFGSS化，システム共通情報のコモンバンク化により，メモリの有効利用およびI/O負荷軽減を図っている．また，XIS-FGSSは機能単位にOM分割されており，必要な機能のみメモリ常駐させることが可能である．

5) 保守性（解析性/変更作業性/安定性/試験性）

① 使用者プログラム障害時，XISコンソールにエラー・メッセージが表示され，エラー情報ファイルにプログラム制御情報（ラン名/トランザクションコード，入力端末名，エラーコード，XIS使用者インタフェース遷移/ステータスコード/XIS内モジュール遷移/HVバンクヒストリ等）がとられる．メッセージコードには，XIS内カテゴリ識別コード，設計書/仕様書番号が埋め込まれており，XIS内の該エラー検出箇所が追跡しやすくなっている．また，外部指示によりユーザ・トレース，XIS使用者インタフェース詳細トレースを取得することができ，エラー検出を容易にしている．

XIS障害については，XIS-FGSSのOM分割にて影響範囲の局所化が図られている．

② 本番環境のテスト使用を可能とするテスト・モードを実現している．

③ プロダクト面では，ソフトウェアについてはCCF(Change Correction Form)/PCF(Permanent Correction File)による変更管理がなされており，ドキュメントとして修正情報追補/リリース・メモ追補が出状される．単体/結合/総合テストおよび効率測定ツールを保持しており，テスト・データの蓄積等，テストの生産性向上を図っている．

また，トラブル，改造要求，開発項目について文書管理がなされており，ユーザでのレベルアップ時のレベル間相違（機能，バグの有無，非互換）等が明確となっている．

6) 移植性（環境適応性/移植作業性/規格準拠性/置換性）

① 汎用的な統合オンライン支援パッケージであり，システムの規模，業種を問わず使用可能である．

② 各種機能は明確に分割されているので必要機能のみ選択使用することが可能である．

③ ユーザあるいはシステムごとに変化のあるシステム情報（データベース，端末，プログラムおよび規模情報）はすべてテーブルウェア化されており，全システム同一XISが使用可能である．

また，業務量拡大/縮小のためのデータベース/ネットワーク等環境変更に際し，使用者プログラムの変更は不要である．

④ XISシステムの生成と実行は独立しているので，別ホストで構築した環境を容易に移植できる．

⑤ ソフトウェア自体はUNISYS 2200シリーズのみで使用可能な言語で作成されており，他のオペレーション・システムへの移植はできない．ただし，各種基準に則った技術資料はベーシック・ソフトウェアとのインタフェース部分を除き有効である．

5.2 今後の課題

XIS は，AIS 1100 II の後継プロダクトとして開発された．NPE/XTPA 環境支援等大幅な機能拡張を実現したが，AIS 1100 II からの移行を考えると不足する機能がある．また，新ハードウェア，ベーシック・ソフトウェアの新機能対応および更なるスケール拡大をめざし，XIS 3 R を開発する．

以下に XIS 3 R の主な課題を示す．

1) BM アダプト機能……XIS のユーザ基盤となる既存 AIS ユーザは，BM 環境下で開発を行っている．

XTPA 機能等 XIS の新機能を使用したいが，全ての業務プログラムを EM 化し，テストするのは負荷がかかり過ぎる等，AIS から XIS の移行に対して使用者負荷を極力減少させるため，既存 AIS(BM) プログラムからの XIS アクセスを許す．

2) AIS 互換インタフェース……XIS 2 R は，多くの機能を AIS/XIS 1 R とは非互換のプログラム・インタフェースにて提供している．

AIS のもとで永年蓄積された使用者ソフトウェア資産を引き続き使用できるよう AIS(BM)/XIS 1 R(EM) の AIS インタフェースにて XIS 2 R-XTPA 機能を使用可能とする．また，XIS 2 R で未対応の AIS 機能－集配信機能，ホットスタンバイ機能，メモエリア機能－を追加提供する．

3) 新ハードウェア対応……新ハードウェア XPC(eXtended Processing Complex) を有効活用し，さらなる大規模システムを支援するとともにノーダウン・システムの充実を図る．

4) ベーシック・ソフトウェア新機能対応……OS 2200 の新機能として TIP セキュリティが発表された．XIS では該機能を有効活用し，オンライン・システム下のセキュリティを追求する．

また，OPEN/OLTP(Online Transaction Processing) 2200 配下の使用者プログラムが XIS を使用可能とする．

5) スケール拡大対応……NPE 環境の適用により多くのスケールリミットが拡大されたが，未だ BM/EM ミックスモードのためネックとなっている部分がある．とくに，TCDBF，XIS 作業領域等システム規模が大きくなるにしたがって必要となるエリアの EM 化はベーシック・ソフトウェアを含め大きな課題である．また，OS 2200 の APG 拡大対応(8 APG → 16 APG) に追随する．

6) 資源効率性の追求……XIS 2 R は，大規模 XTPA システムユーザをターゲットに開発が開始された．今後もしばらくは大規模ユーザの使用が見込まれるが，その後の中小規模ユーザも睨み，資源効率性，導入容易性の追求が必要である．

7) その他新機能……埋め込み SQL(Structured Query Language) の使用を可能にする新 RDMS インタフェースおよび SFS(Shared File System) インタフェースを提供する．

6. おわりに

平成元年から 4 年の歳月をかけ開発してきた XIS 2 R は，平成 5 年 5 月のファース

ト・ユーザ2社の本番により，一応の成功が確認できた．

長期間に及ぶソフトウェア要求仕様書のレビュを経て，設計段階に入る．設計基準をはじめ各種基準書，手順書の作成，標準化により，延べ200人に及ぶ開発メンバの統一を計る．ハードウェア(2200/900, RLP)，ベーシック・ソフトウェア(XTC-TIP, XTC-UDS, UCS)および業務アプリケーションすべてが新規ということもあり，当初の要求仕様あるいはインタフェースはかなり変更/追加された．

4ホストのXTPAシステム，新規ハードウェアRLP, HLC付加ということで，従来の1ホスト・システムとは比較にならないテストケースの多さ，テスト時間の長さであり，トラブル時の切り分けにも相当の時間を要した．

世界初のXTPAシステム構築の一端を担ったXIS 2 Rは，後続XTPAシステムユーザに適用されていくとともに，さらに機能強化され次世代のオンライン・システム支援パッケージとして発展していく．

また，副資産として生まれた各種基準書，手順書は，当社開発標準との整合性をとり，今後ISO 9000-3の品質マニュアルとして発展させる．

XIS 2 R開発にあたり，ご協力をいただいたユーザの方をはじめとして，当社プロダクト担当者に深く感謝の意を表する．

執筆者紹介 育 野 准 治 (Junji Ikuno)

1977年電気通信大学電気通信学部電子計算機学科卒業．同年日本ユニシス(株)入社．銀行システム・サービスを経て，インフラストラクチャの企画，開発に従事．現在システム企画開発二部プログラムマネジメント課に所属．

XTPA 環境における XIS のデータ通信制御

Data Communications Control by XIS in the XTPA Environment

石 川 清 人

要 約 統合オンライン支援システムである XIS(eXtended Information System)は，XTPA システムの特徴を十分に活用するためのさまざまな機能を提供している．

本稿は XIS のデータ通信制御機能が提供する XTPA 対応機能について記述する．

XIS のデータ通信制御機能が提供する主な XTPA 対応機能は以下のとおりである．

1) 処理を実行するホストを意識することなく，使用者プログラムのオンライン業務処理を可能とする．

2) XTPA システムを構成する各ホストで負荷を分散し，使用者プログラムが各ホストで並列的に業務処理を実行することを可能とする．

3) 障害発生時にもオンライン業務処理に支障をきたすことなく，端末業務処理を連続稼働させる．

Abstract As a solution tool to support online transaction processing, XIS (eXtended Information System) provides a wide range of functionalities that help users make good use of XTPA features. This paper describes XIS's data communications control module so designed as to meet XTPA requirements. The module makes it possible in multi-host environments:

1) for any user program to be executed online without users being aware which host computer will process it.

2) for computing loads to be automatically distributed among/between hosts so any user program can be processed concurrently.

3) for uninterrupted online processing on the side of terminals to be guaranteed even when a failure happens to a host.

1. はじめに

当社は，より高い水準の大量高速処理や無停止連続処理の要求に応えるべく拡張トランザクション処理体系である XTPA(eXtended Transaction Processing Architecture)を提供している．XTPA は，フォールトトレランスの冗長技法の一つである疎結合に拡張性を考慮し，メモリはプロセッサごとに独立し，データベースはプロセッサ間で共有させた不完全疎結合[1]のアーキテクチャを取り入れている．XTPA に基づいて構築されるシステム（以降 XTPA システム）は，最大 4 台の独立したホストを結合して，データベースを密接に共有させ，システムの成長性，各システム間の完全独立性，複数システムによる業務分散，無停止連続運転などを実現することができる(本特集号別稿“XIS の XTPA システム運用”参照)．

XTPA システムは，OS 1100(Operating System 1100)，TIP 1100(Transaction Interface Package 1100)およびデータ通信を制御する CMS 2200(Communication Management System 2200)，MCB 1100(Message Control Bank 1100)とデータベー

スを制御する UDS 1100(Universal Data System 1100)などの基本ソフトウェア群から構成されている．しかし，これらのソフトウェアが提供している機能だけではXTPA の特色を発揮したオンライン・システムを構築することはできない．

統合オンライン支援システムである XIS(eXtended Information System)は，XTPA の特徴を十分に活用した環境でオンライン・システムを構築するための，さまざまな機能を提供している(本特集号別稿“統合オンライン支援システム XIS の概要”参照)．本稿は，XIS のデータ通信制御機能が提供する XTPA 対応機能について報告する．

2. XIS データ通信制御機能の XTPA 対応

XTPA システムは，RLP(Record Lock Processor)を使用してデータベースを共有するとともに，運用/リカバリの単位であるアプリケーション・グループ* も全構成ホスト上で共用される．このコンカレント・アプリケーション・グループ（以降コンカレント APG）により，使用者は XTPA システム全体を透過的に意識することができる．XTPA システム環境では，このような特色を活用して，使用者プログラムに処理を実行するホストを意識させず(ホスト透過性)，共有データベースを効率的に並列処理し（並列制御)，ホスト障害時にも業務処理に支障をきたすことなくオンライン・システムを連続稼働させる（無停止運転）ことができる．

しかしながら，データ通信制御関連の基本ソフトウェアである MCB 1100 と CMS 2200 は各構成ホスト上で独立して稼働するため，通信を行うプログラムは，ホストを意識せざるをえない（図 1)．

XIS のデータ通信制御機能は，これらの要件を満たすために，それぞれ以下の機能を提供している．

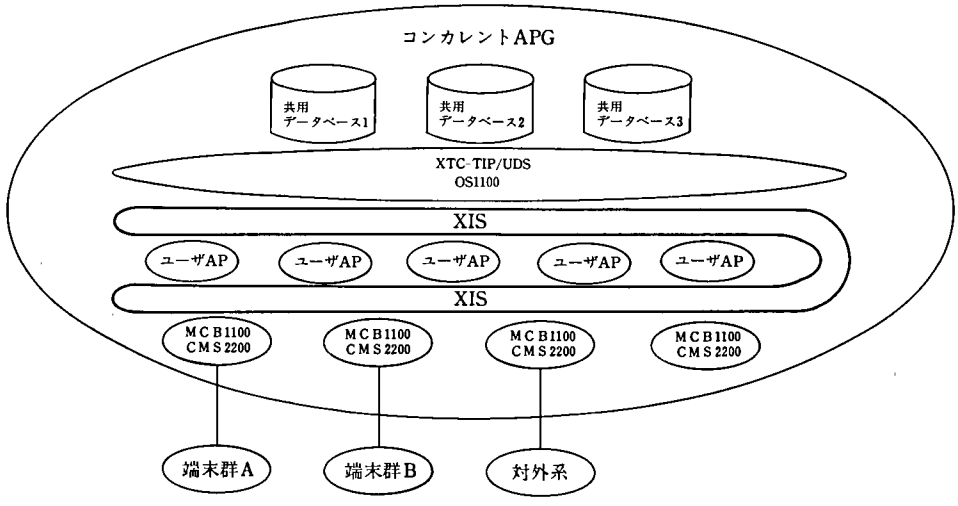


図 1　コンカレント APG の論理構成

* アプリケーション・グループ(APG)はリカバリの単位であり，ローカルとコンカレントの二つの型がある．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．

1)　ホスト透過性……ホスト透過性のために「内部転送機能」を提供している．内部転送機能は，使用者プログラムから処理を要求されたホストと，実処理を行うホストが異なる時に，使用者プログラムに意識させることなくホスト間で処理を転送して実行する機能である．使用者プログラムが端末へ送信要求したホストと，宛先端末が物理的に接続しているホストが異なる場合などに使用される．

2)　並列制御……並列制御のために「資源のホスト割り付け機能」を提供している．資源のホスト割り付け機能は，XTPA システムを構成する全てのホストにおいて頻繁に使用される資源をあらかじめ各ホストに一定数づつ割り当て，エントリを取得する時のホスト間競合等による処理効率の低下を防止する機能である．回復メッセージ送信時に取得するメッセージ保存ファイルなどが対象となっている．

3)　無停止運転……無停止連続運転を支援するために「デュアル・セッション制御機能」，「ネットワーク・イベント制御機能」および，「パスオフ/チェックポイント・リカバリ機能」を提供している．

デュアル・セッション制御機能は，DCP (Distributed Communications Processor) とホストの間にオンラインとバックアップのシステム・セッション* を設定し，TELCON および TH (Terminal Handler) と連携しながらホスト障害時などに自動的に切り替えや再設定を行い，端末とホストの間の通信を途絶えさせない機能である．

ネットワーク・イベント制御機能は，オンライン・システムで使用している通信ネットワーク内で発生したネットワーク・イベント** をオンラインとバックアップのホストで制御し，オンライン制御ホスト障害時にバックアップ制御ホストが自動的に制御を引き継ぎ，ネットワーク・イベントの欠落を防止する機能である．

パスオフ/チェックポイント・リカバリ機能は，ホスト障害発生時に仕掛り中であったパスオフ・メッセージとチェックポイント・メッセージを，他のホストでリカバリする機能である．

本稿では，これらの機能のうちデュアル・セッション制御機能，内部転送機能，および資源のホスト割り付け機能について記述する．

また，以降 A ホストから D ホストまでの 4 ホスト XTPA 環境を前提として例示する．

3. デュアル・セッション機能

3.1 デュアル・セッションの概要

XTPA システム環境では，システムを構築している複数ホストのうち，1 台のホストに障害が発生してもオンライン・システムとして業務処理を継続することが可能である．しかし従来の TELCON/TH および CMS 2200 の機能では，障害が発生したホ

---

*　システム・セッション：DCA (Unisys Distributed Communications Architecture：通信を行うためのユニシス独自のネットワーク・プロトコル) にもとづいて TELCON と CMS 2200 の間に割り当てられる論理的な通信経路．

**　ネットワーク・イベント：通信ネットワーク内で発生した事象は，TELCON からの CENLOG メッセージ，TELCON/TH からの TH ステータス・メッセージ，TELCON/CMS 2200 からのオペレーショナル・メッセージとしてホスト・システムへ通知される．本稿で記述しているネットワーク・イベントとは，CENLOG メッセージ，TH ステータス・メッセージおよび，オペレーショナル・メッセージのことである．

ストとの間にシステム・セッションが設定されていた端末は，ホストとの間の通信がいったん途絶えてしまう．そのため，端末業務処理システムは無停止運転が不可能となってしまう．

そこでDP-M 1* など一部の TH では，端末との通信を行っていたホストに障害が発生した場合も，端末とホスト・システム間の通信を途絶えさせないため，デュアル・セッション機能を提供している．

デュアル・セッション機能とは，TELCON と CMS 2200 の間にオンライン・セッションとバックアップ・セッションの二つのシステム・セッションを設定しておき，通常はオンライン・セッションを使用して通信を行い，オンライン・セッションがホスト障害などにより使用不能となった場合には，それまでのバックアップ・セッションをオンライン・セッションとして通信に使用する機能である．

この機能により，端末とホストの間のシステム・セッションは常に設定されていることとなり，端末業務処理システムの無停止運転が実現される．

デュアル・セッション機能を使用する場合，XIS はオンライン・セッション/バックアップ・セッションの設定ホストや設定状況を管理し，制御する機能を提供する．

デュアル・セッションの接続例を図 2 に示す．

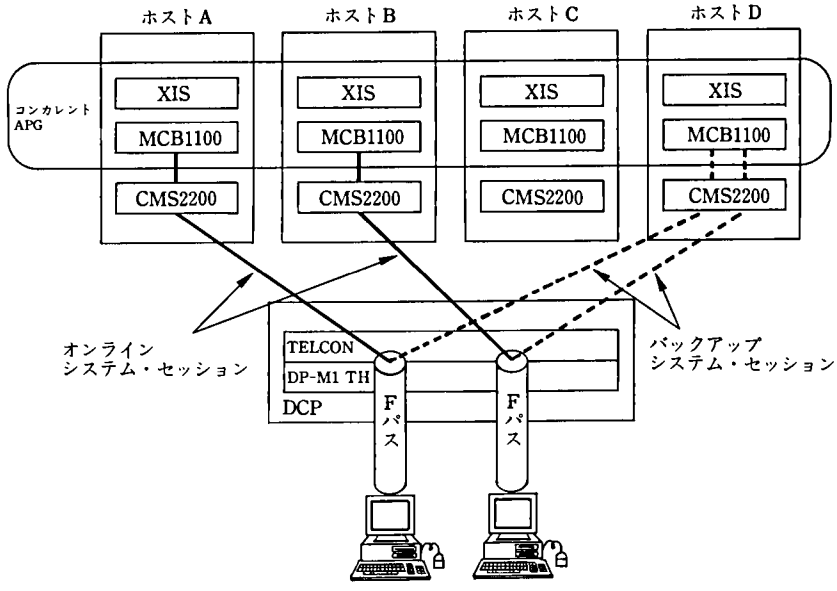


図 2　デュアル・セッションの接続

3.2　デュアル・セッションの初期設定

デュアル・セッションにより無停止運転を実現するためには，端末業務処理に先だってオンライン・セッション，バックアップ・セッションの両システム・セッションを設定しておく必要がある．

デュアル・セッションを設定するには，TELCON/TH からインバウンド・オープ

* DP-M 1: HLP(Higher Level Protocol: DCNA(Data Communication Network Architecture)に従ったプロトコル)に準拠した通信手順．

ン* にて設定する方法と，ホストからアウトバウンド・オープン** にて設定する方法とがある．

インバウンド・オープンにより設定すると，オンライン/バックアップのセッションが設定されるホストは，それぞれ TELCON のシステム生成により設定されたホストに限られてしまう．バックアップ・セッションが設定されるようシステム生成されているホストが障害により稼働しない場合，バックアップ・セッションが設定されないため，無停止運転が実現できないことになる．

アウトバウンド・オープンによりデュアル・セッションを設定する場合，TELCON/TH は，最初にアウトバウンド・オープンが要求されたホストにオンライン・セッションを設定し，オンライン・セッション設定後にアウトバウンド・オープンが要求されたホストにバックアップ・セッションを設定する．使用者は，ホストの稼働状況などを考慮し，セッションを設定するホストを動的に変更することにより，端末業務処理システムの無停止運転を実現することができる．

デュアル・セッションの設定をアウトバウンド・オープンにより行うことで，オンライン・システムの無停止運転，障害時の柔軟な対応が可能となる．

3.3 ホスト障害時のデュアル・セッションの設定

オンライン・システム稼働中にホスト障害が発生すると，TELCON/TH は障害が発生したホストとの間にオンライン・セッションが設定されていた端末のオンライン/バックアップ・セッションを変更する．つまり，障害発生以前のオンライン・セッションをバックアップ・セッションに，障害発生以前のバックアップ・セッションをオンライン・セッションに変更する．以降の端末からの入力メッセージは，オンライン・セッションとなったセッションが設定されているホストへ入力される．

XIS は，端末のシステム・セッションの設定状況および設定ホストを，XIS のシステム・テーブルであるセッション管理テーブル上で管理している（表 1）．XIS は，ホスト障害発生時，セッション管理テーブル上のセッション設定ホスト情報を，TEL-

表 1　セッション管理テーブルの管理情報

管理項目	管理内容
オンライン・セッション設定ホスト情報	オンライン・セッション設定ホストを，ホスト識別子で管理する．
オンライン・セッション設定状況	オンライン・セッションが設定済か，未設定かを管理する．
バックアップ・セッション設定ホスト情報	バックアップ・セッション設定ホストを，ホスト識別子で管理する．
バックアップ・セッション設定状況	バックアップ・セッションが設定済か，未設定かを管理する．
オリジナル・オンライン・セッション設定ホスト情報	TELCON のシステム生成など環境設定時に定義されているオンライン・セッション設定ホストを，ホスト識別名で管理する．
オリジナル・バックアップ・セッション設定ホスト情報	TELCON のシステム生成など環境設定時に定義されているバックアップ・セッション設定ホストを，ホスト識別名で管理する．

* インバウンド・オープン：TELCON/TH とホスト間のシステム・セッションを TELCON/TH 側から設定する方法である．
** アウトバウンド・オープン：TELCON/TH とホスト間のシステム・セッションをホスト側から設定する方法である．

CON/TH と同様に変更し，障害発生以前のバックアップ・セッションをオンライン・セッションとする．以降の端末への出力メッセージ送信処理は，オンライン・セッションとなったセッションが設定されているホストで行われる．

障害が発生したホストの障害復旧時に，XIS は，セッション管理テーブルより該ホストにバックアップ・セッションを設定する端末を特定し，バックアップ・セッションをアウトバウンド・オープンにより再設定する．

3.4 デュアル・セッションの状態管理

XIS は，デュアル・セッションのオンライン/バックアップの両セッションの設定ホスト，および設定状況を使用者が参照/変更できる各種インタフェースを提供する(表2)．

表2 システム・セッション関連コマンド

コマンド・インタフェース名	プログラム・インタフェース名	処理内容
XNSTAT	XNETSSRD	システム・セッション設定ホストを参照する．
XNSCHG	XNETSSWR	システム・セッション設定ホストを変更する．
XNSSOP	——	システム・セッションをアウトバウンド・オープンにより設定する．
XNSSCL	——	システム・セッションの設定を解除する．

XTPA システムでは，業務処理を行う各ホストで均等に業務処理を行うことが要求される．使用者は，セッションの設定ホストを変更/再設定することにより，XTPA システムを構成するあるホストに端末業務処理が集中することを防止し，特定のホストへの業務処理の集中を防止することが可能である．

3.5 デュアル・セッションの運用方法

デュアル・セッション機能は，オンライン・システムの無停止運転のために非常に有効な機能である．しかし無停止運転のためには，ホストなどの障害発生時に，バックアップ・セッション設定ホストの変更および再設定を行わなければならないため，運用には十分な考慮が必要である．

ホスト A からホスト D までの 4 台のホストで XTPA システムを構成し，各ホストで以下のような処理を行うシステムにおけるデュアル・セッションの運用例を記述する．

ホスト A：A 群端末による業務処理を行う．

ホスト B：B 群端末による業務処理を行う．

ホスト C：外接系などの業務処理を行い，端末業務は行わない．

ホスト D：A, B, C 各ホストのバックアップ・ホストとして運用される．

端末業務処理を行っているホストに障害が発生した場合，バックアップ・セッションが設定されている D ホストにて端末業務処理が行われる．A ホストに障害が発生した場合，D ホストには A 群端末のオンライン・セッションと B 群端末のバックアップ・セッションが設定されている．この状態で B ホストに障害が発生した場合，A 群端末，B 群端末共に業務処理が D ホストで行われることになり，D ホストの負荷が増加する．そこで，端末業務処理を行うホストに障害が発生した場合，D ホストにバッ

クアップ・セッションが設定されている端末のバックアップ・セッションの設定を解除し，バックアップ・セッション設定ホストを障害発生ホストに変更している．つまり，A ホストに障害が発生した場合，B 群端末のバックアップ・セッションの設定を一度解除し，バックアップ・セッション設定ホストを A ホストに変更する．このよう

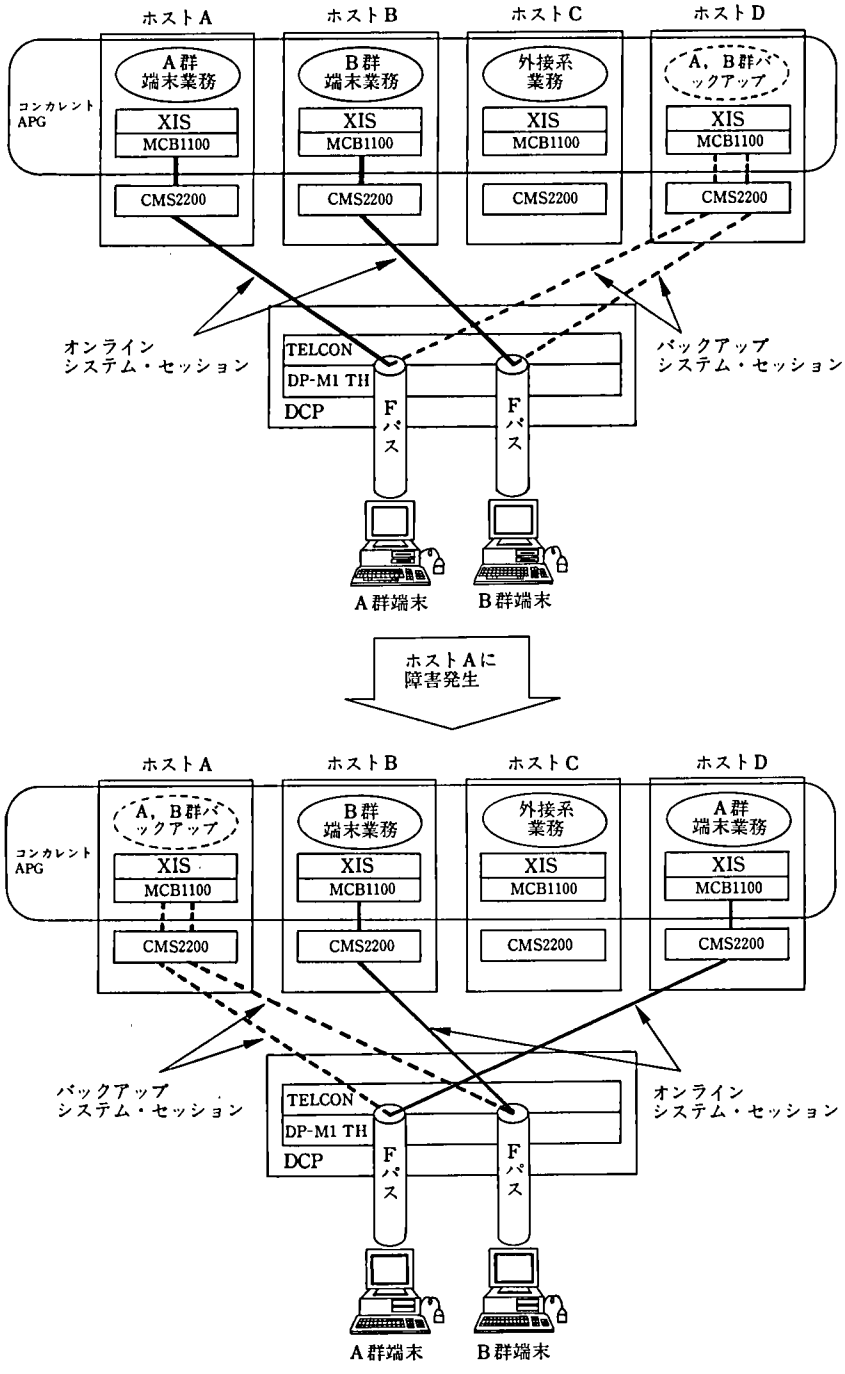


図 3　デュアル・セッションの運用例

な運用により，一つのホストでA群端末，B群端末双方の端末業務処理を行うことを防止している(図3)．

4. 内部転送機能

4.1 内部転送機能の目的

XTPAシステム環境における通信制御システム構成の概念図を図4に示す．

DCP/TELCONおよびDCP/TELCON配下のネットワーク構成要素は，XTPAシステムを構成する全ホストとDCPの間にAMS(Application Management Service)セッションを設定しておくことにより，すべてのホストで制御可能である．

CMS 2200/MCB 1100は，XTPAシステムを構成するホスト各々で独立して稼働している．また，端末との通信のためのシステム・セッションの設定/解除およびシステム・セッションの設定状況の管理は，各ホストのCMS 2200/MCB 1100が行っている．端末との通信に使用できるシステム・セッションは唯一のCMS 2200との間に設定されるため，CMS 2200/MCB 1100に対し端末へのメッセージ送信要求を行う場合，および端末とのシステム・セッションの設定/解除要求を行う場合に，処理を実行するホストを，処理要求プログラムが意識する必要がある．また，CMS 2200に対してコマンド要求を行う場合，どのホスト上で稼働しているCMS 2200にコマンド要求を行うかを，処理要求プログラムが意識する必要がある．

使用者プログラムが処理を実行するホストを意識することは困難であるため，XISは内部的に処理実行ホストを特定し，処理実行ホストへ処理を内部的に転送する内部転送機能を提供する．

4.2 内部転送機能使用時の環境

端末とのシステム・セッションは，XTPAシステムを構成する全てのホストのうち，

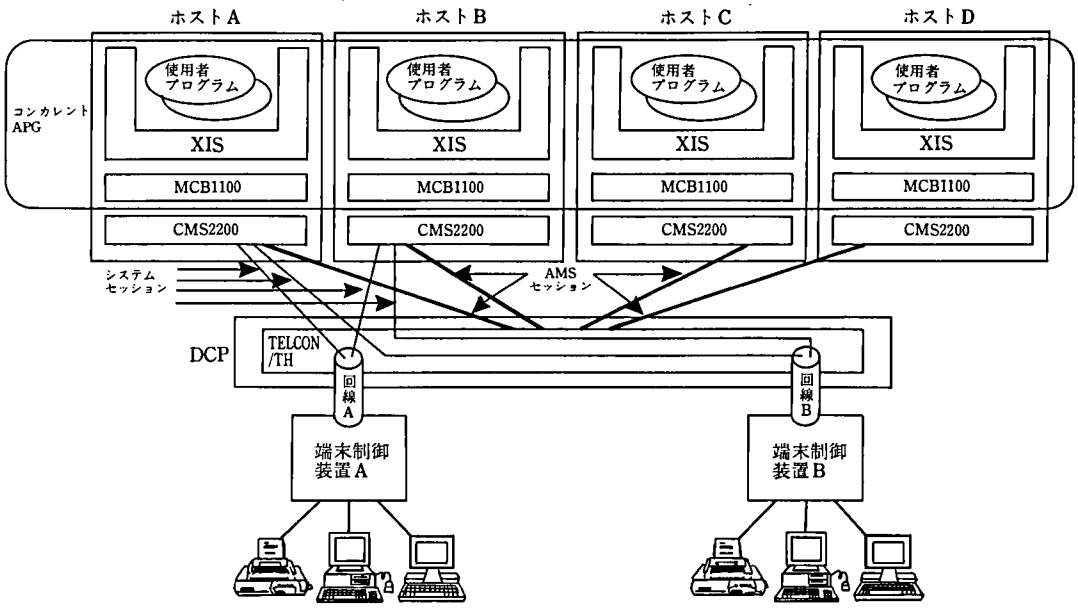


図 4　通信システム構成の概念図

任意のホストに設定可能である．つまり，使用者プログラムが端末へメッセージ送信要求を行う際に，XTPA システム環境を構成する全てのホストが処理実行ホストとなる可能性がある．したがって，XTPA システム環境を構成する全てのホストの組み合わせで内部転送処理が行えるよう環境を設定しなければならない．また，すべてのホストの組み合わせで内部転送を行うホスト間セッション（以下内部転送セッションと記述する）を設定する必要がある．

内部転送セッションは，XIS に登録するとともに CMS 2200 のネットワーク構成定義にホスト間のセッションとして定義する必要がある．XIS 通信制御機能は，内部転送セッションで使用する通信制御装置は，HLC(Host Lan Controller)または DCP を前提としている．

XIS は，内部転送機能により転送されるメッセージに，転送される処理内容，転送メッセージ送信元ホストの識別子などを設定し，内部転送先のホストにおいて処理内容が明確となるよう制御している．

4.3 メッセージ送信処理における内部転送処理

一つの端末に対してメッセージ送受信処理を行うことが可能なシステム・セッションは，唯一の CMS 2200 との間にのみ設定可能である．そこで XIS は，システム・セッションが設定されているホスト情報，システム・セッションの設定状況を管理している．

使用者プログラムは，XIS のメッセージ送信要求インタフェースである 'XMSGSEND' を呼び出すことにより端末にメッセージを送信する．XIS は，使用者プログラムにより送信要求時に指定された端末が，メッセージ送信要求が行われたホスト以外のホストとシステム・セッションを設定している場合に，内部転送機能を使用する．

XIS は，端末を以下のような二つの種類に分けて管理している．

1) 問い合わせ応答型端末……端末から入力されたメッセージに対し，入力端末への応答メッセージが出力されるような，一連の問い合わせ応答処理を行う端末である．
2) 交換端末……応答メッセージを必要としない一方的な入力が行われる端末，または入力メッセージが存在しない端末へホストより一方的な出力が行われる端末である．

4.3.1 問い合わせ応答処理における内部転送処理

問い合わせ応答型端末からの要求メッセージは，端末とのシステム・セッションが設定されているホストにおいて処理される．したがって，通常，応答メッセージ送信処理は，端末とシステム・セッションが設定されているホストにおいて行われるため，内部転送処理は行われない．

しかし，端末からの要求メッセージがホストに入力された後，何らかの障害によりシステム・セッションが切断され，別ホストと該当端末との間に新たにシステム・セッションが設定された場合には，応答メッセージ送信処理要求ホストと，端末とシステム・セッションが設定されているホストが異なることになる．このような場合に，XIS は内部転送機能により処理要求メッセージを転送し，端末とのシステム・セッシ

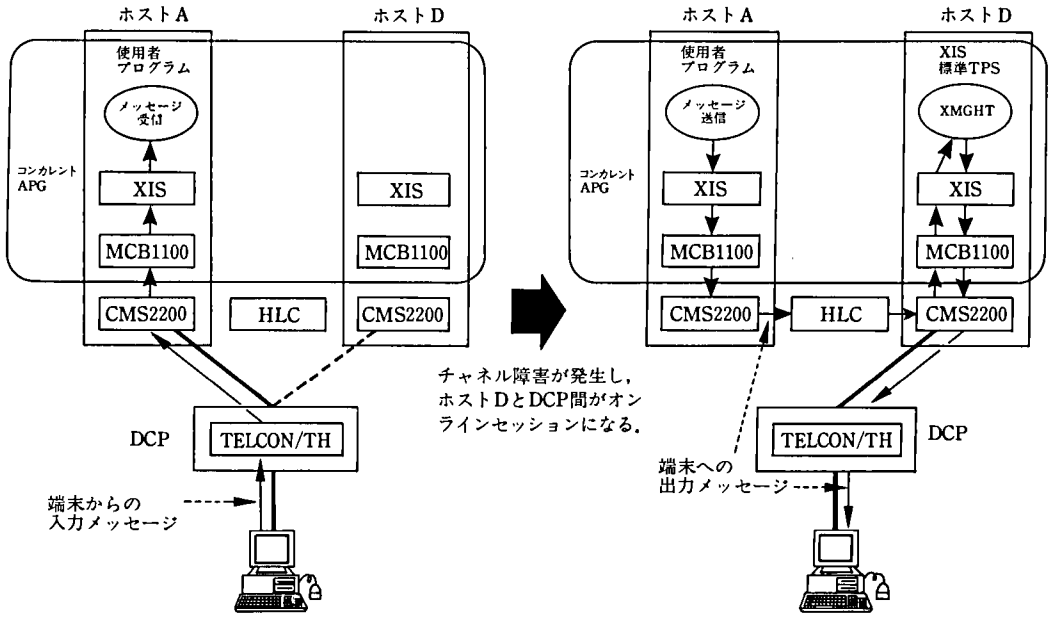


図 5　問い合わせ応答処理における内部転送処理

ョンが設定されているホストにおいて応答メッセージを送信する（図5）．

端末とシステム・セッションが設定されているホストでは，XIS 標準 TPS である 'XMGHT' が，転送されたメッセージを受信し，端末へメッセージを送信する．

4.3.2　交換処理における内部転送処理

交換型端末への出力メッセージの送信は，問い合わせ応答型端末への応答メッセージの送信とは異なり，端末とのシステム・セッションを設定していないホストにおいて一方的な端末へのメッセージ送信要求が発生するため，頻繁に内部転送処理が行われる可能性がある．

また，交換型端末への出力メッセージは，複数のプログラムから並列的に一つの端末にメッセージ送信要求が行われることがある．交換メッセージは連続帳表出力などに使用される場合があり，メッセージ送信要求順序で端末へメッセージを送信することが重要となる．XIS は，XTPA システムを構成する複数ホストの使用者プログラムから同時期に一つの端末へ出力メッセージ送信要求が行われた場合に，送信要求された順序で端末へメッセージを送信するために，メッセージ保存ファイル(MRF:Message Retention File)を用いて制御を行っている．

使用者プログラムからメッセージ送信要求が行われたホストでは，メッセージ保存ファイルへのメッセージ保存のみ行い，メッセージ送信処理ホストで転送されたメッセージを受信した XIS 標準 TPS である 'XISSD' が，流量制御などの XIS 内部処理を行うとともに，メッセージ実送信処理を行う（図6）．

4.4　システム・セッション制御における内部転送処理

端末とホストの間での通信に使用されるシステム・セッションは，通信に先だって必ず設定処理を行う必要がある．

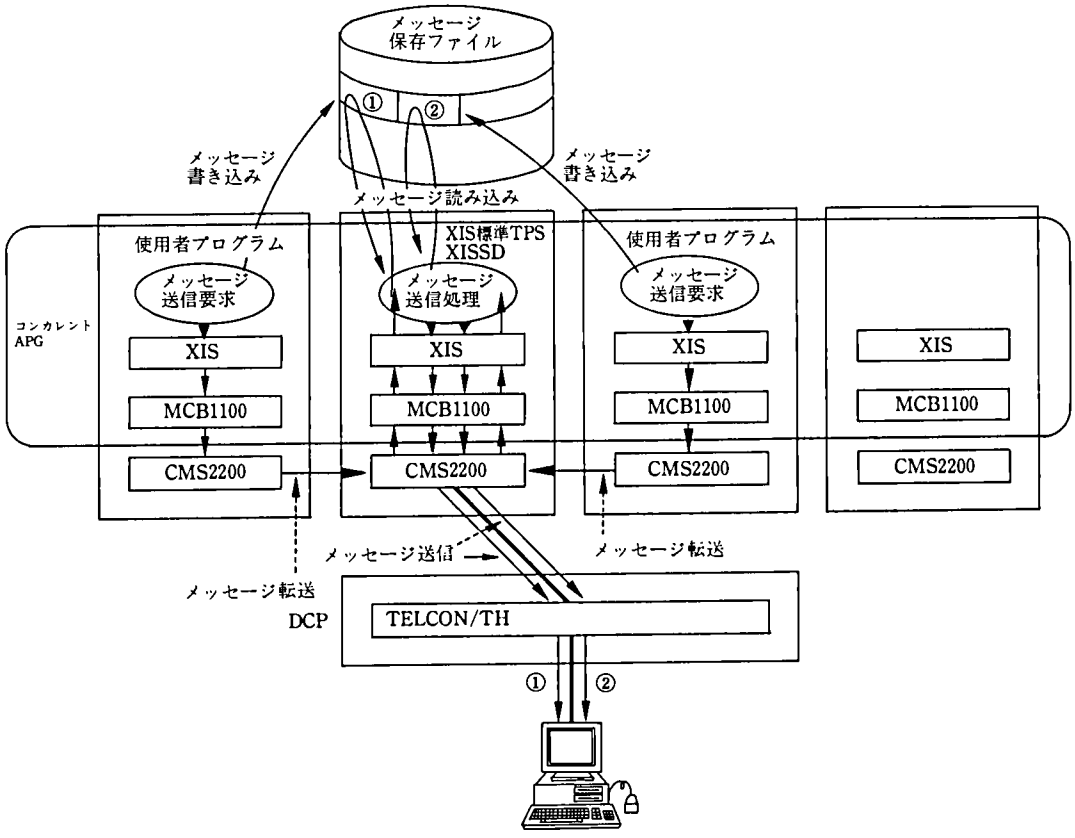


図 6　交換処理における内部転送処理

システム・セッションの設定は，TELCON/TH からインバウンド・オープンにて設定する方法と，ホストからアウトバウンド・オープンで設定する方法がある．

XIS は，ホストからアウトバウンド・オープンにより端末とのシステム・セッションを設定するコマンド・インタフェースを提供する．使用者がシステム・セッション設定のコマンド要求を行うと，XIS は CMS 2200/MCB 1100 に対してアウトバウンド・オープン要求を行う．システム・セッションはアウトバウンド・オープン要求が行われたホストの CMS 2200 と TELCON/TH との間に設定され，以降の端末との通信は，システム・セッションが設定されているホストで行われる．そのため，どのホストでアウトバウンド・オープン要求を行うかが重要となる．

XIS は，XIS が管理しているセッション管理テーブルの情報により，アウトバウンド・オープンを実行するホストを特定する．XIS は，使用者が XIS にシステム・セッション設定要求コマンドを行ったホストと，XIS がアウトバウンド・オープンを実行するホストが異なる場合に，内部転送機能を使用して処理を転送する．

XIS は，XIS が管理しているセッション管理テーブルの情報を変更するコマンド・インタフェース/プログラム・インタフェースを提供する．使用者は，これらのインタフェースを使用して端末とのシステム・セッションを設定するホストを変更することにより，システム・セッションを設定するホストを動的に変更することが可能である．

4.5 ネットワーク制御コマンドにおける内部転送処理

ネットワーク制御コマンドには，DCP/TELCON に制御を要求する NMS コマンドと CMS 2200 に制御を要求する CMS 2200 コマンドがある．XIS は，これらのコマンドを XIS のコマンド・インタフェースを使用することにより実行できる機能を提供する．

ホストから DCP/TELCON に対し NMS コマンドにより制御を要求する場合，ホストと DCP の間に制御の通信を行うための AMS セッションが必要である．XIS は，XTPA システムを構成する全てのホストと DCP の間に，AMS セッションを設定することを前提としている．そのため，XTPA システムを構成する全てのホストから，DCP/TELCON に NMS コマンドにより制御を要求することが可能である．

CMS 2200 は，XTPA を構成する各ホストにおいて独立して稼働しているため，CMS 2200 コマンドを実行する場合は，どのホスト上で稼働している CMS 2200 に対してコマンド要求を行うかを意識する必要がある．XIS は，使用者から CMS 2200 コマンドを実行するよう要求された際に，コマンド・パラメタおよびコマンド要求対象となるネットワーク構成要素から，処理実行ホストを特定する．使用者からのコマンド要求が行われたホストと，コマンド処理を実行するホストが異なる場合は，XIS が内部転送機能を使用し処理を転送する．

5. メッセージ保存ファイルのホスト割り付け機能

XTPA システムでは，XTPA を構成する全ホストで共有する必要のある資源がある．

XIS が管理しているメッセージ保存ファイルは，XTPA システムを構成する全ホストで共有し，アクセスするファイルである．ホストや回線などのネットワーク構成要素に障害が発生した場合，ホストから端末へ送信されたメッセージが消失する可能性がある．そこで XIS は，独自のメッセージ保存ファイルを管理し，回復メッセージの送信要求を使用者プログラムが行った際に，メッセージ保存ファイルに保存している送信メッセージを使用する．データベース等のシステム資源を更新したプログラムからの送信メッセージを非回復メッセージとすると，送信メッセージがメッセージ保存ファイルに保存されないため，障害発生時に送信メッセージの回復処理が不可能となる．そこで，データベース等のシステム資源を更新したプログラムからの端末への送信メッセージは，通常回復メッセージとして送信される．端末へのメッセージ送信要求は，XTPA システムを構成する各ホストにおいて，非同期に発生することが考えられる．そこで，メッセージ保存ファイルへのアクセス時の処理効率が問題となる．

XIS 通信制御機能は，XTPA システム環境において，メッセージ保存ファイル使用時の処理効率の低下を防止するため，各ホストが独立してメッセージ保存ファイルを管理する並列制御を行っている．

5.1 交換メッセージの保存

5.1.1 メッセージ保存ファイル

XIS は，障害発生時などに端末への送信メッセージの回復を行うために，メッセージ保存ファイルに端末への送信メッセージを保存する機能を提供する．

XIS は，メッセージ保存ファイルをセグメント，MRF ブロックという単位で分割し，管理している（図 7）．

① セグメント…メッセージを保存するために，各端末へ割り当てる単位

② MRF ブロック…複数セグメントをまとめた単位

XIS は，一つのメッセージ保存ファイルに複数の端末への送信メッセージを保存する．

問い合わせ応答型端末は，最後に端末へ送信された応答メッセージを保存することにより，端末とホスト間のメッセージ回復処理が行える．そのため，各々の端末に，応答メッセージ保存のために，一意に 1 セグメントを割り当てている．

交換型端末へのメッセージ送信処理において，XIS は独自に流量制御を行いながら，同時に一つの端末に複数のメッセージを送信する．そこで，障害発生時に複数のメッセージの回復処理を行う必要がある．つまり，交換型端末では，メッセージ回復処理のために複数のメッセージを保存しておく必要がある．また，ある交換型端末への送信メッセージ要求を行うトランザクション量は，時期により発生頻度が異なるため，メッセージ回復に必要なセグメントを各交換型端末に一意に割り当てると，メッセージ保存ファイルの容量が増大する．そこで XIS は，各々の端末への送信メッセージをメッセージ保存ファイルのどのセグメントに保存するかを一意に決定しておかず，使用者プログラムが XIS にメッセージ送信要求を行った際に，未使用のセグメントを獲得し，送信メッセージを保存する．

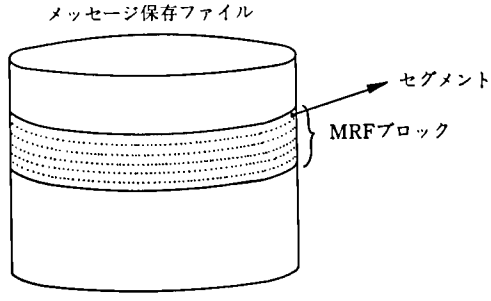


図 7　メッセージ保存ファイルの構造

5.1.2　セグメントの獲得

使用者プログラムが交換型端末へメッセージ送信要求を行った時，XIS は，メッセージ保存ファイルへ送信メッセージを保存するために，メッセージ保存ファイルの未使用セグメントを獲得する．

交換型端末へのメッセージ送信要求は，端末からの入力がなければ発生しない応答メッセージ送信要求とは異なり，端末に対するメッセージ送信が複数プログラムにより並列的に発生する．メッセージ保存ファイルは複数の端末で共有するため，未使用セグメント獲得時の排他制御による効率が重視される．XIS は，メッセージ保存ファイルのセグメント使用状況をホストのメモリ上に配置された 1 セグメントを 1 ビットに対応させたビットマップで管理することにより，セグメント獲得処理の効率化を図っている．

5.2 XTPA システム環境でのメッセージ保存ファイルの使用方法

5.2.1 メッセージ保存ファイルのホスト割り付け

XTPA システム環境では，XTPA を構成する全てのホストからの交換メッセージ送信を可能としなければならない．そのため，全てのホストからセグメント獲得処理が行えることが必要である．

前述したように，XIS はホストのメモリ上でメッセージ保存ファイルのセグメント管理を行っているが，ホストのメモリは XTPA システムを構成している各ホスト独自の資源であるため，他のホストのメモリ上のビットマップを使用することはできない．つまり，XTPA システム環境では，メッセージ保存ファイルの全てのセグメントの使用状況をホストのメモリ上に存在する一つのビットマップで管理すると，メモリ上にビットマップが存在するホストではセグメント獲得処理を行えるが，他のホストではメモリ上にビットマップが存在しないためセグメント獲得処理が行えない．

XTPA システムを構成する全てのホストからの，メッセージ保存ファイルのセグメント獲得を可能とする実現方法として，以下の方法が挙げられる．

1) セグメント管理を行うビットマップをホストのメモリ上ではなく，ファイルとして管理し，全ホストで共有する．

2) 一つのメッセージ保存ファイルを使用するホストを，1 ホストに限定する．つまり，一つの端末への送信メッセージを，複数のメッセージ保存ファイルに保存する．

3) メッセージ保存ファイルを複数のブロックに分割し，一つのブロックを使用するホストを 1 ホストに限定する．

1)案では，メッセージ保存ファイルの全てのセグメントを，XTPA システムを構成する全ホストで管理できるが，メモリ上のビットマップを使用しないためセグメント獲得処理の効率が低下する．

2)案では，セグメント獲得処理の効率化は図れる．しかし，交換型端末へのメッセージ送信処理は，XTPA システムを構成する全ホストで均等に発生するとは限らないため，各ホストで使用するメッセージ保存ファイルは，送信メッセージ回復に必要な全メッセージが保存可能な容量を確保しておく必要がある．1)案，3)案に比べ，全メッセージ保存ファイルの総容量は，飛躍的に増大する．

3)案では，セグメントの管理に加えて，メッセージ保存ファイルを一定単位に分割したブロックの管理を行わなければならないが，1)案，2)案で問題となる効率的な問題，全ファイル容量の問題は解決される．

そこで XIS は，XTPA システム環境において 3)案を採用し，メッセージ保存ファイルのセグメント管理を実現している．XIS は，メッセージ保存ファイルを複数のセグメントをひとまとめとした MRF ブロックとして管理し，あらかじめ MRF ブロックごとに，XTPA を構成する各ホストに割り当てておく方法をとっている．XIS は，XTPA を構成する各ホストに割り当てられている MRF ブロック中のセグメントの使用状況をホストのメモリ上のビットマップで管理する．この制御により，使用者プログラムは，XTPA システムを構成する全てのホスト上で，並列的かつ効率的に交換型端末へのメッセージ送信処理が行える(図 8)．

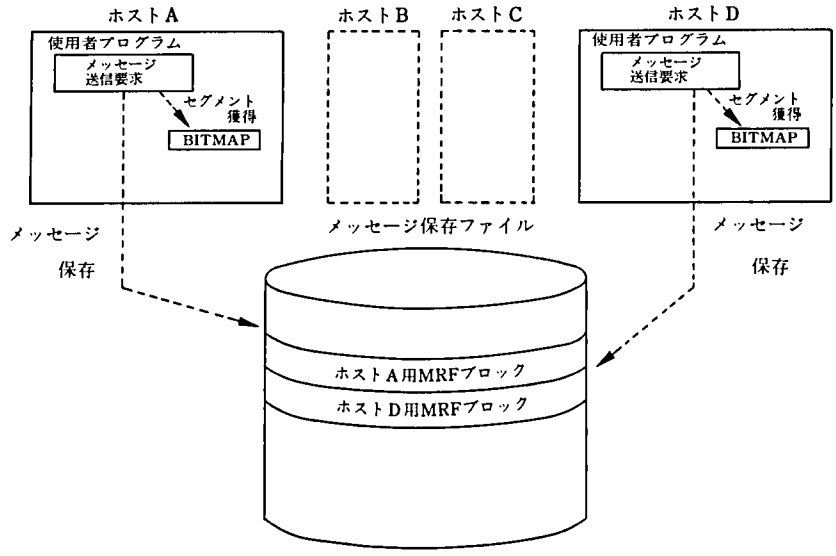


図 8　XTPA システムでのメッセージ保存

5.2.2　XTPA システム環境でのセグメント管理

前述したように XTPA システム環境では，XIS はホストのメモリ上に，割り付けられている MRF ブロックのビットマップを構築し，管理している．XIS は，ホストに割り付けられている MRF ブロックの全セグメントが使用された場合，他ホストに割り付けられていない MRF ブロックを新たに割り付ける．しかし，この方法では，XIS が新たな MRF ブロックをホストに割り付ける処理中に，使用者プログラムよりメッセージ送信要求が行われた場合，新たな MRF ブロックのビットマップがメモリ上に存在しないため，セグメント獲得処理の効率が低下する．

そこで XIS は，XTPA を構成する各ホストにメッセージ保存ファイルから二つの MRF ブロックを割り付け，それぞれの MRF ブロックのビットマップをメモリ上に展開し管理する．XIS は二つの MRF ブロックのうち，一つを使用する．もう一つの MRF ブロックは，使用中の MRF ブロックの全てのセグメントを使用した場合の予備として管理している．予備の MRF ブロックを管理することにより，使用中の MRF ブロックの全セグメントが使用された場合の処理効率の低下は防止される．

XIS は二つの MRF ブロックのうち，どちらの MRF ブロックを使用しているかを管理し，使用中の MRF ブロックの全てのセグメントが使用中となると，使用 MRF ブロックを変更し，予備の MRF ブロックを使用する．この時 XIS は，全てのセグメントを使用した MRF ブロックに替わり，新たな MRF ブロックをホストに割り付けるために，XIS 標準 TPS である 'XMGRV' を起動する．XIS は，この新たな MRF ブロック割り付け処理を XIS 標準 TPS が行うことにより，使用者プログラムの負荷軽減を図っている(図 9)．

6.　お　わ　り　に

本稿では，XTPA システムにおけるホスト透過，並列制御，無停止運転を支援する

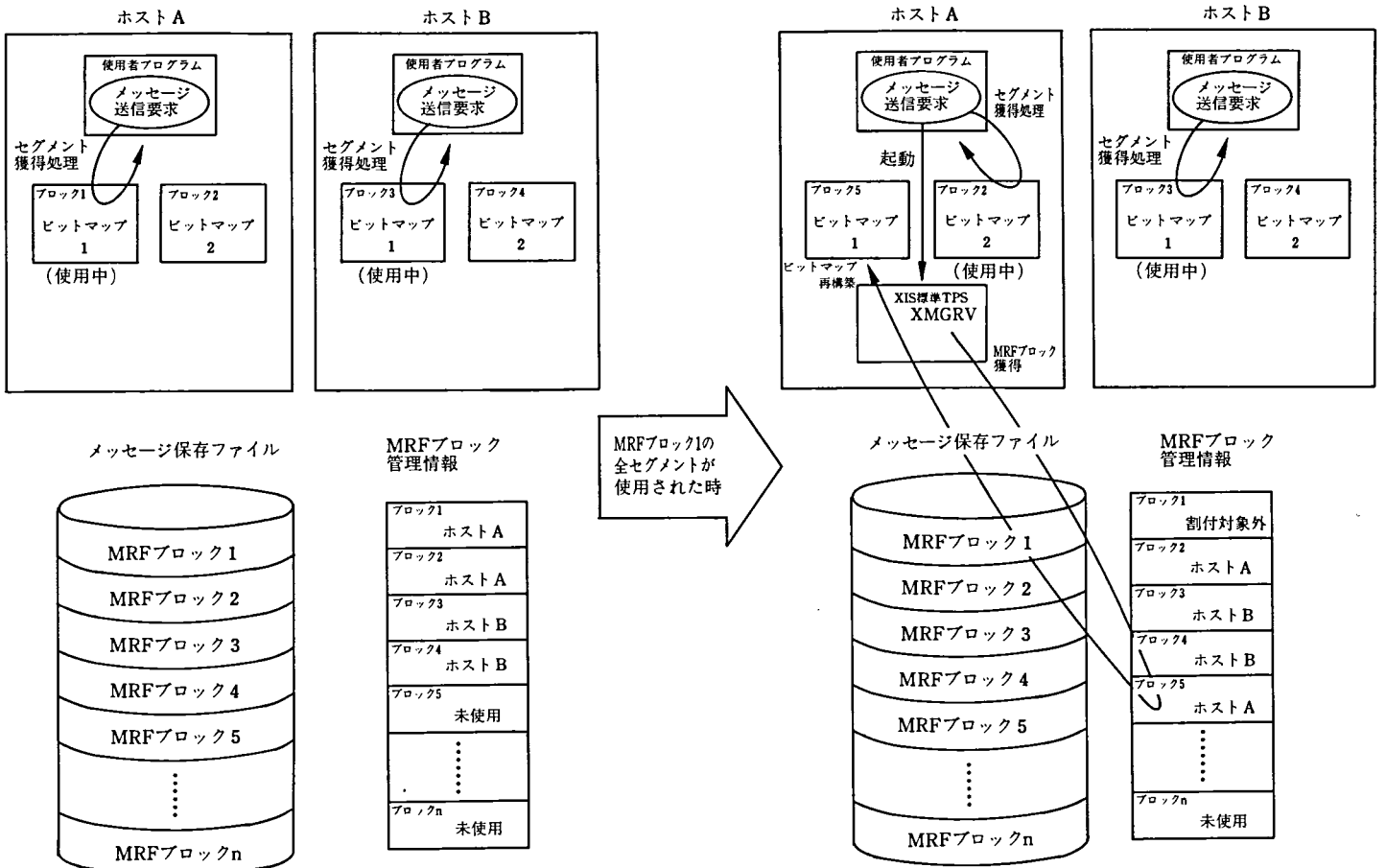


図 9　XTPA システムでのセグメント管理

XIS データ通信制御機能の各機能について記述してきたが，XIS データ通信制御の当初の機能目標については達成できた．

今後の課題として，各機能の効率改善が挙げられる．XIS データ通信制御機能では，拡張データ処理装置 XPC(eXtended Processing Complex)などのハードウェアや OS 2200 などの基本ソフトウェアの新機能に期待しつつ，順次対応し効率改善を行っていく．

また，デュアル・セッション機能により，ホスト障害時にも端末業務に支障をきたさないことが可能であるものの，DCP/TELCON 障害時には端末業務は一時的に停止してしまう．このため，DCP の次機種として出荷予定されている DCP 600/FTX, ILM 40 にも XIS は対応し，さらに信頼性の向上を目指していく．

参考文献
［1］ J. N. Gray 他編，渡辺榮一訳，"OLTP システム"，マグロウヒル出版，1991.
［2］ 沢田啓，"XTPA に基づくノーダウンシステム"，技報通巻 40，日本ユニシス(株)，1994.2.
［3］ 宮村洋，"銀行システムと XIS"，技報通巻 40，日本ユニシス(株)，1994.2.

執筆者紹介
石 川 清 人 (Kiyohito Ishikawa)
1988 年横浜市立大学商学部卒業．同年日本ユニシス(株)入社，統合オンライン・ソフトウェア AIS 1100 II および AIS 1100 II ファミリ・ソフトウェアの開発，サービスに従事．現在システム企画開発 2 部オンライン技術課に所属．

XISの分散トランザクション処理

Distributed Transaction Processing under XIS

栗原幸代

要約 分散処理システム化の進行に伴い，日本ユニシスのユーザにおいても，集中型で構築されている既存ホストシステムを効果的に生かし，分散システムを構築するライトサイジングが求められている．

本稿は，一般的な分散トランザクション処理の方式と要件について記述し，さらに，統合オンライン支援システム—XIS(eXtended Information System)—が提供する分散トランザクション処理制御(ACLES/DTP: Advanced Communication eLEments and Support system/Distributed Transaction Processing)の機能と実現方式について記述する．

ACLES/DTPは，

・分散した使用者プログラム間で複数回データ転送を行う「会話処理」を可能にし，

・その一連の処理がトランザクションとして完結できるよう，プログラム間の「同期処理」を実現する．

「会話処理」では，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にはない「データ送受信中のロックの継続(同一ステップの継続)」，同一データバンクの継続使用を可能とする．

また，「同期処理」では，各使用者プログラムのステップ更新処理を2フェーズ・コミットメントにより同期を取って行う．また，障害が発生した場合でも，整合性がとれるよう回復処理を行う機能を提供する．

Abstract With the wider user acceptance of distributed transaction processing (DTP), Nihon Unisys users are not exceptional in going for systems lightsizing in an attempt to build distributed systems by making effective use of their existing centralized mainframes. Besides referring to the general method and requirements of DTP, this paper describes the functions of the Advanced Communication eLEments and Support system (ACLES/DTP) provided by the integrated on-line support system——XIS (eXtended Information System)—— as well as the way of implementing systems using it.

ACLES/DTP makes possible

・"conversational processing" whereby data are repeatedly transferred among and between distributed user application programs (UAPs), and

・cross-program "synchronous processing" whereby that series of processing processes can be finalized as a transaction.

"Conversational processing" allows "continued locking during send-receive processing (for iteration of the same transaction steps)" and continued use of the same data bank, which have not been provided by the conventional inquiry/answer type or switch type of transaction processing. "Synchronous processing" helps synchronize the step renewal of UAPs on a two-phase-commitment basis, and also provides a function for synchronized recovery processing even in the case of a system failure.

1. はじめに

分散処理システムは確実に普及しはじめており，業務単位/組織ごとの情報処理システムの構築が浸透している．当社ユーザにおいても，集中型で構築されている既存ホストシステムを効果的に生かし，分散システムを構築するライトサイジングが求められている．

分散システムにおいては，処理対象となるデータがコンピュータ・ネットワークに分散しており，このような環境下でオンライン・トランザクション処理を実現するためには，分散した複数のプログラムにより処理された一連の結果が一つのトランザクションとして意味を持つような仕組みが必要となる．このようなトランザクションを「分散トランザクション」と呼ぶ．

統合オンライン支援システムXISは，増え続ける分散システムに対応するために，XISファミリとして分散処理支援ソフトウェア群を提供している（図1）．

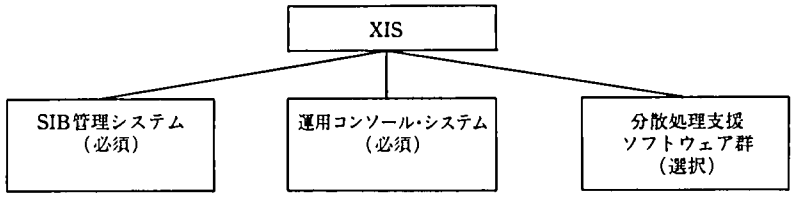


図1　XISファミリ・ソフトウェア・プロダクト

XISファミリの分散処理支援ソフトウェア群のうち，「分散トランザクション」を実現するサブシステムとして開発されたのがACLES/DTP（分散トランザクション処理制御）である（図2）．

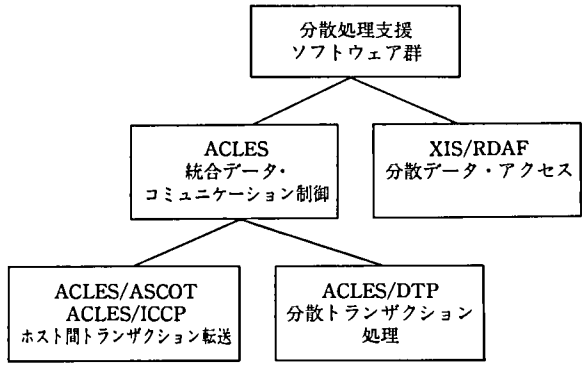


図2　分散処理支援ソフトウェア・プロダクト

本稿では，分散トランザクション処理の方式と要件，ACLES/DTPの機能および実現方式を報告する．

1.1 ACLES/DTP開発の背景

トランザクションの大規模化，業務処理の複雑化に対応するため，より高い処理能力，信頼性を求め，複数のホストに業務処理を分散したマルチホストシステムは従来から数多く存在している．各種端末との接続，対外システムとの接続を担当するフロ

ントエンドシステムと実業務処理を行う業務ホストに処理分担する形態が代表的なマルチホストシステム形態である．このような形態では，大量高速処理のための同時並行処理機能，メッセージリカバリ機能が要求される．AIS/XIS では，当初より該形態を支援するホスト間接続プロダクト(ASCOT 1100, ACLES/ASCOT など)を提供している．

一方，複数の業務システム間あるいは UNIX* システムとホストシステム間など複数のシステムにまたがってデータベース更新を伴う処理が必要となるより複雑な分散処理システムでは，該トランザクションを処理する複数のプログラム間の同期が必要となる．このように分散処理システムの形態が複雑化してきており，従来型の分散システム対応のシステム形態だけでは対応できなくなっている．ACLES/DTP は，分散トランザクションとして処理を完結するため必要となる機能，すなわち，プログラム間の同期処理，障害時の回復処理機能を提供し，トランザクションの特性を保証できる高信頼性システムを実現する．

1.2 プロダクトの目標

ACLES/DTP は，

- 分散した使用者プログラム間で複数回データ転送を行う「会話処理」を可能にし，
- その一連の処理がトランザクションとして完結できるよう，プログラム間の「同期処理」を実現する．

「会話処理」では，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にはない「データ送受信中のロックの継続 (同一ステップの継続)」，「同一データバンクの継続使用」を可能とする．

また，「同期処理」では，各使用者プログラムのステップ更新処理を 2 フェーズ・コミットメント(2.2.2 項の“複数プログラム間の更新同期機能”参照)により同期を取って行う．また，障害が発生した場合でも，整合性がとれるよう回復処理を行う機能を提供する．

ACLES/DTP の機能目標は以下の通りである．

- 通信プロトコルに依存しない独立した統合使用者インタフェース
- アプリケーション・プログラム間の容易な会話処理
- 効率的なデータ転送
- アプリケーション・プログラム間の同期処理
- 障害時の論理通信路，ステップ** 自動リカバリ処理
- 物理的な通信路から独立した会話宛先の仮想化

2. 分散トランザクション処理の概要

本章では，一般的な分散トランザクション処理の実現方式と分散トランザクション

---

* 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．

** ステップとは，XIS の定義するリカバリの単位である．
XIS のステップ管理では，メッセージおよびデータベースの完全な保全をめざすため，使用者プログラムのデータ処理をいくつかのリカバリ単位に区切り，その間の種々の処理要求をリカバリ単位の終了時に一括して実行する遅延処理方式をとることにより，そこでの障害波及範囲の局所化を図っている．

処理の要件について記述する．

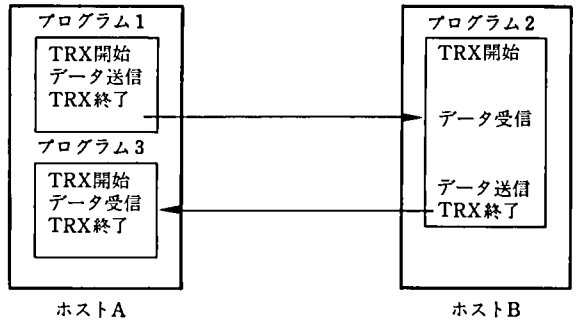
2.1 分散システムにおけるトランザクション処理方式

分散システムにおけるトランザクション処理の実現方式は複数あげられるが，ACLES/DTPのトランザクション処理方式との比較の観点からその一部を記述する．

2.1.1 基本トランザクション転送とACLES/DTPの処理方式

XISは，分散処理支援機能として，ACLES/ASCOTの基本トランザクション転送機能とACLES/DTPの分散トランザクション処理制御機能の二つの形態を提供する．

1) 基本トランザクション転送処理形態……基本トランザクション転送機能は，問い合わせ応答型，交換型などのトランザクション処理形態であり，一つのステップ（プログラム）内でのみトランザクションが完結する形態である（ローカルトランザクション）．したがって，複数のプログラムで一つのトランザクションを処理する場合，使用者がその整合性を保証する必要がある(図3)．障害時は，メッセージの再送による処理のロールフォワード，あるいはアプリケーションプログラムレベルでの処理取り消しを行う必要がある．

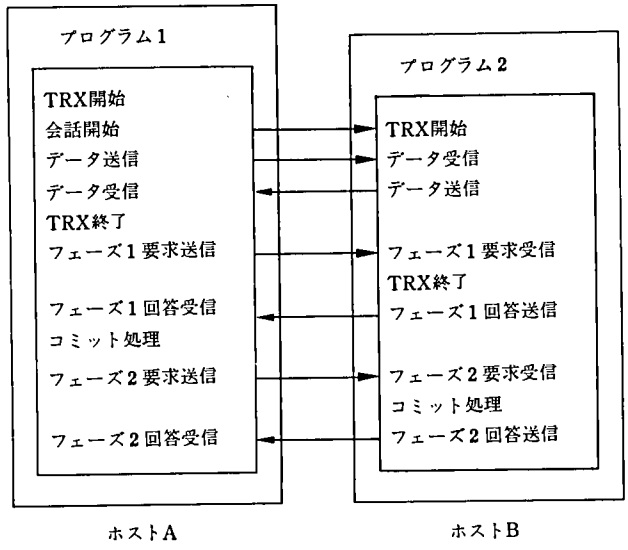


・一つのプログラムは，一つのデータのみ受信する．
・データの送信は，トランザクション終了（TRX終了）時に行われる（遅延送信）．

図3 基本トランザクション転送方式

2) 分散トランザクション処理形態……分散トランザクション処理制御機能は，ネットワークに分散している複数の処理プログラムが相互通信し，複数プログラム間で同期を取り一つの処理として意味を持つ単位にする処理方式である（グローバルトランザクション）．メッセージの送受信はPeer to Peer型で行われ，同期を取った処理が可能である(図4)．障害時は，分散トランザクション処理制御機能がトランザクションのロールバック/ロールフォワードを自動的に行う．ただし，複数のプログラム間で協調して更新処理を行うための制御メッセージの送受信を行うので，基本トランザクション処理形態に比べ効率は落ちる．

3) 基本トランザクション転送形態による疑似会話処理……基本トランザクション転送形態では，一つのプログラムは複数のメッセージを受信することができないため，プログラム間で複数回データの送受信をする場合，三つ以上の複数プログラム間で送受信する必要がある．そのため，「データ送受信中のロックの継続(同



・一つのプログラムは，複数のデータを受信できる．
・データの送信は，即時に行われる．

図 4　分散トランザクション処理方式

一ステップの継続)」,「同一データ・バンクの継続使用」ができないため，使用者が考慮する必要がある．

基本トランザクション転送形態による疑似会話処理例を示す（図5）．

4) ACLES/DTP による会話処理……ACLES/DTP は，データの引き継ぎのような使用者の負荷を軽減するため，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．使用者プログラム間で複数回データ転送を行うことを「会話処理」と呼ぶ．二つの使用者プログラムが「会話処理」を行いながら協調して処理を行うため，両者間に成立する関係を「会話」と呼ぶ．「会話」の確立を起動した使用者プログラムを「ルート」,「会話」の確立を受動した使用者プログラムを「リーフ」と呼ぶ．また，このような「会話処理」を行う使用者プログラムを，DTP－AP (Distributed Transaction Processing－APplication) と呼ぶ．

ACLES/DTP による「会話処理」例を示す（図6）．

2.1.2 Peer to Peer 型処理とクライアント・サーバ型処理方式

プログラム間会話処理には，Peer to Peer 型とクライアント・サーバ型がある．

Peer to Peer 型は OSI トランザクション処理 (OSI—TP)，SNA LU 6.2 などで採用される方式であり，ACLES/DTP も該方式を提供している．Peer to Peer 型は，プログラム対等通信方式でありリカバリが複雑になるが，プログラム処理の自由度が高い．一方，使用者プログラムの設計がクライアント・サーバ型に比べ複雑になる(図7)．

クライアント・サーバ型は，使用者プログラムを，処理を要求する側（クライアン

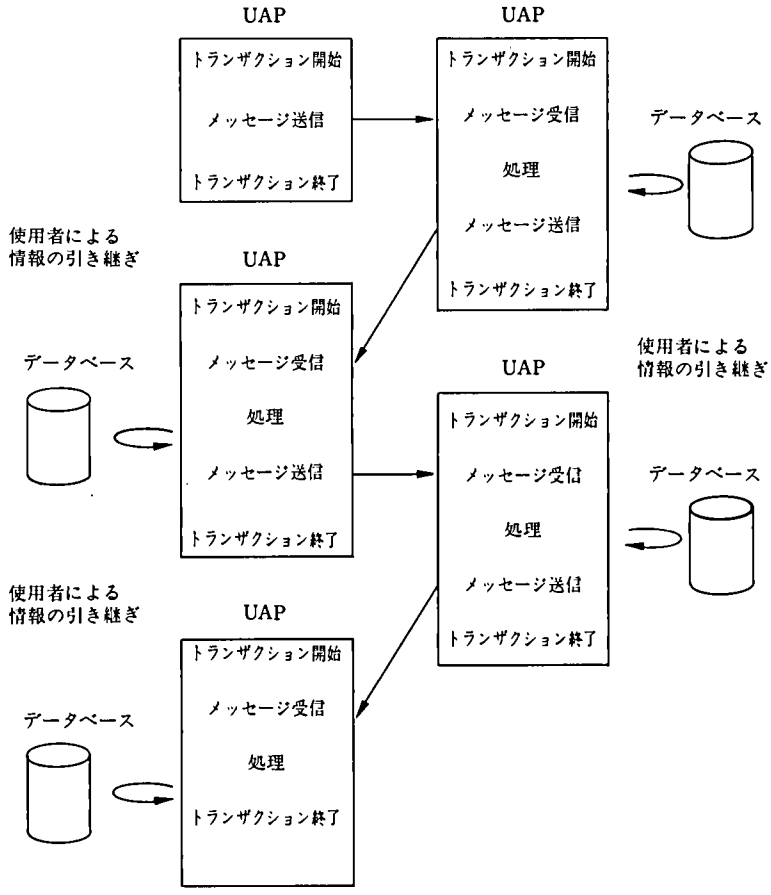


図 5　基本トランザクション転送形態による疑似会話処理

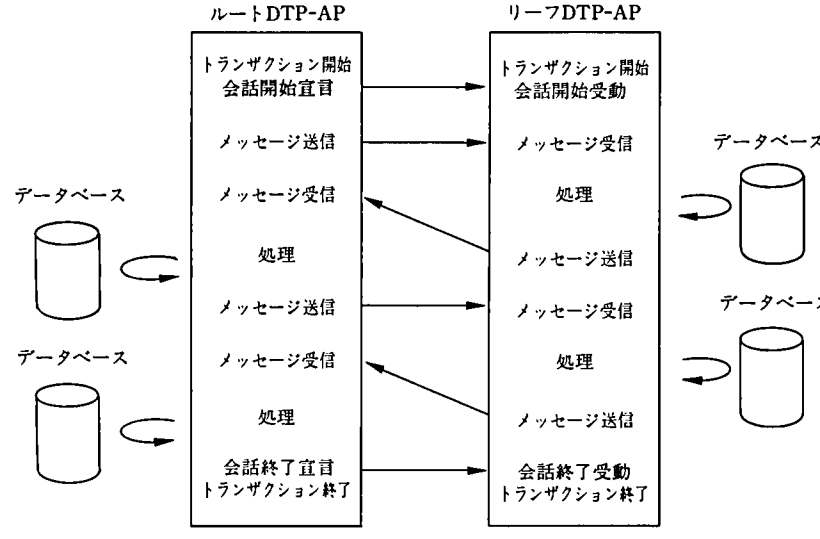


図 6　ACLES/DTP による会話処理

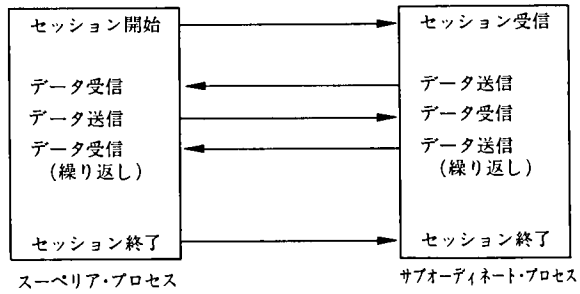


図 7　Peer to Peer 型処理方式

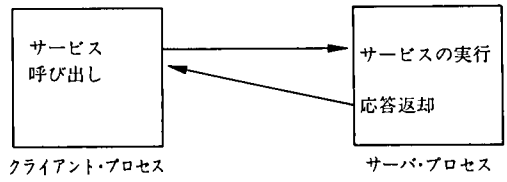


図 8　クライアント・サーバ型処理方式

ト）と，これを実際に処理する側（サーバ）とに分割した処理形態である．処理はサーバが実行するサービスが行う．クライアントはサービスを要求し，サーバがサービスを実行し応答を返す．

クライアント・サーバ型は，アプリケーションをモジュール化できるため，使用者プログラムの設計が容易になり，システムの拡張も容易になる（図 8）．

2.2　分散トランザクション処理の要件

トランザクションは次の特性（ACID 特性）を持つと定義づけられる．

- 原始性(Atomicity)：全体として成功するか失敗するかいずれかの，それ以上分解することのできない単位である．
- 一貫性(Consistency)：処理として完結するとその内容は変更されることはない．
- 独立性(Isolation)　：他のトランザクションから影響を受けない．
- 耐久性(Durability)：障害が発生しても完了した内容は消滅しない．

トランザクション処理の一つの形態である分散トランザクション処理の ACID 特性を保証するための，主な機能は次の通りである．

- 複数プログラム間通信機能
- 複数プログラム間の更新同期機能
- 排他制御機能
- 障害に対する回復機能

2.2.1　複数プログラム間通信機能

分散したシステム間のトランザクション処理を実現するためには，それぞれのシステムに配置されている複数のプログラム間の通信機能が必要である．複数プログラム間通信機能の要件は次の通りである（図 9）．

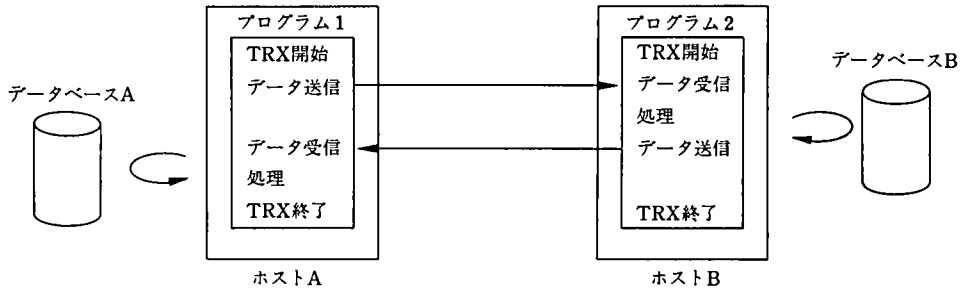


図 9　複数プログラム間通信機能

・会話相手プログラムの特定と起動
・会話を特定した会話メッセージの送受信
・プログラム障害を他の会話相手に通知する

2.2.2　複数プログラム間の更新同期機能

複数プログラム間の更新同期機能について前提であるトランザクションの保証範囲とトランザクション識別子を含めて説明する．

1)　トランザクションの保証範囲……一般的にトランザクション処理システムは，トランザクション処理プログラムが「トランザクション開始」を宣言してから「トランザクション終了」を宣言するまでの処理をそれ以上分割できない一つの単位(ACID特性の「原始性（A)」）として扱える機構を持つ．通常，データベースの更新処理，回復型メッセージの送信を，「トランザクション終了」まで遅延することによって原始性を保証する．

トランザクション処理が正常に終了することを，トランザクションが「コミット」するという．一度トランザクションがコミットすると，トランザクションがアクセスしたレコードのロックは解除される．更新されたレコードは，その情報を保存するログ（audit trail）管理により「耐久性（D)」が保証される．

2)　トランザクション識別子……「トランザクション開始」により，固有の「トランザクション識別子」が割り当てられる．このトランザクション識別子に基づき，更新同期処理，ログ管理（audit trail)，また後述の排他制御処理，回復処理が行われる．

ACLES/DTPではこのトランザクション識別子をステップIDと呼ぶ(図10)．

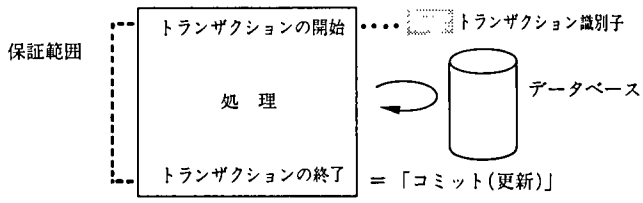


図 10　トランザクションの保証範囲と識別子

3)　複数プログラム間の更新同期処理……分散した複数のプログラムにより処理された一連の結果がトランザクションとして意味を持つためには，複数のプログラ

ム間の更新処理の同期が取られなければならない．

たとえば，図9の処理でデータベースAとデータベースBは「A，Bともに更新される」か「A，Bともに更新されない」場合は，更新処理の同期が取られている．しかし，「一方が更新されて，もう一方が更新されない」場合，ACID特性の「一貫性（C)」が保証されていない．

こうした不正が生じないために，一般的に採用されている処理方式が2フェーズ・コミットメント方式である．

2フェーズ・コミットメント処理方式について，ACLES/DTPの構成要素であるルート（会話の起動側)/リーフ（会話の受動側）を使用して説明する（図11)．

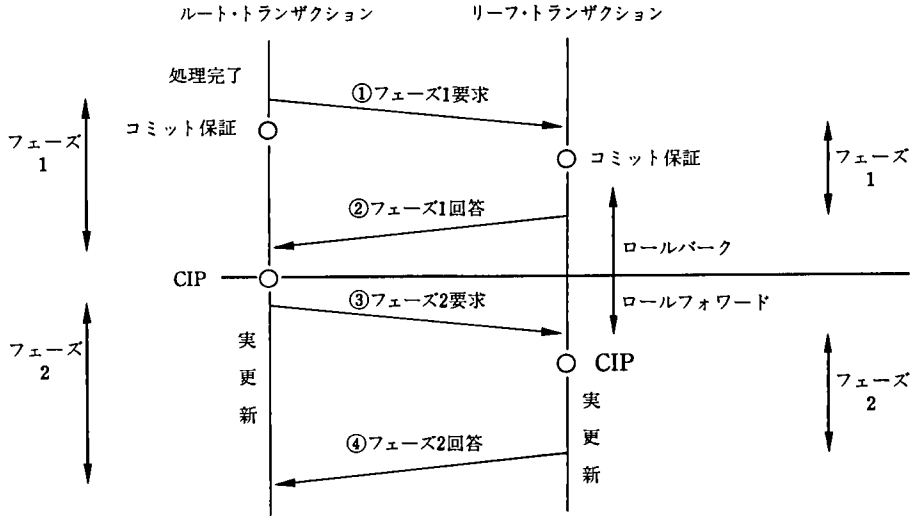


図 11　2フェーズ・コミットメント

「トランザクションの終了」が要求されると更新処理を

・コミット保証（フェーズ1）

・実更新　　（フェーズ2）

に分割し，各フェーズごとにルートからリーフへ問い合わせを行い両者の同期を制御する．これを2PC（2フェーズ・コミットメント処理方式）と呼ぶ．

・フェーズ1：① ルート主導でリーフにコミット保証の問い合わせーコミット指示を行う（フェーズ1要求)．

② リーフはこれを受信しコミット保証の応答を返す（フェーズ1回答)．

・フェーズ2：③ ルートはコミット保証応答を受信しコミット宣言（CIP要求）を行い，リーフにコミット要求を送信し実更新する（フェーズ2要求)．

④ リーフはこれを受信し実更新後コミット回答を返す（フェーズ2回答)．

ルート・トランザクションのCIP要求が行われた時点で，リーフ・トランザクションも含め，処理が完了したと判断し，障害が発生した場合でもすべての処理

を完結する（ロールフォワード）．ルート・トランザクションのコミット宣言が行われていない場合は，ルート/リーフ共にロールバック処理する．

2.2.3 排他制御機能

一般的にある任意の時点で，システムには複数の端末装置から入力データが到着し，複数のトランザクション処理が並行に進められる．そこで，複数のトランザクションが同一のレコードをアクセスすることを避けるために，アクセスに先だってレコードがロックされる．これを排他制御機能と呼び，ACID属性の「独立性 (I)」が保証される．

2.2.4 障害に対する回復機能

トランザクション処理中に障害が発生した場合，ログ管理(audit trail)により保存された「前イメージ/後イメージ」によりデータベースの回復処理を行う．

分散トランザクション処理中，障害が発生した場合は回復処理（リカバリ処理）を行う必要がある．

前述したように，トランザクションは遅延更新されるため，「トランザクション終了」要求以降の2フェーズ・コミットメント処理中の障害が回復処理の対象となる．「トランザクション終了」要求前の障害の場合，更新前のためルート/リーフ共にロールバックする．問題となるのは，ルートのコミット宣言（実更新）後，フェーズ2要求（図11の③）がリーフに届いてリーフがコミット宣言（実更新）するまでの間に障害が発生した場合である．「ルートは更新して，リーフは更新しない」非同期な状態が発生して，ACID特性の「一貫性 (C)」が保証されなくなる．2フェーズ・コミットメント処理方式では，ルートのコミット宣言（実更新）時点でリーフも含めトランザクション処理が完了したと判断するため，このような状態が発生した場合はリカバリ処理でリーフを実更新（ロールフォワード）する．

3. ACLES/DTPの概要

本章ではXISのサブ・システムであるACLES/DTPの分散トランザクション処理の機能および実現方式について記述する．

3.1 XIS内での位置づけ

ACLES/DTPは，XISを前提としており，XISのDC機能の一部に位置づけられる．

XIS内での位置づけは，前述のXIS/メッセージ制御（基本トランザクション転送）を介してMCB 1100とインタフェースをとり，使用者プログラムのスケジュールを制御し，XIS/ステップ管理使用者インタフェースにより使用者プログラムのトランザクション処理を管理する．またXIS/RDAFと環境を共有し，メッセージ送受信および同期制御を行う（図12）．

3.2 会 話 処 理

ACLES/DTPは，ネットワークに分散している複数の処理プログラムが相互通信し，複数プログラム間で同期を取り一つの処理（トランザクション）として意味を持つ単位にする処理方式である．

ACLES/DTPは，こうしたプログラム間の同期処理や障害時の回復処理機能を提供し，使用者が特に意識しなくてもトランザクションの4特性(ACID特性)を保証す

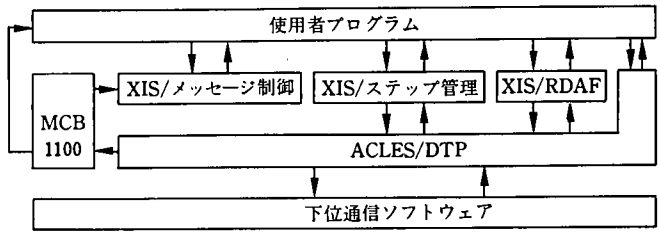


図 12　XIS 内での位置づけ

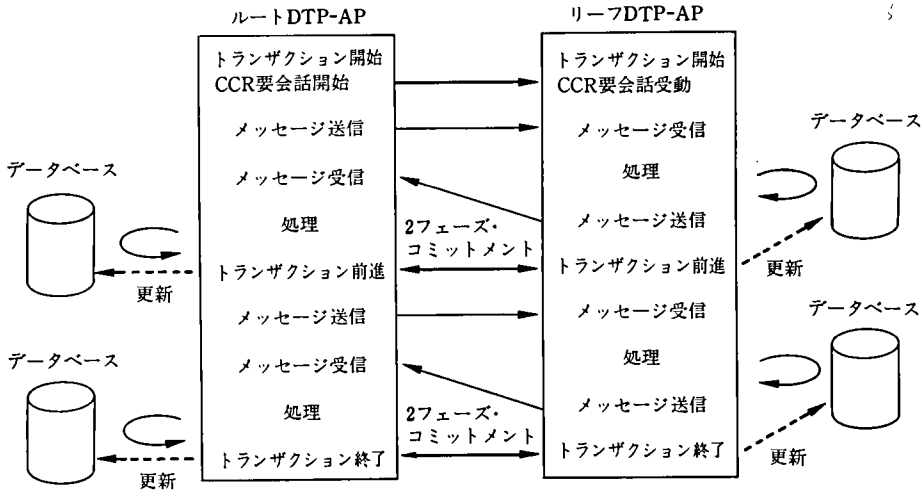


図 13　CCR 要会話処理

る．ACLES/DTP では，会話関係にある DTP-AP ごとのステップ更新処理の同期をとり，分散トランザクションとして DTP-AP 間の処理を完結させる．

使用者は「ルート」の会話確立要求（XDTPOPEN）時に，ACLES/DTP が提供する同期処理機能（CCR 機能）を使用するか否かを選択することができる．CCR 機能とは，使用者がステップ更新要求を行った場合，2 フェーズ・コミットメントにより DTP-AP 間の同期をとって更新する機能である．CCR 機能「要」として確立した会話を「CCR 要会話」と呼び，CCR 機能「不要」として確立した会話を「CCR 不要会話」と呼ぶ．

図 13 に，CCR 要会話処理例を示す．

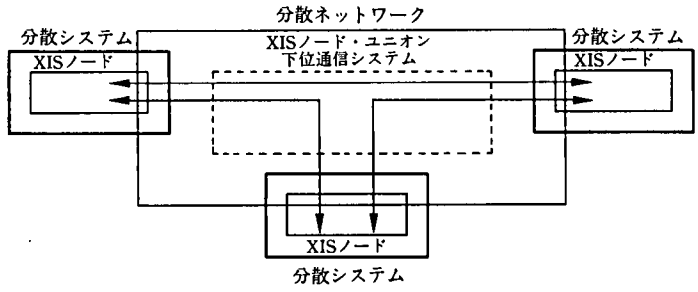
3.3　論　理　構　成

ACLES/DTP は，複雑かつ多様なコンピュータ通信ネットワークの発展に対応できるよう，分散ネットワークをモデル化し柔軟性，拡張性に富んだシステムの実現をめざしている（図 14）．

次に，ACLES/DTP システムの論理構成要素を示す（図 15）．

1)　XST（Xis STation）

ひとかたまりの業務処理の運用・始終業の単位．OSI の応用プロセスをモデル化したもの．XIS ノードに複数の XST を設定することができ，XIS ノード・ユニオン内でユニークな識別子を持つ．



・XISノード
　ネットワークで識別可能なシステムの導入，運用，リカバリの単位．
　XISにとってはIntegrated Recoveryのアプリケーション・グループ(APG)に相当する．ただし，コンカレント構成の場合は，コンカレントAPG内ホスト固有APGに対応する．
・XISノード・ユニオン
　XISノード間の通信を実現するための論理的な結合関係．
　ACLESが規定する構成要素によりネットワーク・ハンドリングを行う．

図 14　ネットワーク構成

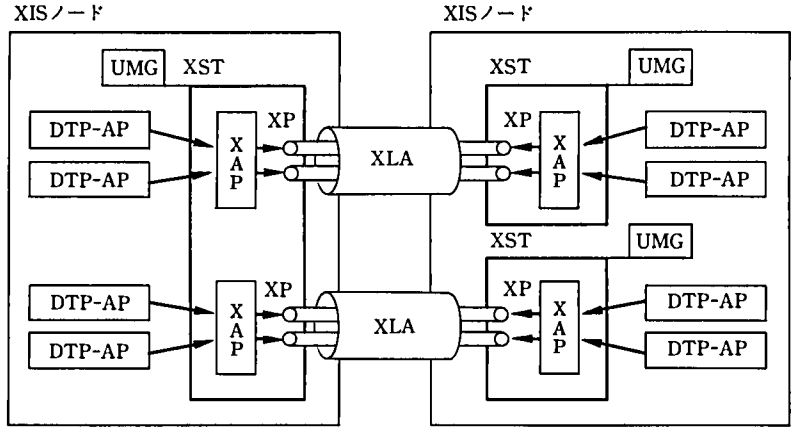


図 15　論理構成

2)　XLA (Xis Logical Association)

XST間の論理的な関連，通信路．自XSTと相手XST間の機能，構成を調整する単位となる．メッセージ転送の論理的通路（会話）の集合体である．XISノードでユニークな識別子を持つ．

3)　XAP (Xis service Access Point)

使用者プログラムがACLESのサービスをアクセスする窓口．メッセージを送受信する宛先，通信相手(XST)の識別となるものである．XAPとXLAは1対1に対応する．

4)　XP (Xis end Point)

メッセージの論理的な通り路のエンド・ポイント＝会話を識別するもの．XLAに複数のXPを設定することによりXLAを多重利用することができる．

5)　UMG (User ManaGer)

XST ごとに一つ存在が可能で，業務の運用を司る運用処理ユーザ・アプリケーション．XST 配下の XLA や XP の開始，終了，異常を把握することができる．

ACLES/DTP は，「物理的な通信路から独立した会話宛先の仮想化」を機能目標の一つに掲げ，使用者プログラムが使用者インタフェースを呼び出す際の会話宛先の指定を次のように仮想化している．

・XAP 識別名　　：通信相手（XST）側を識別する．

・会話識別子（XP）：相手使用者プログラムと 1 対 1 に対応する．

すなわち，XAP は XLA と 1 対 1 に対応するため，一つの XAP 識別名に複数の会話識別子を対応させることが可能である．

3.4 接 続 形 態

ACLES/DTP の接続可能な形態を図 16 に示す．

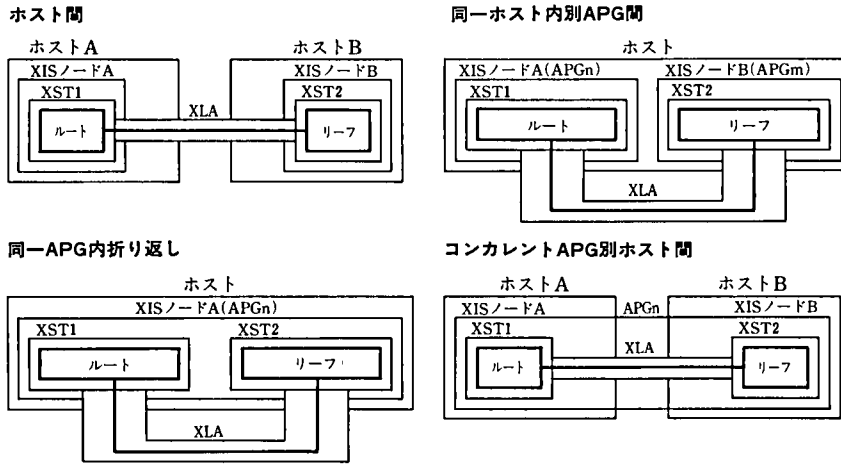


図 16　接続形態

ホスト-ホスト間の接続の他，UNIX-ホスト間の接続形態も，図 16 のホスト-ホスト間と同様の論理構成要素の定義で実現可能である．

UNIX-ホスト間の分散トランザクション処理は，ホスト～サーバ（UNIX）の垂直分散トランザクション処理要求への対応として実現された．UNIX の Open/OLTP 下の AP とホスト側の XIS 下の AP との会話処理を，UNIX 側に ACLES/DTP プロトコルのゲートウェイを実装することで可能としている．

UNIX-ホスト接続では，Open/OLTP のクライアント・サーバ型処理方式を採用し，ホストを巨大なデータベース管理者（サーバ）として位置づけている（図 17）．

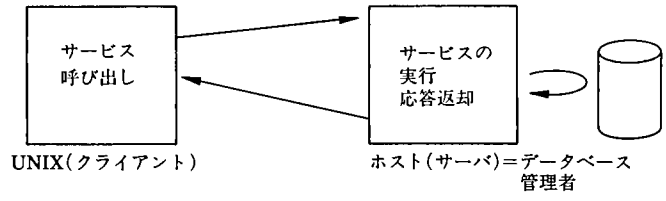


図 17　UNIX-ホスト接続形態

3.5 使用者インタフェースとサービス・プロトコル

ACLES/DTPの会話処理を実現するために，表1に示す使用者インタフェースが提供されている．

表1　ACLES/DTP使用者インタフェース一覧

No	インタフェース名	機　能
1	XDTPOPEN	会話を開始する．
2	XDTPCLOS	会話を終了する．
3	XDTPRCVE	ACLES/DTP制御メッセージを受信する．
4	XDTPSEND	会話メッセージを送信する．
5	XDTPABRT	会話を異常終了する．
6	XDTPINFO	会話状態を取得する．

ACLES/DTPは，XIS/ステップ管理インタフェースにより使用者プログラムのトランザクション処理を管理する．

表2に，XIS/ステップ管理インタフェースを示す．

表2　XIS/ステップ管理インタフェース一覧

No	インタフェース名	機　能
1	XSTPXINT	トランザクション処理の開始
2	XSTPXTRM	トランザクション処理の正常終了
3	XSTPXERR	トランザクション処理のエラー終了
4	XSTPXDRN	トランザクション処理の正常終了(多重受信)
5	XSTPXADV	トランザクション処理の前進
6	XSTPXCAN	トランザクション処理の無効化
7	XSTPXRQU	トランザクション処理の再開始要求

ACLES/DTP環境において，分散システム上のプログラム間会話処理を実現するために使用される通信規約をACLES/DTPプロトコルと呼ぶ．本プロトコルはOSI 7層モデルのアプリケーション層（7層）に位置づけられるACLES/DTP固有のプロトコルである（図18）．

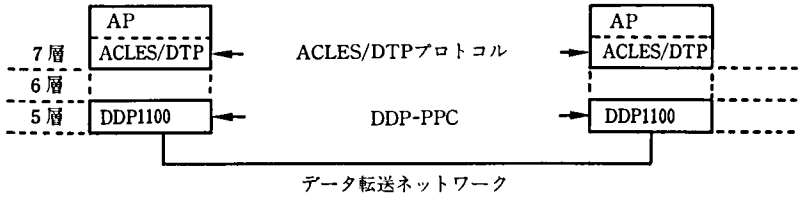


図18　ACLES/DTPプロトコル階層

表1，表2の使用者インタフェース（API）と対応するACLES/DTPプロトコルのサービス・プリミティブとプロトコルデータ一覧を表3に示す．

前述したように，ACLES/DTPはOSI-TPの規定するPeer to Peer型のプログラム間会話処理を採用しており，プロトコルのサービスプリミティブもOSI-TPサービ

表 3　サービス/プロトコル・データ一覧

機能種別	サービスプリミティブ	API	プロトコル・データ	RQ	ID	RP	CF
会話管理	会話確立	XDTPOPEN	XD-OPEN	○	○		
	会話解放	XDTPCLOS	XD-CLOSE	○	○		
	会話異常解放	XDTPABRT	XD-ABORT	○	○		
		XDTPINFO					
会話処理	会話メッセージ転送	XDTPSEND	XD-DATA	○	○		
		XDTPRCVE					
	会話制御権委譲		XD-TRANSFER	○	○		
同期制御	フェーズ1要求	XSTPXADV	XD-COMMIT	○	○		
	フェーズ1回答	XSTPXADV XSTPXTRM XSTPXDRN	XD-CONTINUE	○	○		
	フェーズ2要求		XD-DONE	○	○		
	フェーズ2回答		XD-COMPLETE	○	○		
	会話終了& フェーズ1要求	XSTPXTRM XSTPXDRN	XD-COMMIT- CLOSE	○	○		
	ロールバック通知	XSTPXCAN	XD-ROLLBACK	○	○		
	会話異常終了& ロールバック通知	XSTPXERR XSTPXRQU XDTPABRT	XD-ROLLBACK- ABORT	○	○		
回復処理	ステップ通知		XD-STEP	○	○		
	コミット情報		XD-RECOVER	○	○	○	○
	回復完了		XD-RECOVERY- COMPLETE	○	○	○	○

RQ：要求　ID：指示　RP：応答　CF：確認　○：必須
注）APIがないサービスは，ACLES/DTPが内部的に作成し送受信を行うものである。

スプリミティブに対応して設計されている．

表4に，ACLES/DTPプロトコルのサービスプリミティブとOSI-TPサービスプリミティブの対応表を示す．

3.6　ACLES/DTP会話処理シーケンス

ACLES/DTPサービスプリミティブを使用した会話処理シーケンス例について，正常処理シーケンスを図19に，エラー処理シーケンスを図20に示す．

図19の正常処理シーケンス例について，次に各シーケンスを説明する．

① ルートが「XSTPXINT」でトランザクション処理の開始を要求する．

② ルートが「XDTPOPEN」で会話の開始を要求する．ACLES/DTPは，会話確立（XD-OPEN）を作成し送信する．

③ XD-OPEN受信を契機にスケジュールされたリーフは「XSTPXINT」でトランザクション処理の開始を要求し，「XDTPRCVE」で受信データを受け取る．

④ CCR要会話の場合，ACLES/DTPは内部的にステップ通知（XD-STEP）を作成し送信する．XD-STEPにはリーフのステップ識別子が設定されこれをルート側で保存することにより，障害時の回復処理に使用される．

⑤ リーフはXD-OPENに付加されたデータより，データベース更新等の処理

表 4　ACLES/DTP プロトコルと OSI-TP サービスプリミティブの対応表

ACLES/DTP サービスプリミティブ			OSI-TP サービスプリミティブ		
機能	サービスプリミティブ	プロトコル・データ	機能	サービスプリミティブ	プロトコル・データ
会話管理	会話確立	XD-OPEN	カーネル	ダイアログ開始	TP-BEGIN-DIALOGUE
	会話解放	XD-CLOSE		ダイアログ終了	TP-END-DIALOGUE
	会話異常解放	XD-ABORT		提供者異常終了	TP-P-ABORT
				利用者異常終了	TP-U-ABORT
				提供者エラー報告	TP-P-ERROR
				利用者エラー報告	TP-U-ERROR
				提供者ダイアログ開始拒否	TP-P-REJECT
会話処理	会話メッセージ転送	XD-DATA		データ転送	TP-DATA
	会話制御権委譲	XD-TRANSFER	半重二型	制御権の譲渡	TP-GRANT-CONTROL
				制御権譲渡依頼	TP-REQUEST-CONTROL
			ハンドシェイク	処理の同期を取る	TP-HANDSHAKE
					TP-HANDSHAKE -AND-END
					TP-HANDSHAKE -AND-GRANT-CONTROL
同期制御			コミット	コミット完了時制御権譲渡	TP-DEFERRED -GRANT-CONTROL
	フェーズ1要求	XD-COMMIT		コミット準備要求	TP-COMMIT TP-PREPARE
	フェーズ1回答	XD-CONTINUE		準備完了報告	TP-CONTINUE-COMMIT TP-READY
	フェーズ2要求	XD-DONE		コミット完了要求	TP-DONE
	フェーズ2回答	XD-COMPLETE		コミット完了報告	TP-COMMIT-COMPLETE
	会話終了& フェーズ1要求	XD-COMMIT-CLOSE		コミット完了時ダイアログ解放	TP-DEFERRED -END-DIALOGUE
	ロールバック通知	XD-ROLLBACK		ロールバック要求	TP-ROLLBACK
				ロールバック完了報告	TP-ROLLBACK -COMPLETE
	会話異常終了& ロールバック通知	XD-ROLLBACK-ABORT			
回復処理	ステップ通知	XD-STEP			
	コミット情報	XD-RECOVER			
	回復完了	XD-RECOVERY-COMPLETE			
			非連鎖トランザクション	アトミックアクション範囲縮小	TP-DEFERRED-NEXT -TRANSACTION TP-UNCHAIN -TRANSACTION
				アトミックアクション範囲拡大	TP-BEGIN -TRANSACTION

注）空白に対応するサービスは提供されていない。

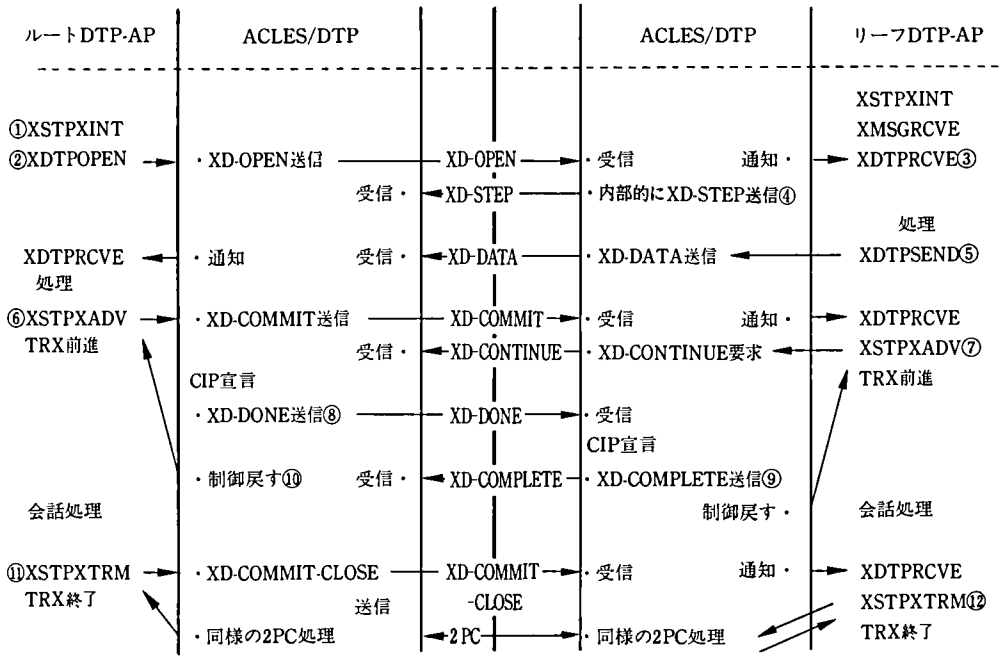


図 19　正常処理シーケンス例

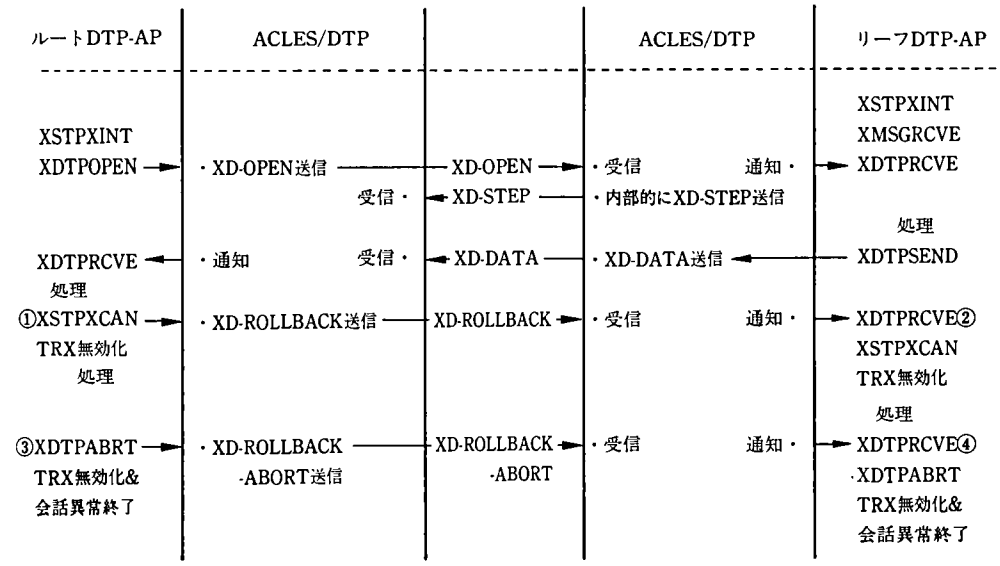


図 20　エラー処理シーケンス例

を行い，処理結果を「XDTPSEND」で送信する．

ACLES/DTP は会話メッセージ転送（XD-DATA）を作成し送信する．

⑥ XD-DATA を受信したルートはデータベース更新等の処理を行い，処理を正常に終了すべきと判断した場合，「XSTPXADV」でトランザクション処理の前進を要求する．これを契機に，更新同期のための 2 フェーズ・コミットメント

処理を開始する．

ACLES/DTPは，フェーズ1要求（XD-COMMIT）を作成し送信する．

⑦ XD-COMMITを受信したリーフは，処理を正常に終了すべきと判断した場合，「XSTPXADV」でトランザクション処理の前進を要求する．ACLES/DTPはフェーズ1回答（XD-CONTINUE）を作成し送信する．

⑧ XD-CONTINUEを受信したルート側ACLES/DTPは，CIP宣言(実更新)を行い，フェーズ2要求（XD-DONE）を作成して送信する．

⑨ XD-DONEを受信したリーフ側ACLES/DTPは，CIP宣言（実更新）を行い，フェーズ2回答（XD-COMPLETE）を作成して送信する．制御をリーフに戻し，新ステップを開始する．

⑩ XD-COMPLETEを受信したルート側ACLES/DTPは制御をルートに戻し，新ステップを開始する．

⑪ ルートは会話処理を行った後，「XSTPXTRM」でトランザクション処理の終了を要求する．これを契機に，2フェーズ・コミットメント処理が開始し，ACLES/DTPは会話終了&フェーズ1要求（XD-COMMIT-CLOSE）を作成し送信する．

⑫ XD-COMMIT-CLOSEを受信したリーフは，「XSTPXTRM」でトランザクション処理の終了を要求する．以下同様の2フェーズ・コミットメント処理が行われる．

図20のエラー処理シーケンス例について，次に各シーケンスを説明する．

① 会話処理中のルートが「XSTPXCAN」でトランザクション処理の無効化を要求する．ACLES/DTPはロールバック通知（XD-ROLLBACK）を作成し送信する．この時点でステップは「XSTPXINT」直後の状態までロールバックされる．

② XD-ROLLBACKを受信したリーフ側ACLES/DTPは「XDTPRCVE」の処理結果に「ロールバック通知の受信」と「ロールバック識別子」を設定することにより，リーフにロールバックすべきことを通知する．リーフは処理結果より判断して，「XSTPXCAN」でトランザクション処理の無効化を要求する．

この時点でステップは「XSTPXINT」直後の状態までロールバックされる．

③ 会話処理を続行したルートが「XDTPABRT」でトランザクション処理の無効化と会話異常終了を要求する．ACLES/DTPは会話異常終了&ロールバック通知（XD-ROLLBACK-ABORT）を作成し送信する．

④ XD-ROLLBACK-ABORTを受信したリーフ側ACLES/DTPは「XDTPRCVE」の処理結果に「会話異常終了&ロールバック通知の受信」と「ロールバック識別子」を設定することにより，会話が異常終了したため，ロールバックすべきことを通知する．リーフは処理結果より判断して，「XDTPABRT」でトランザクション処理の無効化と会話異常終了を要求する．

3.7 更新同期機能とリカバリ機能

3.7.1 更新同期とリカバリの範囲

前述したように，ACLES/DTPの更新同期処理は2フェーズ・コミットメント方式

を採用している．

2フェーズ・コミットメント処理ではルート・トランザクションのCIP要求が行われた時点でリーフ・トランザクションも含め処理が完了したと判断し，障害が発生した場合でもすべての処理を完結する(ロールフォワード)．しかし，分散システムに配置されているため，ルート・トランザクションがCIP要求してからリーフ・トランザクションがCIP要求するまでにはタイム・ラグが存在する．よって，その間に障害が発生した場合，処理の同期を取るためのリカバリ処理が必要になる．

2フェーズ中に障害が発生した場合，ルート・トランザクションがCIP要求済みか否かで，リーフ・トランザクションはロールフォワードまたはロールバックを判断する必要がある．リーフが独自で判断不可能な範囲は，リーフのフェーズ1回答（②）送信からフェーズ2要求（③）受信までの間（図21の網掛け部分）である．

3.7.2 リカバリの方法

リカバリ処理のポイントとなるリーフのステップ識別子について示す（図21）．

❶ リーフのステップ識別子はリーフ・トランザクションを識別するIDで，CCR要会話の会話確立（XD-OPEN）の回答として内部的にルート側へ通知される．

ルートはこれを保存し，同期制御（2フェーズ・コミットメント）に備える．

❷ ルートが「XSTPXADV」でトランザクションの前進を要求すると，ルートが会話開始時に保存したリーフのステップ識別子を回復型テーブルへ書き込む要求を出す．但し，この時点では中間バッファに書かれるのみで回復型テーブルは更新されない．

❸ フェーズ1回答（XD-CONTINUE）を受信したルートはコミット宣言（CIP要求）を行い，リーフのステップ識別子が中間バッファより回復型テーブルに書き込まれる（実更新）．

すなわち，リーフのステップ識別子が回復型テーブルに書き込まれている場合，ルートはCIPを取得したためロール・フォワードと判断し，書き込まれて

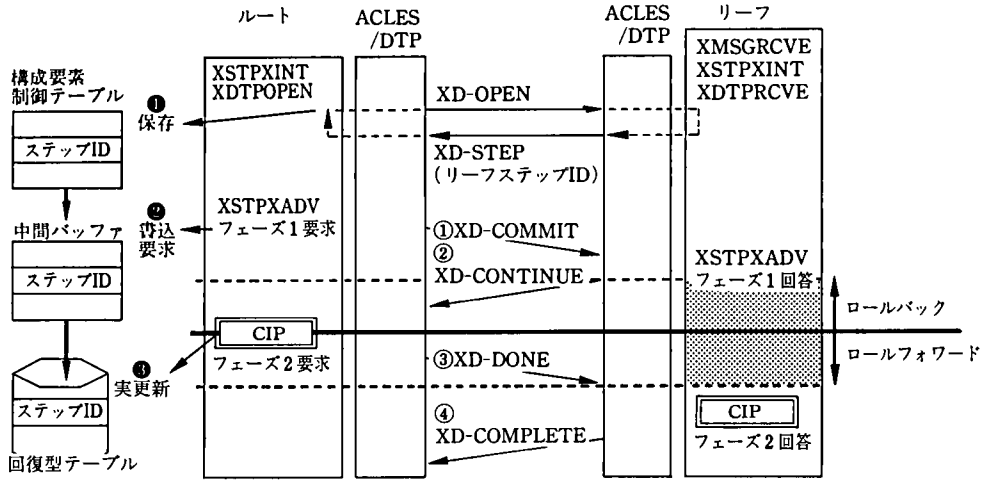


図21 ACLES/DTPの2フェーズ・コミットメント処理方式(▨：網掛け部分)

いない場合はロール・バックと判断する．

ACLES/DTP のリカバリ処理は，

(A)　ルート・トランザクションが配置されたノードでの障害

(B)　リーフ・トランザクションが配置されたノードでの障害

に分けられる．障害発生時のリカバリ対象ステップは図 21 の網掛け部分である．ルートが CIP を取っている場合，リーフのステップをロールフォワードし，取っていない場合，ロールバックする．すなわち，ルート/リーフ共にロールバックまたはロールフォワードが判断可能な場合はリカバリ対象ステップにならない．

(A)の場合，リーフは独自の判断が不可能なため，ルート側の障害がリカバリされて指示（ロールフォワードまたはロールバック）があるまで待ち状態（凍結状態）となる．

ルート側ノードのリカバリ後，ACLES/DTP のリカバリ機能が回復型ファイル(図 21 の❸) に書き込まれた(すなわち，ロールフォワードすべき)リーフのステップ識別子をリーフ側 ACLES/DTP リカバリ機能に渡し，凍結中のリーフのステップ識別子と一致した場合は該リーフをロールフォワードする．

(B)の場合，IRU 1100 主導でルート・トランザクションの状態（ロールフォワードまたはロールバック）を問い合わせる．

リーフ側ノードのリカバリ処理の IRU 1100 にてロールバックまたはロールフォワードの判断不可能なステップ（仕掛かり中ステップ）として認識する．リーフ側 ACLES/DTP のリカバリ処理はルート側の ACLES/DTP リカバリ処理に対しコミット情報(回復型ファイルに書かれたステップ識別子)を要求する．リーフ側 ACLES/DTP リカバリ処理はルート側より取得したコミット情報を IRU に引き渡し，IRU は仕掛かり中ステップがコミット情報と一致した場合は該ステップをロールフォワードする．

3.8 ダイアログ・ツリー機能

分散処理システムが複雑化するに従い，1 対 1 の会話処理では使用者の要求を満たすことができなくなっている．そこで，ACLES/DTP は，ネットワークに分散した三つ以上のプログラム間で会話処理を行うダイアログ・ツリー機能を提供する．

三つ以上の DTP-AP が会話で結合し，木構造をなしたものを「ダイアログ・ツリー」と呼ぶ．一つの DTP-AP は複数の DTP-AP に対して会話を確立することができる(1

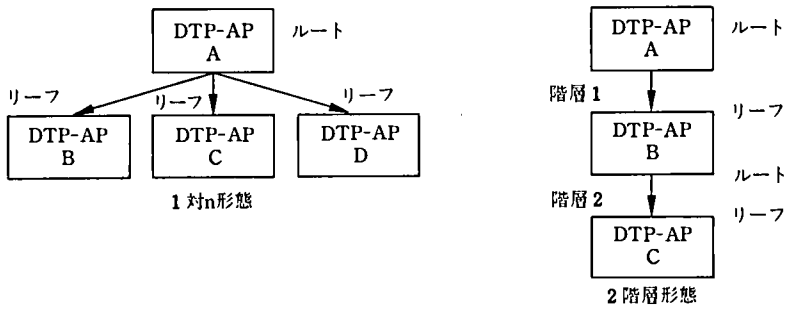


図 22　ダイアログ・ツリー形態

対n形態)．また，あるルートDTP-APより会話を確立されたリーフDTP-APは，さらに別のDTP-APに対して会話を確立し，該会話のルートとなることができる(2階層形態)(図22)．

ダイアログ・ツリーを使用した例としては，UNIX (U 6000)～ホスト1 (2200)～ホスト2(2200)の2階層かつホスト2は複数で分散トランザクション処理環境を構築した例がある (図23)．

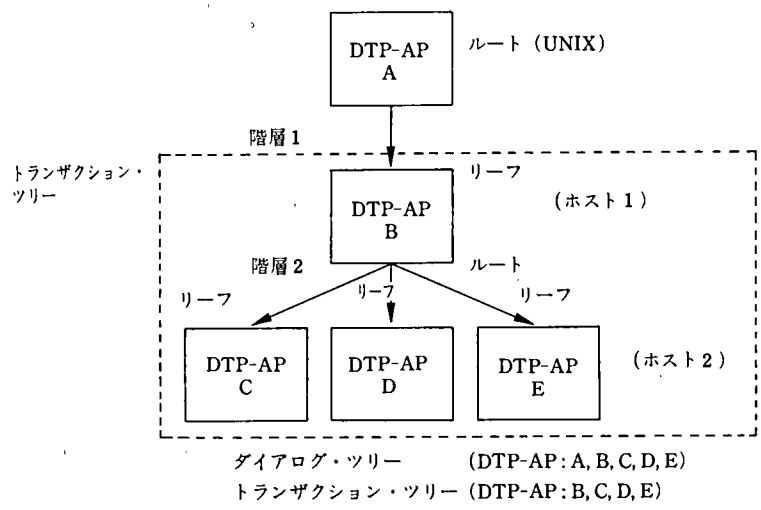


図 23 ダイアログ・ツリー実現例

この構成では，UNIX側がCCR要機能を実装しないため，トランザクション・ツリー (CCR要で確立した部分ツリー) はホスト1～ホスト2の部分の1階層のみとなっている．

4. おわりに

今後コンピュータ・システムの大規模化，分散化がますます進むことにより，分散トランザクション処理の全体に占める割合は増加し重要性が高まると予想される．

ACLES/DTPは，刻々と変わり続ける市場ニーズに対応するために機能を拡張する必要がある．ACLES/DTPの次の課題は2階層以上のトランザクション・ツリー機能の提供である．

参考文献 [1] J. グレイ他著 “OLTPシステム”，マグロウヒルブック，1991.
[2] 棟上昭男他著 “OSIの応用”，日本規格協会，1987.
[3] 情報処理相互運用技術協会作成 “OSI-TP実装規約”，日本規格協会，1993.
[4] 塚本・八田編，大特集 “分散処理技術”，情報処理，Vol. 28, No. 4, 1987.
[5] 黒川茂，“アプリケーション連動の仕組み”，ユニシス技報，Vol. 13 No. 4, 1994.2, pp. 167～178.

執筆者紹介　栗　原　幸　代 (Sachiyo Kurihara)

1990年青山学院大学国際政治経済学部国際政治学科卒業．同年日本ユニシス（株）入社．分散トランザクション処理制御プロダクトの開発，サービスに従事．現在システム企画開発2部オンライン開発2課に所属．

XISのシステム運用記述言語XOL

XOL: The System Operation Description Language for XIS

米津政紀, 徳永路晴

要 約 XIS(eXtended Information System)が提供するオンライン・システムの運用を記述するための言語XOL(Xis Operating Language)を紹介する. システム運用を記述する言語はJCLが一般的であるが, JCLはコンピュータに対するヒューマン・インタフェースが単純なキーボード入力に限られていた頃に開発されたまま進歩していない. 一方, システム運用はコンピュータ利用形態の多様化にともない複雑化している. XOLは, 複雑化したシステム運用, とくにXTPAのシステム運用を記述するために開発された言語である.

本稿は, まず現状の問題点を明確にし, その上でXOLの狙い, 機能, 実現方式に言及することにより, XOLが複雑化したシステム運用を記述する能力があることを明らかにする.

Abstract This paper refers to XOL, which is the operation description language for XIS-supported on-line systems. JCL has generally been used as the language to describe system operation. JCL was developed in the days when the human interface was simply limited to keyboard input operations and has seen no enhancements since then. In the meanwhile, system operation has grown more complicated in proportion as the modes of computer utilization have been diversified. XOL is the language developed to describe the complex operation of systems, especially supported by XTPA.

Aside from mentioning where JCL stands now, this paper is meant to clarify the fact that XOL is capable of describing complex systems operation by discussing XOL's objectives, functions and implementations.

1. はじめに

XOLは, オンライン・システムの運用処理（オンライン・システム開始/終了処理, 各種障害回復処理）を記述するための言語であり, XISのサブシステムであるシステム運用支援機能として開発されたソフトウェアである.

システム運用ソフトウェアの歴史は1970年代後半から始まる. システム運用操作員のコスト削減とコンピュータの安全運転に対する強い要請から, コンピュータの自動運転を狙ったシステム運用ソフトウェアの需要が発生した. 当初, バッチ系ジョブの省力化を図るジョブ管理ソフトウェア, 磁気テープ管理ソフトウェアがコンピュータ市場に登場し, その後のシステム運用ソフトウェアの発展は無人運転機能にまで至っている. 当社のシリーズ1100においても1980年代前半には, COSTAR*, UOSS*, SAFE* 等のシステム運用ソフトウェアをそろえ, 一通りのシステム運用の自動化に対する市場要求に応えた.

1980年代後半に始まるXISの開発において最大の目標は, XTPAを基盤とする大規模トランザクション・システムの実現である. XTPAでのシステム運用は, さまざ

* COSTAR (Computer Operating management System for Total Automation & Reliance) はコンピュータ総合運用管理システム, UOSS (Unattended Operation Support Software) は無人運転支援ソフトウェア, SAFE (System for Automated Failsafe Environment) はホット・スタンバイ・システム支援ソフトウェア・パーケージである.

まな面においてホスト間で整合性をとる必要があり，有機的なシステム運用が要求される．XTPA のオンライン・システムの開始/終了処理においては，MHFS* 環境の初期化状態，ブート・タイプ，ホスト共通処理（VALTAB**，共用 TIP** ファイル・ディレクトリ等），ホスト固有処理（IRU*** のショート・リカバリ等）のホスト間競合等を考慮しなければならない．マルチ・アプリケーション・グループ[1][2]の処理フローにこれらの考慮を加えると，3 次元的な（ホスト固有アプリケーション・グループ処理，ホスト固有アプリケーション・グループ共通処理，ホスト共通アプリケーション・グループ処理，ホスト共通アプリケーション・グループ処理が組み合わされた）同期制御がないかぎり運用操作員の労力は多大である．まして，システム障害の回復処理はさらに複雑な処理フローであり，安全で確実なシステム運用を構築するためには XTPA 対応システム運用ソフトウェアが必須である．従来のオンライン・システム・ソフトウェアでは，システム運用支援機能はオンライン・システムの運用 JCL 生成，処理フローを制御するプログラム等のシステム運用ソフトウェアを提供することであった．しかし，JCL の記述能力は定型的であり，XTPA のシステム運用を記述するのは困難であると予想された．振り返ってみると，飛躍的に進歩したシステム運用ソフトウェアの歴史の中で，なぜか JCL だけがその進歩からとり残されていた．このような背景のもとに，XOL は XTPA のシステム運用を記述することを目的として開発された．

2. XOL の設計方針

従来のオンライン・システム・ソフトウェアのシステム運用支援機能に XTPA の考慮を加える場合の問題点を洗い出す過程で，JCL に代わる言語を開発するという方針が決定した．本章はその開発過程に沿い，まず問題点を洗い出し，そこから導き出された XOL の設計目標について述べる．

2.1 問題点の洗い出し

XIS の設計は，AIS 1100 II (Advanced Information System for UNISYS Series 1100：統合オンライン・ソフトウェア)の開発保守で蓄えられた技術資産を利用して進められた．

AIS 1100 II は 200 を越すユーザ・システムで稼働している安定したオンライン・システム・ソフトウェアであり，システム運用支援機能としてシステムの開始・終了・回復を行う JCL の生成機能を提供している．システムの開始形態は，使用者が独自に設定，変更することができ，開始・終了・回復にあたってはオペレータの判断や手操作を削減し，操作ミスを防止している．また，各 JCL では使用者独自の処理を組み込むことができるため，使用者のオンライン・システムに適したシステム運用が可能となる．

しかし，AIS 1100 II のシステム運用支援機能の延長線上で XTPA 対応する場合，次の問題点があった．

---

* MHFS (Multi Host File Share) はマルチホスト・ファイル・シェアである．

** VALTAB (VALidation TABle) は TIP プログラム妥当性検証テーブル，TIP (Transaction Interface Package) はトランザクション処理システムである．

*** IRU (Integrated Recovery Utility) は統合回復ユーティリティである．

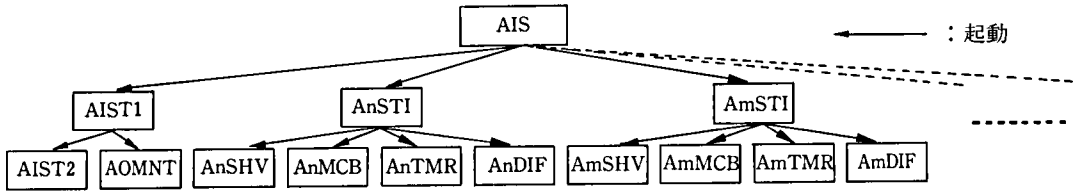


図 1　初期開始処理の場合のラン・ストリーム

1)　現実のシステム運用とシステム運用記述の不一致……たとえば，使用者がシステムの開始処理を行う際に，オペレータは“AIS”というラン・ストリームを起動するだけでよい．このラン・ストリームは使用者の事前の設定，およびシステムの状態を内部的に判断し，複数の開始ラン・ストリームを起動することによりシステムの開始処理を自動的に行う．このように，“AIS”ではそのラン・ストリーム自身が内部的に処理の判断を行うことによってオペレータの負荷を軽減している．

“AIS”から起動されるラン・ストリームは図1のようになる．

このように，“AIS”は複数のラン・ストリームを起動し，かつ起動された各ランもまた複数のラン・ストリームの起動を行っている．また，各ラン・ストリーム同士で同期をとりながら処理を実行している．しかし．実際の各ラン・ストリームの JCL を直接見ただけでは，以上のような各ラン・ストリームの処理の流れを理解するのは不可能である．

このように，あるラン・ストリームが他のラン・ストリームを起動したり，各ラン・ストリームで同期を取っている，という実際の処理と JCL での記述がかけ離れたものとなっている．

2)　各処理ごとに異なる表現形式……“AIS”より起動され各アプリケーション・グループの初期処理を行う“AnSTI”では，同様に“AIS”より起動されるラン・ストリーム“AIST 1”と同期をとる処理を行っており，JCL 上では次のようになる．

```
@XQT VSETUP
ACT=WAIT,AIST1 FILE=AIS$,AIST1$$
RMCONS=AnSTI,usrid,password
```

また，各ラン・ストリーム内では起動するラン・ストリームの判断，各ラン・ストリーム中の分岐，障害時のリカバリ，等のためにラン・ストリーム中でファイル AIS—CPF にチェック・ポイントを書き込み，かつ書き込まれたチェックポイントの参照を行っている．チェックポイントの書込み，およびチェックポイントの参照による処理の分岐はそれぞれ次のように行っている．

```
@XQT VSETPT
INF RUN=AnSTI LABEL=S3ST, ST
ACT CKPT=WRITE
```

```
@XQT VSETPT
INF RUN=AnSTI LABEL=S3ALL
ACT CKPT=NONE CHK=FCSS
```

このように，各ラン・ストリーム中では，各種の運用操作を行うために特別に作成したユーティリティを用いることにより，運用操作の自動化を図っている．

また，ラン・ストリーム中でアプリケーション・グループ状態の変更，MCB の初期化終了待ち，等のために，システム・コンソールへのキーイン，およびシステム・コンソール・メッセージの監視を行っているが，それぞれの処理は次のようになる．

```
@XQT VSETUP
ACT KEYIN,AP FORM=AP,n,UP
ANS=INIT
RMCONS=AnSTI,usrid,password
```

```
@XQT VSETUP
ACT=WAIT,MCB APG=n
RMCONS=AnSTI,usrid,password
```

以上で述べたように，各ラン・ストリームでは特別に作成したユーティリティを使用して各種の運用操作を行っている．各ユーティリティはパラメタの指示に従い内部的に処理を行っているため，使用者がその処理を理解するためには各ユーティリティの機能およびパラメタの意味をそれぞれ理解する必要がある．しかし，ユーティリティのパラメタは上記で例に出した JCL のように機械的な表現しかできず，また各ユーティリティごとに形式，意味が異なっている．それぞれに機能，実行形式，意味が違う各ユーティリティの処理内容を理解するのは困難である．

3) データの管理が困難……JCL によるシステム運用ソフトウェアでは時系列的な処理の流れで処理を分割し，それを単位としてラン・ストリームを作成する．そのため一つのラン・ストリームを形成するのは，必ずしも物理的な処理単位でなく，処理を行う時間的な処理単位となる．AIS の開始処理では，

アプリケーション・グループ共通の初期化処理

↓

各アプリケーション・グループの初期化処理

という時系列的な流れに沿い，それぞれの処理を行うラン・ストリームを起動している．そのため，たとえば TIP 関連の初期設定では，VALTAB の登録はアプリケーション・グループ共通の初期化処理を行うラン・ストリームで行い，TIP プログラムの登録は各アプリケーション・グループの初期化処理を行うラン・スト

リームで行う，というように別のラン・ストリームで実行されている．このことは，ラン・ストリームを使用しない場合の，運用操作員がVALTABの登録やTIPプログラムの登録を行う，という実際の動作とはかけ離れたものとなり，またこのことによりTIPプログラムに関するデータが各ラン・ストリームに分散して存在することになり，保守の煩雑さを招いている．

4) 各処理の独立性が困難……JCLでは定型的な処理しか行えないため，たとえばAISでは，各アプリケーション・グループの開始処理を行うAnSTI（n：アプリケーション・グループ番号）を開始処理を行うアプリケーション・グループの数だけ作成しなければならない．図1でわかるようにAnSTIからは“AnSHV”，“AnMCB”，“AnTMR”，“AnDIF”がそれぞれ起動されるため，これらのラン・ストリームをアプリケーション・グループの数だけ用意しなければならない．このように，JCLはその処理の数だけラン・ストリームが必要となり構造的にシステム運用を構築することが困難である．

以上で述べたような処理の理解の困難さに対してAIS 1100 IIでは運用マニュアルである“AIS 1100 II解説書　運用管理編”にて対応している．

しかし，XISが対象とするXTPAのオンライン・システムの開始/終了処理においては，開始/終了処理における処理フローは複雑化し，それに対応してJCLの制御構造も複雑化せざるを得ない．また，各処理に対応するラン・ストリームをそれぞれに作成する必要があり，たとえばシステムの開始だけをとってもAISに比べ起動する開始ランの数は飛躍的に増大することとなり，もはや運用マニュアルだけでは対処できないほど使用者への負担が増大する．

2.2 設 計 目 標

洗い出された問題点を見てみると，ソフトウェア・クライシス以来語られてきた問題点と同じものがある．現在，これらの問題点を解決する最も有力な方法論はオブジェクト指向である．JCLに代わりオブジェクト指向型言語を開発することにより問題を解決することにした．ただし，JCLとオブジェクト指向型言語があまりにもかけ離れているため，JCLとの親和性を図ったものを設計することになり，次の設計目標を設定した．

1) すべてのシステム運用を統一した表現形式で記述できる．

JCLのラン・ストリーム上で，その処理が複雑化する原因となっている実行の選択，分岐，およびデータの保持をすべて可視的に表現できるようにする．

2) 日本語が容易に使用できる．

日本語によるシステム運用操作の記述を可能にすることにより，運用マニュアルの参照なしに，記述された各種操作を直接参照することで理解できるようにする．

3) 部品化とカプセル化が可能である．

開発，保守の容易性のために，各処理ごとの部品化とカプセル化ができるようにする．

4) 過去の資産（JCL）が使用できる．

JCLの制御（起動）を記述できるようにすることで，既存のJCL使用を可能と

する．

3. XOLの狙い

システム運用が統一した表現形式で記述できるように，まず基本概念を設けた．本章は，その基本概念によりXOLがどのようなシステム運用の実現を狙っているのかを述べる．

3.1 基本概念

XOLでは，システム運用に関わる物理的/論理的構成要素を“アクタ”，その操作/処理を“アクション”という概念で抽象化している．すべてのシステム運用は“アクタがアクションを実行する”という形式で表現され，“アクタにアクションを依頼する”ことによりアクションが実行される．この依頼のインタフェースを“メッセージ・パッシング”と呼ぶ．

3.1.1 アクタとアクション

アクタは，データとアクションをひとまとめにしたものであり，アクションの実行主体でもある．アクタに内在するデータとそのアクションは，アクタ単位にカプセル化されており，データおよびアクションの変更による影響範囲を局所化することができる．アクタは，複数種類のアクションを実行することができる．各アクションは依頼されるメッセージと1対1に対応しており，依頼するメッセージを変えることにより，同一のアクタに別のアクションを実行させることができる．たとえばアプリケーション・グループのアクタを考えてみると，アクタ‘APG’の中にはアプリケーション・グループ状況をデータとして持っている(図2)．アプリケーション・グループのシステム運用には開始，終了および障害回復がある．これらのシステム運用処理はアプリケーション・グループ状況に応じて変わってくるが，アクタに処理（アクション）を依頼する人は常に同じ依頼メッセージを渡すだけでよい．また，アプリケーション・グループのタイプ（ローカル，コンカレント）が変わった場合，システム運用処理内容は変更しなければならないが，依頼メッセージは‘開始’，‘終了’，‘障害回復’のように普遍的なものであり，変更する必要がないので，変更による影響範囲がアクタ内に局所化される．

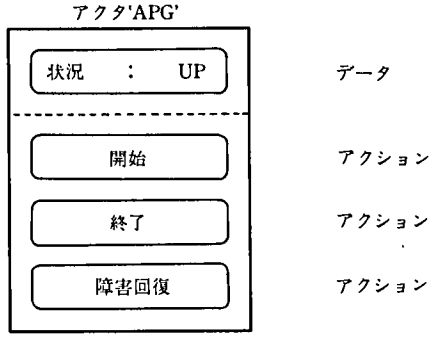


図2　アクタの例

3.1.2 メッセージ・パッシング

アクタにアクションを依頼するためには，アクタに依頼内容をメッセージとして渡す．アクタにメッセージを渡すことをメッセージ・パッシングという．メッセージ・パッシングは，アクションを実行させる唯一のインタフェースである．メッセージ・パッシングされたアクタは，受け取ったメッセージによりアクションを選択し，アクションを実行する．たとえば，アクタ 'APG' に '開始' メッセージをメッセージ・パッシングした場合は開始アクションを実行し，'終了' メッセージをメッセージ・パッシングした場合は終了アクションを実行する(図 3)．

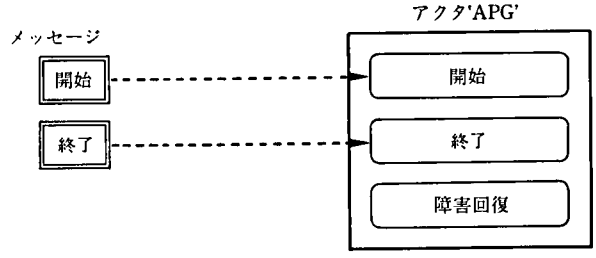


図 3　メッセージ・パッシングの例 1

また，メッセージ中に可変情報を含ませてアクションに可変情報を反映させることができる．これにより共通モジュール的な部品化が可能になる．たとえば，アクタ 'APG' が特定のアプリケーション・グループのシステム運用処理ではなく，可変情報として識別子となるアプリケーション・グループ番号を付加するメッセージ定義であればアプリケーション・グループ共通のアクタにすることができる(図 4)．

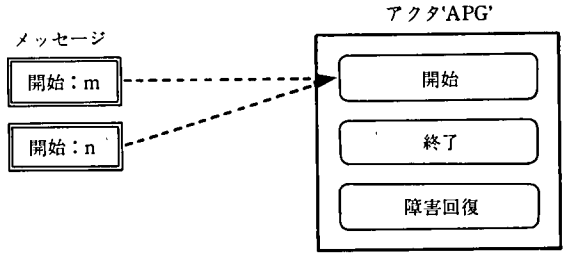


図 4　メッセージ・パッシングの例 2

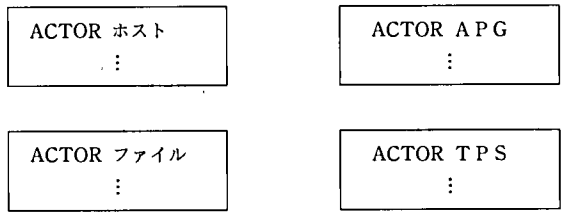
3.2 言語仕様による狙い

3.2.1 日本語での記述

XOL は日本語による記述が可能である．そのため XOL の原始プログラムを直接見ることでシステム運用の処理内容を理解することができる．

3.2.2 現実のシステム運用操作との構造の同一化

XOL においては，すでに述べたように実際のシステム運用における物理的・論理的な構成要素を中心としたシステム運用アプリケーションの構築が可能となる．つまり，現実世界に存在するシステムの構成要素を“アクタ”とすることで，コンピュータの世界の中で現実世界からイメージした自然な発想でシステム運用アプリケーションの構築が行える(図 5)．

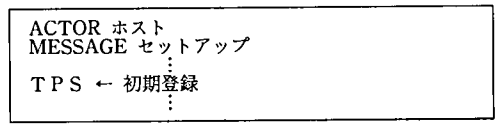


"ホスト", "APG", "ファイル", "TPS"というシステムの構成要素をそのまま，アクタ "ホスト", アクタ "APG", アクタ "ファイル", アクタ "TPS" として作成する．

図 5 アクタの作成例 1

3.2.3 処理の独立性(カプセル化)

XOL におけるシステム運用アプリケーションでは，複数のアクタがそれぞれの処理を担当している．システム運用における一連の処理（各処理がからみあった処理）においても各アクタがそれぞれ自分が行う処理を自律的に判断して行う．処理の中で自分の担当以外の処理を行う場合には，他のアクタにメッセージを介して処理の依頼を行うだけである．そのため，他のアクタでその処理がどのように実現されているのかを知る必要がない．このことにより，システム運用の操作/処理の変更に伴うシステム運用アプリケーションの変更は対象となるアクタの変更にとどまる(図 6)．



アクタ "ホスト" は，アクタ "TPS" に "初期登録" というメッセージを送ることで TPS の初期登録を行うことができ，アクタ "TPS" 中で TPS の初期登録がどのように実現されているかを知る必要がない．

図 6 アクタの作成例 2

3.2.4 データの独立性

アクタが使用するデータはそのアクタ内でのみ更新/参照が可能である．つまり，あるデータの内容，構造を管理しているのは，ある特定のアクタのみである．そのためデータの変更にともなう変更は，そのデータを管理するアクタのみで済む(図 7)．

3.2.5 状況に応じた処理(部品化)

アクタは起動の契機となるメッセージを変更することにより同じアクタに全く別の処理を行わせることができる．また，メッセージに可変情報を含めることができ，その可変情報を処理に反映させることができる．つまり，メッセージから得た可変情報をアクタでは変数として扱い，実行に際しその変数に値を設定することで，同一のアクタで多様な処理を可能としている．そのため従来の JCL ではできなかった状況に応じた処理が，同一のアプリケーションで可能となる(図 8，図 9)．

3.3 開発環境での狙い

開発環境では，システム運用構築の負荷を軽減させることを狙い，次の二点の実現を図る．

```
ACTOR ホスト
CONSTANT ＡＰ１形態='コンカレント',
         ＡＰ２形態='ローカル'                    ・・・①
          ⋮
VARIABLE ＡＰ１状態：CHAR,                        ・・・②
         ＡＰ２状態：CHAR.
          ⋮

MESSAGE セットアップ
[IF ＡＰ１形態='コンカレント' THEN
          ⋮
   ELSE
          ⋮

   ＡＰＧ←セットアップ：'1'
        運用形態    ：ＡＰ１形態；              ・・・③
   ＡＰ１状態：＝セットアップ済    ；            ・・・④
          ⋮

MESSAGE ＡＰＧ状態参照；ＡＰＧ番号                ・・・⑤
  TEMPORARY 参照結果：CHAR
  [IF  ＡＰＧ番号='1'  THEN
     参照結果：＝ＡＰ１状態；
          ⋮
     RETURN  参照結果          ]                ・・・⑥
```

アクタ“ホスト”はアプリケーション・グループの運用形態（コンカレント/ローカル）を管理しており，このアクタのみで参照可能な定数（“CONSTANT”で指定・・・①参照）で保持している．運用形態を処理に反映する必要がある他のアクタには，そのアクタに対するメッセージ（引き数）に運用形態を付加することにより（・・・③参照）行う．また，このアクタのみで参照・更新可能であり，かつその値が保持されるアクタ変数（・・・②参照）にて各アプリケーション・グループの状態を保持しており（・・・④），他のアクタはこのアクタに“APG状態参照”というメッセージをおくることによって（・・・⑤）アクタ変数により保持されているAPG状態を参照することができる．

図 7　アクタの作成例3

```
ACTOR ＡＰＧ
MESSAGE セットアップ：ＡＰＧ番号,
          運用形態    ：ＡＰＧ種別

          ①

MESSAGE ターミネーション：ＡＰＧ番号,
          運用形態          ：ＡＰＧ種別

          ②

MESSAGE リカバリ：ＡＰＧ番号,
          運用形態：ＡＰＧ種別

          ③
```

アクタ“APG”に“セットアップ，運用形態”のメッセージを送った場合の処理，“ターミネーション，運用形態”のメッセージを送った場合の処理，“リカバリ，運用形態”というメッセージを送った場合の処理を行わせる例である．

図 8　アクタの作成例4

1)　標準運用アクタのフレームワークを提供することにより，システム運用構築の負荷を軽減する．標準的なシステム運用の骨組みをシンボリック提供し，使用者は独自のシステム運用を組み込み，アクタを作成する．

2)　作成したアクタのテスト負荷を軽減する．システム運用のテストは，独占的なテスト環境を必要とする場合が多い．しかし，ほとんどのユーザでシステム運用のテスト環境は，本番環境または業務の開発環境と同居している．XOLはアクタを疑似実行させる機能を提供することにより，他の環境に影響を与えずにそのアクタの動作を検証することができる．

```
ACTOR    APG
CONSTANT
MESSAGE  セットアップ：APG番号,
         運用形態    ：APG種別
[IF APG種別='コンカレント' THEN
   ADD 'XIS$*USRJCL.CON$I/'&APG番号      …①
 ELSE
   ADD 'XIS$*USRJCL.LOCAL$I/'&APG番号    …②
                  ⋮
```

アクタ“APG”へのメッセージに含まれるAPG番号とAPG種別(コンカレントかそうでないか)の引き数によりアクタ“APG”が実行する(@ADDするJCL)が異なる．まず，IF～THEN～ELSEによる分岐として，この“APG”というアクタが受け取るメッセージ中に含まれる“APG種別”により処理の分岐が起こる．“APG種別”が‘コンカレント’の場合，上記①のJCL(@ADD XIS$*USRJCL.CON$I/APG番号)が実行され，‘コンカレント’でない場合には②のJCL(@ADD XIS$*USRJCL.LOCAL$I/APG番号)が実行される．それぞれのJCLエレメントのバージョン名の“APG”番号には，メッセージ中に含まれる“APG番号”が反映される．

たとえば，上記“APG”というアクタが受け取るメッセージが，“メッセージ種別＝コンカレント，APG番号＝1”の場合には，
@ADD XIS$*USRJCL.CON$I/1
が実行され，“メッセージ種別＝ローカル，APG番号＝2”の場合には
@ADD XIS$*USRJCL.LOCAL$I/2
が実行される．

図9　アクタの作成例5

3.4　実行環境での狙い

実行環境では，突発的に発生する障害の回復処理を自動化させることを狙っている．XISではオンライン・システムの運行に必要なシステム資源を監視しており，障害発生時には自動的に回復することができる．この自動化の制御構造を使用者も利用することができ，アクタと連動させることにより，使用者が監視から障害回復までのシステム運用構築が可能になる．この制御構造をイベント制御とよび，イベント駆動型のアクション実行を可能としている(図10)．

図10イベント・アクション構造

4.　XOLの実現方式

前章で述べてきた基本概念を軸としてXOLの言語仕様を決定した．アクタを記述するための言語仕様である．XOLで記述されたアクタは，JCLと補完プログラムの実行時パラメタとして翻訳され，JCLを実行することによりアクションを実現している(図11)．

4.1　言　語　仕　様

XOLは，左から右へ文字を連ねた行を上から下へ配し，2次元的に記述する．1行は最大132桁までで，文法上の行数の制限はない．また，行中の桁位置および改行には特別な意味をもたない．アクタは，図12で示す構成で記述する．右側は対応するアクタの記述例である．

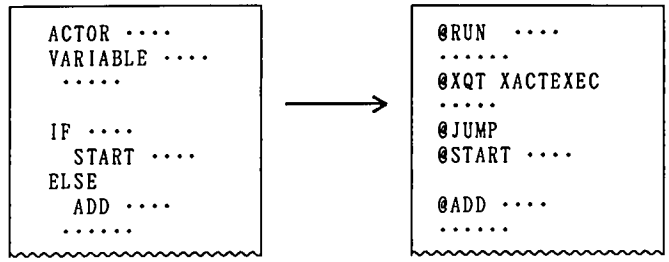


図 11 翻　　訳

①表題部			ACTOR xxxxxxxx
②データ宣言部	定数定義		CONSTANT c=0
	アクタ変数宣言		VARIABLE a: INTEGER
③スクリプト部	スクリプト	メッセージ定義	MESSAGE msg: b
		テンポラリ変数宣言	TEMPORARY d: INTEGER
		アクション	[IF a=b THEN d: =c ・・・

①表題部……表題部は，記述するアクタのアクタ識別子を定義する部である．アクタ識別子とはアクタを識別するためのもので，一意にならなければならない．
②データ宣言部……データ宣言部は，定数の定義とアクタ変数の使用宣言を行う部である．
③スクリプト部……スクリプト部は，アクションを記述する部である．スクリプト部には，スクリプトを1個～20個まで任意に記述できる．スクリプトには，メッセージ定義とアクションを記述する．アクタがメッセージを受けた際，スクリプトに記述されたアクションが実行される．また，必要に応じてテンポラリ変数の使用宣言を記述する．

図 12 XOL 記述構成

アクションを記述するための要素として次のものがある．

1) 定数と変数……定数と変数は，定義/宣言を行ったアクタ内においてのみ有効である．定数と変数は定義/宣言されるとデータ型をもつ．データ型とは定数，変数がもつ値の属性であり，XOL では整数型，実数型，文字型および論理型がある．定数は値をもち，定数を参照した場合にはその定数のもつ値が設定される．変数にはアクタ変数とテンポラリ変数がある．アクタ変数は，そのアクタ内のすべてのスクリプトから参照，更新可能な変数である．また，そのアクタに制御がない時でもアクタ変数の値は保持される．つまり，あるアクションにおいて設定されたアクタ変数の値を，次のアクションで参照すると前回に設定された値を得ることができる．テンポラリ変数は，テンポラリ変数宣言を行ったスクリプト内のみで参照/更新可能な変数である．また，そのアクションが終了すると，テンポラリ変数の値は保持されず，次にアクタが起動された時に前回に設定されたテンポラリ変数の値は保証されない(表1)．

2) 式……式には，算術式，文字式，論理式および関係式がある．各式の演算結果は表2の通りである．

3) 文……文には，表3に示すものがある．また，複数の文を“[“と”]”で囲むことにより，複合文として一つの文とすることができる．

以下に語彙規則と構文規則を記述する．

表 1 アクタ変数とテンポラリ変数

	有効範囲	存在期間
アクタ変数	全域	長期
テンポラリ変数	局所	短期

表 2 式と演算結果

式	意 味	被演算子	演算結果
算術式		整数型	整数型
+	加算	実数型	実数型
−	減算		
*	乗算		
/	除算		
//	整数除算		
文字式		文字型	文字型
&	結合	整数型	
論理式		論理型	論理型
OR	論理和		
XOR	排他論理和		
AND	論理積		
NOT（単項）	否定		
EQV	同値		
関係式		整数型	論理型
=	同等	実数型	
<	左辺小	文字型	
>	左辺大		
<>	不同等		
=<	左辺小または同等		
>=	左辺大または同等		

表 3 文

文	意 味
代入文	変数に値を代入する.
PACK 文	変数に空白を削除して圧縮した文字列を代入する.
IF 文	条件により処理を分岐する.
復帰型メッセージ文	他のアクタにメッセージ・パッシングする. 他のアクタのアクション終了後, 制御が戻される.
分岐型メッセージ文	他のアクタにメッセージ・パッシングする. メッセージ・パッシング後すぐに制御が戻され, アクタは並行してアクションを実行する.
復帰文	アクションの結果を, メッセージを送信したアクタに返す.
START 文	JCL エレメントを@START する.
ADD 文	JCL エレメントを@ADD する.
ELT 文	JCL エレメントを作成する.
WAIT 文	指定された時間, アクションを中断する.

表 4　語彙規則

語彙	説　　明
基本記号	基本記号は，それ自体が特定の意味をもつ記号であり，次の基本記号がある． + − * / // = < > <> =< >= ( ) [ ] { } . , := ; ' " : <- -> < = & ACTOR ADD AND CASE CHAR CLONE CONSTANT DO DOWNTO ELSE ELT END EQV EXCEPT EXCLUSIVE FALSE FOR IF INNER INPUT INTEGER MESSAGE NOT OF OR RANGE REAL RETURN START TEMPORARY THEN TO TRUE VARIABLE WAITWHILE XOR XACTSHRDATA XTCAPG
識別子	識別子は，英字で始まる英数字の列か日本語文字の列のいずれかである．識別子の長さ（文字数）は２０文字（日本語文字：８文字）までに制限される．
数値	数値には，整数と実数があり，１０進数表記を行う．整数は数字の列で，１０桁の数まで表現できる．実数は整数部と小数点と小数部からなり，小数部は数字の列である．整数部および小数部は３０桁の数まで表現できる．整数，実数とも，先頭に符号を置くことにより負の数値であることを表すことができる．符号が省略されている場合，正の数値として解釈される．
文字列	文字列は，前後を引用符にくくられた文字の列である．文字の列中に文字としての引用符を表現するには，引用符を連続して２文字表記する．この場合，１文字目の引用符はくくり記号として解釈されず，２文字で文字としての引用符１文字が解釈される．１つの文字列中に日本語文字と日本語文字以外の文字を混在して使用することはできない．
分離記号	分離記号には，空白，行末および注釈がある．注釈は二重引用符にくくられた文字の列である．文字列中の空白および注釈は分離記号とみなさない．文字列中に行末を入れてはいけない．

① 語彙規則……XOL プログラムは，英字，数字，特殊文字および日本語文字で記述される．文字列中の英字を除き，大文字と小文字の区別は行わない．語彙には表 4 に示すものがある．

② 構文規則……図 13 に XOL の代表的な構文を BNF (Buckus Normal Form) で記述する．

4.2 機 能 構 成

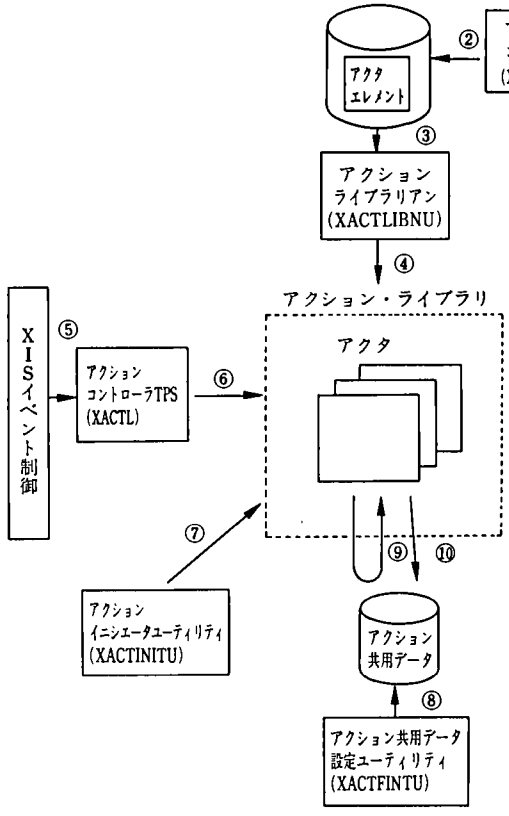
翻訳から実行までに必要なソフトウェアをアクション・ライブラリとしてまとめて提供している．アクション・ライブラリには以下の機能がある．

1) XOL 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，維持
3) アクタの登録，管理
4) アクタの実行を制御する実行時制御
5) イベント制御からのアクタの自動起動
6) ラン・ストリームからのアクタの起動
7) 使用者支援機能
   - 組み込みアクタ
   - アクション共用データ
   - フレームワーク
   - シミュレーティブ・アクタ

アクション・ライブラリのソフトウェア構成は図 14 に示す通りである．

```
〈プログラム〉 ::= ACTOR〈アクタ識別子〉〈本文〉
〈本文〉 ::=〈スクリプト部〉|〈データ宣言部〉〈スクリプト部〉
〈データ宣言部〉 ::=〈定数定義〉|〈変数宣言〉|〈定数定義〉〈変数宣言〉
〈定数定義〉 ::= CONSTANT〈定数定義リスト〉
〈定数定義リスト〉 ::=〈定数名〉=〈定数値〉
                    |〈定数定義リスト〉;〈定数名〉=〈定数値〉
〈変数宣言〉 ::= VARIABLE〈変数宣言リスト〉
〈変数宣言リスト〉 ::=〈変数リスト〉:〈変数型〉
                    |〈変数宣言リスト〉;〈変数リスト〉:〈変数型〉
〈変数リスト〉 ::=〈変数名〉|〈変数リスト〉,〈変数名〉
〈スクリプト部〉 ::=〈スクリプト〉|〈スクリプト部〉;〈スクリプト〉
〈スクリプト〉 ::= MESSAGE〈メッセージ定義〉〈アクション〉
                | EXCEPT〈アクション〉
〈メッセージ定義〉 ::=〈セレクタ〉|〈セレクタ〉:〈仮引数名〉
                    |〈メッセージ定義〉,〈セレクタ〉:〈仮引数名〉
〈アクション〉 ::=〈文〉| TEMPORARY〈変数宣言リスト〉〈文〉
〈文〉 ::=〈代入文〉
        |〈ＰＡＣＫ文〉
        |〈ＩＦ文〉
        |〈ＳＴＡＲＴ文〉
        |〈ＡＤＤ文〉
        |〈復帰型メッセージ文〉
        |〈分岐型メッセージ文〉
        |〈復帰文〉
        |〈ＥＬＴ文〉
        |〈ＷＡＩＴ文〉
        |〈複合文〉
〈代入文〉 ::=〈変数名〉:=〈算術式〉|〈変数名〉:=〈文字式〉
〈ＰＡＣＫ文〉 ::=/〈変数名〉/:=〈文字式〉
〈ＩＦ文〉 ::= IF〈論理式〉THEN〈文〉| IF〈論理式〉THEN〈文〉ELSE〈文
〈ＳＴＡＲＴ文〉 ::= START〈文字式〉
〈ＡＤＤ文〉 ::= ADD〈文字式〉
〈復帰型メッセージ文〉 ::=〈アクタ識別子〉<-〈メッセージ〉
                        |〈アクタ識別子〉<-〈メッセージ〉->〈変数名〉
〈分岐型メッセージ文〉 ::=〈アクタ識別子〉<=〈メッセージ〉
〈メッセージ〉 ::=〈セレクタ〉|〈セレクタ〉:〈実引数〉
                |〈メッセージ〉,〈セレクタ〉:〈実引数〉
〈復帰文〉 ::= RETURN〈文字式〉
〈ＥＬＴ文〉 ::= ELT〈文字式〉INPUT〈行イメージ〉END
〈行イメージ〉 ::=〈文字式〉|〈文字式〉,〈行イメージ〉
〈ＷＡＩＴ文〉 ::= WAIT〈算術式〉
〈複合文〉 ::= [〈複合文本体〉]
〈複合文本体〉 ::=〈文〉|〈複合文本体〉;〈文〉
〈算術式〉 ::=〈算術項〉|〈算術式〉〈加法演算子〉〈算術項〉
〈算術項〉 ::=〈算術因子〉|〈算術項〉〈乗法演算子〉〈算術因子〉
〈算術因子〉 ::=〈数値〉|〈変数名〉|(〈算術式〉)
〈論理式〉 ::=〈論理項〉|〈論理式〉〈加法論理演算子〉〈論理項〉
〈論理項〉 ::=〈論理因子〉|〈論理項〉〈乗法論理演算子〉〈論理因子〉
〈論理因子〉 ::=〈論理値〉|〈関係式〉|〈否定論理演算子〉〈論理因子〉
              |(〈論理式〉)
〈関係式〉 ::=〈算術式〉〈関係演算子〉〈算術式〉
            |〈文字式〉〈関係演算子〉〈文字式〉
            |〈論理式〉〈関係論理演算子〉〈論理式〉
〈文字式〉 ::=〈文字因子〉|〈文字式〉〈結合演算子〉〈文字因子〉
〈文字因子〉 ::=〈文字列〉|〈変数名〉
```

図 13 構文規則



① XOLで記述した原始プログラムをアクション・コンパイラに入力する.
② アクション・コンパイラの翻訳の結果をプログラム・ファイルに作成する.
③ アクション・ライブラリアンにアクタの登録を指示する.
④ アクション・ライブラリにアクタを登録する.
⑤ イベント制御がアクション・コントローラ TPS をパスオフする.
⑥ アクション・コントローラがイベントに対応したアクタを起動する.
⑦ アクション・イニシエータによりアクタを起動する.
⑧ アクション共用データ初期登録ユーティリティにより，アクション共用データを初期登録する.
⑨ 組み込みアクタ“XACTSHRDATA”によりアクション共用データを参照する.
⑩ 組み込みアクタ“XACTSHRDATA”によりアクション共用データを更新する.

図 14　アクション・ライブラリのソフトウェア構成

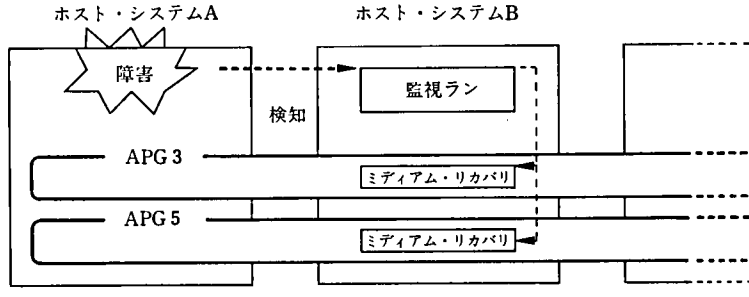


図 15　XTPA 回復処理例

```
ACTOR       ＨＯＳＴ

VARIABLE    ホスト識別名 , ホスト状況 : CHAR

MESSAGE     初期設定 : 自ホスト                                    ・・・①
[  ホスト識別名   :=  自ホスト ;
   ホスト状況     :=  'DOWN'                                  ]

MESSAGE     初期化                                             ・・・②
[                    .
                     .
   ＡＰＧ   <=  初期化 : 3 ;
   ＡＰＧ   <=  初期化 : 5 ;
                     .
                     .
   ホスト状況   :=  'DOWN'                                    ]

MESSAGE     開始                                               ・・・③
[  IF  ホスト状況 = 'UP'  THEN
     [  DISPLAY  'ホスト・システムは稼働中です。'  ]
   ELSE
     [               .
                     .
        ＡＰＧ   <=  開始 : 3 ;
        ＡＰＧ   <=  開始 : 5 ;
                     .
                     .
        ホスト状況   :=  'UP'                              ]   ]

MESSAGE     回復 : 障害ホスト                                  ・・・④
[  IF  障害ホスト = ホスト識別名   THEN                        ・・・⑤
     [               .
                     .
        ＡＰＧ   <=  回復 : 3 ;                                 ・・・⑥
        ＡＰＧ   <=  回復 : 5 ;
                     .
                     .                                  ]
   ELSE
     [               .
                     .
        ＡＰＧ   <=  回復 : 3 , ホスト : 障害ホスト ;           ・・・⑦
        ＡＰＧ   <=  回復 : 5 , ホスト : 障害ホスト ;
                     .
                     .                                  ]   ]

MESSAGE     終了                                               ・・・⑧
[                    .
                     .                                  ]
```

① HOSTアクタの初期設定を行うアクション．HOSTアクタのアクタ変数に初期値を代入する．HOSTアクタには，他に ②初期化 ③開始 ④回復 ⑧終了の4アクションがあり，①初期設定のみが環境設定時のアクションで，他は運用時のアクションである．

⑤ HOSTアクタが受信した'回復'メッセージのパラメタに指定されたホスト識別名が自ホストシステムのものであれば，自系回復処理を行う．そうでなければ他系回復処理を行う．この例では，他系回復処理のアクションが実行される．

⑥ ホスト・システム内のアプリケーション・グループ3，5の回復処理をAPGアクタに依頼する．

⑦ ホスト・システム内のアプリケーション・グループ3，5がコンカレント・アプリケーション・グループなのでホスト・システムA内のアプリケーション・グループ3，5の回復処理をAPGアクタに依頼する．

図 16　HOSTアクタ例

```
ACTOR       ＡＰＧ

VARIABLE    ＡＰＧ状況(16) : CHAR

MESSAGE     初期設定                                          ・・・①
[  ＡＰＧ状況     :=  'DOWN'                                  ] ;

MESSAGE     初期化 : ＡＰＧ番号                               ・・・②
[                    .
                     .
                     .
    ＡＰＧ状況(ＡＰＧ番号)  :=  'DOWN'                        ] ;

MESSAGE     開始 : ＡＰＧ番号                                 ・・・③
[  IF  ＡＰＧ状況(ＡＰＧ番号) = 'UP'  THEN
     [  DISPLAY  'ＡＰＧ' & ＡＰＧ番号 & 'は稼働中です。'  ]
   ELSE
     [               .
                     .
                     .
         ＡＰＧ状況(ＡＰＧ番号)  :=  'UP' ;          ]    ]

MESSAGE     回復 : ＡＰＧ番号                                 ・・・④
[                    .
                     .
    ADD 'XIS$*XISJCL.IRU-SHORT/' & ＡＰＧ番号  ;              ・・・⑤
                     .
                     .                                 ]

MESSAGE     回復 : ＡＰＧ番号 , ホスト : ホスト識別名         ・・・⑥
 [                   .
                     .
     ADD 'XIS$*XISJCL.IRU-MEDIUM/' & ホスト識別名 & ＡＰＧ番号 ;⑦
                     .
                     .                                 ]    ]

MESSAGE     終了 : ＡＰＧ番号                                 ・・・⑧
[                    .
                     .
                     .                                      ]
```

① APGアクタの初期設定を行うアクション．APGアクタのアクタ変数に初期値を代入する．APGアクタには，他に ②初期化 ③開始 ④回復 ⑥回復，ホスト ⑧終了の5アクションがあり，①初期設定のみが環境設定時のアクションで，他は運用時のアクションである．

⑤ あらかじめ作成されているIRU 1100のショート・リカバリ JCLをラン・ストリームに挿入（@ADD）して実行する．

⑥ ④のメッセージ定義と似ているが，XOLでは別メッセージとして扱うことができる．ホスト指定が無い場合，自ホスト・システム内のアプリケーション・グループの回復処理(④)を行い，ホスト指定された場合，そのホスト・システム内のアプリケーション・グループの回復処理を行う．

⑦ あらかじめ作成されているIRU 1100のミディアム・リカバリ JCLをラン・ストリームに挿入（@ADD）して実行する．ミディアム・リカバリ JCLは，ホスト・システムごとに異なるため，JCLエレメントのバージョン名で管理している例である．

図 17　APGアクタ例

5. XOLでのシステム運用記述

実際のシステム運用をXOLで記述してみよう．XTPA形態のマルチ・ホスト・システムでのホスト・システム障害回復運用を中心に例示する．図15にあるようにマルチ・ホスト・システム上にコンカレント・アプリケーション・グループ3,5の2種のオンライン・システムが稼働していた時にホスト・システムAが障害になり，ホスト・システムBで回復処理を実行する運用の記述例である．XTPAの回復処理については本特集号別稿の“XISのXTPAシステム運用”に詳述されているので，IRUのミディアム・リカバリを実行させる部分をクローズアップして記述する．

ホスト・システムB上のXIS監視ランがホスト・システムAの障害を検知するとホスト・システム運用アクタ‘HOST’に次のメッセージをメッセージ・パッシングする．

回復：A

図16はホスト・システム運用アクタ‘HOST’の記述例である．

ホスト・システム運用アクタ‘HOST’はホスト固有な回復処理を行い，アプリケーション・グループの回復処理はアプリケーション・グループ運用アクタ‘APG’に依頼する．アプリケーション・グループ運用アクタ'APG'に次のメッセージをメッセージ・パッシングする（アプリケーション・グループ3の例）．

回復：3，ホスト：A

図17はアプリケーション・グループ運用アクタ‘APG’の記述例である．

6. おわりに

JCLの記述能力に疑問を抱いたところから始まったXOLの開発は，その初期目標を達成できたと確信する．XOLの翻訳結果がJCLであるため，実行時の見地からは，従来のシステム運用ソフトウェアの例外ではないともいえるが，複雑化したシステム運用を記述する能力の向上は明らかである．生産性についての定量的な評価は未だ行われていないので，今後の活動の一環としてデータを収集していきたい．表5は，シ

表5　XOLの機能の充足度

機能	機能要件	充足度
記述	データ処理	○
	日本語	○
	部品化	○
	カプセル化	○
	JCLインタフェース	○
翻訳	診断メッセージ	○
	相互参照表	○
	目的コード・リスト	○
	最適化	×
環境	構築・維持	○
	照会	○
	開発支援	×
	テスト支援	○
実行	効率	△
	動的連結	○
	障害診断	△

ステム運用記述言語の機能要件に対する現在の充足度を表している．

今後の課題としては，標準的なオンライン・システム運用のフレームワークをそろえ提供することがあげられる．また，最適化や静的連結による効率改善も必要と考えている．

XOL は，現在数ユーザで使用されている．本稿により，ユーザが XOL をさらに積極的に，より効果的に使用することができるのであれば，本稿の目的を遂げたといえるであろう．われわれが提供したソフトウェアの価値判断は常にユーザによりされるべきものであり，ユーザからの改造要求を反省の糧となし，より良きソフトウェアの提供に務めたい．XOL に対して忌憚なき意見をいただければ幸いである．

参考文献 ［1］ 沢田 啓，“XTPA に基づくノーダウンシステム”，ユニシス技報，Vol. 13，No. 4，1994 年 2 月，pp. 102～118.
［2］ 黒川 茂，“アプリケーション連動の仕組み”，ユニシス技報，Vol. 13，No. 4，1994 年 2 月，pp. 167～178.

執筆者紹介 米 津 政 紀 (Masanori Yonetsu)
1983 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．システム効率コンサルティング，ホット・スタンバイ・システム構築支援，分散データ・アクセス・ソフトウェアの開発を経て，XIS の開発保守サービス業務に従事．現在システム企画開発二部オンライン技術課に所属．情報処理学会会員．

徳 永 路 晴 (Michiharu Tokunaga)
1989 年横浜国立大学教育学部社会学専攻課程卒業．同年日本ユニシス(株)入社．統合オンライン・ソフトウェア XIS の開発を経て，1994 年 4 月より統合運用システム IOF の開発に従事．現在，システム企画開発二部運用開発課に所属．

XISのXTPAシステム運用

XTPA System Operation by XIS

竹　内　　久

要　約　XTPA (eXtended Transaction Processing Architecture) システム環境では，新しいソフトウェア・アーキテクチャとハードウェア・アーキテクチャとによって，システムの能力とシステムの可用性が増大する．また，複数ホストの使用者プログラムに，TIP(Transaction Interface Package) および汎用データベース・システム(UDS(Universal Data System))のデータベースを共用させて，システムが処理できるトランザクション量を増大させることから，複数ホストを1システムとして開始，終了，回復などのシステム運用に柔軟に対応しなければならない．

本稿では，このXTPAシステム環境のもとで容易かつ柔軟にシステムを運用していくためのXIS (eXtended Information System) 運用機能，およびノーダウン・システムの実現方式について言及している．

該機能の適用により複数ホスト・システムから構成される大容量システムの容易な運用が可能となり，さらには従来のホットスタンバイ・システムと比較して，耐障害性および可用性のより高いノーダウン・システムの構築が可能となった．

Abstract　In the XTPA (eXtended Transaction Processing Architecture) system environment, its new software and hardware architectures are empowered to augment systems capabilities and availability. Also, in there, inevitable is flexibility in system operations including system initialization, termination and recovery where a multi-host system is treated as only a single system because of the allowed shared access by user application programs running on multiple hosts to a TIP (Transaction Interface Packge) data base and/or a UDS (Universal Data System) data base.

This paper describes both the functionalities provided by XIS to make possible easy, flexible system operation in the XTPA environment, and how to build non-stop systems. The use of XIS functions makes it feasible to easily operate high-volume systems consisting of more than one host, and also to create non-stop systems with greater fault tolerance and higher availability than former hot-standby systems.

1. はじめに

当社では拡張トランザクション処理体系，すなわちXTPAを提供している．XTPAは，最大4台のホストを疎結合し，RLP (Record Lock Processor) のもとに，複数ホスト・システムの使用者プログラムにTIPおよびUDSのデータベース・ファイルを共用させて，システムが処理できるトランザクション量を増大させる．

複数ホスト・システム化によって処理トランザクション量を増加させることができるが，その一方では，ホスト・システム台数の増加に伴ってシステム運用がさらに複雑化することが予想される．

XTPA構成からなるシステム（以降，XTPAシステム）では，複数ホスト・システ

ムを一つのシステムとして位置づけることが必要となる．XISでは，使用者に対して複数ホスト・システムをできるだけ意識させずに1システムとして運用させる機能を提供している．

また，オンライン・トランザクション処理のアプリケーションでは，コンピュータは高度なアベイラビリティ*，仮想的にはまったく故障しないシステムが要求される．すなわち，システムのある部分は故障するかもしれないが，システムの他の部分は故障に耐えてサービスを提供し続けるようなシステムである．

従来，ホスト・システムの障害（フォールト**）に対するフォールト・トレランス***技法としてホット・スタンバイ・システムが採用されてきた．しかし，この方式でもシステムは，リカバリ処理が完了するまでアプリケーションに対するサービスの提供が一定時間停止する．

XTPAシステムでは，1ホスト・システムの障害はシステムの部分障害であり，他の健全ホスト・システムでアプリケーション実行を継続することができる．このことによって，よりフォールト・トレランスの高いノーダウン・システムの構築が可能となった．

本稿では，使用者が効果的なXTPA構成のシステムを容易に構築/運用できるように，当社が開発したXISの運用支援機能が提供するXTPA対応機能のうち，システム運用およびノーダウン・システムの実現方式について報告する．

2. XTPAシステム

本章では，XTPAシステムの機能およびその利点を簡単に記述する．

2.1 XTPAの機能

データをホスト間で共用し，かつ矛盾なく更新するために，XTPAでは新しく複数ホスト・システムにまたがる共用アプリケーション・グループ（コンカンレントAPG）が提供された．これまでは，アプリケーション・グループ（APG）は一度に1台のホスト・システム上でしか動作しなかった．XTPAのコンカレントAPGは，XTPAを構成しているすべてのホスト・システム上に生成され，すべてのホスト・システム上で同時に作動する．コンカンレントAPGは，そのデータベースを共用ディスク上に持ち，すべてのホスト・システムがデータベースを同時にアクセスできる．

XTPAシステムを実現するためには，以下のハードウェアおよびソフトウェアが必要である．

図1にXTPAシステムの全体構成を示す．

■ハードウェア

RLP

UTC（Universal Time Coordination：複数ホストの時刻合わせを行う）

HLC（Host Lan Controller：マルチホスト・ファイルシェアリング機能を実現するためのホスト間通信機能を提供）

■ソフトウェア

---

* ある期間中に機能を維持する時間の割合（JIS Z 8115参照）

** 誤りもしくは障害となりうるシステムの状態および条件．

*** フォールトの存在にもかかわらず，外部から見る限りあらかじめ定められた状態を維持するようなシステムの能力．

MHFSF 1100 II (Multi Host File Sharing Facility for UNISYS 1100 series II)
XTC-TIP (eXtended Transaction Capacity-Transaction Interface Package)
XTC-UDS (eXtended Transaction Capacity-Universal Data System)
DP-M 1 TH (Dual Session)

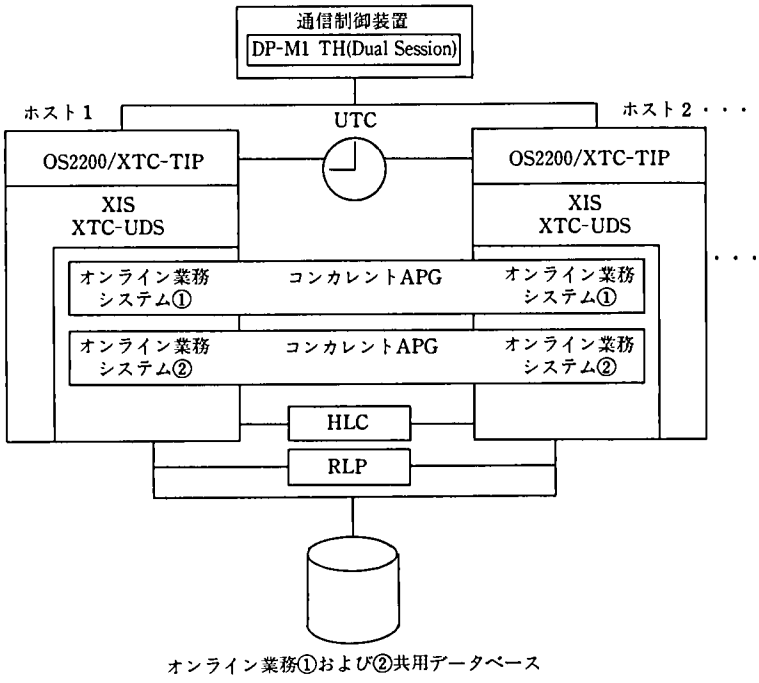


図 1　XTPA システムの全体構成

2.2 XTPA の利点

XTPA 構成の特定のホスト・システムにあるコンカレント APG のことを APG/ホストと呼ぶ．コンカレント APG は，複数の APG/ホストから構成される．

同一アプリケーションを複数のホスト・システムで共用できるため，大容量のトランザクション・システムを構築することができる．また，長いスパンでの業務トランザクション量の増大に対して，ホスト・システムの追加により最大 4 台構成までの XTPA システムとすることができ，業務システムの成長に対して柔軟に対応することができる．

その他にも，日々のトランザクション量に応じて稼働させるホスト・システムの台数を 1〜4 台の範囲で自由に選択でき，従来のシングル・ホスト・システムより柔軟性が高い．

ところで，コンカレント APG の一つの APG/ホストに障害が発生した場合，他の APG/ホストは，障害 APG/ホストが障害発生時にロックしていたデータベースのレコードにアクセスできない．しかし，データベースのその他の部分は，残りの APG/ホストによってアクセスと更新ができる．障害 APG/ホストの RLP ロックは，その APG/ホストのミディアム・リカバリ（詳細は，4.2 節参照のこと）終了時に解除され

る．ミディアム・リカバリは，IRU(Integrated Recovery Utility)を使用して健全ホストから実行するコンカレント APG の回復方法の一つである．

このことにより，あるホスト・システムに障害が発生しても，他の健全ホスト・システムにより処理を続行することが可能となった．

3. XIS による XTPA システムの運用

コンカレント APG は，複数のホスト・システムを共用して使用者業務が稼働する環境であり，使用者プログラムは，どのホスト・システム上で実行されてもよい．そのため，使用者データベースは，どのホスト・システムからもアクセス可能なように共用ファイルとして配置される．一方，ある 1 台のホスト・システムに着目すると，全 APG のプログラムがアクセスするテーブルや TIP の VINDEX（Validity INDEX）のように，APG 間で共用される資源も存在する．また，コモンバンクのような資源は，ホスト・システムごとに存在する．XTPA システムでは，このようなシステムを構成する資源が複雑に関係しており，それぞれの状態を管理しなければならない．

また，障害回復運用についても APG およびホスト・システムごとにその障害状況を判断し，関連する APG およびホスト・システムと連携をとって適切な回復処理を行う必要がある．

このように XTPA システムのシステム運用（開始/終了/回復）は，シングル・ホスト・システムに比較して複雑かつ困難となる．XIS では，複数ホスト・システムから構成される XTPA システムをあたかも 1 システムとして運用できるような機能を提供している．本章では，XIS による XTPA システム運用の実現方法を記述する．

3.1 XTPA 環境での運用構成

XIS では，XTPA システムを構成する資源を図 2 に示した単位でとらえ，各々の単位の状態を管理することによって，XTPA システム全体の運用を制御している．

1) ホスト共通/APG 共通……全ホストに共通，かつ全 APG に共通の資源であり，XTPA システムが稼働する場合には，必ずセットアップされていなければならない部分である．
2) ホスト固有/APG 共通……1 ホスト内で固有，かつ全 APG に共通の資源であり，あるホストが稼働する場合には，必ずセットアップされていなければならない部分である．
3) ホスト共通/APG 固有……全ホスト内に共通，かつ 1 APG に固有である資源であり，あるコンカレント APG が稼働する場合には，必ずセットアップされていなければならない部分である．
4) ホスト固有/APG 固有……1 ホスト内で固有，かつ 1 APG に固有である資源であり，業務があるホストのある APG で稼働する場合に，その APG/ホストで必ずセットアップされていなければならない部分である．

上記の運用単位を構成する要素を表 1 に示す．

3.2 XIS の開始/終了

XIS では，これらの運用単位から構成されているシステムの運用を行うため，表 2 に示す運用 JCL を提供している．

表1　運用単位の構成要素

		ホスト・システム	
		共　通	固　有
APG	共通	使用者テーブル(FCSS, TCDBF) XIS テーブル(FCSS, TCDBF)	コモンバンク・テーブル TIP の VINDEX, FLAGBOX 常駐ジョブ(XIS モニタ)
	固有	使用者データベース(DMS, RDMS) 使用者テーブル(FCSS, TCDBF) XIS テーブル(FCSS, TCDBF)	コモンバンク・テーブル XIS テーブル(AIS メモリ・ファイル) 常駐ジョブ(XIS タイマ, MCB 等)

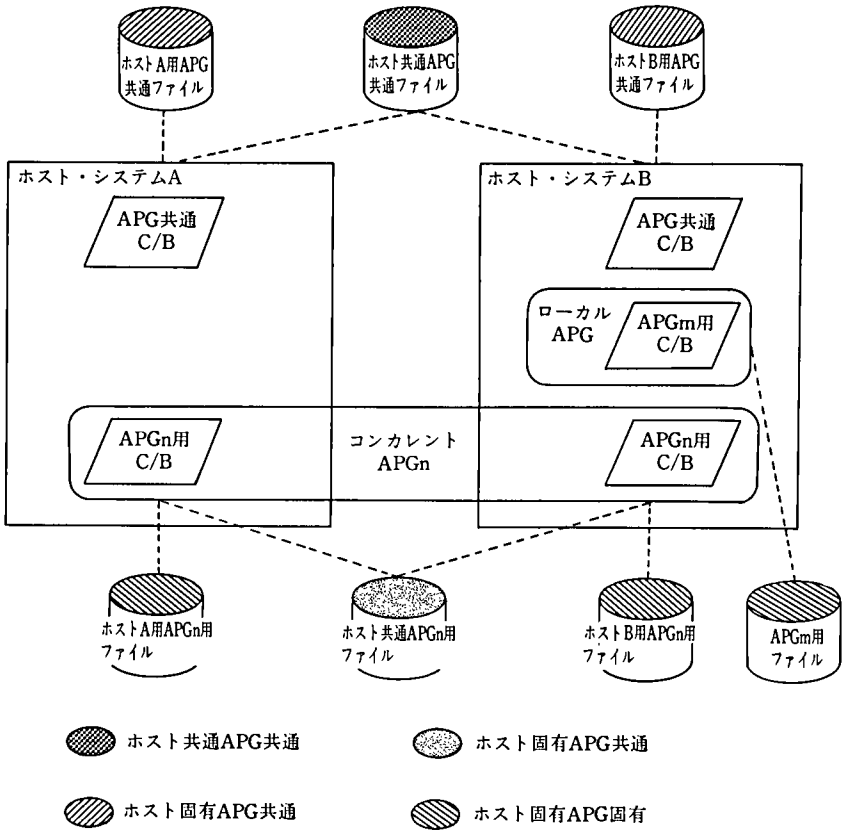


図 2　XTPA システムの運用単位

これらの JCL は，ホスト固有処理はホストの台数分だけ存在し，各々のホストのホスト識別名でジョブ名が修飾される．JCL 名はジョブ名と同一である．また，APG 固有処理は APG の個数分だけ存在し，ジョブ名の 2 桁目の数字が APG 番号を表している．APG 共通処理のジョブ名の 2 桁目は 'Z' で表す．

ホスト共通の開始処理は，最初に立ち上がったホスト・システムでのみ実行される必要があり，終了処理は最後に終了するホスト・システムでのみ実行されなければならない．

同様に，APG 共通の開始処理は最初に立ち上がった APG でのみ実行される必要があり，終了処理は最後に終了する APG でのみ実行されなければならない．また，ホス

表2 XIS 運用 JCL

運用単位	ジョブ名	処 理 内 容
ホスト共通 APG 共通処理	XZCNS	ホスト共通 APG 共通の XTPA システム開始処理
	XZCNZ	ホスト共通 APG 共通の XTPA システムの MHFS 初期化後回復処理
	XZCNT	ホスト共通 APG 共通の XTPA システム終了処理
ホスト固有 APG 共通処理	XZCHS	APG 共通のホスト・システム開始処理
	XZCHR	APG 共通のホスト・システムの回復ブート後回復処理
	XZCHZ	APG 共通のホスト・システムの初期ブート後回復処理
	XZCHT	APG 共通のホスト・システム終了処理
ホスト共通 APG 固有処理	XnSIC	コンカレント APG のホスト共用部分開始処理
	XnSZC	コンカレント APG のホスト共用部分 MHFS 初期化後回復処理
	XnTMC	コンカレント APG のホスト共用部分終了処理
ホスト固有 APG 固有処理	XnSIL	コンカレント APG のホスト非共用部分開始処理
	XnSRL	コンカレント APG のホスト非共用部分回復ブート後回復処理
	XnSZL	コンカレント APG のホスト非共用部分初期ブート後回復処理
	XnCSR	コンカレント APG の APG ダウン後自系回復処理
	XnCMR	コンカレント APG のミディアム・リカバリ処理
	XnTML	コンカレント APG のホスト非共用部分終了処理

n：APG 番号

表3 システム運用インタフェース

ジョブ名	ジョブの処理内容
XISN	ホスト内の特定 APG 開始処理起動
XAPn	APG 開始処理起動
XIMn	APG 終了処理起動

n：APG 番号

ト・システムに障害が発生したのかどうかなど，種々のシステム状況を判定して適切なジョブを選択して起動する必要がある．

しかし，使用者がシステム状況を調べ，それに合わせてこれらの JCL を的確に起動することは非常に困難であり，かつ誤操作を招く可能性が非常に高い．XIS は，これらの制御を行う機能を提供しており，使用者は，ただ「この APG を立ち上げる」，「このホストを立ち上げる」，あるいは「この APG を終了させる」という非常に単純化された形態でシステム運用を行うことができる．このように，使用者判断の必要性を少なくしているので，操作ミスが少なくなり，的確かつ迅速にシステムを運用することが可能となっている．

XIS は，表3に示すシステム運用インタフェースを提供している．これらのジョブの処理を行いたいホスト・システムにて起動することによって，容易に XTPA システムを運用することができる．

このような単純な使用者インタフェースは，システム稼働状況自動判定機能および運用処理同期機能により実現されている．

3.2.1 システム稼働状況自動判定機能

XISが提供している各運用単位に対するJCLは，その時のシステムの状態によって各種存在し（表2参照），必要に応じて選択され実行される必要がある．

XISでは，下記に示したシステムの状態をシステム稼働情報ファイルに一元管理している．

- APG/ホストの運用状況（未稼働/開始処理中/稼働中/終了処理中）
- APG/ホストの稼働中運用ジョブ
- ブートフラグ（直近の開始/終了処理以降ブートされたか否か）
- 直近に行われたブートのブート種類（初期ブート/回復ブート）
- ブート時刻
- MH IN* フラグ(直近のホスト共通APG共通開始処理終了後，MHFSが初期化されたか否か)
- MH IN時刻

システム稼働情報ファイルは，全ホストから参照/更新が行えるよう共用EXECファイルに配置されている．参照/更新はそれぞれ，ER IOW$のファンクションRDL$およびW$を使用して排他制御を行っている．

一方，ホスト識別情報はホストごとの非共用EXECファイルに配置されたホスト識別情報ファイルにて管理している．ホスト識別情報ファイルにより，ジョブが起動されたホスト・システムを自動的に判定し，JCLのジョブ名（ホスト識別名）を決定する．

表4　ブート後の開始JCL

システム稼働状況		起動ジョブ 初期ブート	起動ジョブ 回復ブート
そのホストのAPGがすべて未稼働状態の場合		XnSIL	XnSIL
そのホストに異常終了のAPGがある場合	XnSIL実行中	XnSIL	XnSIL
	XnSRL実行中	XnSZL	XnSRL
	XnSZL実行中	XnSZL	XnSZL
	XnTML実行中	XnSZL	XnSZL
	稼働中	XnSZL	XnSRL
	未稼働	—	—

表5　ノン・ブート時の開始JCL

システム稼働状況			起動ジョブ
そのホストのAPGがすべて未稼働状態の場合			XnSIL
そのホストに異常終了のAPGがある場合	XnSIL実行中		XnSIL
	XnSRL実行中		XnSRL
	XnSZL実行中		XnSZL
	XnTML実行中		XnSZL
	稼働中	MCB：UP	—
		MCB：DN	XnCSR
	未稼働	—	—

* MH IN：MHFS（ファイル・シェア）環境の初期化を行うキーイン

●ホスト識別情報（ホスト識別名，MHFS ホスト識別名，MHFS ホスト指標）

使用者が APG 開始起動処理を起動した場合，起動 JCL（XISN, XAPn）中の XIS 開始ジョブ起動ユーティリティ（XSTUKICKV）は，表 4 および表 5 に示した条件に従って APG/ホストの開始ジョブを決定し起動する．

3.2.2 運用処理同期機能

ホスト共通/APG 共通開始処理は，システムの初期開始時に最初の APG/ホストの開始を行う時にのみ，一回だけ実行される必要がある．同様にその終了処理は，最後の APG/ホストの終了処理を行う時のみ実行されなければならない．また，あるタスクの実行において，別のジョブの特定のタスクの終了が前提条件となる場合がある．

これらの判定および同期取りを行う機能が運用処理同期機能であり，JCL 中に組み込まれている同期ユーティリティ（XSTUCTRLV）が行っている．

運用処理同期機能は，以下の機能から構成されている．

- ●ジョブのチェックポイントを採取する．
- ●ジョブのチェックポイントを参照し，指定されたチェックポイントが採取されている場合，指定されたラベルにスキップする．
- ●他の特定ジョブのチェックポイントを待ち合わせ，指定されたラベルにスキップする．
- ●運用単位の開始処理/終了処理状況を参照し，指定されたラベルにスキップする．

これらは，システム稼働情報ファイルに運用単位ごとに状況を管理することによって行っている．運用処理同期機能がシステム稼働情報ファイルに管理している情報は，表 6 のとおりである．

表 6　システム稼働情報

		ホストシステム	
		共　通	固 有 × 8
APG	共通	直近の MH IN 時刻 実行中ジョブの種類 開始処理状況 終了処理状況 ジョブごとのチェックポイント情報	ブート時刻 ブート種別 実行中ジョブの種類 開始処理状況 終了処理状況 ジョブごとのチェックポイント情報
	固有×32	MH IN 実行フラグ 実行中ジョブの種類 開始処理状況 終了処理状況 ジョブごとのチェックポイント情報	APG 種別(コンカレント，ローカル) ブートフラグ 実行中ジョブの種類 開始処理状況 終了処理状況 ジョブごとのチェックポイント情報

3.2.3 ファイル障害対応

ホスト識別情報ファイルおよびシステム稼働情報ファイルは，ファイル障害に備えて物理的に 2 重化しており，片系障害時には正常系を使用して運用可能である．

また，障害系を再カタログし，正常系から複写することにより，簡単に障害復旧が可能である．

4. XTPAに基づくXISのノーダウン・システム

XISは，XTPAの機能を使用してノーダウン・システムの実現をはかった．本章ではXTPAに基づいたノーダウン・システムの機能および実現方式について記述する．

4.1 従来のリカバリ・システム

従来，ホスト・システムの障害に対するフォールト・トレランス技法としてホットスタンバイ・システムが採用されてきた．ホットスタンバイ・システムとは，二つ以上の独立したホスト・システムから構成され，あるホスト・システムが障害の時に自動的に他方のホスト・システムに切り換えて稼働させるシステムであり，ホスト・システムのハードウェア冗長構成をもとに構築されたホットスタンバイ・スペアリング技法の一形態である．

ホットスタンバイ・システムの形態は，次のように分類される．

1) 待機型……待機型は，ホットスタンバイ・システムの基本型であり，I/O機器を本番系ホストに切り換えて稼働させる方式である(図3)．

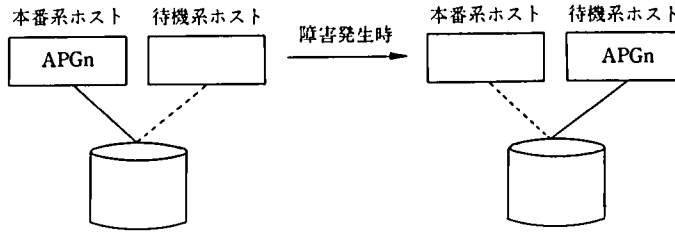


図3　待機型ホットスタンバイ・システム

2) 切り換え型……切り換え時にリブートを介在させる．本番系に接続されていたI/O機器をすべて切り換え系ホストに切り換えて，再開始を行う方式である(図4)．

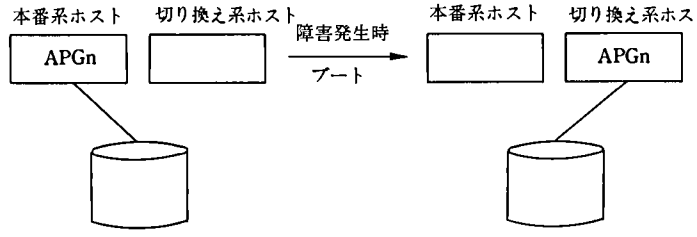


図4　切り換え型ホットスタンバイ・システム

当社では両形態を提供しているが，一般的に切り換え型ホットスタンバイ方式の方が待機型に比較してアプリケーションの停止時間が長い．

待機型ホットスタンバイ・システム（以降単純に'ホットスタンバイ・システム'と記述する）を実現するためには，待機用のホスト・システムの他に以下のハードウェアおよびソフトウェアが必要である．

●ハードウェア

ASU（Automatic Switching Unit：自動システム切り換え装置）

●ソフトウェア

SAFE 1100 (System for Automated Fail-safe Environment: ノーダウン支援ソフトウェア・パッケージ)

MHFSF 1100 II

AIS 1100 II (Advanced Information System for UNISYS series 1100 II: 統合オンライン支援システム)

本章では，従来のフォールト・トレランス方式である待機型ホットスタンバイ・システムについて，その実現方式および問題点について記述する．

4.1.1 ホットスタンバイ方式の実現方法

ホットスタンバイ・システムは，以下の機能から成り立っている(図5)．

1) ホスト・システム障害自動検知

ホスト・システム障害の自動検知には，次の形態がある．

① 自系障害監視

重要ジョブ障害，キュー・パニック等を監視し，ホットスタンバイ・システムを作動させてホストを切り換える．

② 他系障害監視

本番系システムから一定時間間隔で送信されてくるハートビートを待機系システムで監視し，ハートビートが一定時間途切れた場合に，本番系システム障害とみなして切り換え処理を行う．

2) ハードウェア機器の自動切り換え

SAFE 1100 が ASU に対して切り換え指令を行い，ASU は各ハードウェア機器を本番系へと切り換える．

3) 回復処理の自動化

本番系ホストから待機系ホストへの機器の切り換えが完了すると，待機系の SAFE 1100 は回復ジョブを起動し，回復処理を行う．

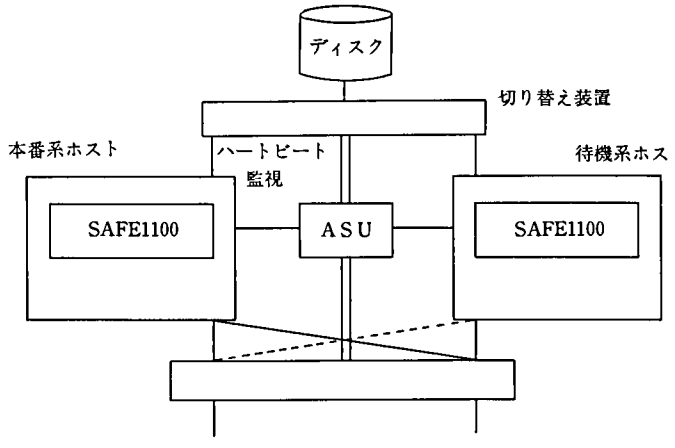


図 5 ホットスタンバイ・システムの全体構成

4.1.2 ホットスタンバイ方式の問題点

ホットスタンバイ方式では，ホスト・システムに障害が発生してからアプリケーシ

ョンが再開できるまで，以下の処理に必要な時間の総和がかかる．

① 障害を検知するまでの時間

② 機器切り換え処理にかかる時間

③ 回復処理の起動および処理にかかる時間

これは，ホットスタンバイ方式の原理上避けらない．

これまで，上記の時間を短縮させるために種々の技術が考案・開発されてきた．しかし，現実的には約1分のアプリケーションの停止が発生する．アプリケーションを実行しているのが1台のホスト・システムであるために，ホスト・システム障害はシステム全体障害となるからである．

すなわち，ホットスタンバイ方式では，ホスト・システム障害に対するフォールト・トレランスという課題に対して，

「障害発生からリカバリ終了まで，アプリケーションの続行が完全に不可能となる．」

という問題点がある．

また，切り換えを物理的に行うためハードウェアとしてASUが必要となる．

4.2 XTPAに基づくノーダウン・システムの実現方法

前述のように，XTPAシステムではその大きな特徴の一つとして，同一APGが複数のホスト・システムで稼働し，あるホスト・システムが障害になった場合でも他の健全ホスト・システムでは業務処理が続行できるということがあげられる．これは，従来のホットスタンバイ方式に代わりうるホスト・システム障害に対するリダンダンシ確保の方法である．

一方，ホスト・システム障害発生時には，障害ホスト・システム上の稼働プログラムが保持していたデータベースのロックは解放されず，健全ホスト・システムから該領域は参照不能となる．

したがって，XTPAシステムではホスト・システム障害を迅速に検知し，ミディアム・リカバリを実行し，障害ホスト・システムが保持しているロックを解除し，障害ホストを切り離す必要がある．

XISでは，ホスト・システム障害に対し，フォールトの検出，位置決定，隔離，そしてリカバリを行う機能を提供しており，これをXISミディアム・リカバリ機能と呼ぶ．XISミディアム・リカバリ機能の機能構成は，図6のとおりである．

XISミディアム・リカバリ機能は，以下の機能から成り立っている．

- ホスト・システム稼働状況監視機能
- ミディアム・リカバリ起動判定機能

4.2.1 ホスト・システム稼働状況監視機能

ホスト・システム稼働状況監視はXTCモニタが行う．XTCモニタは，システム運用APGの常駐ジョブであるXISタイマの一機能である．被監視ホスト・システム上に常駐するXISタイマのタイムスタンプ・ライト・アクティビティは，共用FCSSファイル上のタイムスタンプを一定時間ごとに更新している．一方，監視ホスト・システム上に常駐するXISタイマのタイムスタンプ・リード・アクティビティは，共用FCSSファイル上のタイムスタンプを一定時間ごとに参照している．各々のアクティビティは，リアルタイム・レベルで稼働させている(図7)．

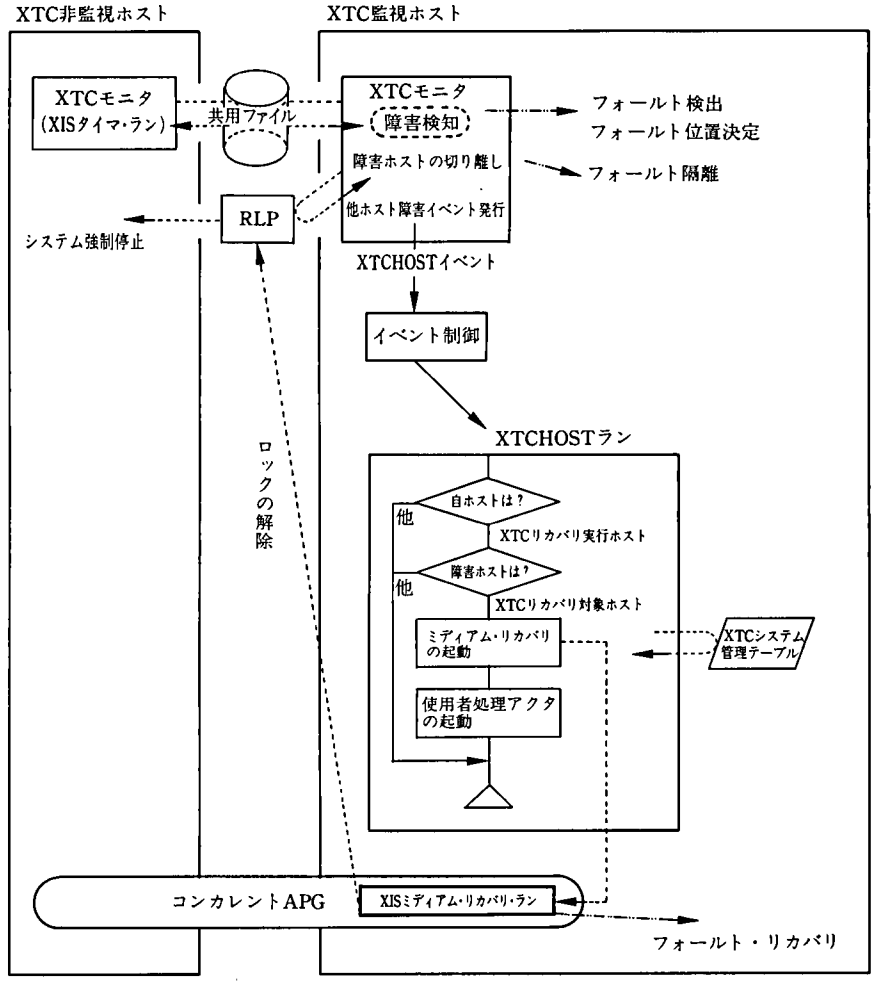


図 6　XIS ミディアム・リカバリの機能構成

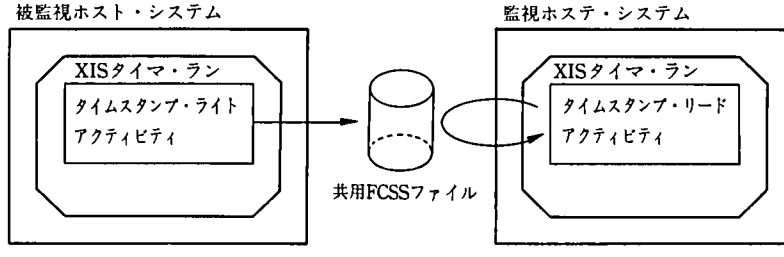


図 7　ホスト・システム稼働状況監視（XTC モニタ）

ホスト・システムの稼働状況監視を，通信回線を使用したハートビート方式でなく，共用 FCSS ファイルを使用したタイムスタンプ方式を採用した理由は以下のとおりである．

1)　以下の方法により，リダンダンシが簡単に確保される．

①　TIP 2 重化ファイル（対ファイル障害，ディスク障害，CU 障害）

② I/O チャネル多重化（対 I/O チャネル障害）

2) 信頼性が高い．

① 上記リダンダンシが確保されている場合，全体障害の場合以外，FCSS ファイルに対する I/O は失敗しないと考えられる．

② リアルタイムレベルのアクティビティだけで機能が実現されているため，ホスト・システムのスロー・ダウンや他の障害に影響されにくい．

被監視ホスト側のホスト・システム稼働状況監視がタイムスタンプを一定時間ごとに更新し，監視ホスト側のホスト・システム稼働状況監視が被監視ホストのタイムスタンプを一定時間ごとに参照することによって，タイムスタンプの更新，未更新からホストの稼働状況（障害の有無）を検知する(図 8)．

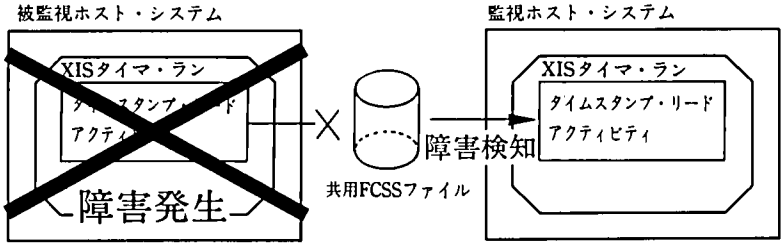


図 8　ホスト・システム障害の検知

なお，XTPA システム内の各ホストは，相互に被監視ホスト/監視ホストの関係になっており，あるホスト・システムが障害になった場合，稼働中の全ホスト・システムで検知が可能である．

タイムスタンプ処理間隔，タイムスタンプ監視処理間隔，タイムスタンプ未更新許容回数は設定を変更することが可能である．

障害発生から障害を検知するまでの時間 T は，タイムスタンプの参照間隔を α，障害発生時からタイムスタンプを参照するまでの時間差を β，タイムスタンプの未更新許容回数を n とした場合，次式で示される．

$$T=\alpha*(n+1)+\beta \quad (0<\beta\leqq\alpha)$$

例を図 9 に示す．

4.2.2 障害発生ホスト・システムの強制切り離し

被監視ホスト・システムの障害を検知した場合，障害と判定したホスト・システムがスローダウンしている可能性があり，この状態で回復処理を行うとディスク上のデータが破壊される可能性がある．このためホットスタンバイ方式では，ASU を切り換えることにより，物理的に障害ホストを切り離し，データの破壊を防いでいる．XTPA システムでは，EXEC に対する障害ホストの強制切り離し命令を実行することによって，上記問題に対処している．

EXEC のホスト強制切り離しでは，以下の処理を行う．

① 障害ホスト・システムがスローダウン等で停止しきっていない場合を考慮し，RLP 経由で障害発生ホスト・システムを強制的に停止させる．

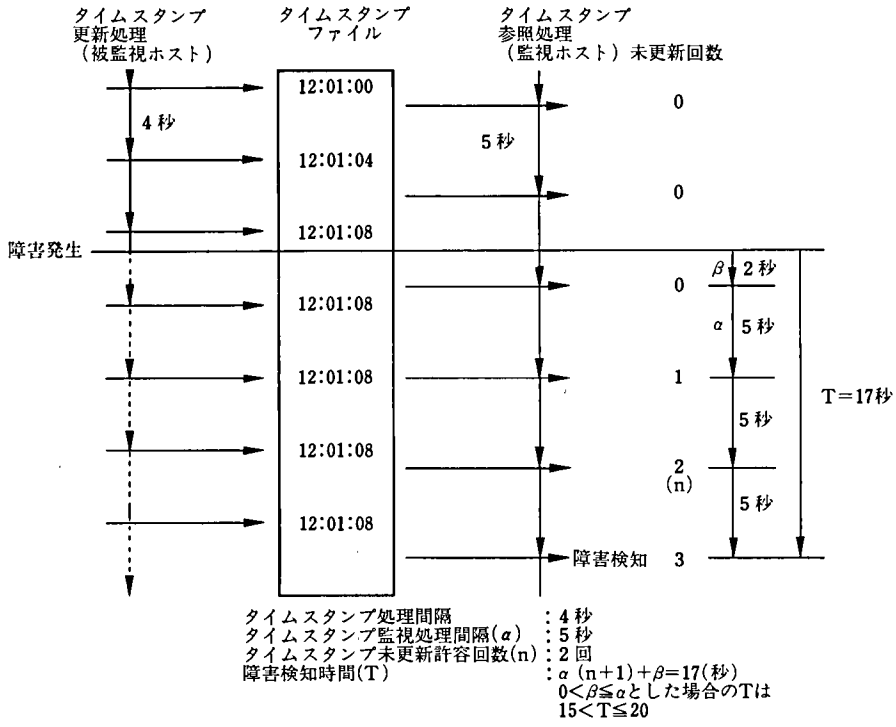


図 9　ホスト・システム障害検知時間例

② 障害発生ホスト・システムを XTC 環境から切り放す.

③ 障害発生ホスト・システムを MHFS 環境から切り放す.

4.2.3 ホスト・システム障害イベント

障害発生ホストの回復処理の自動化・短縮化の一手段として，被監視ホストの稼働状況の遷移をイベント（事象）として発生させる．ただし，監視制御中に限りイベントを発生させる．イベントには，ホスト・システム障害のフォールト位置決定を行うために，以下の情報が含まれている．このイベントにより，イベント制御機能を経由して，回復制御ジョブ（XTCHOST ジョブ）を起動する．

オブジェクト　：発生 MHFS 識別名(A～H)
事象発生ホスト：発生ホスト識別名
作成ホスト　　：検知ホスト識別名

4.2.4 ミディアム・リカバリ起動判定機能

ホスト・システム稼働状況監視機能がホスト障害を検知するとイベント制御を経由し，回復制御ジョブ（XTCHOST ジョブ）を起動する．XTCHOST ジョブでは，自ホストが XTC リカバリを実行するかどうかを検査し，XTC リカバリを実行するよう設定されている場合のみ，XIS ミディアム・リカバリ・ジョブおよび使用者処理アクタ（ジョブ）を起動する．

使用者が XTC リカバリを実行するかどうかは，あらかじめ障害ホストとリカバリ実行ホストの関係を XTC リカバリ・パターンとして設定しておく必要がある(表 7).

表7　XTCリカバリ・パターン

障害ホスト＼リカバリ実行ホスト	A	B	C	D
A		○		
B	○			
C	○			
D	○			

表の見方
障害ホストがホストB～Dの場合ホストAがリカバリを実行し，障害ホストがホストAの場合にはホストBがリカバリを実行することを示す．

4.2.5　XISミディアム・リカバリ・ジョブ

XISミディアム・リカバリ・ジョブは，XISによってあらかじめ自動的に生成されており，IRUのミディアム・リカバリ，およびXISのシステム・テーブルのリカバリを行う．

IRUのミディアム・リカバリが終了すると，障害ホスト・システムで更新中のデータベースはすべてリカバリされ，ロックも解除される．この時点で，他ホスト・システムのアプリケーションに対する影響は完全に無くなる．

4.2.6　DP-M 1 THのデュアル・セッション機能

DP-M 1 THのデュアル・セッション機能とは，一つの端末に対してシステム・セッションを2重化（オンライン・セッションとバックアップ・セッション）し，ホスト・システム障害に対してオンライン・セッションからバックアップ・セッションに自動的に切り換える機能である．詳細は，本特集号別稿の“XTPA環境におけるXISのデータ通信制御”参照のこと．

4.3　ノーダウン・システムの実現

図10に示すような端末を使用した問い合わせ応答型のトランザクション・システムを仮定する．

端末（TH：DP-M 1）は，コンカレントAPG/ホスト：n/Aに接続されており，使用者プログラム：TPSpが起動され，TPSpは，データベースを更新する処理を行う．また，端末のバックアップ・セッションは，コンカレントAPG/ホスト：n/Bに接続

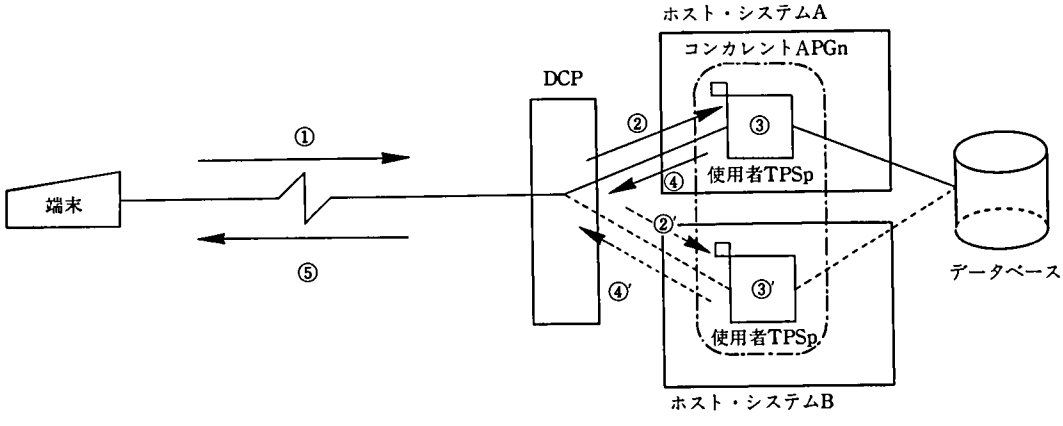


図10　問い合わせ応答型トランザクション・システムの例

されている．

上記の仮定において，ホスト・システム：A に障害が発生した場合，そのトランザクションの処理フェーズは以下の現象となる．

1) ①に示される段階の途中あるいは送信前でホスト・システム障害が発生すると端末の接続が DCP 内で自動的にバックアップ・セッションに切り換えられ，電文は，②'の経路をたどってホスト・システム:B で TPSp によって処理されデータベースを更新し(③')，④'の経路で端末に応答が返される．この結果，端末使用者にはホスト・システム障害の影響はでない．

2) ホスト・システム障害発生時に障害ホスト・システムで処理が行われている（②～③）場合，端末に応答電文が返らず，端末使用者はホスト・システム障害の影響を受ける．この場合，データベースは更新されていない．端末から次の電文を送信すると 1) と同様，端末の接続が自動的にバックアップ・セッションに切り換わってアプリケーション処理が続行できる．

3) ホスト・システム障害発生時に，ホスト側で使用者 TPS は終了しているが，CMS (Communication Management System) 内に応答電文が滞留中（④）である場合，2) の場合と同様，端末に応答電文が返らない．しかし，この場合，データベースは更新されている．端末から次の電文を送信すると 1) と同様にアプリケーション処理が続行できる．

4) DCP (Distributed Communication Processor) から端末に応答電文を送信中(⑤)にホスト・システム障害が発生しても端末使用者に影響はない．この場合も，端末から次の電文を送信すると 1) と同様にアプリケーション処理が続行できる．

5) 仮に，健全ホスト・システム：B に端末が接続されているとすると，その端末群は，基本的に処理が続行できる．ただし，障害ホスト・システムで実行中のプログラムが保持しているレコードに対してロック命令を出すと，そのプログラムはデッドロックとなり，アプリケーション処理はできない．

ミディアム・リカバリが終了すると，すべての障害ホスト・システムが保持しているレコードはすべて回復され，ロックも解除されるため，アプリケーションは完全に回復される．

上記 2) と 3) では端末使用者に対する現象が同一であるので，データベースが更新されたか否かを判断できるように，端末ごとに最終電文を保存しておき，使用者からの要求によってそれを渡す機能を提供している．

従来のホットスタンバイ・システムの場合，①～④の段階でのホスト・システム障害はすべて端末使用者に影響がでてしまうので，アプリケーションは続行不可能となる．

4.4 ノーダウン・システムの構築形態

XTPA システムでは最大 4 台のホスト・システムを使用できるため，いろいろな形態のフォールト・トレランス・システムを構築することが可能である．本章では，フォールト・トレランス・システムの形態について説明する．

1) 待機型 XTPA システム……待機型 XTPA システム (図 11) は，1～3 台の本番ホスト・システムと待機ホスト・システムから構成されるシステムである．本番

系ホスト・システムに障害が発生すると，待機ホスト・システムは新たな本番ホスト・システムとなる．障害ホスト・システムは，新たに待機系ホスト・システムとして立ち上げることも可能である．待機ホスト・システムが常態稼働時にはテスト系のアプリケーションを稼働させることも可能である．

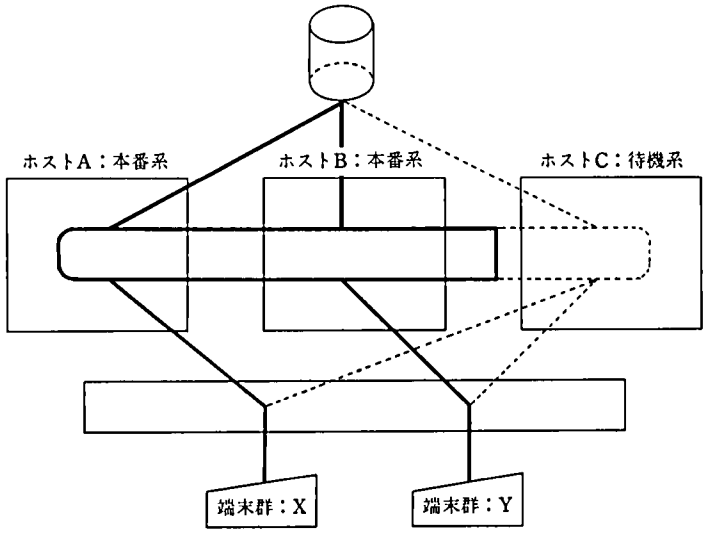


図 11　待機型 XTPA システム

2)　吸収型 XTPA システム……複数のホスト・システムが相互にバックアップし，ホスト・システム障害発生時には，相手ホストの業務を吸収して，アプリケーションを続行する方式である(図 12)．この場合，吸収した形態で稼働できるだけのホスト・システムのキャパシティが必要である．

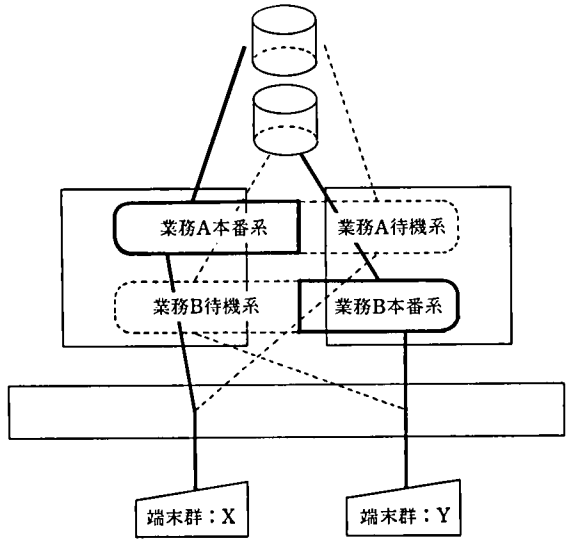


図 12　吸収型 XTPA システム

5. お わ り に

XTPA システムにおいて XIS 運用機能を使用することによって，使用者には以下の利点がある．

1) 業務システムの成長に合わせて，ホスト・システムあるいは APG の増設を行っても，統一的なインタフェースでシステム運用を行うことができ，使用者の負担が少ない．

2) 曜日による業務量の多少に応じた柔軟なホスト構成でシステムを運用することができ，コストの削減を行える．

3) 複雑な XTPA システムの運用を簡単/確実に行うことができ，使用者の負担が減らされている．

また，ホスト・システムに障害が発生しても少なくとも他の健全ホストで，アプリケーションを続行することができ，事実上ノーダウン・システムを実現できる．

表 8 にリカバリ方式によるフォールト・トレランスの違いを示す．

表 8 リカバリ方式によるフォールト・トレランスの違い

方式	アプリケーション中断時間	フォールト・トレランス	アベイラビリティ
リブート型	長	小	x%
待機型	↓	↓	∧
XTPA 型	無し	大	≒100%

現在，ホスト・システム障害が発生してから，その障害がミディアム・リカバリが終了し無害化されるまで，約 60 秒かかっている．今後，拡張データ処理装置 XPC (eXtended Processing Complex) を使用すると，ホスト・システムの障害検知は，EXEC によって行われることになる．また，非回復型のファイルに対するレコード・ロックはホスト障害発生時に即時に XPC によって解除される．XIS の XPC 対応機能は，94 年 10 月のリリース予定で現在開発が進行中である．これにより，障害検知時間の短縮および障害波及範囲の縮小が見込まれ，よりアベイラビリティの高いノーダウン・システムが実現されるであろう．

また，ホスト・システム側では稼働するホスト・システムは仮想化されており，どのホスト・システムでプログラムが稼働しても，データベースを更新しても，一貫性は保たれる．これは，柔軟なシステムを容易に運用するためのシステムとしての重要な資質である．この点について，端末を考えてみると，わずか DP-M 1 TH および Level 2 A/2 B TH でのみホスト・システムを切り換える機能を持っているだけである．しかし，これも 2 台のホスト・システムが対象であり，ホスト・システムの仮想化という目標にはまだ不十分である．今後，ホスト・システムに対する端末のより完全な仮想化が望まれる．

XIS 運用機能の開発にあたって，また，本稿の執筆にあたって数々の貴重な助言を頂いた関係者各位に感謝の意を表したい．

参考文献　［1］ 沢田啓，“XTPAに基づくノーダウン・システム”，ユニシス技報，Vol. 14，No. 4，1994. 2.

執筆者紹介　竹　内　　久（Hisashi Takeuchi）

1987年東京工業大学工学部化学工学科卒業．同年日本ユニシス(株)入社．同社客先であるN金庫システム・サービスを経て，統合オンライン・ソフトウェアXISの開発，サービスに従事．現在システム企画開発二部オンライン開発一課に所属．

3階層分散システムにおける協調分散トランザクション処理

Cooperative Distributed Transaction Processing on a Three-tier Distributed System

北川達朗, 宮下　真

要　約　コンピュータ・システムは，単一ホスト集中型処理から複数マシンを使用した分散処理に移行しつつある．また，UNIX* における業務処理環境は著しく向上し，UNIX マシン間の分散処理も一般化しつつある．

そのような状況の中で要求されたのは，ホスト～サーバ～ワークステーションといった3階層にわたるオンライン・トランザクション処理の実現であった．なかでも，ホスト～サーバ間の協調処理の実現とその信頼性が必要とされた．

本稿ではシリーズ2200とU6000といった異なるプラットホーム間の協調分散トランザクションの実現方式，ゲートウェイ・システムの処理構造および機能を述べると共に，今後の問題を挙げている．

ゲートウェイ・システムの主な機能は，以下の通りである．

1) プロトコル・マッピング
2) トランザクション・マッピング
3) 宛先管理
4) コミット同期制御

Abstract　Computer systems are in the process of evolving from centralized single-host data processing to distributed multi-machine processing. Such marked progress has been seen in UNIX-based data processing that it has made distributed processing in the UNIX environment even more commonplace. What was called for in such transitional circumstances was to build a three-tier on-line transactions processing system made up of hosts, servers and workstations. What was required most of all was reliable cooperative processing between host computers and servers.

In addition to mentioning the architecture and functionality of the gateway system, this paper describes the implementation of cooperative distributed transaction processing between different platforms: the 2200 system and the U 6000 server, including what problems remain to be solved. The major features provided by the gateway system are for (1) protocol mapping, (2) transactions mapping, (3) address management, and (4) synchronized commitment control.

1. はじめに

近年，UNIX 上での業務処理環境は著しく改善され，ビジネス・ユースのためのプラットフォームとして，UNIX マシンの位置づけは，非常に重要なものとなりつつある．とくに，OA などの部門処理におけるサーバ・マシンとして急速に広がりを見せている．

また，基幹業務処理を実行するオンライン端末に対しては，ホスト処理の一端を担

* 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．

うインテリジェント端末としての役割のみならず，部門OA処理などと共存できるワークステーション機能が要求され，そのプラットフォームとしてDOS-PCやUNIXワークステーションが注目されている．

このような背景のもとで，ホストを中心とした基幹業務処理と部門処理とを同一のネットワーク環境の中で実現させるために，ホスト～サーバ～ワークステーションの3階層構造の考え方が浸透してきた．そして，その階層構造の中に，トランザクション処理環境を構築したいという要求が発生するのも当然の成り行きといえる．

しかし，オンライン処理の分野とりわけトランザクション処理環境の提供に関しては，高い信頼性を要求されるということもあって，UNIXが提供できる環境は十分とはいい難く，長年の蓄積を背景とした汎用機の信頼性にせまるまでには至っていないのが現状であろう．

ホスト（シリーズ2200）～UNIXサーバ～UNIXワークステーションの3階層構造の中で分散トランザクション処理環境を実現するためには，各々のプラットフォーム上にあるトランザクション処理環境を結合し協調動作させることを考える必要がある．

そこで，UNIXの環境における代表的トランザクション・モニタであるTuxedoと，2200ホストの統合トランザクション処理パッケージであるXISをベースに如何にして3階層分散トランザクション処理環境を構築したかを述べるとともに，今後の問題点についても触れたい．

2. 分散トランザクション処理

本論に入る前にまず，『分散トランザクション処理』について定義することにしたい．分散トランザクション処理の定義は様々にされているが，ここでは前述の3階層ネットワークを前提として，下記項目を満足することを必要十分条件と考えた．

① 一つの業務処理（トランザクション）が複数のプロセスに分割されながら，一つの不可分な処理単位を構築すること

② プロセスはホスト，サーバ，ワークステーションなどの複数のノード上に分散されうること

③ ホスト，サーバ上に分散トランザクション制御機能プロセスが存在すること

①②は，極く一般的な条件であるといえる．③の条件は，単純なプロセス間のデータ連携処理は範疇外であることを意味しており，重要な要件としてとらえる．ここで，『分散トランザクション制御機能』とは

ⓐ トランザクション制御機能（グローバル・トランザクションおよびローカル・トランザクションの管理）

ⓑ 分散ノード間のプロトコル・ハンドリング機能

のことである．

この前提で，各ノードにおいてベースとなるプロダクトの選定を行ったが，候補となるトランザクション・マネージャ（以降，TM）は

- U 6000上ではTuxedo（プロダクト名；Open/OLTP）
- 2200上ではホストレベルでの分散トランザクション処理に対応しているXIS

があげられた．ここで，おのおのの TM としての特徴を整理してみよう．

2.1 Tuxedo が提供するトランザクション処理環境

Tuxedo は，X/Open* が規定している DTP (Distributed Transaction Processing) モデルに基づいた分散トランザクション処理環境を提供する．

X/Open では，DTP の構成要素をトランザクション・マネージャ (以降，TM)，リソース・マネージャ (以降，RM)，アプリケーション (以降，AP) に分類している (図1)．TM は，AP 間のメッセージの授受，複数 RM 間のコミット制御などの，トランザクション全体を管理する．RM は，データベースなどの共有リソースを管理し，TM が管理する AP プロセス（サーバ・プロセス）内での共有リソースへのアクセスを可能とするものである．

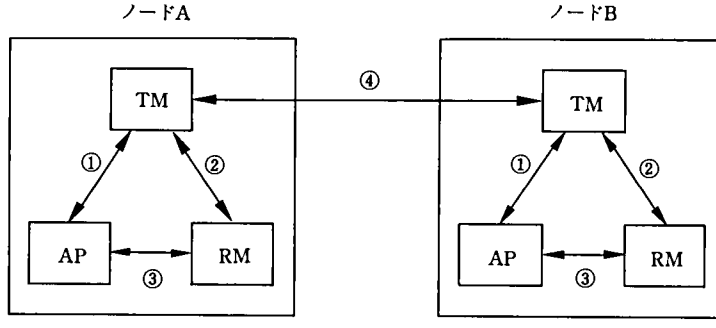


図 1　X/Open DTP モデル

① AP～TM 間のインタフェース

ATMI (Application to Transaction Manager Interface) と呼ばれる，TM が AP に対して提供する標準インタフェースである．インタフェースの種類としては，

- トランザクション・バウンダリ規定（グローバル・トランザクションの開始/終了）
- サービス呼出し/応答受信（AP 間のデータ送受信）

などがある．

② TM～RM 間インタフェース

トランザクション内で行われるリソースアクセスに対するコミット制御のために，TM-RM 間で実行される標準インタフェースであり，XA インタフェースと呼ばれる．

③ AP～RM 間インタフェース

RM が AP に対して提供する RM 固有のインタフェースである．通常，データベースを管理する RM では，SQL インタフェースとなる．

④ TM～TM 間通信プロトコル

ノード間通信のための TM 固有のプロトコルである．

* X/Open は X/Open 社の登録商標である．

Tuxedo システムはまた，機能面から下記三つの主要コンポネントからなる．

- SYSTEM/T
- SYSTEM/WS
- SYSTEM/HOST

これらのコンポネントを組み合わせることにより，さまざまなネットワーク構成における分散トランザクション処理が実現できる．

2.1.1 SYSTEM/T

SYSTEM/T はトランザクション管理のカーネル部分であり，下記の基本機能を持つ．

- クライアント・サーバ処理モデルを基本としたトランザクション管理
- プロセス稼働状況監視
- 宛先管理/経路ルーティング
- 優先制御/負荷分散
- 2相コミットメント制御

クライアント・サーバ型トランザクションでは，図2に示すように，AP はサービスの要求者であるクライアントとサービスの提供者であるサーバに分割される．

クライアント AP は TM ライブラリとリンクしてクライアント・プロセスを構成し，サーバ AP は TM ライブラリおよび RM ライブラリとリンクしてサーバ・プロセスを構成する．サーバはサービス（＝業務処理）を実装しているプロセスであり，サービス内で RM の提供するインタフェース（SQL など）を使用してリソースへのアクセスを行う．

この形態では，サーバ・プロセスが行うリソースへのアクセスのみが，コミット制御の対象となる．

2.1.2 SYSTEM/WS

Tuxedo では，上記のカーネル機能で実現されるサーバノード間の水平分散処理環境に加えて，SYSTEM/WS コンポーネントによりワークステーション～サーバノー

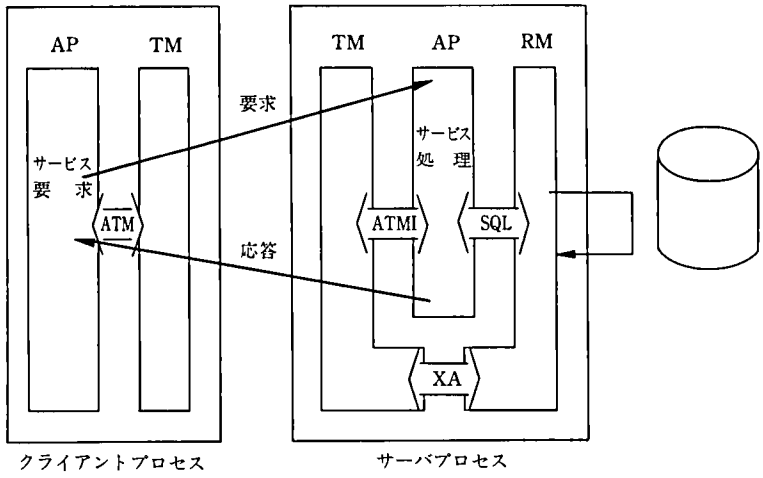


図 2　Open/OLTP クライアント/サーバ処理例

ド間の垂直分散処理環境を提供している（図3）.

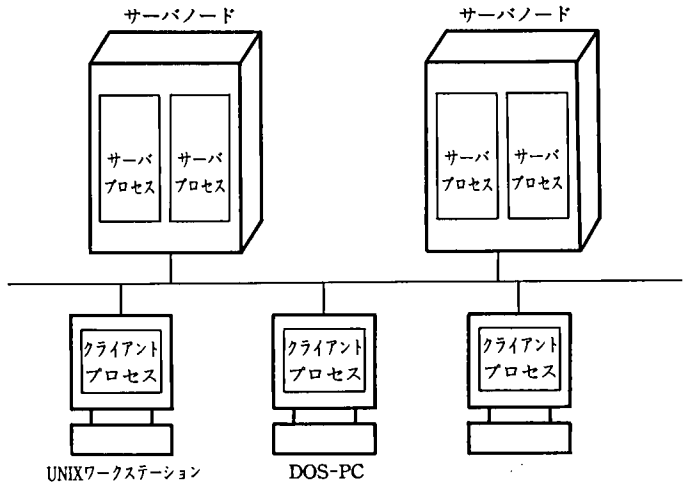


図 3　SYSTEM/WS ネットワーク環境

ワークステーション上には，クライアントプロセスのみが存在し，サーバノード上のサーバプロセスに対する処理要求を行うことができる．これにより，サーバノードの負荷をワークステーションに分散させることが可能となる．

また，SYSTEM/WS は，DOS マシン，UNIX マシン上で稼働することが可能であり，おのおののプラットフォーム上で稼働するフロントツールと連携することが可能である．

2.1.3　SYSTEM/HOST

Tuxedo では，SYSTEM/HOST コンポーネントにより IBM 社の MVS/CICS* ホストとの分散トランザクションを実現することが可能である（図4）.

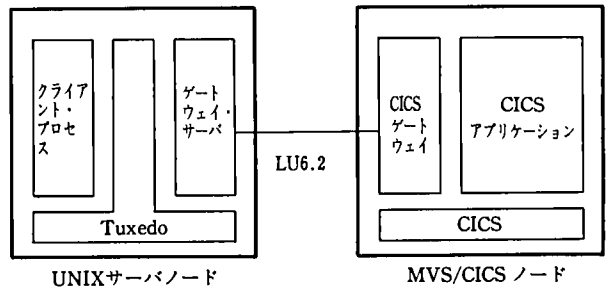


図 4　SYSTEM/HOST 構成

このコンポーネントは，CICS 制御下のアプリケーションをほとんど変更することなく（メッセージの入出力インタフェースは変更要），Tuxedo が制御する分散トランザクションのサーバプロセスとするためのゲートウェイ・システムとして機能する．ただし，このコンポネントは

① 2相コミットメントを支援しない

* MVS/CICS は IBM 社の登録商標である．

② CICS側からのクライアント処理ができない

という制限がある.

2.2 XISが提供する分散トランザクション処理環境

XIS (eXtended Information System) は ACLES/DTP (Advanced Communication control eLEments and Support system/Distributed Transaction Processing) と協調して, 2200ノード間の分散トランザクション処理環境を提供する. XISは, ノード/アプリケーション・グループ単位にトランザクション制御を行う. ACLES/DTPは, XISのメッセージ制御のサブシステムとして, 複数ノードに分散された分散トランザクションを構成する各プロセス間の会話制御を行う (図5).

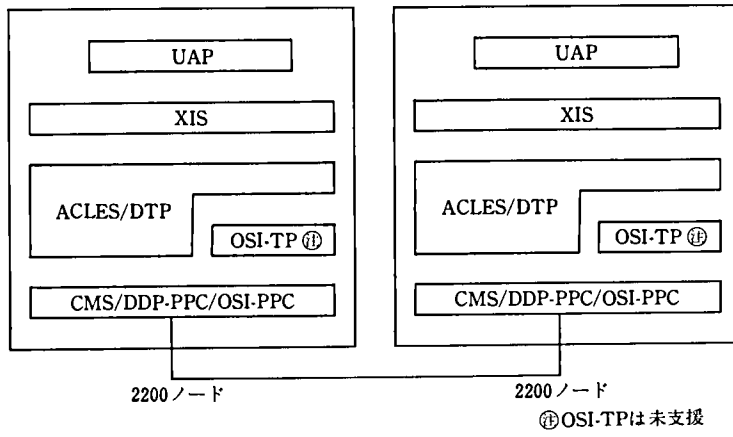


図 5 XIS分散トランザクション環境

XISおよびACLES/DTPにおける分散トランザクションの処理モデルは, OSI-TPが規定しているPeer to Peer型である(図6).

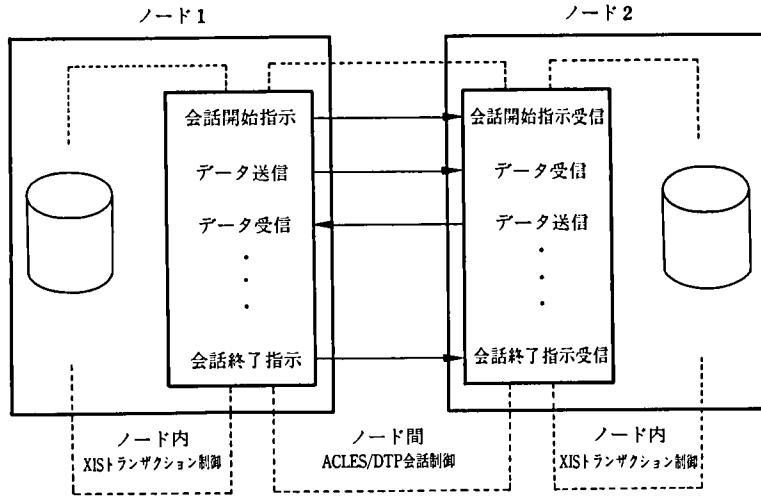


図 6 Peer to Peer型処理形態

この形態では，クライアントサーバ型と異なり，会話処理により分散アプリケーション間で複数のデータの送受信が可能である．また，アプリケーションプロセスは，会話処理の起動者，受動者に分かれるが，クライアントサーバ型のようにサービスの要求者，提供者という区別はなく，個々のアプリケーションプロセスは，トランザクション処理において対等であり，おのおののプロセスにおけるリソースへのアクセスが，コミットの同期制御の対象となる．

ACLES/DTP は，図 6 に示すように OSI プロトコル階層の 7 層(アプリケーション層）において独自のプロトコルを使用してノード間通信を行っている．そのため，異機種接続は，OSI が規定するセション層プロトコル（プロセス間通信プロトコル）を前提としており，7 層の ACLES/DTP 独自のプロトコルも実装することが必要となる(図 7)．

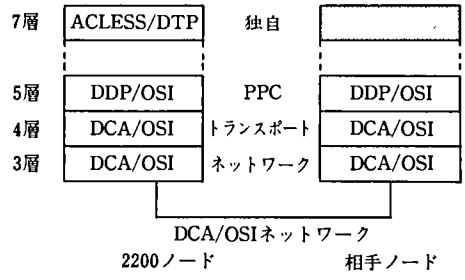


図 7　プロトコル階層

3. Tuxedo-XIS 接続方式

以上のような機能を持つ各 TM のもとで相互接続方式を検討したが，各々の TM に修正を加えないゲートウェイ・システムとして実装すること，を前提とした．

TM の機能を大別すると，①トランザクション制御機能，②通信制御機能となる．②はとくに分散トランザクション処理を支援する TM に必須機能である．異なる TM をゲートウェイ方式で接続するには図 8 の形態となるが，この方法を実現するためには，通信プロトコル 1，通信プロトコル 2 がともに開示されていることが必要条件となる．XIS サイドの ACLES/DTP プロトコルは，当社の独自なプロトコルであるので問題ないが，Tuxedo プロトコルは Novelle から開示されていないため，この形態は実現不可能である．

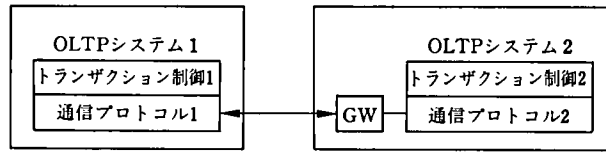


図 8　ゲートウェイによる異種接続方式

上記のような対等なノード接続に替わる方法として，一方のノードを他方のノードのサーバ・システム（たとえばデータベース・サーバ）とみなすことが考えられる．

X/OPEN のトランザクション処理モデルにおいては前述のように，RM を OLTP

システムを構成する一つの独立要素として定義している．また，TM とのインタフェース (XA) を守っている限り任意の RM を追加・作成することができる．この特性を利用することにより図9のような接続方式が考えられる．

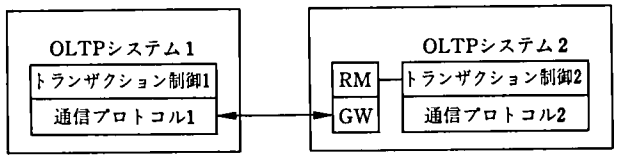


図 9　Tuxedo-XIS 接続方式

この形態によると，OLTP システム1(XIS 2200)側の通信プロトコルを，OLTP システム2 (Open/OLTP) 側のゲートウェイ (GW) で実装し，そのプロセスを，Open/OLTP TM が該当の RM を介して制御することにより，二つの TM を結合することが可能となる．

また，この結合方式は，

・2200 側では，新規開発および XIS 2200 の修正は不要，

・U 6000 側は，Open/OLTP 上の RM の開発としての位置づけであり，

Open/OLTP 本体への修正は不要という特徴を持つ．

これにより，このゲートウェイ・システムによる接続は

・XIS TM にとっては同種接続

・Tuxedo TM にとっては，RM アクセス

とみなすことができることになり，実質的に両 TM ともに異種接続を意識することなく相互接続が実現できることになる．

4. サーバ～ホスト間のゲートウェイ・システム概要

XIS ゲートウェイは，図10に示すように，サーバ～ホスト間の分散トランザクション処理を可能にするためのゲートウェイ・システムとして U 6000 サーバ上に実装される．

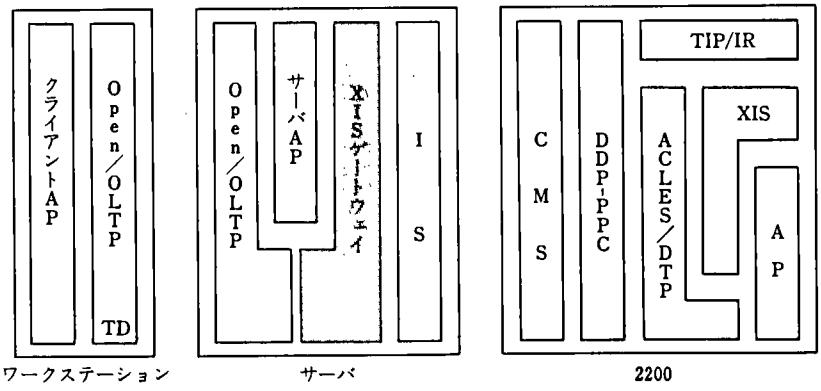
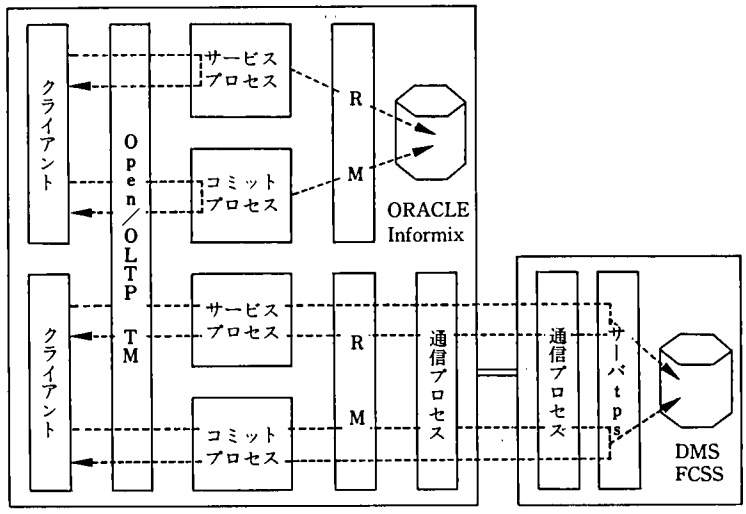


図 10　ゲートウェイ・システム構成

この仕組みでは，図11に示すように，UNIXサーバ上のクライアント・プロセスは，2200上のアプリケーション（サーバTPS）と接続する場合，UNIXサーバ上のサービス（たとえば，ORACLE*のデータベース・アクセスサービス）を呼び出す場合と同様の意識でサービス呼出しをするだけで良い．



Informixは米国Informix社の登録商標である．

図11　サーバ～ホスト間トランザクション処理形態

4.1 XISゲートウェイ構成要素

U 6000サーバ上のOpen/OLTPにおいて，XISゲートウェイは，2200ホストを一つの更新可能なデータベースとして想定したRMとして位置づけられる．そのため，図12に示すように，Open/OLTPのTMやAPとのインタフェースを持つRMライブラリとホストとの会話制御プロセスで構成される．

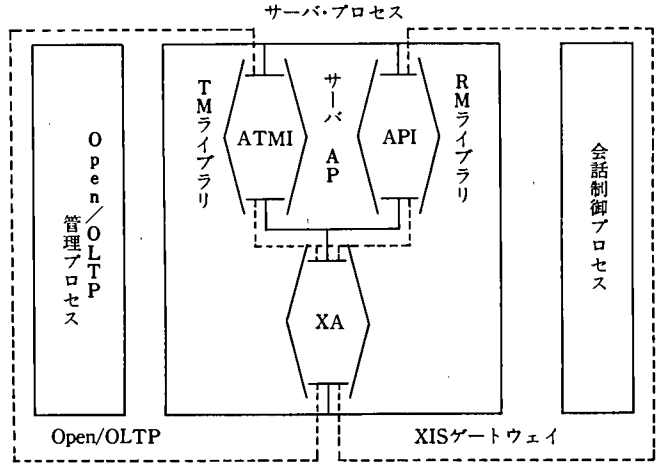


図12　XISゲートウェイの位置付け

* ORACLEは，米国ORACLE社の登録商標である．

4.1.1 RMライブラリ

RMライブラリは，通常，アプリケーション・インタフェース（以降，API）提供部分とXAインタフェース提供部分で構成される．

Open/OLTPのサーバ・プロセスには，サーバ・アプリケーション・プロセス（以降，サーバAPプロセス）とTMS（Transaction Manager Server）と呼ばれるコミット制御プロセスの2種類があり，それぞれ構造と処理内容が異なる．サーバAPプロセスは，Open/OLTPのTMライブラリ，RMライブラリ，サーバAPがリンクされ，RMのAPIを使用して行うサーバAPのサービスの実行が主処理となる．これに対し，TMSには，サーバAP部分を含まず，Open/OLTPのTMライブラリとRMライブラリがリンクされ，通常，サーバAPプロセスで行われたデータベース・アクセスのコミット/ロールバック処理を行う（図13）．

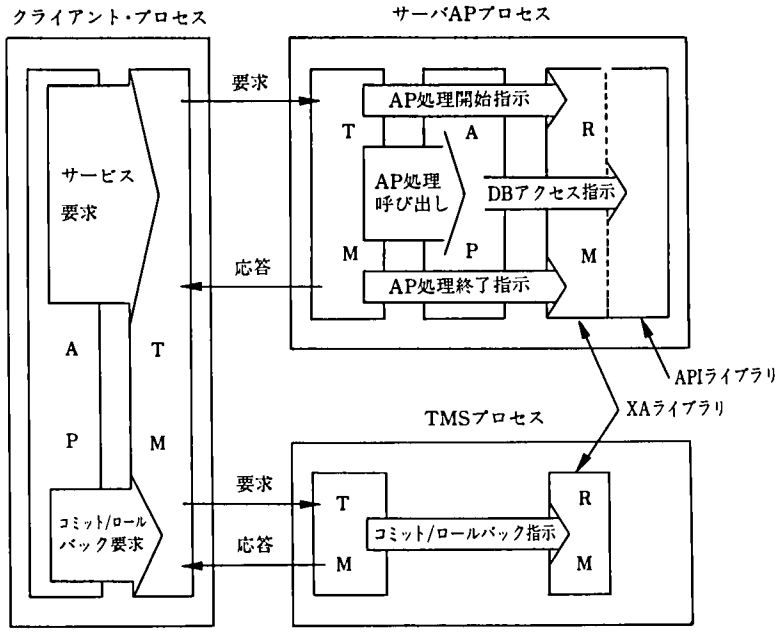


図 13　Open/OLTP トランザクション処理形態

4.1.2 会話制御プロセス

会話制御プロセスは，会話制御プロセス用の構成ファイルに指定された，会話セッションごとの自ノード，宛先ノードのセッション端点のネットワーク内での識別名（ISまたはDDPの構成ファイルに登録されているもの）などの情報により，会話セッションの単位で常駐し，ACLES/DTPとの全二重の会話セッションを確立し，データの送受信を行う（図14）．

4.2 XISゲートウェイ機能概要

XISゲートウェイは主に以下の機能から成り立っている．

- プロトコル・マッピング
- トランザクション・マッピング

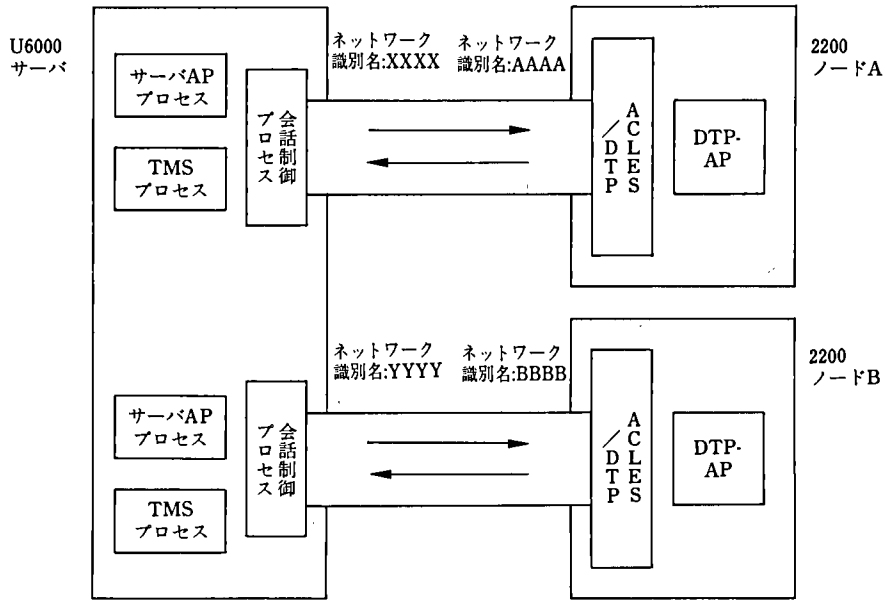


図 14　会話制御プロセス

・宛先管理

・コミット同期制御

4.2.1　プロトコル・マッピング

XIS ゲートウェイでは，API ライブラリ部分のユーザデータの送受信処理において，AP で認識するデータ形式⇔ACLES/DTP データ形式の変換を行う．また，ユーザデータ内の漢字コードの変換（EUC⇔LETS-J）も行う．

XA ライブラリ部分では，Open/OLTP からのコミット/ロールバック制御指示を受け，2200 上 ACLES/DTP アプリケーションへのコミット/ロールバック指示メッセージ，送信，応答の受信を行う．

このため，図 15 に示すように XIS ゲートウェイは，Open/OLTP システムにおいては，サーバ AP プロセス，TMS プロセスとして処理を行うが，XIS 2200，ACLES/DTP の分散トランザクション・システムにおいては，会話処理を行う相手の一つのアプリケーションとして位置づけられる．

4.2.2　トランザクション・マッピング

Open/OLTP では，一つのクライアント AP の要求により複数のサーバ AP が行うデータベースへのアクセス処理を一つのグローバル・トランザクションとして管理する．Open/OLTP は，グローバルトランザクションごとに一つの識別子を割り当て，クライアントのサービス要求時，およびコミット/アボート要求時に TM は RM にトランザクション識別子を通知する．RM では，サービス処理において行われたデータベース更新などの処理をトランザクション識別子に対応させて管理し，TM のコミット/ロールバック指示において指定されたトランザクション識別子に対応したデータベース更新処理のコミット/ロールバックを行う．

XIS ゲートウェイでは，図 16 に示すように，Open/OLTP より通知されるトランザ

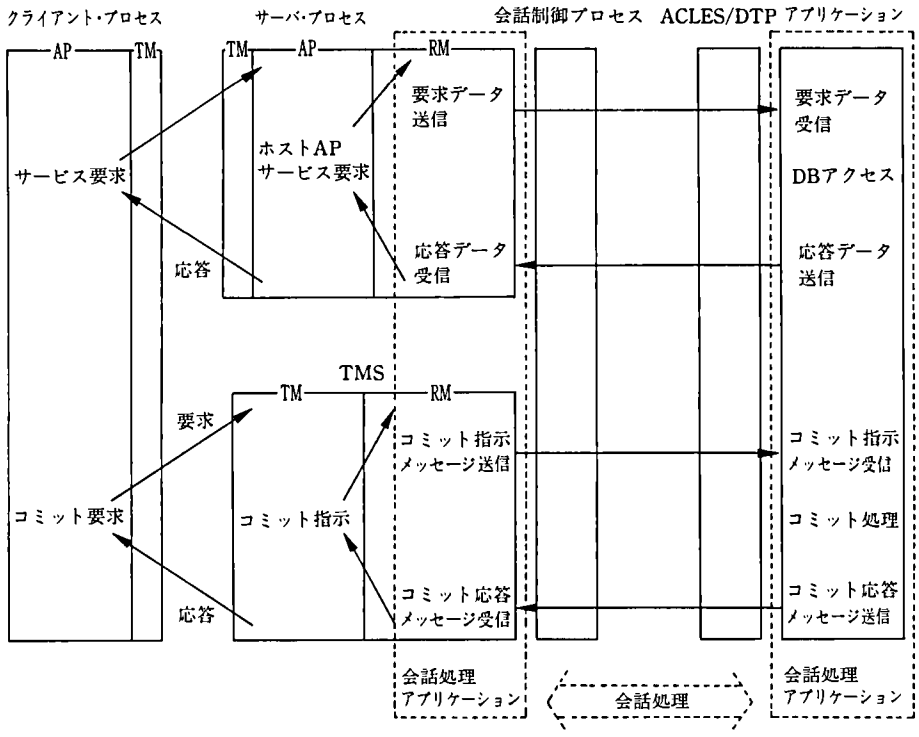


図 15　プロトコル・マッピング

クション識別子と ACLES/DTP との会話処理で使用する会話識別子との対応を管理する.

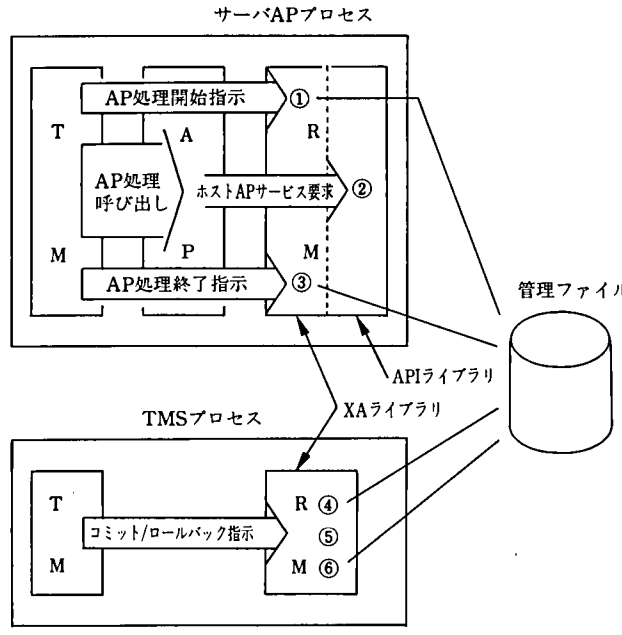
サーバ・アプリケーション・プロセス内の RM ライブラリ処理において，Open/OLTP から通知されるトランザクション識別子ごとに識別子管理ファイルを作成し，会話処理で使用するために割り当てた会話識別子や会話処理に必要な情報を記録する．TMS プロセス内の RM ライブラリ処理ではコミット/ロールバック指示とともに通知されたトランザクション識別子をキーとして識別子管理ファイルから対応する会話識別子やその他の会話情報を取得し，該当会話でのコミット/ロールバックの指示メッセージ送信，応答の受信を行う.

4.2.3 宛 先 管 理

XIS ゲートウェイでは，クライアント AP において，要求の宛先が 2200 上に存在することを意識することなく，通常の U 6000 上のサービス要求と同様に 2200 上のアプリケーションへのサービス要求を可能にするための宛先情報を管理している (図 17).

1) 宛先ノード情報……Open/OLTP では，サーバ・プロセス起動時に RM に対してデータベースのオープンの指示を行う．このとき，Open/OLTP の構成ファイルのパラメタに指定されたデータベースのオープン情報を RM に通知し，通常の RM は Open/OLTP より通知された情報に基いて該当データベースのオープン処理を行う.

XIS ゲートウェイでは，Open/OLTP のデータベースのオープン情報指定パラ



①トランザクション識別子をファイル名として，識別子管理ファイルを作成する．
②会話識別子を割り当て，会話処理を開始し，要求データの送信，応答データの受信を行う．
③①で作成した識別子管理ファイルに②で使用した会話識別子およびその他の会話情報を記録する．
④トランザクション識別子をキーとして，識別子管理ファイルから会話識別子などの会話情報を取得する．
⑤④で取得した会話情報を使用し，コミット/ロールバック指示の送信，応答の受信を行い，会話処理を終了する．
⑥使用した識別子管理ファイルを削除する．

図 16　トランザクション管理

メタに宛先ノードのセッション端点のネットワーク識別名を指定する．RM ライブラリ部分では，サーバ・プロセスの起動時に Open/OLTP から通知され，セッション構成情報は会話制御プロセスで管理しているため，サーバ・プロセスごとに宛先のノード情報を持つことになる．

2)　宛先アプリケーション名……XIS ゲートウェイでは，2200 上のアプリケーション処理を一つのサービス処理とみなし，XIS ゲートウェイのサーバ AP プロセスのサービスと 1 対 1 に対応することを原則としている．この XIS ゲートウェイのサービスとは，ユーザに作成されるルーチンであり，内部で XIS ゲートウェイの提供する API により，2200 上アプリケーションへの要求データの送信と処理結果の受信を行い，2200 上アプリケーションの Open/OLTP での代替サービスとなるものである．このサービスのエントリ名は，2200 上アプリケーション名と同じ名前にすることを原則としており，これにより，クライアントにおいて，サービス要求の宛先が 2200 上にあることを意識せずに，通常の Open/OLTP のサービス要求と同様に 2200 上アプリケーションへのサービス要求を発行することを

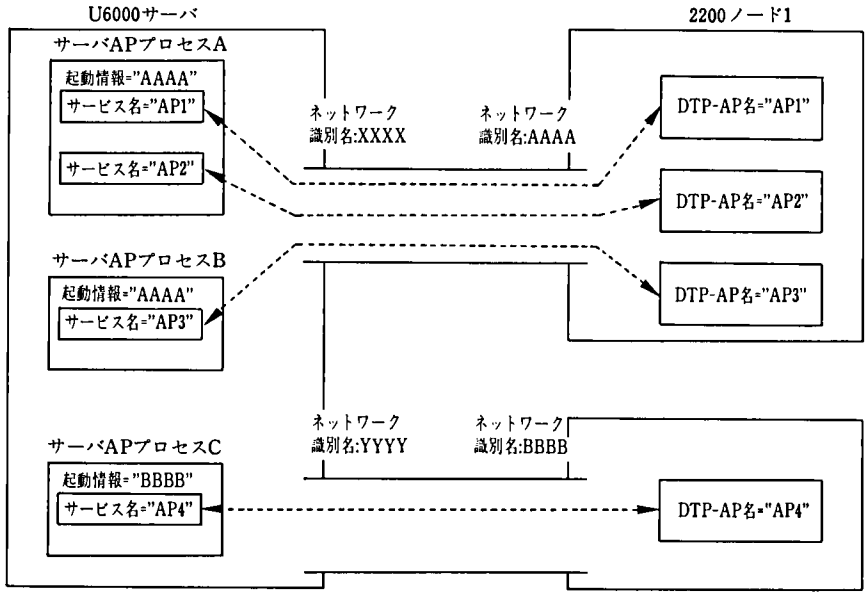


図 17　宛先管理方式

可能としているとともに，XIS ゲートウェイの内部処理でサービスのエントリ名の認識により，宛先となる 2200 上のアプリケーションを特定している．

4.2.4 コミットメント

XIS ゲートウェイでは，Open/OLTP の 2 フェーズ・コミット（以降，2 PC）制御を受け，疑似的な 2 PC 処理を行う．

Open/OLTP における 2 フェーズ・コミットは，データベースに対応して存在する TMS プロセスのプロセス間通信により制御される (図 18)．トランザクションに関与する複数の TMS プロセスのうち，ある一つの TMS プロセスが調整者となり，2 フェーズ・コミットの調整を行う．クライアント・プロセスからのコミット要求には調整者である TMS プロセスが受け，ファースト・フェーズ処理として，トランザクションに関与している他の TMS プロセスに対してコミット準備指示を行う．その応答が全てコミット可能を示すものであるか否かにより，セカンド・フェーズ処理として，それらの各 TMS プロセスに対してコミット指示かロールバック指示を行う．TMS プロセス内部では，ファースト・フェーズ，セカンド・フェーズにおいて，XA インタフェースにより，TM から RM に対して，ファースト・フェーズ，セカンド・フェーズでそれぞれ，コミット準備要求，コミットまたはロールバック要求を行う．

XIS ゲートウェイでは，図 19 に示すように，この Open/OLTP の行う 2 フェーズ・コミットの制御のもとで，2200 上のアプリケーションとの会話処理により，2200 上アプリケーションに対してデータベースのコミット/ロールバックを指示する．

XIS ゲートウェイの RM ライブラリでは，TM からのファースト・フェーズのコミット準備要求により，2200 上アプリケーションおよびシステムからの異常通知の受信確認を行う．このときに異常通知が確認されれば，XIS ゲートウェイの RM ではコミ

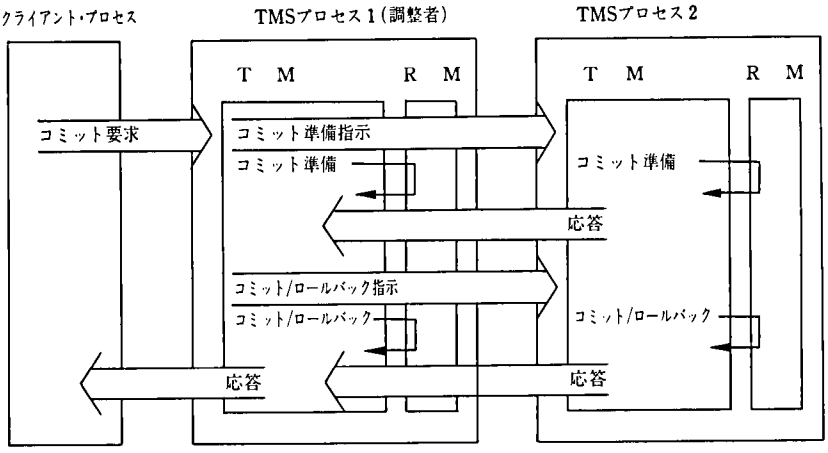


図 18 Open/OLTP の2フェーズ・コミット

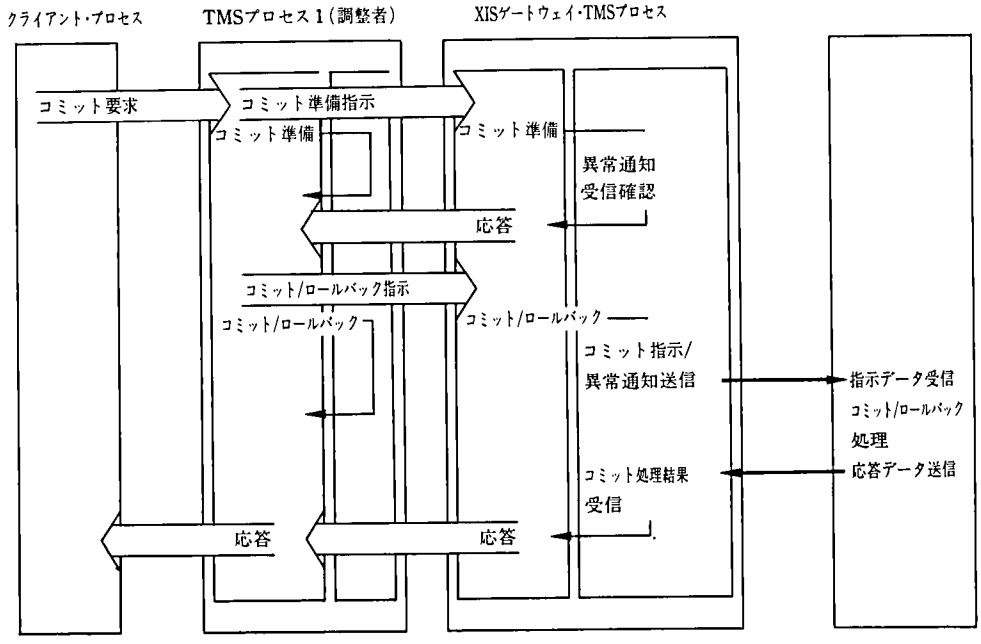


図 19 ゲートウェイにおける疑似2PC

ット準備不可を示すエラーを TM に返すため，Open/OLTP では，セカンド・フェーズでトランザクションのロールバックを行う．異常通知が確認されなかった場合，XIS ゲートウェイの RM はコミット準備 OK を示すステータスを TM へ返す．

セカンド・フェーズで TM からコミットが要求される場合，XIS ゲートウェイの RM は，2200 上のアプリケーションに対して，コミット指示を表すデータを送信する．これにより，2200 上アプリケーションでは，XIS に対してコミット要求を行い，処理結果を表すデータを XIS ゲートウェイに対して送信し，XIS ゲートウェイの RM では，受信したデータの内容により，TM に対してコミット処理結果のステータスを返す．

セカンド・フェーズでTMからロールバックが要求される場合，XISゲートウェイのRMでは，2200上のアプリケーションに対して，異常通知を送信する．2200上アプリケーションでは，異常通知の受信により，XISに対してロールバック要求を行う．

このような疑似的な2フェーズ・コミットを行うことにより，セカンド・フェーズ処理中に障害が発生するケースを除き，サーバノード上のデータベースと2200上のデータベースの不整合の発生を防いでいる．

本来の2PC処理では，ファースト・フェーズのコミット準備処理として，データベースの仮更新イメージを障害が発生しても復旧可能な媒体に保持することが必要であるが，この疑似的な2PC処理では，実際の2200ノード上のデータベースのコミット準備指示は行っていない．また，コミット指示，応答はアプリケーションのみに認識されるもの（ACLES/DTPはユーザデータとして扱う）であるため，セカンド・フェーズで障害が発生した場合のコミット処理のリカバリは不可能となり，この疑似的な2PC処理ではデータベースの不整合を100%防止することはできない．

これに対し，サーバノード上のOpen/OLTPと同様，ACLES/DTPとXISの2200ノード間での分散トランザクションにおいては，本来の2PC処理が可能である．そのため，今後，以下の改善を行うことにより，XISゲートウェイにおけるサーバノード～2200ノード間での本来の2PC処理も可能となる．

1) トランザクション制御データの使用……2200ノード間の分散トランザクション処理で使用されているACLES/DTPが認識するトランザクション制御データによりコミット準備，コミットの指示を行う．
2) ステップ管理……2200上ローカル・トランザクションの管理単位であるステップの識別子をOpen/OLTPのグローバル・トランザクション識別子の対応とそれらの状態管理で行う．

以上のような構造のゲートウェイ・システムを構築することにより，3階層分散トランザクション環境を実現できたわけであるが，システム的にみるとこれは，『SYSTEM/HOSTのUNISYSホスト版』であるといえる．

Tuxedoの三つのコンポネントおよびXISゲートウェイを使用した3階層分散トランザクションのモデルは図20のようになる．

5. 今後の分散トランザクション処理環境

以上，述べてきた方法により，2200/XISシステムを巨大サーバとみなす形態でのU 6000 → 2200間分散トランザクション処理が実現できた．しかし，これはいくつかの制約のもとでのプロトタイプ・システムとして位置づけるべきものである．該システム構築によって明らかとなったU 6000-2200接続における問題点，未解決項目を下記にあげる．

5.1 解決すべき項目

1) 日本語コードの取扱い……異機種接続を行う場合には必ず問題となる項目である．当システムでは，U 6000上のXISゲートウェイ・プロセスにおいて，
   ・送信時 … EUC → Lets'J 変換

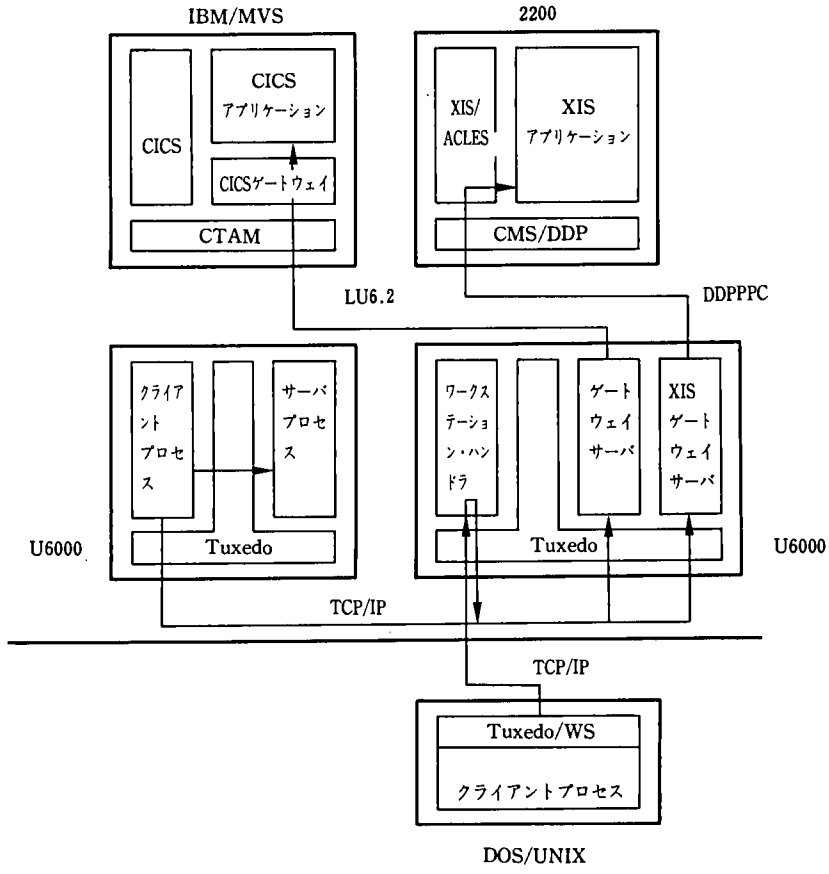


図 20　3階層分散トランザクション・モデル

・受信時　…　Lets'J → EUC 変換

を，ユーザ・データのすべてに対して実施している．

このシステムでの機能範囲(U 6000 側からのクライアント処理のみ)であれば，この実装形態でもとくに問題とはならないが，将来的に U 6000 と 2200 が分散トランザクションを構成する対等なノードとなることを考慮すると，双方のノードに機能を持つべきであろう．方策としては下記2通りが考えられる．

① データ送信側あるいはデータ受信側のいづれで実施するかを規定し，各ノードにおいてローカル・コード⇔リモート・コードの変換を実施する．

② ノード間のデータ転送は標準コードにて実施し，データ送信ノードにおいてローカル・コードから標準コードへの変換，データ受信ノードにおいて，標準コードからローカル・コードへの変換を実施する．

2) 構造化データの取扱い……トランザクション処理に限らず，複数の分散されたノード上に存在する AP プロセス間でデータの授受を行う場合，そのデータ構造を AP 側から設定できる機能は非常に重要である．

・クライアントプロセス，サーバプロセスが構造化データ内の各フィールドを同じフィールド名でアクセス可能なこと

・構造化データ内にキャラクタフィールド，バイナリフィールドの混在が可能であること

を実現すべきである．

3) 管理機能の実現……前述した通り，該システムにおける2200ノードは，Open/OLTPにとってはDBMSでしかない．したがって，Open/OLTPが標準装備する管理対処外である．しかし，クライアントプロセスが要求するサーバプロセスを実行するノードであることもまた事実である．2200上のオブジェクト（サーバ，サービス，DBMS，…）の管理，トランザクション・ステータス管理などは是非とも実現しなければならない．これは，異機種を含むネットワーク・システムにおける統合管理が必要であることを意味している．この実現は非常な難題ではあり，ひとり分散トランザクション処理システムだけの問題ではない．最近話題としてとりあげられることが多くなっている分散システム運用機能による解決が待たれる．

4) 2相コミットメント機能の実現……X/OPEN XAインタフェースが規定している2相コミットメント・プロトコルはTM⇔RM間プロトコル（インタフェース）であり，XISおよびACLES/DTPが実現している2相コミットメント・プロトコルはTM⇔TM間プロトコルであるが，両者間に根本的な差異はない．XISゲートウェイ・プロセスにて，両者プロトコルのマッピングを実装すれば実現可能であり，それにより完全な分散トランザクション処理環境に一歩近づくことができる．しかし，2相コミットメント・プロトコルの実現は，ある意味で異種TMが密結合することであり，障害/リカバリ・システムの共通化も併せて実施しなければならない．

5) 2200クライアント機能の実現……2200を巨大サーバとしてインプリメントした該システムでは，クライアント・プロセスを起動する環境は考慮していない．これは，Open/OLTPのコミット制御プロトコルが明らかになっていないために，2200クライアント形態での2相コミットが困難であるためである．しかし，今後，2200とU 6000を対等ノードとして接続することを考えると，この機能の実現のための策を講じる必要がある（OSI-TPプロトコルの支援が有力な解決策と思われる）．

6. おわりに

今後2〜3年のうちに，コンピュータ処理のダウンサイジングが進む中で，分散トランザクション処理はますます脚光を浴びるであろう．そのなかでも，既存汎用機のDBと既存あるいは新規のUNIX/マイクロ系システムのDBとの間のトランザクション処理は非常に重要な位置を占めるであろう．

今回のU 6000〜2200上のトランザクション結合は，プロトタイプ・システムとは言え，実際に二つの異なるTMをまたがって一つのトランザクション処理を実現しており，分散トランザクション処理に一石を投じたと言うことができる．今後は，米ユニシスにて開発中のOLTP 2200接続により徐々に機能アップが図られると考えられるが，機能拡張もさることながら，分散システムに与えられたキーワードの一つである

メインフレームとUNIX相互運用性の追求がポイントとなろう．これは，とりもなおさず，過去10数年にわたって蓄積してきた汎用機におけるトランザクション処理の運用・管理・リカバリといったインフラ・システムをUNIX機に移植することにほかならない．その対応が今後の大きな課題となると考えられる．

執筆者紹介　北　川　達　朗（Tatsuro Kitagawa）

昭和28年生，52年埼玉大学理学部数学科卒業．同年日本ユニシス（株）入社．証券・金融機関にてSEサービスに従事した後，1100シリーズ゛系のオンライン・インフラ・プロダクトの企画・開発，．UNIX系のインフラ・プロダクトの企画・開発を経て，平成5年よりオープン・システム関連の利用技術サービス業務に従事．現在オープン・システム・サービス第一本部SI利用技術部に所属．

宮　下　　真（Makoto Miyashita）

昭和42年生，平成2年千葉工業大学工業経営学科卒業．同年日本ユニシス（株）入社．分散インフラ・ソフトウェアの開発に従事．現在システム技術第一本部システム企画開発一部分散システム技術課に所属．

Aシリーズのトランザクション処理の実現方法

An Implementation of Transaction Processing on the A Series

森　良行, 原　広仁

要　約　近年のオンライン・トランザクション処理環境は，分散処理を含めてさまざまなトランザクション形態をとるようになってきている．このため，OLTP (OnLine Transaction Processing) の基盤システムの中核をなすTPモニタ (Transaction Process Monitor) は，種々のトランザクション形態に対応できる多様性，システムの拡張や業務処理の変更に伴って変更できる柔軟性，トランザクション特性を保証できる信頼性を持ち，大量トランザクション処理を可能とする高処理能力とそれに伴う負荷分散の考慮がなされている必要がある．

Aシリーズ上のCOMS (COmmunication Management System) は，TPモニタとして位置づけられ，このような機能特性を実現するための仕組みを兼ね備えている．また，現在提供されているCOMSでは，分散処理環境に対する機能に課題を残しているが，今後の機能拡充によって順次整備されることが期待できる．

本稿では，OLTPを支える機能を明らかにし，将来的にもAシリーズOLTP環境の中核となるCOMSでこれらの機能がどのような実装技術によって実現するかを紹介する．

Abstract　All different patterns of transaction processing, including distributed data processing, are seen today in on-line transaction processing environments. This has caused the transaction process (TP) monitor, the kernel of infrastructure systems for on-line transaction processing (OLTP), to be equipped with versatility high enough to deal with various transaction modes, flexibility that allows modifications according to systems expansion and changes in applications, and reliability capable of assuring transaction properties as well as high-level performance which guarantees mass-volume processing and the associated capability of distributing processing loads.

The COmmunication Management System (COMS) for the A Series, positioned as the TP monitor, is designed in such a way that those functional characteristics can be well implemented. The COMS product, currently available, still runs short of a complete set of functionality for distributed data processing environments, but it is expected to gradually develop into a full-fledged version through the continued addition of required functions.

Besides clarifying the OLTP-supporting functions, this paper describes what technology can be used to implement those functions in COMS, the core of A Series-based OLTP systems.

1. はじめに

ネットワークを介したコンピュータ利用がさまざまな分野で一般化しつつあるなか，OLTPと呼ばれるオンライン・トランザクション処理システムは，様々な利用形態をとる．OLTPは，端末からのコンピュータの処理要求を一つの取引（トランザクション）として，複数の端末から同時にその要求を受け付けて逐一処理を行う形態であるが，現在は，端末（ターミナル）というよりは，端末（パソコンを含む）やワークステーションが処理要求を行う対象（クライアント）として広がりを見せ，またクライアントからの直接の要求に基づいて処理を行う形態のみならず，サーバを介した

要求への処理形態も一般化しつつある．さらに分散処理環境では，多種多様なサーバが別のサーバのクライアントとなってきた．

OLTPのシステムは，業務処理から考えると使用する分野によって確かに様々な形態をとっているが，トランザクション制御という観点でその機能対象を整理し，抽象化し，それに基づいて現在の実装技術を捉え直すことは，多様な処理形態に対するトランザクション制御プログラムの適応性を探る上で必要なことであり，また今後の方向付けも確かなものとなる．

本稿では，「Aシリーズにおけるトランザクション処理の実現」に焦点を当て，まずトランザクション処理に対する考え方を述べ，それに基づいた実装技術を紹介する．実装技術の章では，トランザクション環境の物理的な多様性を業務処理プログラムがどう論理的なレベルで一貫した扱いができるのかという「トランザクション制御」の観点，耐障害性の観点，高いスループットを得るための機構上の観点，さらにトランザクション処理環境における開発・保守の観点から紹介する．最後に，現在の実装技術の分散トランザクション処理における拡張性を展望する．

2. OLTPを支える機能

2.1 トランザクション

一般にオンライン・トランザクションという場合には，オンライン入力されるメッセージを示すことが多いが，OLTPのシステムでは，メッセージの中で，以下の四つの特性を持ったものをトランザクションとして定義する．この特性は，OSIの応用層の国際標準規格として定義されている（参考文献[1][4]を参照）．

- 原子性(Atomicity)　：一つのトランザクションに対して処理が正常に完了するか，または処理が全て取り消されるかの二者択一の処理の単位であること．
- 一貫性(Consistency)　：トランザクションの処理結果が一定していること．すなわち，トランザクション処理結果が論理的矛盾を起こさないこと．
- 独立性(Isolation)　：並列にトランザクション処理を実行している場合に，相互のトランザクションが干渉しないでトランザクションとして独立していること．他のトランザクションによって処理結果が左右されないこと．
- 耐久性(Durability)　：完了したトランザクションについては，障害によってそのトランザクションの処理が消滅することはない．すなわち，完了したトランザクションの情報がデータベース上に反映されることを保障すること．

クライアントからの要求が一つのトランザクションの発生として処理される形態ばかりでなく，複数のトランザクションを発生させることも考えられる．すなわち，クライアントは，処理形態上，複数の処理要求を一回のメッセージで要求し，その返答として要求の受け付け可否を受けるだけで，その結果の照会を別の要求として行うことがある．この場合には，要求の受け付け処理が一つのトランザクションとして扱わ

れ，受け付けた要求各々が別のトランザクションとして扱われる．このような形態のトランザクションにおいても，各々のトランザクションでみれば，上記の四つの特性を持っていなければならない．

また，クライアントからの要求が一つのトランザクションとして，複数のホストシステム上で分散して処理され，その結果の回答が戻される形態もある．この場合に上記の特性を保証するには，いずれかの分散ホストでエラーが発生した場合は，他のホストの処理が正常に終了していても，そのトランザクションの戻し処理が必要となり，その同期性を保証しなければならない．

この様にトランザクションは，複数のトランザクション（分散トランザクションと呼ぶ）を生成して，そのトランザクションをもとに実際の処理を行うことがある．これは，原子性と一貫性といった特性を保証するために，元のトランザクション処理が解消された場合には，生成された複数のトランザクションも解消しなければならない．このようなトランザクションの特性を保証することにより，様々なオンライン環境に対応したトランザクション処理システムを構築することができる．

2.2 OLTPのソフトウェア構成と機能

OLTPのソフトウェア構成は，機能別にみると以下のようなモジュールがあり，図1にその構成図を示す．なお，以下の「ステーション」は，ワークステーションや端末を含めた総称として使用している．

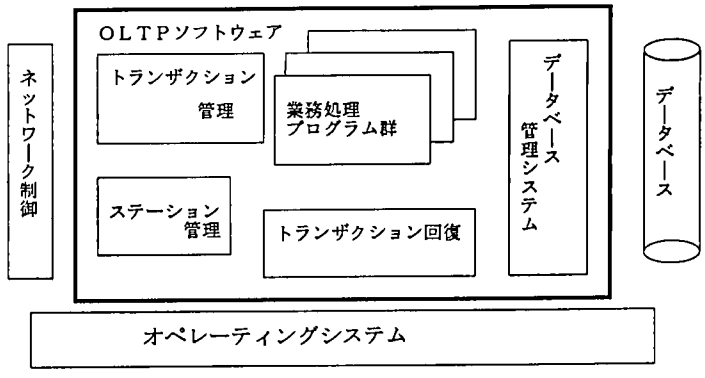


図1　OLTPソフトウェア構成図

1)　業務処理プログラム群

業務処理プログラムは，業務やトランザクションの種類に応じて存在する．これらのプログラムは，トランザクション管理モジュールより自動的に起動され，各トランザクションは，トランザクション管理モジュールを介して業務処理プログラムに渡される．また，同一プログラムが複写されて多重に稼働することもある．これらの業務処理プログラムをトランザクション・プロセッサ（TP）と呼ぶ．

2)　ステーション管理モジュール

このモジュールは，ステーションからのメッセージ形式の変換とステーションとの送受信に関する操作を行う．この機能によって，トランザクション・プロセッサは，扱うトランザクション形式の統一性が保証され，ステーションとの送受

信にともなう物理的操作から解放される．

ステーションからの入出力メッセージの形式がステーションのタイプによって異なる場合には，トランザクション・プロセッサが同じ形式のトランザクションとして処理できるように，業務処理要求に合わせトランザクション形式への変換が必要になる．このモジュールは，トランザクション・プロセッサ別，ステーションタイプ別，あるいは，トランザクション別に形式変換できる機能を提供する．また，形式変換では，ステーションからの入力データのタイプ検査（数値型とか文字型など）や値の範囲検査なども行い，形式上，妥当でないデータをトランザクション・プロセッサに渡さず，エラーメッセージをステーションに戻す機能やステーション固有の出力データの加工などの機能も併せ持つ．

また，ステーション管理モジュールは，トランザクション・プロセッサとステーションとの送受信において入力メッセージの保存，出力メッセージの到達確認，未送信メッセージの保存と出力管理などの物理的な送受信を補う機能も持つ．したがって，トランザクション・プロセッサは，業務処理と直接関係しないステーション操作から解放され，送受信先を論理的に扱うことが可能となる．さらに，利用者がステーションを介して扱える業務を保証するために，トランザクション単位での安全保護（セキュリティ）の管理機能も提供する．

3) トランザクション管理モジュール

このモジュールは，業務に応じてトランザクション・プロセッサの起動・終了を実行管理し，トランザクション・プロセッサとステーション間のトランザクションの経路管理を行うモジュールでOLTPの中核をなす．このモジュールは，トランザクション管理（TM: Transaction Manager）プログラムとかTPモニタ(Transaction Process Monitor）とも呼ばれる．

このモジュールの機能であるトランザクション・プロセッサの実行管理においては，一般に利用者がどの業務にどのトランザクション・プロセッサを稼働させるかを設定する登録機構を提供しており，業務開始要求があれば設定に従って対象のトランザクション・プロセッサを起動する機能を持つ．また，稼働中のトランザクション・プロセッサが処理中に異常終了した場合は，自動的に再起動して他のトランザクションの処理を継続するためにトランザクション・プロセッサの稼働監視が必要となる．この起動・終了の機能には，トランザクション処理待ち行列の長さを監視して，当該トランザクション・プロセッサの複写数を動的に変更する機能も必要となる．すなわち，トランザクションの滞留量に応じて実行されるトランザクション・プロセッサの複写本数が変更され，トランザクションの処理待ち時間を平準化する機能である．とくに，複数の業務処理を取り扱うOLTPでは，日々の業務処理の中でトランザクションの増減に応じて対象となるトランザクション・プロセッサの複写数を変更しなければならないので，この機能が有効になる．

第二の機能は，トランザクションの経路制御である．トランザクションの経路付けにおいて，ステーションからの入力トランザクションは，当該ステーションが行っている業務が確定していれば，その業務に基づいた経路付けを行う必要が

ある．すなわち，トランザクション上のある欄を検査することで，当該業務内のどのトランザクション・プロセッサに渡すべきかを判断する仕組みである．TPモニタは，単純な判断形式であればトランザクション上の欄とその設定値の登録とそれに伴う経路選定機構を持ち，複雑な判断形式であれば利用者の手続きを組み込み，経路選定時点で呼び出す機構を提供する必要がある．これらの機構は，トランザクション・プロセッサが他のトランザクション・プロセッサへトランザクションを渡したり，入力とは異なるステーションに出力したりする場合の経路付けにも使用できる．このように経路選定についても，登録による経路付けと，利用者が作成した経路付けの手続きの呼び出しの二面的な要求を満たす機能が必要となる．

4)　データベース管理システム

データベースに対する操作や回復処理を行うモジュールで，OLTPシステムとしてトランザクションの特性を保証できる回復処理とそのための業務処理インタフェースを持つ．

5)　トランザクション回復モジュール

トランザクションデータのロギング機能とそのログファイルを使用した回復処理の機構を持つモジュールである．このモジュールは，データベース管理システムとトランザクション・プロセッサ間のインタフェースを持ち，トランザクション特性を保証するためにデータベース管理システムの回復処理に対する補完機能を司る．

トランザクション・プロセッサは，トランザクションの処理を行うためにデータベースに対する更新操作の始まりや終了をトランザクション単位にTPモニタに知らせる必要がある．これは，トランザクション・ログの作成をデータベース処理と同期して行い，データベース回復に伴って回復対象トランザクションを識別する必要があるからである．

このインタフェース機能は，トランザクション・プロセッサの記述によってトランザクションとデータベースの同期性が壊されないように，TPモニタとデータベースシステムを統合してインタフェースできる関数を提供することが望ましい．これは，コンパイラに対する機能であってもTPモニタの機能であってもトランザクション・プロセッサからみれば同じであるので，そのいずれかで実現する必要がある．

また，TPモニタは，一つのトランザクション処理を契機に実行されるバッチ処理についても，このトランザクション特性を保証する必要がある．データベース障害により，当該トランザクションの処理が取り消され，トランザクションログより再処理になった場合においてもこのバッチ処理が同期して実行する必要がある．このため，TPモニタは，バッチ処理の起動終了もトランザクション・プロセッサと同様に管理する機能を持つ必要がある．

2.3　分散環境のトランザクション制御

分散処理環境におけるトランザクション処理では，複数のシステムを使用して一つのトランザクションを処理する形態が考えられる．この様な処理形態では，トランザ

クションの原子性を考えると，あるシステムで処理が正常に終了しても，他のシステムで処理が正常終了しない場合には，そのトランザクションの処理はすべて解消しなければならない．このために，この同期制御を行う仕組みが必要となる．その代表的な方式として二相コミットメントがある．

二相コミットメントは，トランザクションを主に処理する一つのシステムと従に処理する複数のシステムに分離して，処理の相を二つに分けて主と従のシステムの処理確認を行う．第一相では，主のシステムから従のシステムに対して処理要求を行い，従のシステムがトランザクション処理として資源確保または資源排他まで行いその結果を戻す．この時点で，処理が正常に終了するか否かが主システムでは確認できる．第二相では，主システムが従システムに対して処理継続か取り消しかを通達し，従システムは，その処理が完了した時点で主システムに通知する．これによりトランザクションの原子性が保証される．このような同期制御機構は，TP モニタと DBMS が連携した方式で実現し，そのインタフェース機能も単純化する必要がある．また，トランザクション・プロセッサによって同期制御の整合性が失われないような保護や，分散トランザクションが処理できるデバッグ環境の提供も必要となる．

3. A シリーズにおけるトランザクション処理環境

3.1 ソフトウェア構成

A シリーズにおけるトランザクション処理に関わるソフトウェアを図 2 に示す．前章の OLTP 環境でのソフトウェア構成との関係は次の通りである．

	A シリーズでの実装ソフトウェア名
オペレーティングシステム	MCP(Master Control Program)/AS
ネットワーク制御	BNA(Burroughs Network Architecture) A シリーズのネットワーク制御系のソフトウェアで LAN 制御も司る．
トランザクション管理	COMS(COmmunication Management System) A シリーズでは，TP モニタをメッセージ制御システムとも呼ぶ．
ステーション管理	実装のソフトウェア構成では COMS に組み込まれている．
トランザクション回復	実装のソフトウェア構成では COMS に組み込まれている．
業務処理プログラム群	COMS で実行管理されるトランザクション・プロセッサ（業務プログラム）．
データベース管理システム	階層，ネットワーク型の DMS II，リレーショナル型の SQLDB，意味モデル型の SIM の三つのデータモデルがある．

3.2 TP モニタとしての COMS

A シリーズのトランザクション処理における TP モニタは，COMS (COmmunication Management System) と呼ばれるプログラムが担当する．COMS は，業務処理プログラムへのアプリケーション・インタフェースを提供するとともに，トランザクション環境の設定情報に基づいて動的リンクライブラリ (DLL-Dynamic Link Library) を多用してトランザクションの経路制御を行う．COMS によるトランザクションの処理の流れを図 3 に示す．COMS は，全部で 18 種類の設定要素(エンティティ)を持つが，このうち，図 3 に示した基本の要素について，処理の流れに従って説明する．

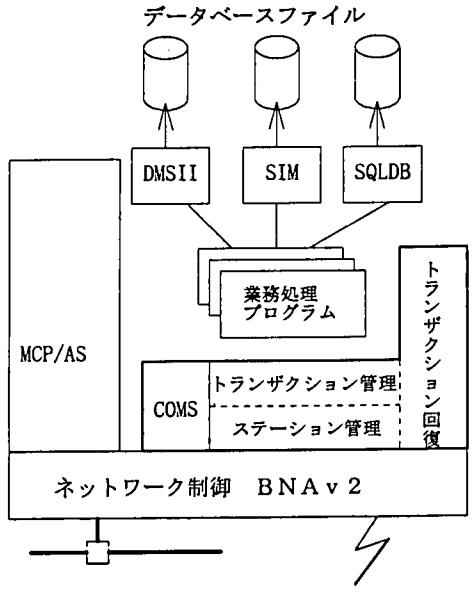


図 2　ソフトウェア構成図

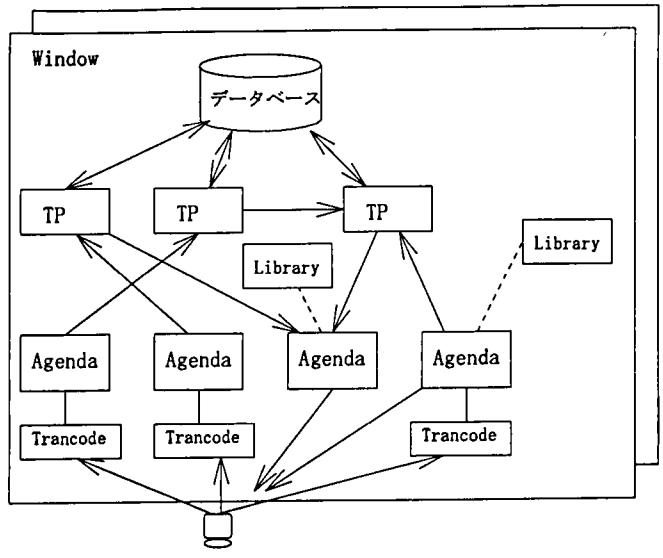


図 3　COMS を用いたトランザクション処理

1)　Window (必須定義要素)……トランザクション処理環境を識別する識別名であり，トランザクション・プロセッサは，少なくとも一つの Window に関係づけられる．この Window 定義では，使用者数，Trancode (後述) の最大長，出力保護 (ステーションへの送信をトランザクション処理完了まで行わない機能) の有無のようなグローバル情報も定義される．最大 255 Window を定義できる．

2)　Trancode(任意定義要素)……トランザクションの一部で，経路付け用のシンボリックとなるキーである．Window あたり最大 2048 Trancode を定義できる．

Trancode には直接，到達すべきトランザクション・プロセッサを指定するのではなく，経路テーブルである Agenda を関連づけるといった間接的な定義を行う．未定義のトランコードを持つトランザクションの入力は，「省略時 Agenda」と関連づけられる．

3) Agenda（必須定義項目）……トランザクション経路付け用テーブルであり，最大 4095 Agenda を定義できる．Agenda には，到達すべき宛先（トランザクション・プロセッサ）のほか，トランザクション・プロセッサに先だって，あるいは，後処理として経由しなければならない Processing Item（処理項目：後述）などが定義される．

4) Processing Item/Processing Item List（任意定義要素）……「処理項目」と訳している．Agenda に関連づけられ，主としてメッセージとトランザクション間の形式変換のために，トランザクション・プロセッサの前処理・後処理用の手続きを動的リンクライブラリの入口点のリストとして定義する．最大 1023 個の Processing Item List を持つことができ，一つのリストには最大 12 個の Processing Item を含めることができる．

5) Library（任意定義要素）……前処理，後処理用の「処理項目」を介して実行するライブラリプログラムを定義する．最大 1023 Library を定義できる．

6) Program（必須定義要素）……トランザクション・プロセッサとなる業務処理プログラムであり，最大 7500 Program を定義できる．これらプログラムは，COBOL 74，COBOL 85，ALGOL，PASCAL，C などの高級言語で書かれ，各コンパイラが提供している COMS のための拡張機能を利用する親言語インタフェース方式で COMS とインタフェースする．

7) Database（任意定義要素）……DBMS の回復処理と同期してトランザクション回復を行う必要があるデータベースを定義する．A シリーズが実装する DMS II（階層，ネットワーク型），SQLDB（リレーショナル型），SIM（意味モデル型）のいずれのデータベースでもよい．最大 255 Database を定義できる．Database には更新トランザクションのみを COMS オーディット（Transaction Trail）に記録するか，照会トランザクションも含めるかなどを定義する．

3.3 トランザクション処理環境定義の実際

トランザクション処理環境定義は，COMS 自身の 1 モジュールである「COMS ユーティリティ」を用いて，メニューによる対話形式または指令形式で行う．ここでは指令形式を例にして説明する．

以下のようなトランザクション処理モデルを考える（図 4）．

— トランザクション処理環境名は，“SAMPLE” とする．

— “SAMPLE” 内では三つのトランザクション・プロセッサと，一つのデータベースが稼働する．

— “TR 00 1” で始まるテキストは，経路テーブル “Agenda2” を経て “TP1” に渡され，データベース “DB1” を更新する．“TP1” からの折り返しテキストは，経路テーブル “Agenda1” を指定して出力され，途中，編集のために “List1” に含まれる Processing Item “P1” を通過する．Processing Item “P1” の実体

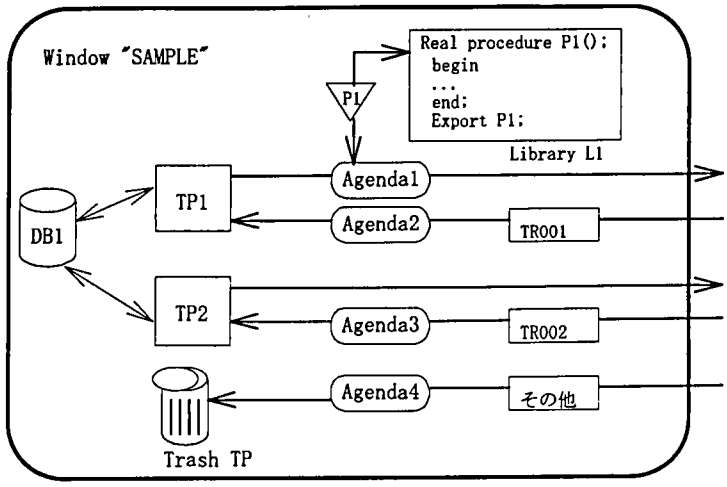


図 4　トランザクションモデル例

```
Create Window "SAMPLE";                 処理環境のｳｨﾝﾄﾞｳ"SAMPLE"を定義する.
Create Database "DB1";                  データベース"DB1"を定義する.
Create Program "TP1",                   ﾄﾗﾝｻﾞｸｼｮﾝﾌﾟﾛｾｯｻとしてﾌﾟﾛｸﾞﾗﾑ"TP1"を定義 する.
        Database = "DB1";               関連するﾃﾞｰﾀﾍﾞｰｽ"DB1"を設定する.
Create Program "TP2",
        Database = "DB1";
Create Program "TRASHTP";
Create Library "L1";                    処理項目の手続きを持つﾗｲﾌﾞﾗﾘ"L1"を定義する.
Create ProcessingItem "P1",             処理項目"P1"を定義する.
        ActualName = "P1",              "P1"の手続き名"P1"を定義する.
        Library ="L1";                  "P1"を持つライブラリを指定する.
Create ProcessingItemList "LIST",       処理項目リスト"LIST"を定義する.
        ProcessingItem = "P1";          処理項目"P1"を登録する.
Create Agenda "AGENDA1",                アジェンダ"AGENDA1"を定義する.
        Window = "SAMPLE",              ｳｨﾝﾄﾞｳ"SAMPLE"のｱｼﾞｪﾝﾀﾞとして定義する.
        ProcessingItemList = "LIST1";   このｱｼﾞｪﾝﾀﾞは, 処理項目ﾘｽﾄ"LIST1"を呼び 出す.
Create Agenda "AGENDA2",
        Window = "SAMPLE",
        Destination = "TP1";
Create Agenda "AGENDA3",
        Window = "SAMPLE",
        Destination = "TP2";
Create Agenda "AGENDA4",
        Window = "SAMPLE",
        DefaultInputAgenda = Yes,
        Destination = "TRASHTP";
```

図 5　トランザクション処理環境定義例

は，Library “L1” の中の手続き “P1” である．

— “TR 00 2” で始まるテキストは経路テーブル “Agenda3” を経て “TP2” に到達し，データベースへの更新を行い結果を端末へ折り返す．

—その他のテキストは，省略時 Agenda を経て “TrashTP” に到達し破棄される．

このモデルを実現するための定義例を図 5 に示す．このような定義作業自体，COMS 稼働中に行うことができ，結果は即座に反映される．現実の業務ではもっと複雑な経路が想定されるが，処理モデルが明確になっていれば構築は容易である．たとえば，LINC はこの例と同じように指令による定義を生成し，COMS に対し読み込みを行わせることでトランザクション環境の自動生成を実現している．

4. A シリーズの実装技術

4.1 トランザクション制御

4.1.1 多　様　性

COMS におけるトランザクション処理環境はウィンドウであると述べた．多様なトランザクション処理環境を実現するものとしてこのウィンドウは最大 255 個まで持つことができる．トランザクションを発生させるステーションは，必ずどれかのウィンドウのもとで操作することになる．また，一つのステーションが複数のトランザクション処理環境で使用する場合には，ウィンドウを切り替えることで操作できる．

通常ステーションでは，一つのトランザクションを発生させ，「Online」処理を行うが，入力トランザクションが契機になってディレイド処理のようなバッチ処理を行うことがある．このようなバッチ処理を，トランザクション処理の一貫として COMS のもとで行うことができる．この処理を COMS では「Batch」プロセスと呼び，トランザクション・プロセッサ間のトランザクションの受け渡しとバッチ処理を一つのトランザクション・プロセッサ内で行うことができる．当然このバッチ処理についても，トランザクション回復の対象となる．図 6 では，親言語構文でこの「Online」と「Batch」の切り替えを行う Enable 文の例を示す．

```
Enable(ComsIn,"ONLINE");
      % 「Online」処理操作を行う
Enable(ComsIn,"BATCH");
      % 「Batch」プロセスに切り替える
Enable(ComsIn,"ONLINE");
      % 「Online」処理に戻す
```

図 6 「Online」と「Batch」の切り替えを行う Enable 文の例

トランザクション処理の中には，他のトランザクション処理環境のトランザクション・プロセッサとのトランザクションの受け渡しが必要となる．たとえば，複数のデータベースの更新を伴うトランザクション処理がある．COMS では，出力先をプログラムやアジェンダに指定できるので，他のトランザクション環境のトランザクション・プロセッサにトランザクションを渡すことができる．また，トランザクション・

プロセッサが別のトランザクション・プロセッサに処理を依頼するネスト構造クライアント・サーバ型のトランザクション処理もトランザクション・プロセッサ間通信(TP-TP) を用いて簡単に実現することができる.

4.1.2 形式変換へのカスタムアルゴリズムの提供

COMS 下のトランザクション処理では, 3.2 節で紹介した前処理, 後処理に単一もしくは複数の Processing Item を用いることで, 入出力トランザクション経路に自由に介入することができ, 柔軟なトランザクション処理を実現できる. Processing Item は実数型の戻り値をもつ手続きとして宣言する. 手続き内に利用者が望む「カスタムアルゴリズム」を記述できる. ALGOL による記述例を図 7 に示す.

```
Real PROCEDURE HandleForeignInput
                (Hdr, State, Text1, Text2, UserText, OutputProc);
  Array Hdr[*];
  Real State;
  Ebcdic Array Text1, Text2, UserText[0];
  Real Procedure OutputProc(Hdr, State, Text1, Text2);
     Array Hdr[*];Real State;Ebcdic Array Text1, Text2[0];
     Formal;
  BEGIN
  Pointer CurPtr, NewPtr;
  IF Input(Hdr) THEN                              % 入力トランザクションの時
       IF TestDesignator(HdrStnDesg, ForeignDevice) = 0 THEN
       BEGIN                                      % 外部デバイスからの入力の時
       GetInputDesciptor(CurPtr);                 %  入力ﾄﾗﾝｻﾞｸｼｮﾝにﾎﾟｲﾝﾀをセット
       GetOutputDescriptor(NewPtr);               %  変換後のﾄﾗﾝｻﾞｸｼｮﾝﾎﾟｲﾝﾀをセット
       Replace Newptr By CurPtr+10 For TextLen(Hdr)-10;  % Edit input
       TextLen(HDR):=*-10;                        % Set new length
       Destination(Hdr):=0;                       % Reset destination
       Agenda(Hdr):=InputRouterAgenda;            % Re-routed by COMS
       Action(State):=1;                          % Set action to"SEND"
       OutputProc(Hdr, State, Text1, Text2);      % Do SEND
       EditforeignInput:=* & 1 [7:8];             % Stop processing
       END;
  END EditForeignInput;
```

図 7　ALGOL による「カスタムアルゴリズム」の記述例

図 7 では, 入力トランザクションが外部デバイスであれば, 先頭 10 バイトをカットした後, 再度 COMS に差し戻して (Trancode によって) 経路を付け直すという処理を行っている. たとえば, UNIX* から入力されたテキストをあたかも既存のダム端末と同じようにトランザクション・プロセッサに扱わせる時は, このような Processing Item が非常に役立つ. また, この処理は, 動的リンクライブラリによって呼び出されるので, トランザクション・プロセッサのトランザクションの受信処理の一環としてプロセス切り替えなしに行われ, Processing Item を組み込む事そのものについてはオーバヘッドがほとんどないといってよい. これら Processing Item の適用例としては

* 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.

・トランザクション内容の編集
・宛先のリダイレクション（経路変更）
・トランザクションの分割，併合
・トランザクションの保存

など様々である．実際，SDF (Screen Design Facility)/SDFPlus と呼ばれる画面の表示や入力項目検査を行う製品は，この Processing Item が持つライブラリプログラムを生成する．Processing Item は，通常は図7のように ALGOL で記述するが，ALGOL ライブラリを経由することで処理の実体を COBOL, C, FORTRAN や PASCAL で記述することができる．言い換えれば利用者は自分の得意な言語で入出力トランザクション経路に自由に介入できるのである．

4.1.3 ルーティング

ルーティング（経路付け）機能に関して，COMS は極めて強力な方法を提供している．経路付けは2章で述べたように大きく分けて2種類ある．一つは，事前に経路を定義しておく静的ルーティング，もう一つは動的に宛先が与えられる動的ルーティングである．一般的には入力トランザクションに対しては前者が，出力ルーティングに対しては後者が用いられる．

1) 静的ルーティング

静的ルーティングは3.2節で紹介した Agenda（アジェンダ）と呼ばれるテーブルを事前に定義することで行われる．トランザクション・プロセッサを一切持たないような特殊なウィンドウは例外として，各ウィンドウには少なくとも一つの Agenda が必要である．各入出力経路には一つの省略時 Agenda を設定することができる．通常の Agenda は，トランコードもしくはトランザクション・プロセッサによって明示的に指定することで関連付けられるが，省略時 Agenda は明示的に指定されない（したがって宛先がわからない）ようなトランザクションに対して適用される．

具体的には，トランザクションとして処理すべきコンテキスト向けにトランコードを登録しておいて，しかるべき宛先のトランザクション・プロセッサに経路付けする一方で，制御通知メッセージや破棄すべき不要メッセージを省略時宛先（省略時 Agenda が示す宛先）で処理する形態が，LINC による生成システムをはじめ，よく使用される．また，特殊デバイスや単純ファイル伝送などに見られる入力トランザクションの源が，トランコードに相当するような一意のコンテキストを含むことができない場合は，「全て」を省略時宛先で処理する形が採用される．この場合は1ウィンドウ1アジェンダ1プログラムといった構成で事足りる．

2) 動的ルーティング

動的ルーティングは，トランザクション・プロセッサないし Processing Item が明示的に宛先を与える方法である．トランザクション・プロセッサからの出力は何も指定しない限り入力源に返されるようになっている．したがって単純なクライアント・サーバ型トランザクション・プロセッサをプログラミングする者にとってはルーティングを気にすることはない．動的ルーティング，すなわち特定の宛先が望まれる場合は，ステーション識別子 (Station designator) またはプログ

ラム識別子（Program designator）をCOMSヘッダ（トランザクションに付随するタグ）に指定する．4.1.2項で述べたProcessing Item（カスタムロジック）群を経由させたい場合は，アジェンダ識別子(Agenda designator)を同時に指定することになる．動的ルーティングから静的ルーティングへの切り替えを望む場合には宛先をCOMS内にあらかじめ定義されているAgendaの「INPUT_ROUTER」を指定することで，再度コンテキスト内容すなわちトランコードに従って静的トランザクションルーティングを行わせることも可能である．

4.2 障害対応

COMSにおけるトランザクション障害対策は，次に述べるような考え方で実装されている．

1) 入力待ち行列の保護

トランザクション・プロセッサによってまだ処理されていないトランザクションは，ホルトロードを越えて保持される．すなわちシステムダウンにより未処理トランザクションが失われることはない．この保存の単位は「経路」であり，ウィンドウやトランザクション・プロセッサ単位とはなっていない．これにより効率的な保存が可能になっている．

2) 二相トランザクション

AシリーズではDMS IIなどのDBMSが二相コミットメントを備えるが，これはあくまでDBMSの整合性を保つためであったり，あるいは汎用機独特の(COMSを通さない昔風の）バッチ処理のためである．OLTPとしては冒頭に述べたようにオンライントランザクションとしての同期，もっといえばトランザクションとしての四つの特性が保証されなければならない．

各言語コンパイラではトランザクションコミットを宣言する「EndTransaction」構文にCOMS用の拡張が実施されている．これを用いることでオンライントランザクション処理はDMS IIなどのDBMSが持つ二相コミットメントと完全に同期できる．図8に例を示す．

図8のEndTransaction With Textは，二相トランザクションの頂点（Mid-Transaction)を実行しリソースの解放フェーズに入る．(1)～(3)でトランザクション・プロセッサが障害を起こした場合，動きは次のようになる．

(1)で中断……何も起こらない．次のトランザクションを受信する．

```
(1)
Receive          トランザクションの入力
(2)
BeginTransaction with COMS-Header
(3) -- データベースの更新処理
EndTransaction With Text
```

図 8　二相コミットメントの例

(2)で中断……処理中のトランザクションが再送される.

(3)で中断……処理中のトランザクションが中断される.

トランザクションの再送については,上記のように自動的に行われるようになっている.もし同じトランザクションが数回の再送によってまだ完了(EndTransaction)しない場合は,トランザクション自身に誤りがあるとみなし,源に送り返されるが,望むならProcessing Itemによってどのようにでも処置できる.

3) トランザクション記録

完了トランザクションはCOMSに登録したデータベース毎にTransaction TrailとしてDBMSとは別にトランザクションログを取ることができる.このTransaction Trailはアーカイバル回復処理によって単独で再適用できるが,DBMSと連動しているので,DBMS側の追跡記録が障害で使用できなくなった場合に,DBMS回復の入力トランザクションとして使用することもできる.なお,この処理もライブラリ呼び出しの一環として行われ,やはりトランザクション記録に伴うプロセス切り替えはない.

一般的に汎用機のTPモニタはこのようにDBMSとは別にトランザクションログを採取する傾向にあり,それをパフォーマンスの点から問題視する意見もあるが,これはTPモニタを介さないバッチ処理によるデータベース更新が存在するためである.金融業界などでは「センターカット」と称して,センターのバッチ入力の全てのトランザクションをTPモニタ経由にすることがかなり以前から行われているが,他ではバッチ処理はミッションクリティカルな部分において現存していることを考えると,トランザクション記録の一本化はやや尚早かもしれない.

4.3 トランザクション処理の効率化

TPモニタには大量のトランザクションを効率的に処理することが求められる.したがって負荷分散をどのように実現するかは重要な課題である.ここでは比較的一般的なパイプライン方式とCOMSが採用しているライブラリ並列方式を比較してみる.

1) パイプライン方式

COMS以前のメッセージ制御システムであるGEMCOS (Generarized Message COntrol System)で採用されていた方法で,GEMCOSを入力トランザクションの受付,経路選択,出力用などの機能別に分割し,それらを並列的に実行させる方式で,図9に示すとおり一定の単位時間で結果を得ることができる点でプロセッサの高速化技術などにも用いられているパイプライン方式に似ている.

この方式では,本来,t1+t2+t3時間かかるGEMCOSのトランザクション処理をProcess A, B, Cの3プロセスで分担して行うことで,Max(t1, t2, t3)時間単位でトランザクション・プロセッサは,トランザクションを得ることが可能になる.

しかし,各パイプラインノード当たりのサービス時間(tn)がある程度均一でないとこの方式は効果を発揮できない.たとえば,トランザクション量増大に伴い,各パイプラインノードに待ちが発生した場合,一般的な待ち行列理論が示すとお

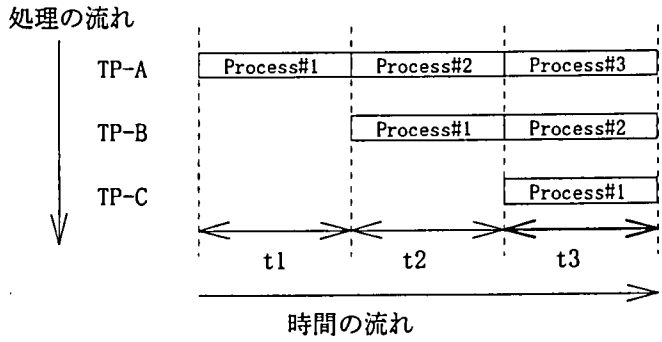


図 9　パイプライン処理

りスループット低下が発生する．また，複数のプロセスが処理を担当するためプロセス切り替えも必須となる．このようにパイプライン方式をソフトウェアの動作に適用した場合にはトランザクションボリュームに対する隘路が存在する．

2)　動的リンクライブラリの並列化

COMS は，パイプライン方式の隘路を図 10 に示すような動的リンクライブラリ呼び出しを並列化して用いることで解決している．

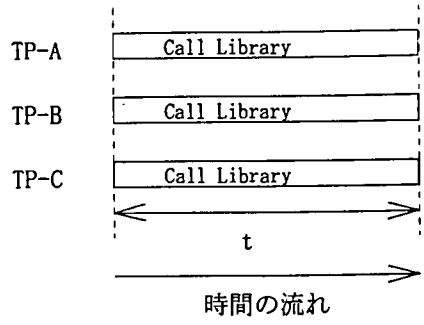


図 10　動的リンクライブラリ呼び出しの並列化

COMS 下のトランザクション・プロセッサは，動的リンクライブラリ呼び出しの連続によって，トランザクションの受け取りである「RECEIVE」から結果を回線に送出する「SEND」までを，連続した 1 プロセスで完結することができる．すなわち，トランザクションの前処理・後処理は，あたかもトランザクション・プロセッサからの手続き呼び出しのような振る舞いになる．したがって，これらの処理待ち行列の滞留時間やプロセス切り替えのオーバーヘッドは存在しない．

さらに，トランザクションの受け取り (RECEIVE) 時点での待ちの解消という点では，COMS における各トランザクション・プロセッサが最大 255 本までの同一コピーを持つことができることによって対処することができる．コピーの増減は COMS がトランザクション・プロセッサの受信待ち行列の状態を判断して自動的に制御する．トランザクション・プロセッサの数だけ COMS が並列に実行できることによってパイプライン方式に見られた隘路というものは存在しなくなっ

たと言える．

4.4 トランザクション処理の開発・保守環境への適用性

一つのトランザクション・プロセッサが取り扱うトランザクションの種類は多種にわたることがある．このため複数のトランコードが一つのトランザクション・プロセッサに経路付けられることになる．この場合，トランザクション・プロセッサがトランザクション種類の識別を行うことが必要になる．COMS ではコンテキストであるトランコードには，Module Function Index（MFI）と呼ばれる一意の整数を事前に割り当てることができる．これにより，入力トランザクション上は意味のある文字列をトランコードとして使用し，プログラム内ではこの MFI を使用して処理手続きの分岐の索引値として扱うことができるので，トランザクションの種別をキーにしてモジュールの呼び出しを行う処理手続きを作成できる．たとえば，ALGOL における手続き参照配列の Procedure Reference Array の索引値にこの MFI を使用すれば，トランザクション種別の判断を行わないで処理手続きを呼び出すことができる．ただ，こういった使用法については徐々に取り入れられつつあるがまだ一般的ではない．柔軟性の面からもこういった COMS の良さを生かしていくべきであろう．

トランザクション処理環境は，業務変更等によりステーションの追加や変更，それに伴うトランザクションの形式変換手続きの追加削除，トランザクション・プロセッサの追加削除や複写数の変更など，多岐にわたる保守が発生する．COMS は，これらの対応を制御ファイルの保守によって行うことができる．すなわち，COMS のプログラムは何等変更することなく対話方式で動的に上記のような修正を加えたり照会することができるし，変更すべきパラメタをあらかじめ作成してバッチ的に制御ファイルを変更したり，現在の状態をパラメタファイルに出力することもできる．このように COMS は，トランザクション処理環境の設定が動的かつ簡易に行えるし，テスト環境や本番環境を保守する上でも設定をあらかじめ確認できるので，トランザクション処理環境の変更に容易に対応できる．

5. 今後のトランザクション環境

5.1 現状の課題

現行の COMS は，ホストシステムに接続されているステーションからのトランザクションを，ステーションの接続形態に依存しないですべて同じ形態のトランザクションとして，トランザクション・プロセッサが取り扱えるような機能を有している．しかしながら，COMS には，トランザクションが他のシステムと同期してデータベースを更新したり，他のシステムでトランザクション処理を並列に行わせ，その同期性を持ってトランザクション処理を完結させるといった機能は持ち合わせていない．複数の A シリーズ間の分散処理システムを構築する場合，ホスト間を通信する機能は BNA によって提供されているので，利用者はその機能を用いて LINC によるアプリケーションレベルで分散処理システムを構築するか，あるいは，分散処理のための基盤システムを作成し，そのソフトウェアとトランザクション・プロセッサがインタフェースを行うようなシステム構築を行っているのが現状である．

また，分散処理環境における一つのシステムの障害が，分散処理環境全体の処理に

影響を及ぼす．したがって，複数のシステムの環境を利用し，障害の発生したシステムに換わる代替のシステムでトランザクション処理が続行できることが望まれる．

本章では，今後のトランザクション環境としての分散トランザクション処理において望まれるCOMSの姿を展望する．

5.2 分散トランザクション処理

まず，LINCの機能を用いた分散処理の現状を紹介する．LINCを使用している場合には，図11にあるようなソフトウェア環境としてHUBROUTERと呼ばれるプログラムがあり，BNAを介してホスト間通信を行っている．COMS制御下のトランザクション・プロセッサとしてのLINCプログラムは，HUBROUTERを経由して他のホストシステムのLINCのCOMSトランザクション・プロセッサと通信できる．この形態の中で一つのトランザクション処理として，ステーションの接続されているホスト(すなわち，入力メッセージを受け付けたホスト)から他のホスト上のデータベースを更新し，自ホスト上のデータベースも更新する処理を行う．このような処理を行うには，二相コミットメントのようなホスト間の同期制御が必要となるので，その制御を両方のホストシステム上のHUBROUTERが監視していて，二相コミットメントの制御を行っている．COMSトランザクション・プロセッサは，他システムとのメッセージの入出力のために，HUBROUTERの呼び出しを行うのみで二相コミットメントを意識する必要がない．しかしながら，この機能は，二つのホストシステム間の機能であり，また図12にあるように一方のホストが処理している間は，他方のホストはその処理が終了するまで待たされる．

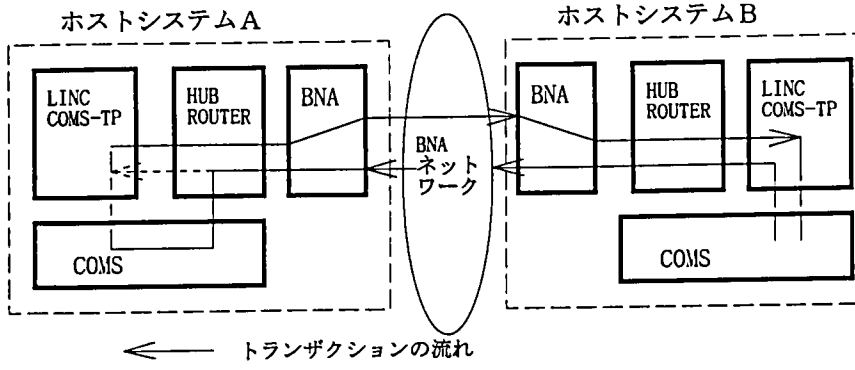


図 11　LINCの分散処理の構成図

しかしながら，分散環境におけるOLTPの利用形態は，LINCが対象としている範囲を越えた広がりを見せている．今後のOLTPの処理形態としては，トランザクションを受信したシステムで完結する場合，受信したシステムが処理しないで他のシステムで処理させる場合，複数のシステムで各々処理を行いその結果を得て受信したシステムが処理を行う場合，一つのトランザクションが業務処理を行うシステムとは異なるシステム上にあるデータベースを使用して処理する場合などが考えられる．さらに，このような処理形態は，多種多様な業務を統廃合して複数のシステムに分散して処理させる水平分散や，部門と全社情報という垂直分散が複合し，ライトサイジングとし

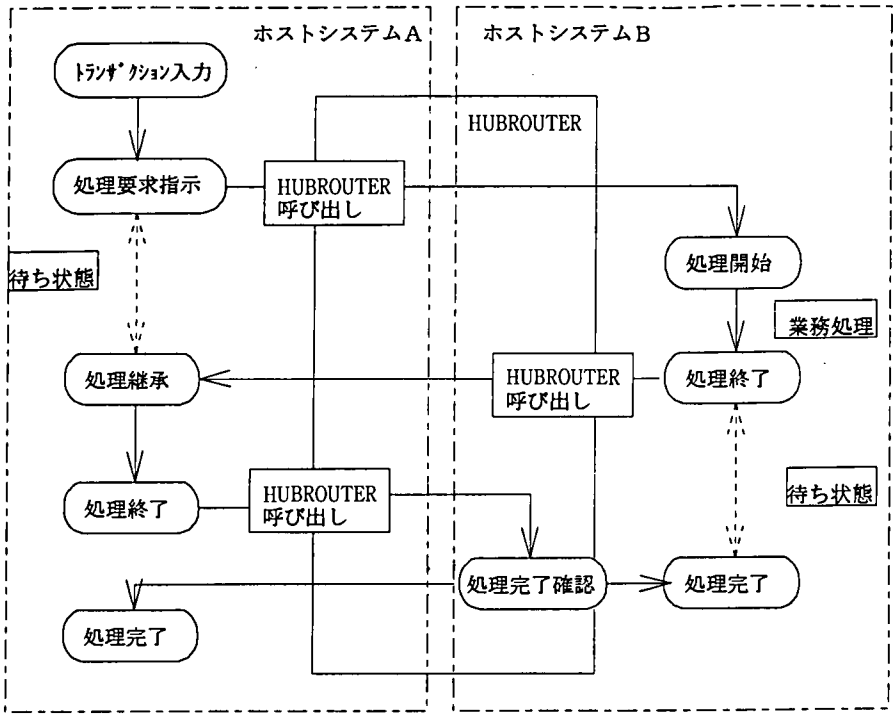


図 12　HUBROUTER の呼び出し

ての役割が遂行できるシステム作りに基づいている．このような処理形態を支援するためには，分散データベースアクセスができることや，ホスト間のメッセージ経路を動的に決定でき，その監視が可能となる基盤システムが必要となる．とくに，分散環境における経路指定や対象業務システムとシステム間のプロトコルに関しては，X/OPEN*（UNIX を中心としたの標準化団体）の分散トランザクション処理モデルや ISO（国際標準化機構）の OSI（解放型システム間相互接続）の応用層のトランザクション処理の標準化に対応できていなければならない．

A シリーズの場合，TP モニタの COMS が，分散環境の経路管理を行う必要があり，データベースの分散アクセスは，DMS II で実現すべき機能である．これらの機能は，ユニシス・アーキテクチャ（UA）としての Open/OLTP に基づき提供されなければならない．

5.3　分散トランザクション処理の継続的サービスの提供

分散トランザクション処理ができる環境では，クライアントの要求を受け付けたシステム（以下「サーバ」と呼ぶ）がその処理結果応答を返す役割を負う必要がない．要求トランザクションは，受け付けたサーバ上で，複数のトランザクション（分散トランザクション）に分解され，各々のトランザクションが対象のサーバに渡され，各サーバ上で処理が行われる．要求トランザクションとしての処理が完了したならば，それを制御したサーバが対象のクライアントに戻すことになる．このような処理形態の一つとして図 13 のようなサーバの機能分離が考えられる．

* X/OPEN は X/OPEN 社の登録商標である．

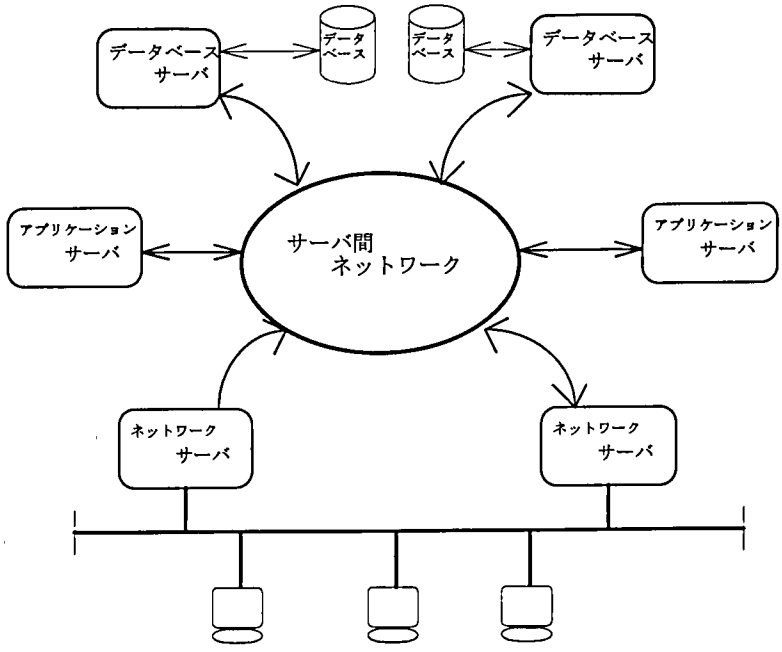


図 13　機能別サーバの構成例

図13の例では，ネットワークサーバがステーションからの入力トランザクションを受け，そのトランザクションを内部形式に変換後，どの業務処理かを判断して対象のアプリケーションサーバに渡す．アプリケーションサーバから受けた出力トランザクションを指定されたステーションに対して出力メッセージに変換後送信する．アプリケーションサーバは，業務処理を主体に行うサーバで，データベース操作指示をデータベースサーバに要求しながら処理を行う．データベースサーバは，要求されたデータベース操作に基づいてデータベース処理を行い要求されたサーバに対してその結果値を戻す処理を行う．

機能を分離した形態の分散トランザクション処理環境では，各サーバ間の制御や処理のためのトランザクション（グローバル・トランザクションと呼ばれる）と，従来のクライアント・サーバの形式のトランザクションを共に取り扱わなければならない．現行のグローバル・トランザクションの主要なものに，データベースの分散化に伴うものがある．これは，データベースのリモートアクセスがデータベース管理システム(DBMS) 内に取り込まれていないために，TP モニタの機能としてグローバル・トランザクションを発生させて他のサーバに処理をさせる方式である．たとえば，LINC の HUBROUTER を使用した処理のように他のサーバ上のデータベースをアクセスし，その結果を得る形態である．

分散トランザクション処理環境の中での障害回復処理を考えたとき，いずれのサーバも一つのトランザクションからみれば必要なサーバになるので，障害が発生した場合には，代替処理を他のサーバで可能となるような仕組みが要求される．この切り替えは，障害が発生した処理中のトランザクションは取り消されて再処理となるが，他

のトランザクションについては，代替処理を行うサーバが処理できるような無停止処理を考慮した形態をとるべきである．この形態がとれないと他の機能処理を行うサーバに対しての影響が発生して分散している意味がない．このため，各機能サーバは，一つの機能サーバが障害を発生させたときその代替として他のサーバが処理を行う機能を持たねばならない．図14に障害の切り替え例を示す．

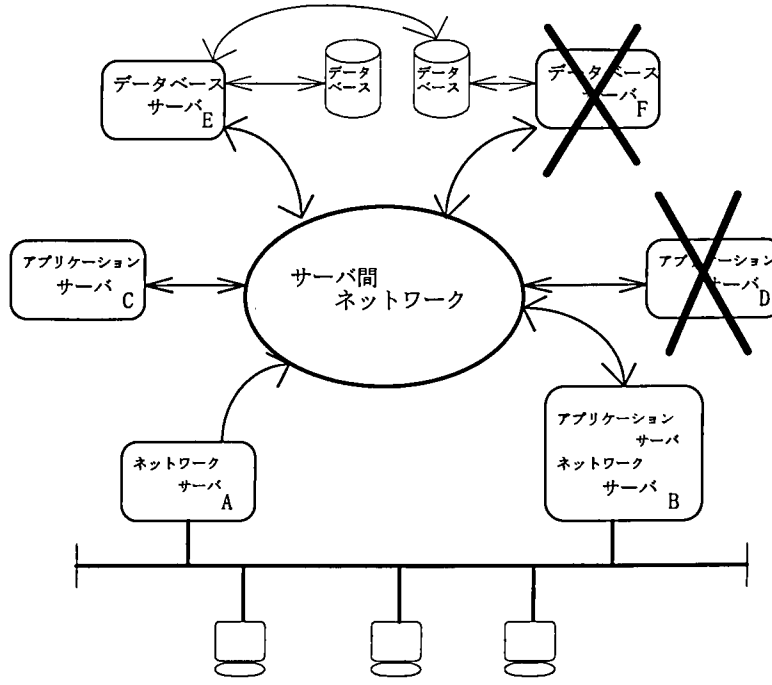


図 14　各サーバ障害の切り替え例

図14では，アプリケーションサーバのシステムDが障害を起こした場合に，ネットワークサーバのシステムBがその代替処理を行うために，アプリケーションサーバとネットワークサーバを兼ねる．また，データベースサーバのシステムFが障害を起こしたときには，システムFが管理していたデータベースをシステムEのデータベースサーバが管理しシステムFの代替処理を行う．このような仕組みであれば，メインフレームで用いられるホットスタンバイ・システムもサーバ単位に構築すればよく，無停止の対応も可能になる．これは，機能別サーバによる分散システムに限らず，グローバル・トランザクションが発生する分散システムにおいて可能になる．

このようなシステム構築を考えると，分散トランザクション処理を行うシステムでは，各システム間での相互運用性や移植性がなければならない．また，各サーバのシステム規模に応じた機種の設置が可能となり，サーバ単位に処理量の増加の対応ができサーバの機種を増強することで可能になるシステム構成が現実のものとなる．

Aシリーズを基にした分散トランザクション処理システムの構築は，COMSのサーバへの動的な経路管理に対する機能拡張やデータベースに対する分散アクセスが必須

となる．今日，すでに提供されているデータベースのシステム間での相互バックアップ機能やPOSIX準拠による移植性の向上などの機能と統合され，分散トランザクション処理におけるサービスの継続的な提供を早期に望むものである．

6. おわりに

Aシリーズにおけるトランザクション処理を行うにあたって，COMSをTPモニタとして位置づけ，OLTP環境のもとにおけるCOMSの提供機能がどのような考えに基づくものか，それの実装技術がどのようなものであるかを提示した．また，最近の業界動向から見た分散処理形態を実現しようとした時には，LINCシステムが持つような外付けの基盤システムを用意しなければならないといったCOMSの機能不足も提示した．しかしながら，分散システムについては，UAで位置づけられているOpen/OLTPの仕様がCOMSの機能として提供が予定されており，機能別サーバの構想も考えられているので，今後のCOMSの機能強化に期待する．

参考文献

[1] 杉山敬三，小花貞夫，鈴木健二，“OSI TP（トランザクション処理）プロトコルソフトウェア　の設計と評価”，情報処理学会論文誌，Vol 34，No 5，1993年5月．

[2] Donald. J. Gregory “Transaction Processing and COMS” Gregry Technical Journal, 1992年10月．

[3] 長谷川聡，阪田史郎，“分散処理の高信頼技術”，情報処理，Vol 28，No 4，1987年4月．

[4] 佐藤健，“トランザクション処理の標準化動向”，情報処理，Vol 28，No 4，1987年4月．

執筆者紹介

森　　良　行（Yoshiyuki Mori）

1951年生．1975年上智大学理工学部数学科卒業．同年日本ユニシス(株)入社．金融機関の客先サービスに従事した後，Aシリーズの利用技術サービスに従事する．現在，システム技術第二本部利用技術二部に所属．

原　　広　仁（Hirohito Hara）

1958年生．1981年京都産業大学経営学部卒業．同年日本ユニシス(株)入社．流通系ユーザにおける利用技術支援を経て，現在関西支社システム技術部に所属．

統合運用システム 拡張IOFの概要

An Overview of IOF (Integrated Operating Facility)

今 西 秀 文

要 約 統合運用システム(拡張IOF)はシリーズ2200の運用全般を支援するソフトウェアであり，単一ホスト環境だけではなく，複数コンピュータ・システムを効果的に運用するための支援機能を実現している．

拡張IOFは，IOF/BASE-II(IOF共通機能)，IOF/MONI(各種障害監視，稼働状況監視)，IOF/WORK(バッチ業務のスケジュール管理と実行管理)，IOF/MASS(マスストレージ・ファイルのバックアップ管理)，IOF/RLS(プログラムのリリース管理)のサブシステムで構成され，サブシステム単位に独立して使用することが可能であるとともに，拡張IOF全体として有機的に結合された運用が可能である．とくに，運用情報の管理(SIB/BASE)とユーザ・インタフェース(運用コンソール・システム)を各プロダクトから独立させることにより，統一された管理体系と操作性を実現している．

Abstract Designed as a software tool to support the whole operation of the Series 2200, IOF (Integrated Operating Facility) allows the effective management of systems operation in both single-host and multi-host environments.

Consisting of five subsystems such as IOF/BASE-II (common basic IOF functions), IOF/MONI (for the centralized monitoring and control of systems faults and computer operation), IOF/WORK (for workflow management) as well as IOF/RLS (for software release management), IOF enables each of these subsytems to run independently, and also makes it possible for all of them to combine together according to user needs. IOF is an integrated operation management system which provides improved operability through its presence independent of SIB/BASE and systems-monitoring consoles (user interfaces).

1. はじめに

コンピュータ・システムの運用管理は，日々の運転計画の作成からバッチ業務の実行や障害発生の監視/回復作業，稼働状況の評価まで多岐にわたりながら，有効な支援システムの欠如から，依然として要員のスキルに頼ることが多い状況にある．当社も，RECS (Run Execution Control System)，SMOCS (Standard Magnetic tape Operation Control System) 等，この分野に対する支援システムの提供を行い，一定の成果をあげてきた．COSTAR (Computer Operating management System for Total Automatic & Reliance) を中心とするコンピュータ運用総合自動化ファシリティ (Unattended Operation Facility，以降UOFと称す) は現在も100を越えるユーザで使用されている．しかしながら，UOFは本質的には個別ツールの集合体であり，ツール間の連携が薄く，管理体系/ユーザ・インタフェースの整合が取れていないこと，使用者独自の運用システムとの連携が困難であること等，コンピュータ運用全般を支援する上で多くの課題を残したままであった．

一方，ユーザ状況を見るに，近年のコンピュータ・システムの大規模化や分散処理の進展に伴い，コンピュータ・システムの運用はますます複雑化している．これに反し，運用/運転要員の若年化や外部委託化が進み，高スキル要員の確保が困難になってきており，運用管理システムに対する要求も高度かつ広範囲に及んできている．すなわち，①コンピュータ運用の全範囲にわたる支援の実現，②複数コンピュータ・システムを効果的に運用するための支援，③習熟を必要としない親しみやすいユーザ・インタフェースの提供，④最新のコンピュータ処理能力に対応したシステム・リミット値の大幅な拡大等，運用支援システムの根幹を見直す必要の高い要求である．統合運用システム (Integrated Operating Facility，以降拡張IOFと称す) は，UOFの不足を補う形式で第一版 (レベル1R) を1991年にリリースし，その後中核となるバッチ業務のモデル化を見直した第二版 (レベル2R) を1993年にリリースした．本稿では，拡張IOF (レベル2R) の基本概念を説明し，合わせて構成サブシステムの概要紹介を行う．

2. 概　　要

2.1 特　　徴

拡張IOFはシリーズ2200の運用全般を支援するソフトウェアである．拡張IOFは，コンピュータ運用を統合的に支援することを目的に，以下の考えに基づき開発を実施した．

1) 統合パッケージ……コンピュータ運用全般を支援するパッケージ・ソフトウェア群であり，個々のソフトウェア・プロダクトごとに独立して導入・運用が可能であるとともに，拡張IOF全体として密に連携した運用が可能である．とくに，運用情報の管理 (SIB/BASE) とユーザ・インタフェース (運用コンソール・システム) を各プロダクトから独立させることにより，一体化した使用感を実現する．

2) 複数ホスト管理……XTPA (eXtended Transaction Processing Architecture)[1]により結合されたホスト群やネットワーク結合されたホスト群を管理・制御可能な運用環境の提供を行う．複数ホストを単一の制御点から監視/制御/管理する機構を提供し，システム運用の分野に対するシングルシステム・ビューを実現する．

3) ユーザ・インタフェースの充実……MS OS/2*プレゼンテーション・マネージャをベースにした運用コンソール・システムを使用することにより，ビジュアルで親しみやすいユーザ・インタフェースを提供する．

2.2 ソフトウェア構成と主要機能

拡張IOFは，図1に示すように，以下のサブシステムから構成されている．

IOF/BASE-II (共通機能)：

SIB/BASEや運用コンソール・システムとのインタフェース，およびホスト間パスの管理等，拡張IOF各サブシステムが作動するために必要となる共通機能であり，拡張IOFを使用する上で必須のサブシステムである．また，拡張IOF

* MS OS/2は米国マイクロソフト社の登録商標である．

の管理情報を保持する運用統合データベースが含まれる．

IOF/MONI（集中監視制御）：

ホストおよび構成機器の障害状況やリソースの使用状況の監視を行い，拡張IOFの他サブシステムを含めた監視情報を収集し，運転員への通知を行う．IOFエリア内のコンピュータ・システムを単一の制御点から監視・制御する機構を提供する．

IOF/WORK（ワークフロー管理）[2]：

定型的なバッチ業務の走行計画の作成からジョブの自動起動，およびジョブ使用ファイルの管理，バッチ運行実績の管理までを実施するバッチ運用の総合管理を行う．

IOF/MASS（マスストレージ管理）：

ディスク上に恒常的に存在するファイル群（データベース，プログラム・ファイル等）の保全を中心とした，ディスク運用の支援を行う．

IOF/RLS（リリース管理）：

ソフトウェアのリリース作業に伴う，リリース対象ソフトウェアの受付/配布/入替えを一連の作業と捉えその管理および自動実施機能を提供する．

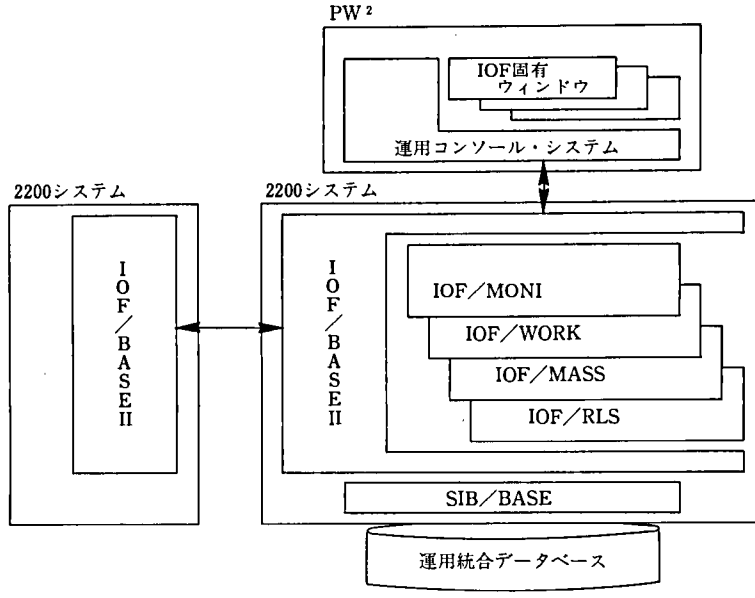


図 1　拡張IOFのソフトウェア構成

他に，必須の関連ソフトウェアとして以下のものが存在する．

SIB/BASE（SIB管理　基本機能）：

拡張IOFの各種サブシステムで使用される運用情報を一括して管理する運用統合データベースに対する操作インタフェースとユーティリティであり，サブシステム間で共通する情報を共有し，運用情報登録作業の統一性/一貫性を実現している．

運用コンソール・システム：

拡張IOF各サブシステムの共通ユーザ・インタフェースとして使用されるPW²上のソフトウェアであり，マウスおよびウィンドウ操作を中心とした統合化された使用者インタフェースを実現する．

3. 基本概念

3.1 エリア管理

拡張IOFでは，複数コンピュータ・システムを効果的に運用するために，管理下の全ホストを唯一のホストで集中管理する方式を採用している．このとき，拡張IOFによる集中管理が実施されるホスト群を総称してIOFエリアと呼ぶ（図2）．

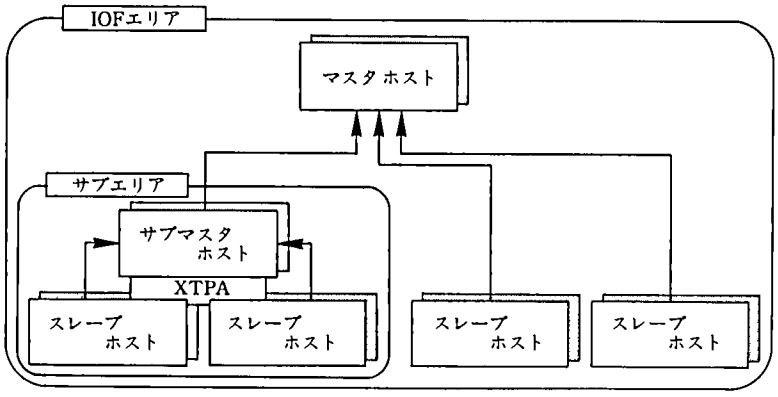


図2 IOFエリア管理

1) IOFエリアとサブエリア

IOFエリアには唯一のマスタホストが存在し，マスタホストはエリア内の全ホストの管理を実施する．マスタホストによる管理の対象となるホストを，マスタホストと対比してスレーブホストと呼ぶ．XTPAにより結合されたホスト群の場合，ファイルの共有やジョブ実行ホストの可動性などネットワーク接続のホスト群よりも密接な連携を行う必要があるため，XTPA結合されたホスト群でサブエリアを形成する．サブエリアはIOFエリア内を分割し独立した制御を可能とする単位であり，サブマスタホストがマスタホストの機能を代行する．マスタホストはサブマスタホストを介してサブエリア内のホストの管理を行う．

なお，XTPAにより結合されたホスト群からなるサブエリアの場合，サブマスタホスト障害時のリカバリを容易にするため，実構成ホストの他に仮想ホストを設定しサブマスタホストの役割を持たせる．

マスタ/サブマスタ/スレーブホスト上のIOFは，IOFエリア内の位置付けに従い機能を分担する．機能分担の典型を，IOF/MONIおよびIOF/WORKを例にして，表1に示す．

2) イベント/コマンド制御

IOFエリア内のホスト間での情報交換や，拡張IOFプログラム間の情報交換のために，イベントおよびコマンドの2種類のメッセージ形式が存在する（図3）．

イベントとは，拡張IOFの管理対象に関する状況変化を表すメッセージであ

表1　IOFエリア内の機能分担

	IOF/MONI	IOF/WORK
マスタIOF	全ホストの監視状況を運用コンソール・システムを介して集中表示する.	IOFエリア内の全ホストのスケジュールを集中作成する. また, 運用コンソール・システムを介して全ホストのバッチ進捗状況を表示し, オペレータ指示を下位ホストに伝達する.
サブマスタIOF	管理下の各スレーブホストの監視状況をマスタホストに通知する.	ジョブの起動条件の管理を行い, 起動可能なジョブの実行ホストの決定とジョブの実行を行う. また, サブエリア内の共通資源(共用ディスク, テープ)の管理を行う.
スレーブIOF	リソース稼働状況の採取や障害発生の検出を行い, 上位ホストに通知する.	ジョブ内のランの順次起動とラン終了状況の判定を行い, 上位ホストに通知する. また, ホスト内のローカル・ディスクの管理を行う.

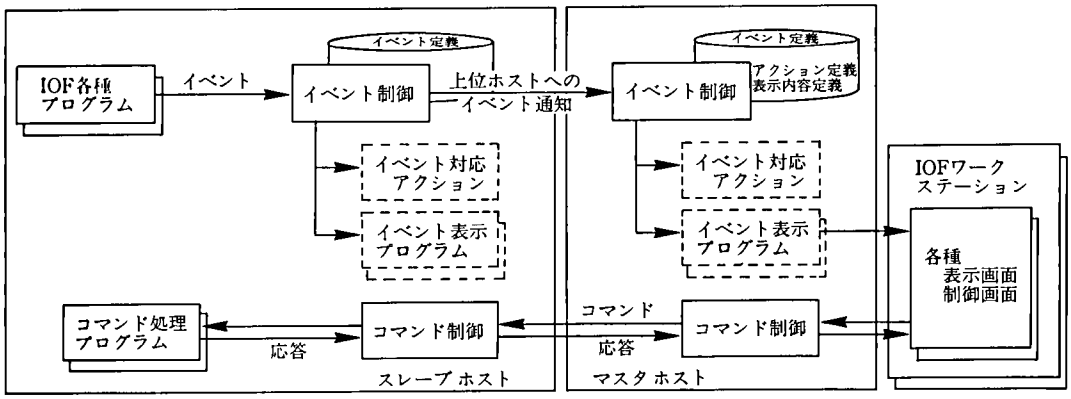


図3　イベントとコマンド

る. イベント・メッセージはイベントの送出者が直接的に制御内容を指示するのではなく, イベント自体に定義されている手続きに従い制御内容が決定される. イベント・メッセージは, IOFエリアの管理体系に従い, 発生ホストから直上位ホストをへて順次マスタホストまで届けられるとともに, 各ホストにて対応する処理が実行される. このことは, イベント・メッセージ自体に定義されている制御手続きに基づき決定される.

コマンドとは, 拡張IOFの特定機能に対し制御を依頼するためのメッセージであり, あらかじめ処理主体が意識されていること, およびIOFエリアの構造に依存しない要求者と処理主体との直接会話であることに特徴がある. コマンド処理は, 原則として同期処理である. IOFコマンドの実行はコマンド・テキストを直接処理プログラムに渡すことで可能だが, コマンド制御処理を介在させることにより, 処理プログラムを意識せずにコマンドの実行を指示することが可能となる. IOFワークステーションを使用した検索・制御操作は, 主にコマンド処理として実行される.

3.2　拡張IOFの管理情報

運用支援ソフトウェアでは, バッチ業務やバッチ業務の実行に必要となるファイル/

プログラム/JCLなどの実行環境からコンピュータ・システムやディスク/テープ媒体まで，数多くの管理対象を含んでいる．拡張IOFは，各サブシステムを使用するために必要となる情報を運用統合データベースにより管理する．運用統合データベースは拡張IOFの各サブシステムで使用される運用情報を統合して管理するデータベースであり，サブシステム間で共通する情報を共有し，運用情報登録作業の統一性/一貫性を実現している．

運用統合データベースで管理される情報には，使用者があらかじめ設定しなければならない基本定義情報と，拡張IOF各サブシステムにより生成されるサブシステム作業情報が存在する．拡張IOFで管理される情報の論理構造を図4に示すとともに，主要な管理単位に対する紹介を行う．

1) アプリケーション/サブシステム情報

アプリケーションとは業務内容，管理者や主管部門などにより分類された業務処理の集合であり，一般には使用者のアプリケーション・システムを意味する．アプリケーションは拡張IOFにより管理されるソフトウェア資産の所属を示す単位であり，プログラム，データ（ファイル），ジョブ/ワーク，などは特定のアプリケーションに所属する形式で管理される．アプリケーションはIOF/WORKにおけるスケジュール作成の独立した作業場所として使用される．スケジュールの作成/変更作業はアプリケーション単位に実施されるものであり，異なるアプリケーション間では独立した制御が可能である．

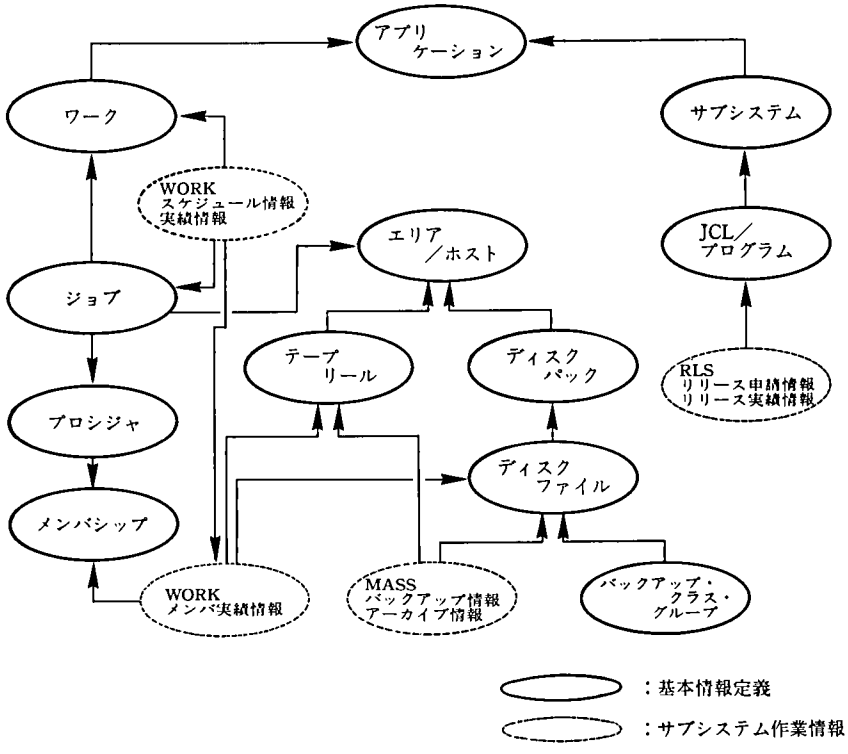


図4 運用統合データベースの管理情報

サブシステムとはアプリケーション内の詳細なメンテナンス単位を意味し，IOF/RLS におけるリリース作業の受付け単位として使用される．

2) ワーク/ジョブ/プロシジャ情報

IOF/WORK により管理されるバッチ業務は，ワーク/ジョブ/プロシジャの3階層の体系に基づき管理される（図5）．

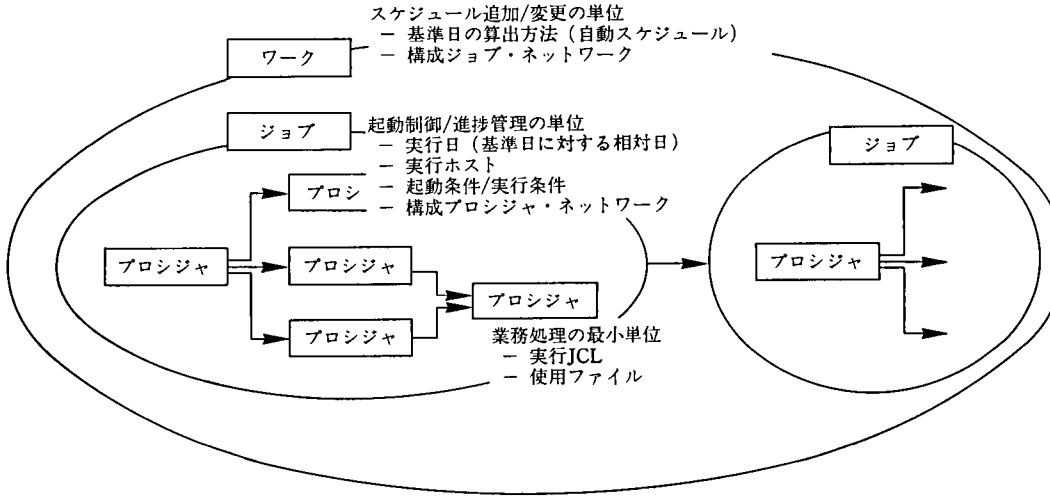


図 5 ワーク/ジョブ/プロシジャ

ワークとは，たとえば勘定系システムの月次処理というような，アプリケーション内で意味のあるひとまとまりの処理単位（プロシジャ）の集合に対する管理名称である．ワークはスケジュールの作成/操作の単位として使用される．ワークは，一連の作業を実行する場合の基準となる日程（基準日）を決定する条件と，処理対象となるプロシジャ群（構成ジョブのネットワーク）を定義したものとして表現される．一つのワークに対し複数の基準日条件と対応するジョブのネットワークを定義可能であり，複数の処理日と複数の実行ホストにまたがって定義することも可能である．ワークを構成するプロシジャのすべてがひとまとまりのネットワークを形成する必要はなく，独立した複数のネットワークであってもよい．

ジョブとは，特別な条件なしに連続して実行し得るプロシジャ・ネットワークであり，運行管理における起動制御/進捗管理の単位として使用される．ジョブは構成プロシジャのネットワークと該ネットワークを実行するために必要となる条件（起動条件/実行条件）により表現される．ジョブは特定の単一ホスト/単一処理日内に実行し得る形態であらねばならない．

プロシジャとは，一つまたは複数のプログラムとその実行手続きからなる業務処理単位をいう．プロシジャは IOF/WORK における最小管理単位として，使用プログラムおよび使用ファイルとその入出力区分を保持している．プロシジャはジョブの再起動ポイントとして使用され，ジョブがエラー停止した場合，エラーとなったプロシジャから再開始される．

プロシジャには，使用者の業務処理内容を定義した一般プロシジャのほかに，

IOF処理を定義したバックアップ・プロシジャ（IOF/MASS）やリリース・プロシジャ（IOF/RLS）が存在する．

シリーズ2200では，ジョブはランのネットワークとして定義される．ラン・ネットワークはジョブ内のラン実行順序を定めるものであり，他ジョブのランへのネットワークは許されない（図6）．

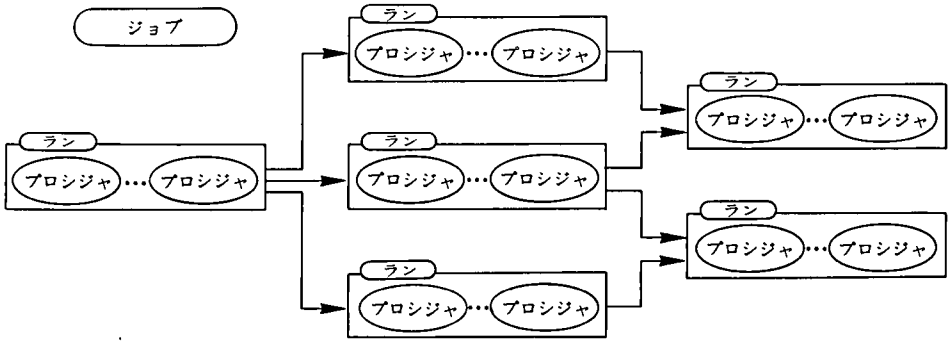


図6 ジョブ/ラン/プロシジャ

3) ファイル/メンバシップ情報

スケジュール・ジョブで使用されるファイルには，ジョブの実行中のみ使用される一時的なファイルから次回処理のために累積・保存されるファイルや他業務との連携で使用されるものなど，種々の使用形態とライフサイクルが存在する．それらのファイルの中には，データベース等恒常的に存在するファイル（パーマネント・ファイル）とともに，多くの中間ファイル等，ジョブ/プロシジャの実行中のみ必要であり特定のジョブ/プロシジャの実行に伴い生成され終了時には消滅するファイル（プライベート・ファイル）が存在する．他に，累積ファイルや長期保存ファイル等，ワーク/ジョブのスケジュールとは独立してファイルのライフサイクルが管理されるファイル群（メンバシップ・ファイル）が存在する（図

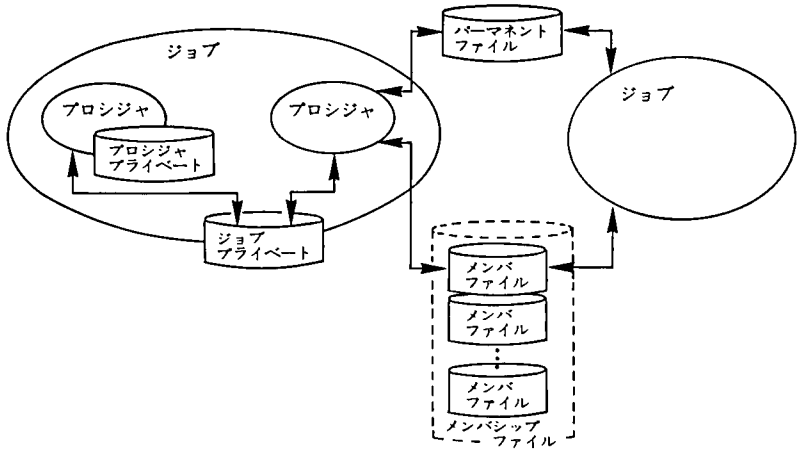


図7 ファイル管理体系

7).

パーマネント・ファイルとは，移動型ディスク・パック上に恒常的に常駐するファイル群であり，EXECファイル(プログラム・ファイル，MSAMファイル等)，FCSSファイル，UDS-TIPファイル，UDS-EXECファイル等が存在する．パーマネント・ファイルに対しては，IOF/MASSにより，バックアップの管理が実施される．

スケジュールされたジョブの実行に伴い，生成/消滅が管理される中間ファイル(ディスク・ファイル）をプライベート・ファイルと呼び，IOF/WORKにより管理される．プライベート・ファイルは，ファイルの有効期間に基づき，プロシジャ・プライベートとジョブ・プライベートに分類される．プロシジャ・プライベートとはソート用スクラッチのように特定のプロシジャの実行時のみに使用されるファイルであり，ジョブ・プライベートとはジョブ内のプロシジャ間で参照/更新が可能なファイルをいう．

メンバシップ・ファイルは，ファイルのライフサイクルがファイルを作成したジョブとは独立して管理されるファイルである．管理的には特定のアプリケーションに所属するが，共用ファイルとしてすべてのジョブから作成/参照/更新することが可能である．メンバシップ・ファイルは，1論理ファイルに対し複数の物理ファイル（メンバ・ファイルと呼ぶ）を作成することが可能である．メンバシップ・ファイルは，個々のメンバを管理するために，論理ファイル一意名（メンバシップ識別名）とメンバ・ファイルの作成順序に従った順序数を持つ．メンバシップ・ファイルは，作成したジョブの実行期間とは独立した有効期限を持つ．

4) バックアップ・グループとバックアップ・クラス情報

バックアップ・グループとは，パーマネント・ファイルの集合であり，パーマネント・ファイルに対するバックアップ作業の単位を規定する．使用バックアップ・ツール，ファイル種別，保存期間，アプリケーション・グループが同一のファイルの集まりであり，バックアップ作業は基本的にこの単位で実施され，バックアップ実績として保存される．なお，一つのファイルが複数のバックアップ・グループに所属することが可能である．

バックアップ・ツール，保存管理方法等が同一であるバックアップ・グループをまとめて，バックアップ・クラスとして定義が可能である．バックアップ・クラスに含まれるバックアップ・グループの省略値は，このバックアップ・クラスに定義されたものがとられる．

4. サブシステム概要

4.1 IOF共通機能（IOF/BASE-II）

4.1.1 運用統合データベース

拡張IOFを構成する各サブシステムの管理情報を統合管理するデータベースであり，各サブシステムのための唯一の情報源としてIOF使用者との接点を担当するものである．データベース管理システムとしてUDS/RDMS 1100を使用し，以下の付加機能を持っている．

1) データの有効期限管理

データの事前登録や過去の時点に遡ってデータ値を参照することを可能にするデータの有効期限管理が使用可能である．データの検索時に検索対象日付を指定することにより，指定日付に対応したデータのみが検索できるよう考慮されている．この機能により，変更情報の事前登録が可能であり，再処理時のデータベース情報の整合性を確保することができる．

2) メソッド

データの登録/更新等のタイミングで起動され，データの検証や関連項目の設定を行う手段としてメソッドを組込むことが可能である．メソッドは運用統合データベースの各エンティティ単位に登録可能である．データの登録/検索/更新/削除の過程で呼出され，データの参照/更新や別データの登録を行うことができる．また，処理ステータスを呼出しプログラムに返すことが可能であり，使用者独自のデータ検証に利用することができる．

3) ユーザ・カスタマイズ支援

使用者運用に基づき，データベースの論理構造のカスタマイズが可能である．属性項目の追加やエンティティの追加，メソッドの変更・追加が可能である．また，RDMS 1100の機能を直接使用して管理情報の抽出を行うことにより，使用者独自の処理を容易に構築することが可能である．

4.1.2 IOFワークステーション

拡張IOF各サブシステムの共通インタフェースとして，PW²上の運用コンソール・システムが使用される．運用コンソール・システムは，OS/2プレゼンテーション・マネージャの機能を使用して，マウスおよびウィンドウ操作を中心としたグラフィカルなユーザインタフェースを実現している．また，運用コンソール・システムは，ウィンドウ・プログラムの開発を容易にするために，多くのひな型ウィンドウを準備しており，それらを利用して使用者独自のウィンドウを開発することも可能である．

以下は，拡張IOF各サブシステムとの連携により使用可能となるウィンドウ群である．なお，これらのPW²上の運用コンソール・システムをベースに構築された拡張IOFのユーザ・インタフェース機能を総称してIOFワークステーションと呼ぶ．

—ホスト状況・構成機器状況監視ウィンドウ（IOF/MONI）
—イベント通知・制御ウィンドウ（IOF/MONI）
—コンソール表示・操作ウィンドウ（IOF/MONI）
—マイクロスケジューラ（IOF/WORK）
—マイクロオペレータ（IOF/WORK）
—リリース申請・入替え制御ウィンドウ（IOF/RLS）

4.2 集中監視制御（IOF/MONI）

IOF/MONIはホストの稼働状況や構成機器の障害状況を監視するとともに，他サブシステムを含めたイベント情報を収集し，運転員への通知を行う．IOFエリア内のコンピュータ・システムを単一の制御点から監視・制御する機構を提供するソフトウェアであり，複数のコンピュータ・システムを少人数で集中監視制御する運用を支援する．監視結果はIOFワークステーションや音声出力装置に出力可能である．

1) ホスト障害監視

コンピュータ・システムの停止やシステム内の主構成機器（CPU/メモリ/各制御装置/ディスク装置，等）の状態異常の検出を行うとともに，TPS のエラーや，常駐ランやセットアップ/リカバリ・ラン等，使用者が設定した重要バッチ・ランの障害（エラー終了）検出を行う．また，統合オペレーション・システム（CONSOLE）と連携し，システム・コンソールに表示される特定メッセージを監視することが可能である．これらの監視項目や監視インターバルは，ホストの負荷に応じて使用者が任意に設定することが可能である．

2) ホスト稼働監視

システム機器の使用状況（CPU 利用率/メモリ利用率/ディスク装置の IO 利用率，等），トランザクション処理数や走行中のバッチラン数等のワークロードを定期的に監視し，状況の通知を行う．また，採取された各項目の現在値に対し使用者が設定した限界値との検査を行い，警告状態発生の有無を監視することができる．障害監視と同様，監視項目/インターバルは使用者で調整可能である．

3) 集中コンソール表示

統合オペレーション・システム（CONSOLE）と連携し，各ホストのシステム・コンソール・メッセージを IOF ワークステーションに表示することが可能である．複数ホストのコンソール・メッセージを IOF ワークステーションに集中表示するとともに，応答型メッセージへの応答や任意型キーインが実施可能となる．また，発生時刻やメッセージ内容に基づき，過去に表示されたコンソール・メッセージを再表示することが可能である．

4.3 ワークフロー管理（IOF/WORK）

定型バッチ業務の走行計画の作成から自動起動，および使用ファイルの管理，実績の管理までを実施するバッチ業務の総合運用管理システムである．IOF/WORK は従

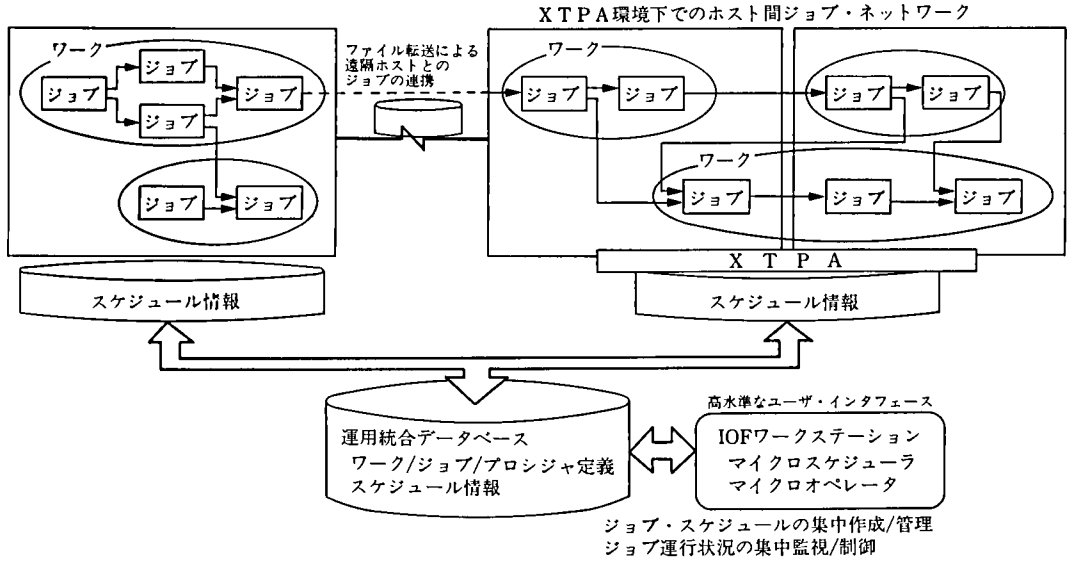


図 8 IOF/WORK

来の運用管理システムに比べ，以下の特徴を持っている（図8）．

—複数ホストのスケジュールの一括作成およびホストをまたがったスケジュールの作成/変更が可能である．

—XTPA運用下における，ホストをまたがったジョブ・ネットワークの制御やホストをまたがったスケジュールの調整を実施できる．

—高水準ユーザ・インタフェースの実現とエンドユーザ・スケジューリングの支援

—24時間・365日連続運転への対応

1) 集中スケジュール管理

IOF/WORKはワーク/ジョブの管理情報をもとに日々のジョブ・スケジュールの自動作成を行う．ジョブ・スケジューリングは，マスタ・ホストにてIOFエリア内の全ホストのスケジュールを一括作成する集中スケジュール方式を採用している．複数ホストのスケジュールを一括作成することにより，スケジューリング作業の集約化・効率化が可能であり，遠隔ホストに対する計画・管理要員を含めた無人化が期待できる．また，XTPA構成におけるホストをまたがったジョブの流れの把握や，ホスト間のデータ交換の整合性の確保等，スケジュールの正確性/精度の向上を図ることができる．なお，スケジュール操作の単位は，一連のジョブ・ネットワークを総称したワークを使用している．

スケジュール作成作業は，スケジュールの検証・補正とスケジュール確定，および確定後の変更作業に分類される．任意期間におけるジョブの実行予定の検索や，実行予定に対する補正を予約する作業をスケジュール検証と呼ぶ．スケジュール検証はアプリケーション単位に独立して実施可能であり，IOFワークステーションを使用した操作やカレンダ上に業務上の特定日を設定することによりスケジュールを補正することが可能である等，運用部門を越えて利用部門が直接スケジュールを操作する方策を提供している．また，スケジュール検証作業を打切り，処理日単位に実行ジョブを特定しジョブ間の実行順序の決定を行う作業をスケジュール確定と呼ぶ．スケジュール確定は暦日単位に連続して，かつ全ホスト/全アプリケーションを一括して実施される．確定処理は運用部門が明示的に実施したり，あらかじめ設定されている確定期間を確保するよう自動的に実施させることが可能である．なお，確定後のスケジュールに対しても検証・補正と同様の手順で変更を行うことが可能である．

スケジュール検証・補正/変更作業のために，マイクロスケジューラと呼ばれる，IOFワークステーションを使用した視覚的かつ高水準なユーザ・インタフェースが提供される．スケジュール検索ウィンドウを中心とするマルチ・ウィンドウ環境にて，マウスを主体にしたポインティング操作により，スケジュール検索/追加/削除やスケジュール基準日の変更，およびジョブ起動条件の変更等の作業が実施可能である．

2) ジョブ実行環境の管理と統合運行制御

ジョブの自動起動は以下の起動条件/実行条件をもとに実施される．

ジョブの起動条件とは，ジョブを起動する上で必須となる条件であり，これらはスケジュール検証・確定時にジョブに付加されたものである．具体的には，所

属アプリケーションの稼働ステータスや先行ジョブの正常終了，入力データの受取りや各種メッセージ条件の受取り完了等があげられる．ジョブの実行条件とは，起動条件を満足するジョブが複数個存在する場合に，起動対象ジョブを選択する条件であり，コンカレンシの満足度やジョブの動的優先度が考慮される．

ジョブは実行可能なホスト環境により，ホスト・ディペンド・ジョブ（HDJ）とエリア・ディペンド・ジョブ（ADJ）に分類・管理される．HDJ とはジョブの実行環境が特定のホストだけに限定されるジョブをいい，ADJ とは，XTPA 構成のホスト下で，実行環境が共有媒体上に存在し，媒体を共有するホスト群のいずれでも実行可能なジョブをいう．ADJ の実行ホストは，標準実行ホストをあらかじめ指定するとともに，スケジュール検証・変更時やジョブ起動時に動的に変更することが可能である．ADJ を使用することによりホストをまたがったジョブ・ネットワークの定義・制御が可能であり，バッチ制御に対するシングルシステム・ビューを実現することができる．ADJ の起動制御を可能にするために，XTPA 結合されたホスト群はサブエリアを構成し，サブエリアを代表するサブマスタ・ホストが全構成ホストのジョブの起動を統合制御する．

ジョブの進捗状況の監視やジョブの再起動等の運行制御操作のために，マイクロオペレータとよばれる，IOF ワークステーションを使用した視覚的かつ高水準なユーザ・インタフェースが提供される．

3) ファイル管理

スケジュール・ジョブで使用されるメンバシップ・ファイルとプライベート・ファイルに対する引当てと，使用媒体の管理を行う．各ジョブは@ASG/@FREE に代わる@UASG/@UFREE プロセッサを使用してこれらのファイルに対する割当て/開放と実績保存要求を行う．

スケジュール・ジョブで使用されるテープ/ディスク媒体はプールに分類されて管理される．プールとは，管理媒体を論理的にグルーピングする単位であり，各ファイルの作成時には指定されたプールに属する媒体が選択される．プールは，利用可能なホストの範囲により，ホスト管理プールとエリア管理プールに分類される．ホスト管理プールは該プールを管理するホストでのみ利用可能なプールであり，プール管理ホストのみが所属リールを使用可能である．エリア管理プールはエリア/サブエリア内のどのホストでも利用可能なプールであり，エリア/サブエリアを構成するホスト間で所属リールを共有することが可能となる．エリア管理プールは，該エリアを管理するマスタ/サブマスタホストにより管理される．

テープ・メンバシップ・ファイルに対しては，ジョブの実行予定に基づき必要なテープ・リールの準備を指示するテープ出庫指示書を作成することが可能である．また，ディスク・メンバシップ・ファイルに対しては，IOF/MASS と連携し，新規に作成されたメンバのバックアップを採取することが可能である．

4.4 マスストレージ管理（IOF/MASS）

IOF/MASS はディスク上に恒常的に存在するパーマネント・ファイル群（データベース，プログラム・ファイル等）の保全を中心とした，ディスク運用の支援を行うソフトウェアであり，以下の機能を持っている（図 9）．

—ファイルの破壊発生に備え，ディスク・ファイルをテープに保存し，保存テープの管理を行う．

—OS 1100 (IR) により採取されるデータベースの更新情報 (ATD) を自動的にテープに保存し，保存テープの管理を行う．

—ファイル破壊の発生時に，上記保存テープをもとにファイルのロードと復旧処理を行うJCLを作成，実行する．

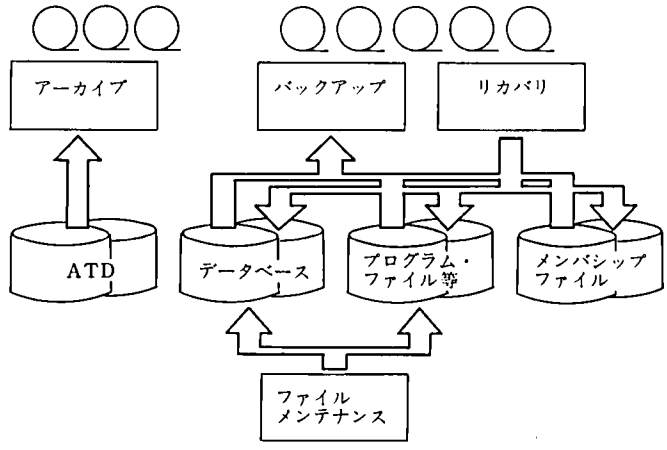


図 9　IOF/MASS

1)　アーカイブ

OS 1100により保存されるATD (Audit Trail Disk) を監視し，テープへのATDのバックアップ(IRU，またはCUPATTプロセッサを用いたアーカイブ処理)を実行する．また，アーカイブの完了したATDをディスク上から削除する．なお，当アーカイブ機能は，拡張IOFが作動するアプリケーション・グループが停止している状況においても実行可能である．

アーカイブに使用されるリールは運用統合データベースでアーカイブ用に登録されたテープ・プールから自動選択され，使用実績が保存される．

ATDのサイクル監視機能において，アーカイブ・ランの起動を行う基準となるATDのサイクル数をリテイン・サイクルと呼ぶ．リテイン・サイクルを越えたサイクル数のATDが作成されると，ATDのサイクル監視機能によって，アーカイブ・ランの起動が行われる．また，ATDのサイクル監視によるアーカイブ・ランの起動の他，業務終了時などにATDを一括してアーカイブする運用も可能である．

2)　バックアップ

パーマネント・ファイル，およびIOF/WORKにより作成されるディスク・メンバシップ・ファイルのバックアップを行う．また，バックアップに使用されるリールとバックアップ実績の保存管理を行う．バックアップ採取ツールとして，FAS 1100, FURPUR, IRU 1100, PALDUM等が使用可能である．ただし，メンバシップ・ファイルのバックアップについては，必ずFAS 1100が使用される．ま

たバックアップ実行以前に使用予定リールの一覧表を出力することが可能である．

バックアップの起動方法は使用者が任意の時点で当機能を起動する随時バックアップ方法と，IOF/WORK にバックアップ・スケジュールを登録し，定期的にバックアップを行う方法の 2 種類が存在する．

バックアップ実行結果と使用されたリール情報を取り込み，指定された期間あるいは世代のバックアップ実績を保存管理する．ただし，ファイル指定のバックアップは期間のみの保存管理を行い，ディスク・メンバシップ・ファイルは最新世代のみ保存管理を行う．

3) リカバリ

当システムで管理するディスク・ファイルに対して障害が発生した場合，指定されたリカバリ指示に基づいてリカバリ JCL を生成する．同時に，使用リールの一覧を印書する．また，指定に応じてファイルの再作成・TIP 再登録の JCL 生成を行う．使用者は，当機能により作成された JCL を任意のタイミングで実行し，ファイルのリカバリを行う．リカバリ対象のファイルを以下のような単位で指定することが可能である．

—バックアップ・グループ単位（グルーピング情報で定義されたグループ）

—ファイル単位

—パック単位

リカバリ対象のパーマネント・ファイルが回復型ファイルである場合，アーカイブ実績より，IRU 1100 を使用したロング・リカバリ JCL を作成する．使用者の指定（リカバリのスタート・ポイント，エンド・ポイント）に基づいて，バックアップ実績，アーカイブ実績からロング・リカバリに必要なアーカイブ・リールを選択する．そして，このアーカイブ・リール，および ATD を使用し，使用者任意の時点（エンド・ポイント）まで復元する．また，ロング・リカバリ時，アーカイブ情報を IRU のヒストリ・ファイルに書き込むことによって，アーカイブ・リールの自動アサインが可能である．

4) ファイル・メンテナンス

パーマネント・ファイル，とくに TIP/UDS ファイルに対する各種運用 JCL/パラメタの作成を行う．運用統合データベースの登録情報を基に，以下の運用 JCL/パラメータが作成可能である．

—ファイルのカタログ/デカタログ

—TIP への登録/TIP からの登録解除/TPASG/TPFREE

—指定されたレグのダウン/アップ指示パラメタ

—指定されたファイルのダウン/アップ指示パラメタ

—FCSS 二重化ファイルの片レグ障害に対するダイナミック・レグ・コピー用パラメタ

4.5 リリース管理（IOF/RLS）

IOF/RLS は，ソフトウェア資産（JCL, プログラム，等）のリリース作業を自動的に実施することにより，運用部門の負荷軽減，人為的な誤り防止を実現する．IOF/RLS

はリリース対象ソフトウェアの受付，配布，入れ替えを一連の作業と捉え，その管理および実施機能を提供する．

ソフトウェア資産のリリース作業は開発部門から運用部門にまたがる作業であり，IOF/RLSではソフトウェアのリリース作業を以下の手順で実施する．それぞれの手順に対し，IOFワークステーションの操作インタフェースが用意されている(図10)．

① 申請（開発部門)……ソフトウェア資産の入替え計画・内容を登録する．
② 編集（開発部門)……入替え対象のソフトウェア資産自体を登録する．
③ 確定（運用部門)……開発部門から登録された申請内容を確認・調整する．
④ 配布（運用部門)……申請時に指定された入替先にソフトウェアを配布する．
⑤ 入替（運用部門)……配布されたソフトウェアを本番環境へ入替える．

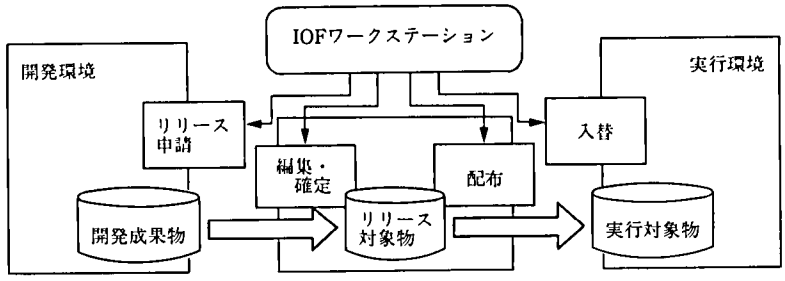


図 10 IOF/RLS

1) リリース申請・編集

開発部門により作成されたソフトウェア資産のリリースを行うため，当システムがリリース作業を実施するのに必要となる各種情報（たとえば，ソフトウェア名/リリース先名/リリース時期等）を登録する機能である．リリース申請はサブシステム単位に実施し，1回の申請にて同一日に入替えが実施される複数資産・複数ホストの登録が可能であり，IOFワークステーションを使用して登録を実施する．以前に実施された申請の内容を参照して操作を軽減することも可能である．また，リリース申請のグループ化が可能である．複数の申請を大きな表題の下に管理し，申請間の作業順序付けや複数サブシステムにわたる申請が可能である．

リリース申請の完了後，リリース対象物の編集を行う．編集とはリリース対象となったソフトウェア資産を開発環境から抜出し，運用環境への配布・入替えを実施するまでの間，一時的に当システム内で保管することを示す．リリース資産の編集完了後は開発環境を切離して運用可能となる．

2) リリース計画の確定

開発部門からの個々の申請をもとに，運用部門にて配布日・入替日・入替条件等のリリース・スケジュールを調整（変更）し，リリース・スケジュールを確定する．確定されたリリース申請は，当システムにより自動配布・入替えの対象として取扱われる．申請確定作業もIOFワークステーションを使用して実行する．確定の方法として個々の申請単位に確定を行う方法と，編集済みのリリース申請をすべて確定する一括確定の方法が存在する．

3) リリース対象物の配布・入替え

リリース申請時に指定された入替え日付・時刻に達すると，すべてのリリース先ホストの指定ファイルに対し，自動的にリリース対象物の入替え作業が実施される．また，リリース先ホストが異なる場合は入替えに先立ち申請情報とリリース対象物の配布処理が実施される．配布作業は入替え実施日以前に実行することも可能であり，すべて申請時の指定に基づく．なお，配布作業はDDA-BASE (DDP; Distributed Data Processing) を使用したファイル転送により実施される．

ソフトウェア資産の入替え処理は．資産の種別ごとにあらかじめ登録されている，リリース・プロシジャにより実施される．リリース・プロシジャとは，IPFプロシジャ（およびIPF SQL）により記述されたプログラム資産の入替え作業手順であり，当システムにより標準プロシジャが提供されるとともに，使用者が任意に変更・作成可能である．

5. 拡張IOFの適用

拡張IOFは以下のような種々のホスト構成での利用が可能である．

—単一ホスト形態

—XTPA構成形態

—ネットワーク構成形態

—XTPAおよびネットワークの複合構成形態

ここでは，既存ユーザでの適用事例を中心に，拡張IOFの利用形態について紹介する．

5.1 XTPA構成ホストの統合運行制御運用

XTPA構成下では各ホスト間でディスク・ファイルを共有することによりバッチ業務をどのホストで実行することも可能であるが，現実的には，ホストをまたがったバッチ処理の流れをオペレータが制御することは困難である．通常の運用ではホスト間の負荷バランスやテープ/プリンタ関係操作の便宜上どのホストでどの業務を実行するかはあらかじめ決められていることが多いが，ホスト障害時や特定ホストで極端な処理遅延が発生した場合などには，バッチ業務の実行ホストを組替えることが可能な運用が必要となってくる．IOF/WORKはXTPAにより結合されたホスト群のバッチ業務（ジョブ）を集中管理することにより，ホストをまたがったジョブのネットワークを制御し自動起動する．ホスト障害時には障害ホストで実行予定であったジョブを他のホストで実行するよう変更したり，オペレータ指示により任意のジョブの実行ホストを変更することが可能である．どのホストでジョブが実行される場合も，1台のIOFワークステーションを使用することにより，全ホスト/全ジョブの進捗状況の監視や起動制御指示が可能である（図11）．

5.2 分散ホストの集中監視制御運用

地域的に離れたホスト群を一か所（マスタホスト）から集中監視・制御を行う運用が可能である．個々のホストは自動運転が実施されるとともに，ジョブの進捗状況や障害状況などはマスタホストに通知され，マスタホストに接続されたIOFワークステ

ーションに通知される（図12）.

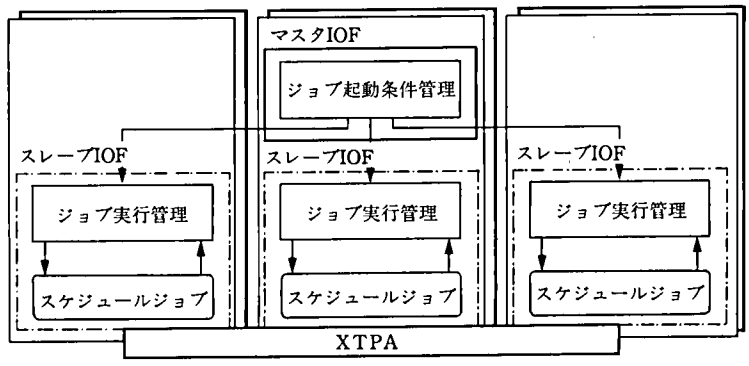


図 11 XTPA構成ホストの統合運行制御

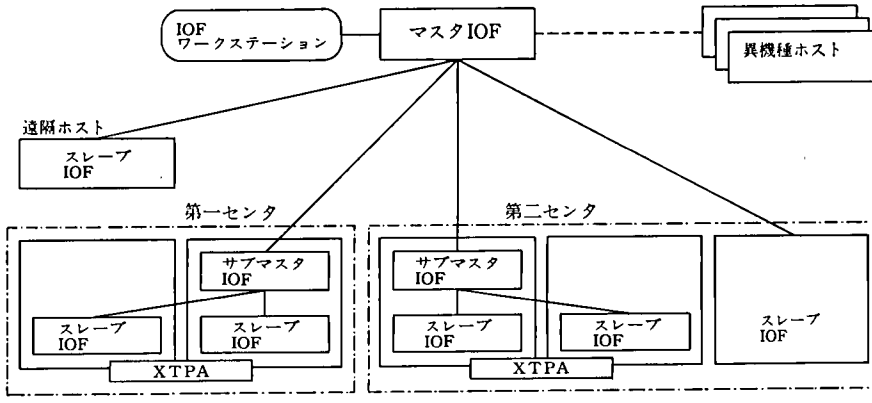


図 12 分散ホストの集中監視制御

6. おわりに

運用支援ソフトウェアの究極の目標は，大胆な要約を行えば，安全性・信頼性の確保にあると思う．機能の網羅性やユーザ・インタフェースの高機能化もそれ自体が重要な課題ではあるが，個々の機能間の整合性・連携が図られ，コンピュータ・システムを安全に運用するための仕掛けとなり得ることがより一層重要である．そのために必要なことは，運用支援システムの管理対象が適切にモデル化されていること，各管理対象物に対する操作と対象物の状態遷移が明確に定義されていることである．統合運用システム（拡張IOF）は，当分野に対するUOF等の実績を踏まえ再構築を行った新しい運用支援システムであり，とくに運用情報の統合管理に意を尽くしたものである．

拡張IOFの今後の課題として，以下の三点がある．

一つは，対象とするコンピュータ・システムの適用範囲の拡大である．コンピュータ・システムの運用環境は複雑・多岐にわたっており，シリーズ2200だけで完結するものではなく，異機種コンピュータ・システムを含めた運用環境全域に対する情報管

理と連携が必要である．次いで，当システムで保持される管理情報・実績情報を活かした評価・計画方面への展開があげられる．コンピュータ運用に関する上流工程である評価・計画業務に対する支援は有効なツールが存在しない未開の分野であり，今後の大きな課題である．最後に，一番重要なことは，実装機能の充実と使用容易性の一層の向上である．

運用支援システムは個々のユーザの個性との調和が必要であり，きめ細かいフォローとともにマニュアル・ガイド等の充実を図ることが必要である．

本稿に有益なコメントをいただいた査読者の方々，当システムを開発するに際し有益な助言をいただいた方々，当システムの初期導入・適用に多大な尽力をいただいた方々に，心から感謝の意を表したい．

参考文献 ［1］ 沢田 啓，“XTPA に基づくノーダウン・システム”，技報通巻 40，日本ユニシス(株)，1994. 2.
［2］ 後 博，“IOF/WORK によるバッチ運用”，技報通巻 40，日本ユニシス(株)，1994. 2.

執筆者紹介 今 西 秀 文 (Hidefumi Imanishi)
昭和 48 年名古屋工業大学卒業．同年日本ユニシス（株）入社．平成元年より運用環境支援プロダクトの開発・サービスに従事．現在，システム企画開発二部運用技術課課長．

IOF/WORKによるジョブ・スケジューリングとユーザ・インタフェース

Job Scheduling and User Interfacing by IOF/WORK

安坂敏秀, 友枝 研

要 約 コンピュータ・システムの高度化・分散化に伴って，バッチ処理の運用も複雑化・多様化の一途をたどっている．運用管理システムに対する要求もXTPA(eXtended Transaction Processing Architecture；拡張トランザクション処理アーキテクチャ）に代表される新アーキテクチャへの対応から高水準なユーザ・インタフェースの提供までと多岐にわたり，従来の運用管理システムでは対応が困難な状況となっている．このような背景のもと，今後のバッチ処理運用を担うべく開発された運用管理システムがIOF/WORKである．

IOF/WORKはこれらの課題に対応するために次の機能を実現している．

1) 集中スケジュール管理
2) 統合運行制御
3) 連続運転モード
4) マイクロスケジューラ／マイクロオペレータ

Abstract In proportion to computer systems getting more advanced and dispersed, batch data processing is steadily becoming more complicated and diversified. Recent user demand regarding systems management tools is for the support by computer suppliers which ranges from software implementations on the lines of a new architecture such as eXtended Transaction Processing Architecture (XTPA) to new availability of higher-level user interfaces, thus making it impossible for traditional systems management products to readily respond to customer needs. Under this pressure, IOF/WORK has been developed as a system destined to meet future batch processing requirements.

With a view to giving the most appropriate solutions, IOF/WORK is so tailored as to provide, as its functions, (1) centralized job scheduling, (2) integrated run control, (3) continued operation control and (4) roles of a micro-scheduler/micro-operator.

1. はじめに

IOF/WORKは拡張IOF (IOF；Integrated Operating Facility) の中で，バッチ処理の自動運行を担当するサブシステムである．バッチ処理の運行はトランザクション処理に比べセンタ・オペレータに依存する部分が格段に高く，当社においても早くからその省力化のためのシステムが作成されてきた．しかし，近年バッチ処理運用に対する要求は，ホストマシンの24時間365日連続運転，XTPAをベースとした複数ホストシステム運用，ワークステーションの高機能化を背景としたより高水準なユーザ・インタフェースの提供と，高度かつ広範囲に及んでおり，従来の運用管理システムでは対応しきれない状況となっている．

本稿では，これからのバッチ処理運用に求められる新たな要求を分析し，その要求に対してIOF/WORKがどのような機能をもって対応したかを記述する．

2. IOF/WORKの概要

2.1 開発の背景

コンピュータ・システムの高度化や広域ネットワーク化とともに，システムの利用形態はますます複雑化・多様化してきている．それに伴いシステム運用部門の作業量も急増し，運用そのものも非常に煩雑化している．こうした状況の中，限られた時間と運転要員のもとで，複雑化したコンピュータ・システムを安全かつ効率的に運用していくことがシステム運用部門の重要課題となっている．

このような背景のもと，今後のコンピュータ運用を担うべく企画された統合運用システム（IOF）は，当社主要ユーザの逼迫する要求を取りまとめる形でフェーズ1（IOF 1 R）が開発された．この中でバッチ運用管理システムにおいては既存の運用管理システム（COSTAR；Computer Operating management System for Total Automation & Reliance）をベースに以下の機能拡張が実施された．

- COSTARの持つバッチ業務管理体系の拡張（COSTAR/WORK）
- 分散処理形態における業務処理連携の強化（COSTAR/DXM）

しかし，次世代の運用管理システムとして具備すべき高度なユーザ・インタフェースや分散処理システムへの対応は十分なものとは言えず，また新しいシステム・インフラストラクチャであるXTPAの取り込みやEM（Extended Mode；拡張モード）環境への対応等，新アーキテクチャの出現にもとずく運用ソフトウェアの体系整備も必須な状況となっていた．

COSTARならびにIOF 1 Rで提供済みの機能を吸収するとともに，これらの課題に対応するための新機能を追加して再構築されたバッチ運用管理システムがIOF/WORKである．

2.2 COSTARの機能と問題点

COSTARは，シリーズ2200のバッチ処理運用を総合的に支援するための運用管理システムであり，表1に示す機能を備えている．これらの機能でバッチ運用全般をほぼカバーしているものの，その管理・制御範囲が単一ホスト内に限定されており，複数ホスト・システムのバッチ運用に適用する場合，次のような問題点が存在した．

1) スケジュールの作成が各ホスト単位に実施されるため，ホストをまたがった業務スケジュールの作成/変更が難しい．
2) 業務処理ホストをダイナミックに変更することが不可能なため，複数ホストを利用した効率のよいバッチ運行が行えない．また，ホスト障害の場合，障害ホストで長時間にわたり業務が中断する恐れがある．
3) 登録情報が各ホスト単位に管理されるため，複数のホスト間で共有するような情報，たとえば，シェアドファイル*やシェアドパック**等の情報が重複して管理されることになり，情報の整合性維持が容易ではない．

また，システムの操作面においても，COSTARはバッチ・ユーティリティ中心のユーザ・インタフェースであるため，ユーティリティの使用方法や業務処理ホスト等のコンピュータ処理に精通していないユーザでは操作が難しいとの問題点も指摘されて

*シェアドファイル：シェアド・ディスク上に作成されたファイル
**シェアドパック：ファイル・シェア（MHFS：Multi Host File Sharing）機能のもとで複数のシリーズ2200システムから共有される磁気ディスク装置

表1 COSTARの主な機能

機　能	サービス内容
スケジュール管理機能	月間スケジュールの自動生成 日別スケジュールの自動生成 スケジュールの変更（追加/削除/処理日変更） スケジュール関連諸表の作成 リラン指示 作業指示書の作成 スケジュールの繰り越し
運行管理機能	ランの自動起動 ランの終了状況の把握と終了処理 ロストランの自動回復 オペレータ・インタフェース ・ラン制御 ・稼働状況表示
ファイル管理機能	ファイルの世代管理 ファイルの期限管理 ファイルの自動割当と解放処理 磁気テープの出庫処理 ディスク・ファイルのバックアップ 障害ファイルの回復

いた．

2.3 IOF/WORKの開発目的と機能目標

IOF/WORKは，単一ホストからネットワーキングされた複数のコンピュータ・システムまでのバッチ処理運用を集中的かつ統合的に管理・制御することを目的として開発された運用管理システムであり，その機能目標は次の通りである．

1) 複数ホストで稼働するバッチ・スケジュールの一括作成/維持/管理
2) スケジュールの作成/検索/変更に対する高水準インタフェースの提供
3) スケジュール関数の提供
4) スケジュール関連作業の並行処理の実現
5) スケジュール情報に基づくジョブの自動起動
6) 24時間365日連続運転下での中断のない起動制御
7) IOFエリアの管理体系に従った運行状況の集中監視/制御
8) ジョブ走行状況の検索/制御における高水準インタフェースの提供
9) XTPA環境下における統合運行制御
10) ファイルのライフサイクル管理
11) ファイルの自動割当と媒体管理
12) ファイルの連携制御
13) ファイルのセーブ/ロード機能の提供

IOF/WORKでは，これらの機能を実現することにより以下のことを可能としている．

- 高水準なユーザ・インタフェースによるエンドユーザ・スケジューリングの実現
- スケジューリング/運行制御の集中化によるコンピュータ・ネットワーク全体としての信頼性の向上

●運用操作の集中化によるオペレーション・コストの削減

2.4 IOF/WORKの管理概念

2.4.1 業務処理の管理体系

IOF/WORKでは対象となるバッチ処理業務を，ワーク/ジョブ/プロシジャの3階層の体系にもとづき管理・制御している．そして，これらの業務処理を管理する領域としてアプリケーションと呼ぶ管理概念を使用する．

1) アプリケーション……アプリケーションとは業務内容，管理者や主管部門などにより分類された業務処理の集合であり，一般には使用者のアプリケーション・システムを意味する．アプリケーションはIOF/WORKを越えた拡張IOF全体の管理概念であり，IOF/WORKの観点からは以下のような特徴を備えている．

- IOF/WORKが管理するファイル，ワーク/ジョブなどの所属を示す単位として使用される．
- スケジュール検証を行う場合の独立した作業場所として使用される．スケジュールの検証作業はアプリケーション単位に実施されるものであり，異なるアプリケーション間では独立して操作が可能である．

2) ワーク……ワークとは，たとえば勘定系システムの月次処理というような，アプリケーション内で意味のある処理単位（プロシジャ）の集合に対する管理名称である．ワークは，一連の作業を実行する場合の基準となる日（基準日）を決定する条件と，処理対象となるプロシジャ群（構成ジョブのネットワーク）を指定したものとして定義され，スケジュールの作成/操作の単位として使用される(図1)．

ワークは複数の処理日ならびに複数のホストにまたがって定義することが可能であり，図1のワークは処理日を軸に表現すると図2のように，また構成ホスト

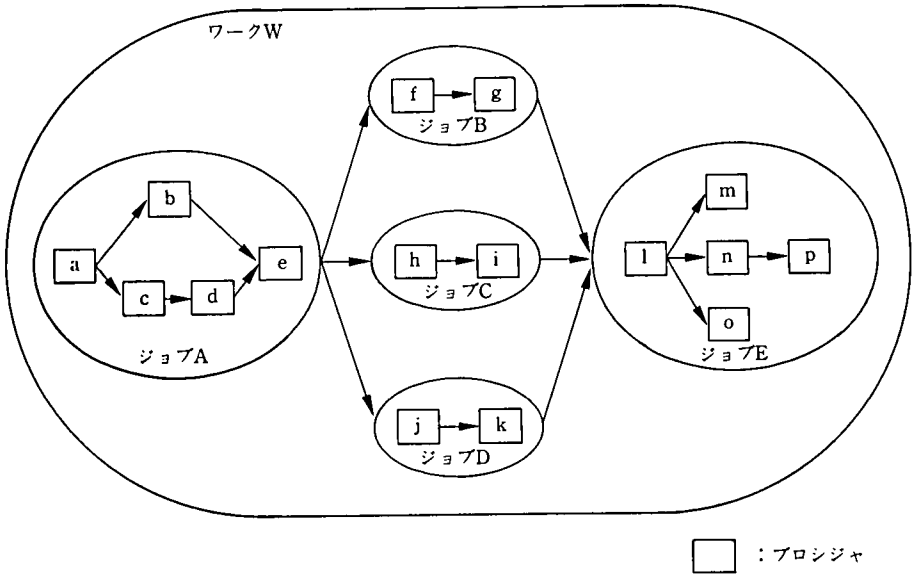


図1 ワークの論理構成

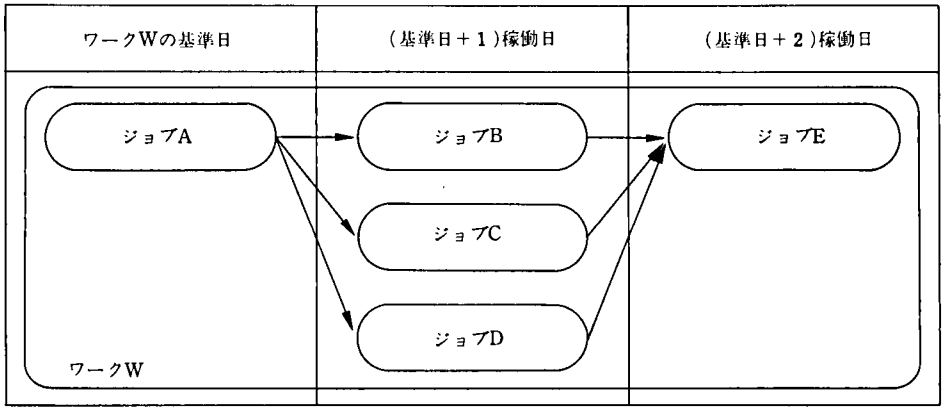


図2　処理日を軸にしたワークの表現例

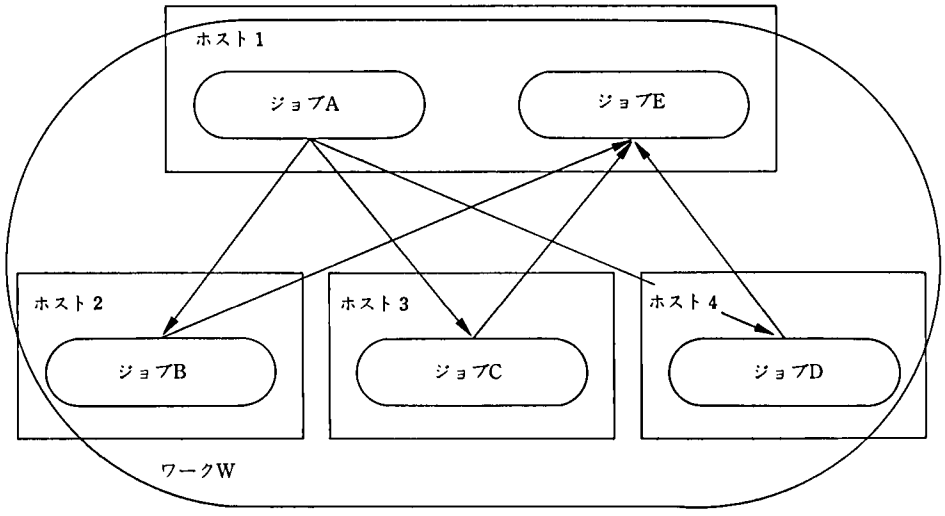


図3　構成ホストを軸にしたワークの表現例

を軸に表現すると図3のように書き直すことができる.

3)　ジョブ……ジョブとは運行管理における起動制御/進捗管理の単位であり,次のような特徴・制限を持つ.

- ジョブは構成プロシジャのネットワークと該ネットワークを実行するために必要となる条件(起動条件)により定義される.シリーズ2200においては,ジョブは一つのランまたは複数のラン・ネットワークで表現される.
- ジョブは特定のホスト/特定の処理日内で実行し得る形態でなければならない.

4)　プロシジャ……プロシジャとは,一つまたは複数のプログラムとその実行手続きからなる業務処理単位をいう.プロシジャはIOF/WORKにおける最小の管理単位であり,業務処理エラー時のリカバリポイントの制御に使用される.

2.4.2　複数ホストの管理体系

拡張IOFの各サブシステムは,分散システム下の各ホストを単一の制御点から管理

制御する手段を提供している．この時の管理/被管理の対応を定義する概念をIOF管理エリアと呼ぶ．IOF/WORKもIOF管理エリアの概念に基づき管理・制御が実施される．

1) IOFエリア……IOFの定めるプロトコルにより接続されたコンピュータ群から構成され，IOFを介した集中管理・集中制御が実施される範囲をIOFエリアと呼ぶ．IOFエリアには唯一のマスタホストが存在し，エリア内の全ホストの管理・制御を実施することができる．マスタホストによる管理・制御の対象となるホストをマスタホストと対比してスレーブホストと呼ぶ．

2) サブエリア……IOFエリア内を分割し，独立して管理・制御することが可能なエリアをサブエリアと呼ぶ．サブエリアには唯一のサブマスタホストが存在し，サブエリア内の全ホストに対して，マスタホストの機能を代行することができる．

IOF管理エリアを構成する主な管理概念を図4に示す．

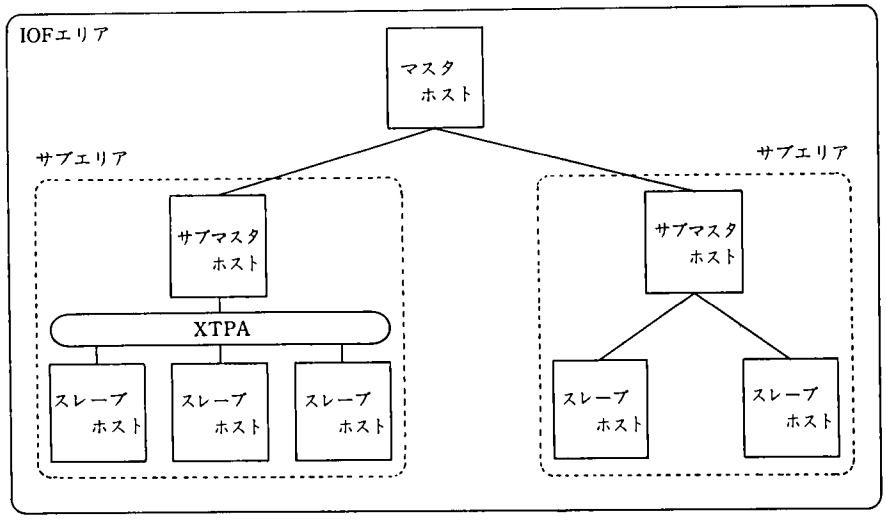


図4 IOF管理エリア

IOF/WORKの稼働する各ホストは，IOFのエリア管理の考え方に従い表2に示すような役割を持つ．

表2 各ホストごとの役割

	役　割
マスタホスト	● IOFエリア内の全ホストのスケジュールを集中作成する． ● マイクロオペレータを介して全ホストのジョブ/ランの制御や進捗監視を行う．
サブマスタホスト	● 統合運行制御のジョブ起動条件管理が稼働し，ジョブの起動制御と進捗管理を実施する． ● サブエリア内の共用資源（共用ディスク，テープ等）の管理を行う．
スレーブホスト	● 統合運行制御のジョブ実行管理が稼働し，スケジュールランの起動と進捗管理を実施する． ● ランの進捗状況を上位ホストに対して通知する． ● ホスト内のローカル資源（ローカルディスク，テープ等）の管理を行う．

2.5 IOF/WORKの機能概要

IOF/WORKの機能は，大別すると1)スケジュール管理，2)運行管理，3)ファイル管理，4)マイクロスケジューラ/マイクロオペレータの四つに分類することができる(図5)．

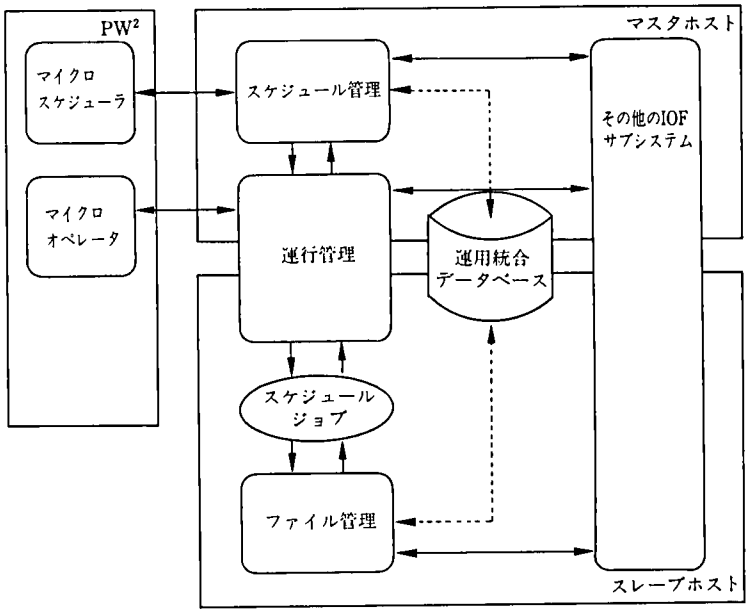


図5　IOF/WORKの機能構成

1)　スケジュール管理……運用統合データベース*に登録されたカレンダ情報，ネットワーク情報，ジョブ情報等の基本情報をもとに，日々の業務処理計画の生成/変更/検索を実施する機能である．スケジュール作成には，全ホストのスケジュールを一括して作成する集中スケジューリング方式を採用し，スケジュール関連作業の集約化・効率化を実現している．スケジュール管理で提供する機能の一覧を表3に示す．

表3　スケジュール管理機能一覧

機　能	内　容
スケジュール検証	マイクロスケジューラを介して，任意期間におけるジョブの実行予定の検索および実行予定に対する変更を予約する機能
スケジュール確定	処理日単位に実行ジョブの決定を行う機能．併せてジョブで使用するテープ媒体の決定も行う．
スケジュール変更	業務処理の変更に応じて確定済みスケジュールの追加/削除/処理日移動を行う機能
スケジュール管理諸表の作成	運用統合データベースの各種定義情報や確定スケジュール情報をもとに各種管理資料を作成する機能
再処理管理	運行実績情報をもとに処理済みジョブの再実行スケジュールを生成する機能

*運用統合データベース：拡張IOFを構成する各サブシステムの管理情報を統合管理するデータベース．データベース管理システムにはUDS/RDMS 1100を使用している．

表4　運行管理機能一覧

機　能	内　容
ジョブ起動条件管理	ジョブの起動条件を管理し，起動可能ジョブの選択・起動を行う機能
ジョブ実行管理	ランの起動条件を管理し，起動可能ランの選択・起動を行う機能
プロシジャ実行管理	各プロシジャの開始/終了を管理し，リラン時の開始ポイントの制御を行う機能
運行制御コマンド	運行制御に対するオペレータからの指示を処理する機能
進捗状況監視	ジョブの進捗を監視し，障害状況や遅延状況の検出を行う機能

表5　ファイル管理機能一覧

機　能	内　容
ライフサイクル管理	ファイルの生成から消滅までの全ライフサイクルにわたり，その有効期限，保存数，存在する媒体等の属性情報を一元的に管理する機能
ファイルの割当/解放制御	ジョブの実行時に，業務処理で必要とするファイルを自動的に選択し割り当てるとともに解放時にファイルの使用実績を取得・更新する機能
媒体管理	ファイル媒体の物理属性とその使用状況の管理ならびにファイルに対する媒体の割り付け/解放を制御する機能
セーブ/ロード管理	マスストレージ管理サブシステム（IOF/MASS）と連携し，業務処理で生成されたファイルのセーブ/ロードを行う機能

2)　運行管理……スケジュール管理により生成された実行予定情報をもとに，ジョブの起動制御ならびに進捗状況の監視を行う機能である．マスタホストより全構成ホストのジョブの進捗状況の監視・制御が可能であり，運行制御操作の集中化を実現している．運行管理で提供する機能の一覧を表4に示す．

3)　ファイル管理……ジョブで使用するファイル，およびそれらのファイルが存在する媒体の管理・制御を行う機能である．ホスト間で共有されるファイルや媒体の一元管理が可能であり，ホストをまたがった業務間でのファイル連携処理を容易に実施することができる．ファイル管理で提供する機能の一覧を表5に示す．

4)　マイクロスケジューラとマイクロオペレータ……IOF/WORKでは，運用コンソール・システム* 上に次の二種類のユーザ・インタフェースを実装し，運用業務全般にわたり簡易かつ安全な操作環境を提供している．

- ●マイクロスケジューラ（スケジュール検証コンソール）
  スケジュール情報の検索・補正・確定等のスケジュール関連作業を実施するためのユーザ・インタフェースを提供する．
- ●マイクロオペレータ（運行制御コンソール）
  ジョブの進捗状況の監視や再実行等，当日スケジュール業務に対する各種オペレーション操作を実施するためのユーザ・インタフェースを提供する．

3. IOF/WORKによるスケジューリング

3.1　スケジュールの作成手順

ここでは，IOF/WORKにおける業務スケジュールの作成手順と各手順ごとの機能概要を述べる．当システムによるスケジュール情報の作成は，以下の手順で実施され

*運用コンソール・システム：拡張IOFやXISで統一ユーザ・インタフェースを構築するためのワークステーションPW²上のソフトウェア．

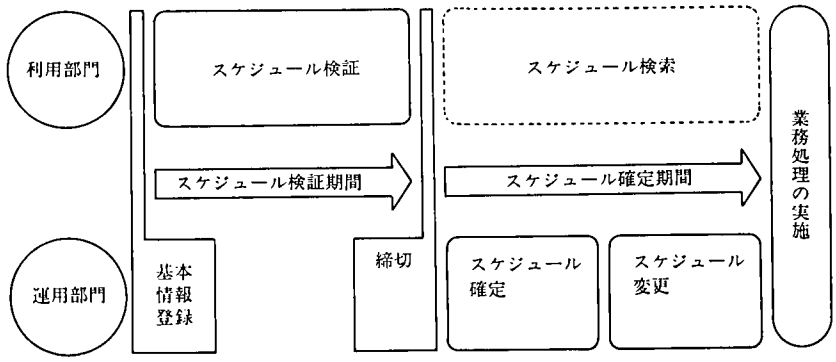


図6 スケジュール作成手順

る（図6）.

手順1：基本情報登録

スケジュールの自動生成に必要となる基本情報を運用統合データベースに登録する.

手順2：スケジュール検証

登録した基本情報をもとにスケジュール情報の検証を実施する.マイクロスケジューラを使用することにより利用部門を中心とした検証作業が可能である.

手順3：スケジュール確定

検証作業を打ち切り，検証が終了した期間のスケジュール情報を確定する.この作業は運用部門を主体に進められる.

手順4：スケジュール変更

確定後に実行予定の変更が生じた場合は,運用部門によるスケジュール情報の変更作業が実施される.

IOF/WORKでは，スケジュール検証からスケジュール変更までの一連の作業を広義にスケジュール作成処理と呼ぶ.IOF/WORKのスケジュール作成処理は,マスタホストにてIOFエリア内の全ホストを対象として一括して実施する.この集中方式のスケジュール作成処理により，次のような効果が期待できる.

- 複数ホストのスケジュールを一括して作成することによるスケジューリング作業の効率化
- ホストをまたがったジョブの流れが容易に把握できることによるスケジュール精度の向上

3.1.1 基本情報登録

IOF/WORKでは，運用統合データベースに登録された業務処理情報をもとにスケジュール作成処理を実施する.スケジュール作成処理を実施するに当たって登録が必要となる基本情報には次のようなものがある.

- カレンダ情報

個々の暦日日付に依存して設定される業務管理情報.ジョブの実行日付を決定

するために必要となる変数（スケジュール変数）等を保持する．

- ワーク定義情報
  ジョブの実行日付を決定するための基準日条件(スケジュール関数により記述)とワークを構成するジョブの情報
- ジョブ定義情報
  ジョブの実行条件や構成ラン，使用プロシジャ等の情報
- プロシジャ定義情報
  プロシジャの実行に必要となる使用ファイルの情報
- ネットワーク情報
  ジョブおよびプロシジャ間の実行順序を既定した情報
- ファイル定義情報
  IOF/WORK により管理されるファイル，およびそれらのファイルで使用する媒体（テープ/ディスク）情報

3.1.2 スケジュール検証

任意期間におけるジョブの実行予定の検索や，実行予定に対する変更を予約する作業をスケジュール検証と呼ぶ．IOF/WORK では，バッチ業務のスケジュール検証作業を利用部門が直接実施できるよう，運用コンソール・システムを使用した視覚的かつ高水準なユーザ・インタフェース（マイクロスケジューラ）を提供している．スケジュール検索ウィンドウを中心とするマルチ・ウィンドウ環境とマウスを主体にしたポインティング操作により，スケジュールの検索・変更作業を容易に行うことが可能となっている．

- スケジュールの操作域と並行処理
  従来の運用管理システムが実行ホスト単位にスケジュールの検証作業を行っていたのに対して，IOF/WORK ではバッチ業務のスケジュール検証作業をアプリケーション単位に実施することを基本としている．各利用部門ごとに，他のアプリケーションとは独立して，自アプリケーションに所属するジョブのスケジュール検証を実施することができ，検証作業の並行処理を実現している．また，ワークを中心としたスケジュール操作により，ジョブの実行ホストを意識することなく検証作業を行うことが可能となっている．
- スケジュールの検索と補正
  従来の運用管理システムでは，確定済みの期間を越えた長期スケジュールを検証する場合，検証する期間のスケジュールを実際に確定させた後，スケジュール・リスト等を出力して確認することが必要であった．しかしこの場合，検証終了後に確定させたスケジュールを取り消す作業が必要となり，利用部門が単独でスケジュールの検証を行うことは困難であった．IOF/WORK では，マイクロスケジューラによりワークステーション PW²上で各ワークの実行日条件を解析することにより，実際に確定処理を実施することなく指定期間のスケジュール状況を検証することが可能となっている．また，検証した結果，スケジュールの修正（追加/削除/処理日の移動）が必要な場合には，確定処理に向けて変更を事前に登録することが可能である．この操作をスケジュール補正と呼ぶ．

表6　マイクロスケジューラより可能な検証作業

検証作業	作業内容
スケジュール検索	既定の実行日条件に従い指定期間のワーク・スケジュールをグラフィカルに表示するウィンドウ（ワーク・スケジュール表示ウィンドウ）を通して，該当アプリケーションに所属するワークのスケジュール予定日（ワーク基準日）を検索することが可能である．また，特定のワークの詳細なジョブ構成や個々のジョブの起動条件をサブ・ウィンドウで検索することができる．
スケジュール基準日の変更	ワーク・スケジュール表示ウィンドウでワーク基準日を直接ドラッグすることにより，ワークの実行予定日を変更することができる．また，ジョブ構成表示ウィンドウからは構成ジョブ個々のスケジュール日を変更することが可能である．
スケジュールの追加/削除	既定の実行日条件ではスケジュールの対象とならないワークを強制的に追加することが可能である．また，ワーク・スケジュール表示ウィンドウで特定基準日のワークを削除することも可能である．
ジョブ起動条件の変更	ジョブ起動条件表示ウィンドウから，該当ジョブに対する起動条件や実行条件の詳細を変更することが可能である．

マイクロスケジューラにより実施することが可能な検索・補正作業を表6に示す．

3.1.3　スケジュール確定

スケジュール検証作業を打ち切り，処理日単位に実行ジョブを特定しジョブ間の実行順序を決定する作業をスケジュール確定と呼ぶ．スケジュール確定は暦日単位に連続して，かつ全ホスト/全アプリケーションを一括して実施する．確定処理は運用部門が明示的に実施したり，あらかじめ設定されている確定期間を確保するように自動的に実施させることが可能である．スケジュール確定は，図7に示す複数の作業から構成される．

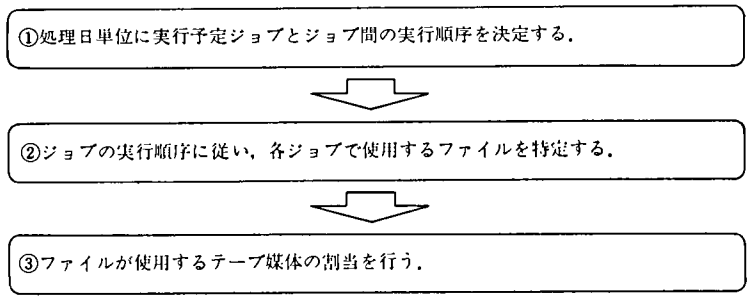


図7　確定作業の手順

確定したスケジュールに対しては，マイクロスケジューラよりスケジュール検証時と同様の検索・補正機能が使用可能である．さらに，運用部門向けには日別/ホスト別の確定スケジュール検索画面が追加される．利用部門においても，検証期間と同一の操作でマイクロスケジューラより確定済みスケジュールを検索することが可能である．ただし，変更は不可であり，必要時には運用部門に依頼し変更を実施してもらわなければならない．

3.1.4　スケジュール変更

確定したバッチ業務の走行予定を変更することをスケジュール変更処理と呼ぶ．確

定スケジュールに対しても検証期間と同様の操作手順により変更作業を実施することができる．スケジュール変更は，変更対象とするワーク/ジョブの実行日当日まで実施可能である．

3.2 ジョブ実行日の決定方法

IOF/WORK では，自動スケジュール生成の精度向上を図るために，ワークや構成ジョブの実行日条件をスケジュール関数と運用カレンダの機能を使用して定義する．

3.2.1 スケジュール関数

スケジュール関数とは，運用カレンダに設定されたスケジュール変数を利用して一定の日付集合を算出する手続きで，拡張 IOF の各サブシステムが日付集合を求める場合に共通して使用する機能である．スケジュール関数には，表7に示されるような日付演算機能が準備されており，従来の運用管理システムでは自動生成の対象外とされていた複雑な実行パターンもその対象とすることが可能となっている．スケジュール関数を使用した実行日条件の定義例を図8に示す．

表7 スケジュール関数一覧

関 数	関 数 の 機 能
@HEAD	日付集合の暦日上の先頭日を示す．
@TAIL	日付集合の暦日上の末日を示す．
@FOR	日付集合を指定された番号に従って昇順に順序付けする．
@BACK	日付集合を指定された番号に従って降順に順序付けする．
@AFT	ある日付集合に対して一定日数後ろにずれた日付を示す．
@BEF	ある日付集合に対して一定日数前にずれた日付を示す．
@PRE	条件によって日付を前にシフトする．
@POS	条件によって日付を後ろにシフトする．
@MOD	一定の間隔をあけた日付の集合を示す．
@SEQ	一定期間連続した日付を示す．
@ALT	ある日付集合の一部を別の日付集合で置き換える．
@ANY	何も条件付けしない（表記上の統一のため用いる）．

```
● 毎月第1営業日の2営業日前
    @BEF [@HEAD [$APL], $APL, 2]
● 営業日ただし非稼働日の場合は直前の稼働日
    @PRE [$APL, $RUN]
● 毎月5日から3稼働日連続して実施
    @SEQ [5,$RUN, 3]
● 毎週土曜日と日曜日。ただし第1稼働日が土曜日の場合は月曜日に実行
    @ALT [$SAT | $SAN, @HEAD [$RUN] &$SAT,
        @POS [@HEAD [$RUN] &$SAT, $MON]]
● 毎月10日。ただし非営業日の場合は直後の営業日
    @POS [10, $APL]
```

図8 スケジュール関数の定義例

3.2.2 運用カレンダとスケジュール変数

運用カレンダとは，営業日や立ち上げ時刻のように，個々の暦日日付単位に設定・変更されるような情報を統合して管理する運用統合データベース上の機構であり，拡張 IOF を構成する各サブシステムから共通して使用される．運用カレンダは，論理的には図9のように暦日をキーとし各管理項目を要素とするテーブルとして表現され

管理項目 / 暦日日付	曜日	稼働日	営業日	立上時刻	終了時刻	・・・・
yy/mm/dd	Sat.	on	off	—	—	
yy/mm/dd	Sun.	off	off	hh：mm	hh：mm	
yy/mm/dd	Mon.	on	on	hh：mm	hh：mm	
yy/mm/dd	Tue.	on	on	hh：mm	hh：mm	
・ ・ ・						

図9　運用カレンダの論理表現

る．個々の暦日日付に対して与えられる管理項目（曜日～終了時刻）をカレンダ変数と呼び，その中でもON/OFFの論理値を持ちスケジュール関数で使用可能な変数(稼働日，営業日）を特にスケジュール変数と呼ぶ．

運用カレンダはスケジュール関数を使用する上での必須情報であり，IOF/WORKのスケジュール作成処理に先立ち必ず必要期間のカレンダ情報が作成されていなければならない．

3.3　ジョブの統合運行制御

IOF/WORKは，スケジュール管理により生成された確定スケジュールをもとにジョブの自動起動を実施する．ジョブの起動処理形態には次の2種類が存在する（図10）．

- ●単一運行制御：それぞれのホストごとにジョブ起動条件管理およびジョブ実行管理の両者が稼働する形態であり，各ホストごとに独立して制御が実施される．
- ●統合運行制御：XTPAで結合されたホスト群（サブエリア）でのみ使用可能な

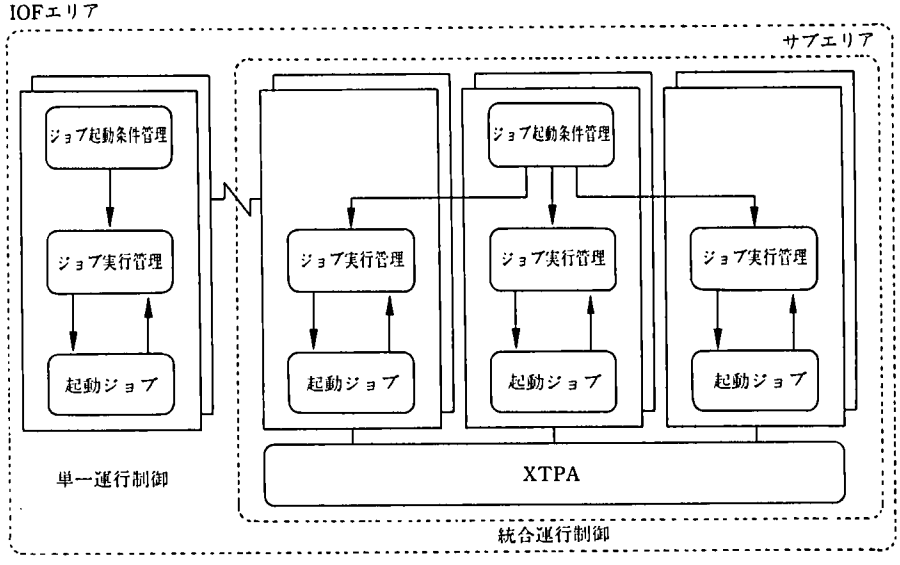


図10　ジョブの起動制御形態

形態であり，サブマスタホストにのみジョブ起動条件管理が稼働し，構成全ホストのジョブ実行管理が一括制御される．統合運行制御を使用することにより，ホストをまたがるジョブ・ネットワークの定義が可能となり，バッチ運行に対するシングルシステム・ビューを実現することができる．

3.3.1 ホスト・ディペンドジョブとエリア・ディペンドジョブ

XTPA 構成下のホスト群で統合運行制御を実施する場合，ジョブは実行可能なホスト環境により，ホスト・ディペンドジョブ (HDJ) とエリア・ディペンドジョブ (ADJ) に分類・管理される．HDJ とはジョブの実行環境 (JCL, プログラム，使用ファイル等) が特定のホストにのみ限定されるものをいい，ADJ とはジョブの実行環境が共有媒体上に存在し，媒体を共有するホスト群のいずれのホストでも実行が可能なジョブをいう．ADJ を使用することにより，任意のジョブの実行ホストを強制的に変更したり，ホスト障害時には障害ホストで実行予定であったジョブを他のホストに振り分けたりすることが可能となる．

3.3.2 仮想マスタ運用

統合運行制御を使用する場合，サブマスタホストでのみジョブの起動条件管理が稼働するため，サブマスタホスト障害時のリカバリ方法が重要となる．サブマスタホストのリカバリを容易にするために，IOF/WORK では起動条件管理の実行ホストをシステムの立ち上げ時に決定し，該ホストをサブマスタホストとして稼働させることを可能にしている．この運用を仮想マスタ運用と呼ぶ．起動条件管理が存在するホスト

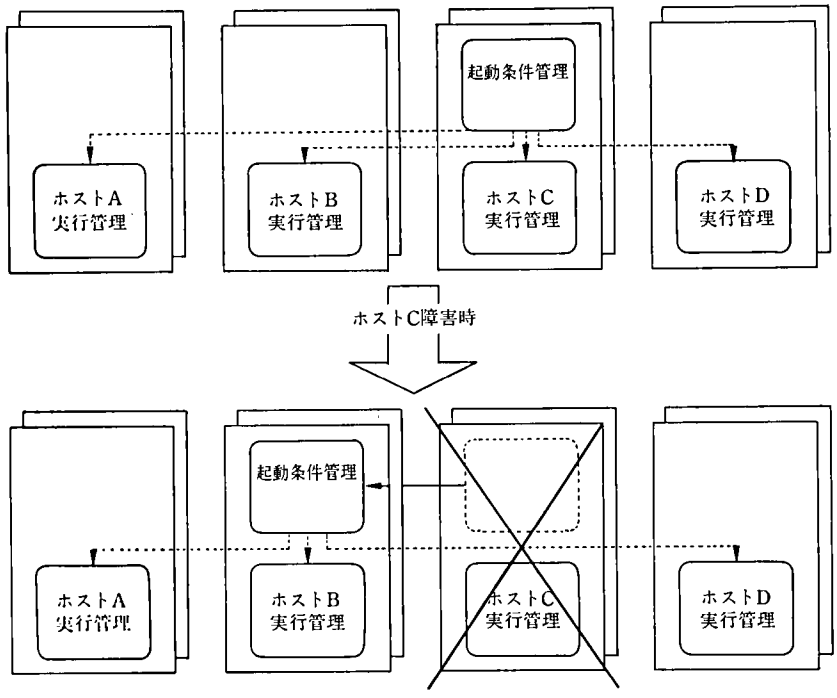


図11 仮想マスタ運用によるリカバリ

(サブマスタホスト)の障害時には別ホストで該処理を稼働させることにより，サブマスタホスト機能の即時リカバリが可能となる．図11に仮想マスタ運用によるリカバリ方法を示す．

3.4 ジョブ進捗状況の監視

IOF/WORKでは，IOFエリアの管理体系に従い，単一の制御点（マスタホスト）から全構成ホストに対する運行状況の監視・制御を行うことが可能である．監視・制御のためのインタフェースは，PW²上にマイクロオペレータ（運行制御コンソール）として実装されている．マイクロオペレータとは，運用コンソール・システムを使用したIOF/WORKの運行制御画面群の総称であり，マウス操作を中心とした簡易なインタフェースで当日のオペレーション業務を実施することを可能にしている．マイクロオペレータを介して以下に述べる監視・制御が可能である．

●進捗状況の監視

ホスト上の進捗状況監視機能によりエラージョブの検出や開始/終了遅延ジョブの検出が可能である．検出された進捗状況は，PW²に通知され以下に示す進捗状況画面群に表示される．

－全体進捗状況表示

－ホスト別進捗状況表示（バーチャート形式/一覧表形式）

－運行状況表示

●詳細状況の検索

個々のジョブ/ランの進捗状況や各種起動条件の詳細を検索することが可能である．

－ジョブ詳細情報検索

－ラン詳細情報検索

●運行制御指示

詳細状況の検索と連動して，運行管理機能に対して以下に示すような各種制御指示を行うことが可能である．

－ジョブの再起動

－ジョブのサスペンド

－ジョブの強制起動

－ジョブ起動条件の設定/解除

3.5 連続運転対応

IOF/WORKでは，コンピュータのマシン日付とは別に独自の処理日(スケジュール日）を管理している．このスケジュール日を1稼働日進めるとともに運行制御用情報を翌日用に切り替えることを処理日の切り替えと呼ぶ．従来の運用管理システムでは，処理日の切り替え時に運行制御処理を終了させる必要があり，ホスト・コンピュータの連続運転に対応することが困難であった．IOF/WORKでは，停止運転モードと連続運転モードの二種類の切り替え方法を提供しており，連続運転下においても中断のない起動制御処理が可能となっている．

1) 停止運転モード……従来の運用管理システムと同様，運行制御処理の終了後に処理日の切り替えを実施し，再び運行制御処理を立ち上げる形態をIOF/WORK

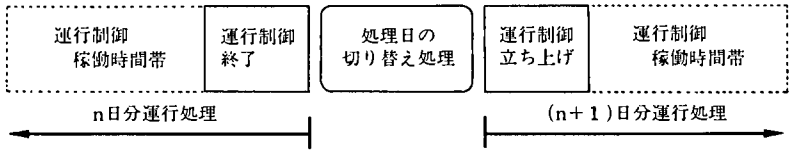


図12　停止運転モードによる切り替え

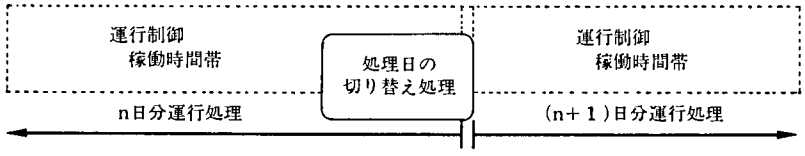


図13　連続運転モードによる切り替え

では停止運転モードと呼ぶ（図12）.

2)　連続運転モード……運行制御処理の立ち上げ/終了を実施せずに,翌日用スケジュールに切り替える運転形態を連続運転モードと呼ぶ．切り替え処理は，システムに指定された一日の稼働開始時刻になると自動的に実施される．処理日の切り替え中は,新たなジョブ/ランの起動および実行中のランは一時的にサスペンドされ，切り替え完了後に自動的に再開される（図13）.

4.　IOF/WORKのユーザ・インタフェース

IOF/WORKでは，スケジュールの作成から当日のオペレーション業務までのバッチ運用全般にわたり高水準なユーザ・インタフェース（マイクロスケジューラ/マイクロオペレータ）を提供している．いずれもマウスによるメニュー選択中心の操作で，コンピュータ処理に習熟していないユーザでも安全かつ容易に目的とする処理を実施することが可能となっている．ここでは，マイクロスケジューラおよびマイクロオペレータの種々の操作の中から代表的な操作について幾つか例をあげて紹介する.

4.1　ワーク・スケジュールの検証

特定アプリケーションに対するスケジュール状況の検索や補正作業を実施する場合，マイクロスケジューラの「ワーク・スケジュール」ウィンドウ（図14）を使用する．該ウィンドウは，検証作業を実施する場合のマイクロスケジューラのメイン・ウィンドウであり，検証対象アプリケーションに所属するワークの基準日情報が表示される.

該ウィンドウから実施可能な検証操作のうち，主なものを以下に示す.

- ●基準日の追加：追加対象となるワーク/Sパタン（＊印で示される項）の該当日付の項をダブルクリック
- ●基準日の移動　移動対象となるワーク/Sパタンの該当日付の＊印を変更後の日付の位置にドラッグ
- ●基準日の削除　削除対象となるワーク/Sパタンの該当日付の＊印をウィンドウの左上部の「ゴミ箱」位置へドラッグ

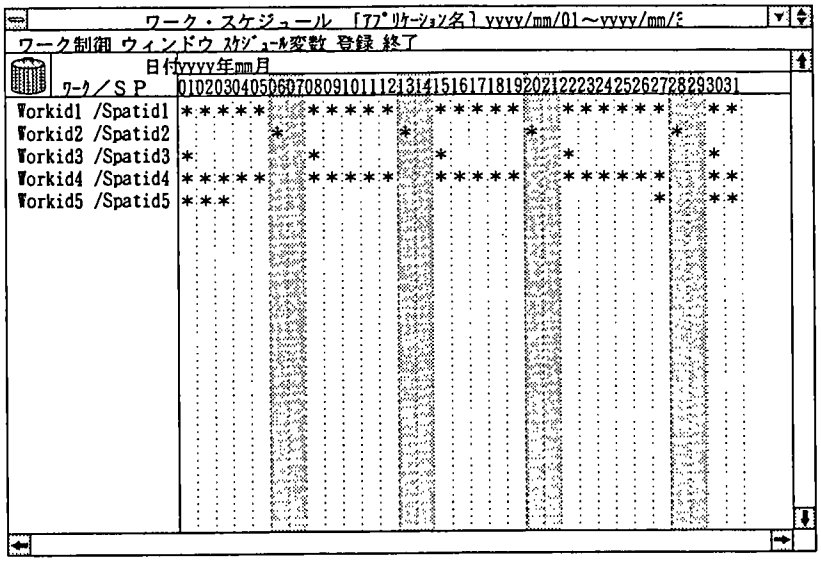


図14 ワーク・スケジュール・ウィンドウ

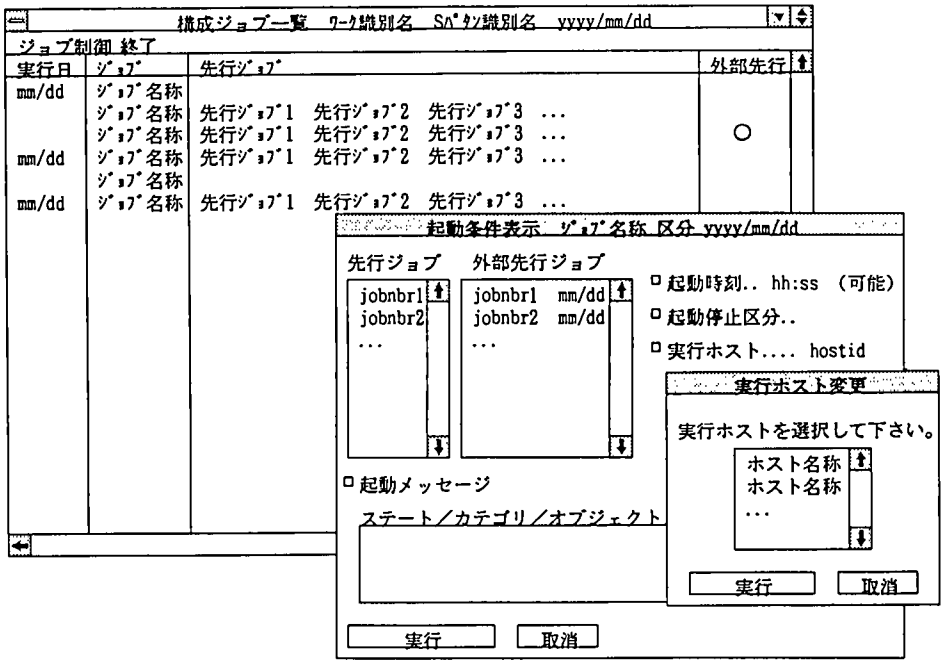


図15 構成ジョブ一覧ウィンドウと起動条件変更手順

また，特定基準日のワーク/Sパタンをダブルクリックすることにより，そのワークの「構成ジョブ一覧」(図15)を呼び出すことができる．「構成ジョブ一覧」ウィンドウには選択したワークを構成するジョブがジョブの実行日順に，先行ジョブ情報とともに表示される．該ウィンドウからは構成ジョブに対して以下の検証操作が可能である．

- ジョブ起動条件の検索/変更
- ジョブ実行日の変更
- ジョブの追加
- ジョブの削除
- ジョブネットワークの変更

該ウィンドウにおいても，マウスによるメニュー選択を中心とした操作により検証作業を行うことが可能となっている．たとえば，ジョブの実行ホストを変更する場合は，以下のような手順となる．

① 変更対象となるジョブをダブルクリックし，「起動条件表示」ダイアログボックスを呼び出す．

② ダイアログボックスの＜実行ホスト＞チェックボックスをクリックし，「実行ホスト変更」ダイアログボックスを呼び出す．

③ 実行ホストを選択し実行ボタンをクリックする．

④ 「起動条件表示」ダイアログボックスの実行ボタンをクリックする．

4.2 特定ホストのジョブ進捗状況の検索

マイクロオペレータの基本ウィンドウである「全体進捗状況」ウィンドウにて検索対象とするホストのホスト名称をクリックすることにより，対象ホストでスケジュールされているジョブの進捗状況を表示することができる．ホストごとの進捗状況表示ウィンドウには，バーチャート形式でジョブの進捗状況を表示する「個別進捗状況」ウィンドウと指定ホストでスケジュールされているジョブを一覧形式で表示する「スケジュール一覧表」ウィンドウの二種類がある(図16)．どちらもジョブの実行状況に応じて表示色が区別され，走行状態が容易に識別できるようになっている．

なお，「スケジュール一覧表」ウィンドウにおいては＜ジョブ名＞フィールドをクリックすることにより，該当ジョブの詳細情報(「ジョブ詳細」ウィンドウ)の表示や実行状況に応じた各種制御(再起動やホールドの設定/解除等)を実施することができる(図17)．

4.3 エラージョブのリカバリ

実行中のジョブがエラー終了した場合，操作員は通常次の手順でジョブのリカバリを行う．

1) エラーランの実行ジャーナルを採取しエラー原因を特定する．

2) エラー原因に応じた対処を行う．

3) エラーランの再実行を行う．

このリカバリ手順をマイクロオペレータを使用して行う場合の操作例を次に示す(図18,図19)．

① 「スケジュール一覧表」ウィンドウの＜ジョブ名＞フィールドをクリックし，「ジョブ詳細」ウィンドウを呼び出す．

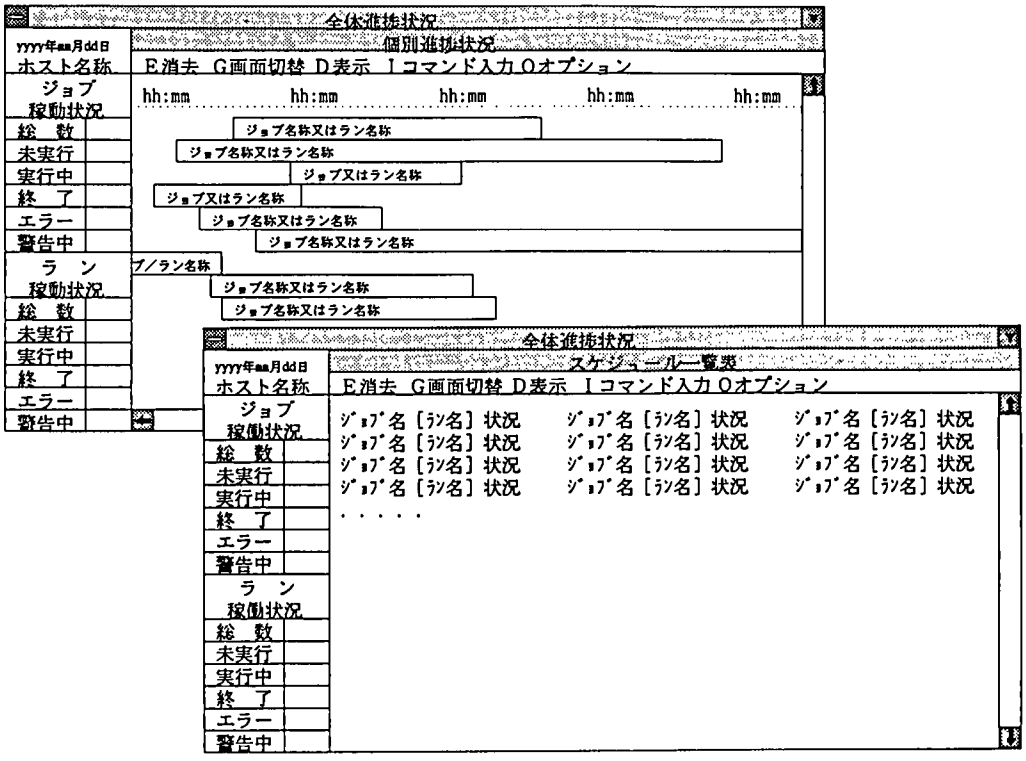


図16 進捗状況表示ウィンドウ群

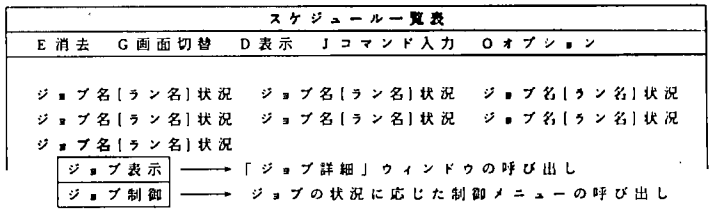


図17 「スケジュール一覧」ウィンドウでの操作例

② 「構成ラン」サブウィンドウで状況がエラーとなっている<ラン名>フィールドをクリックしポップアップ・メニューより［ジャーナル検索］を選択する.

③ 続いて表示されるジャーナル検索メニューより［最終プロシジャ］を選択し，最終プロシジャの実行ジャーナルをそのプロシジャの先頭より表示する.

④ 呼び出された「実行ジャーナル」ウィンドウでエラーとなった処理を特定し対処を行う.

⑤ 再び「構成ラン」サブウィンドウのエラー<ラン名>フィールドをクリックし，ポップアップ・メニューより［ラン制御］を選択する.

⑥ 続いて表示される制御メニューより［ランの再起動］を選択する.

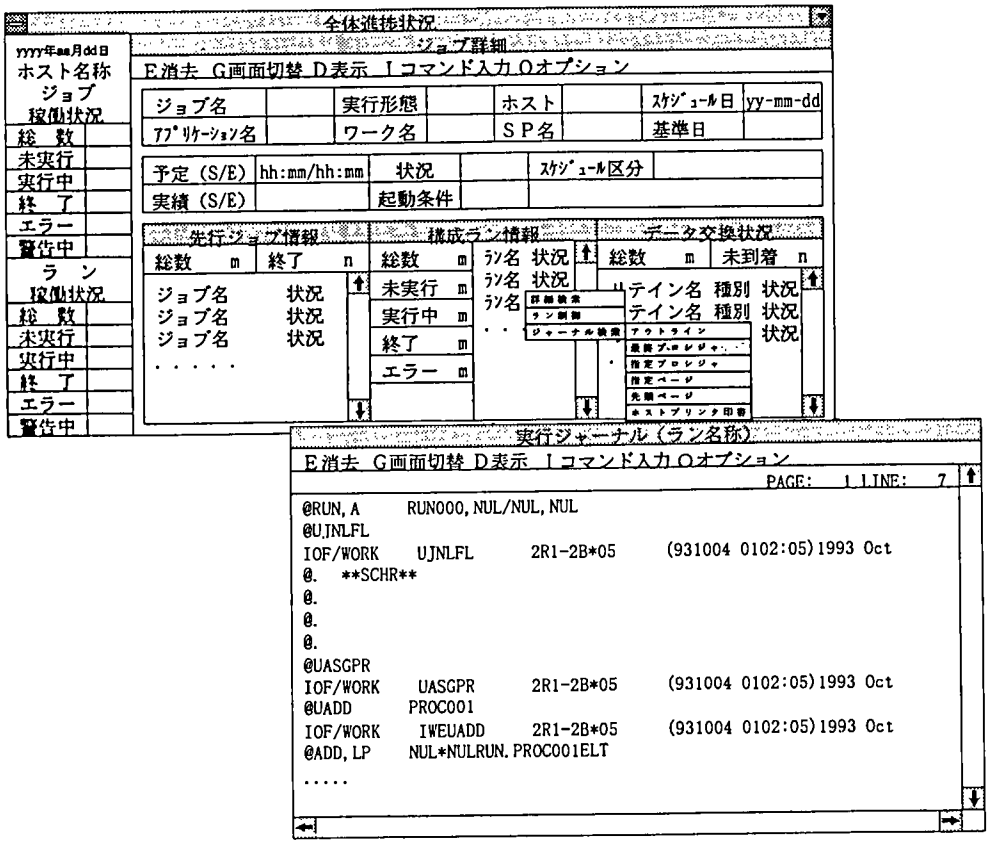


図18 ジョブ詳細検索とジャーナル検索

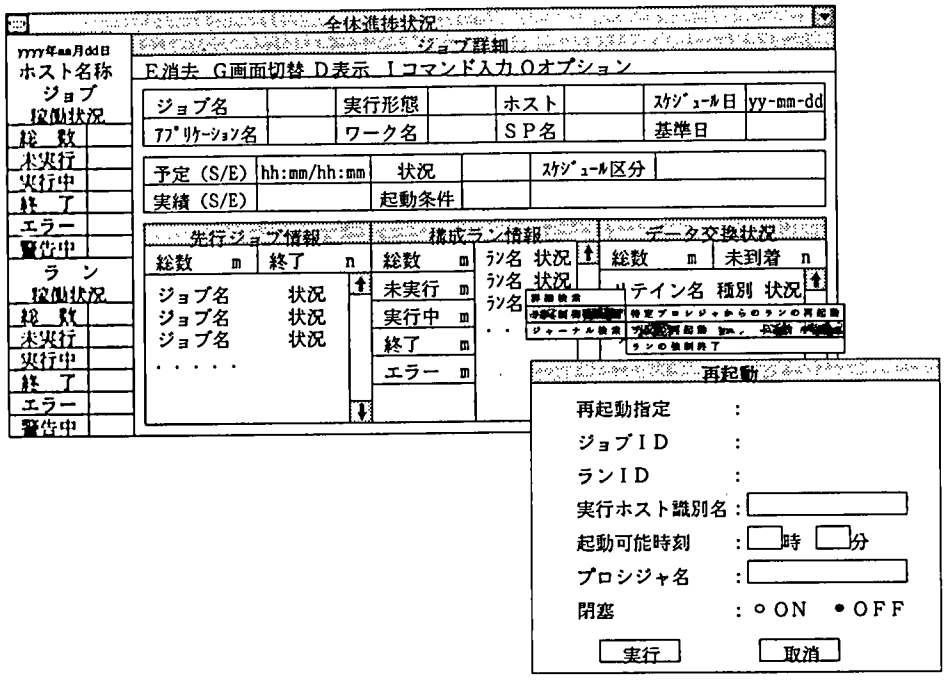


図19 ランの再起動操作例

⑦ この時，[再起動]ダイアログボックスが表示され，再起動条件の詳細設定が可能である．

⑧ 実行ボタンをクリックすることにより対象ランの再起動が実施される．

5. おわりに

以上IOF/WORKによるバッチ運用の自動化についてその対応を述べた．IOF/WORKは平成5年5月に本番を迎え，以降大きなトラブルもなく安定稼働を続けている．開発目的であったバッチ運用の集中化，高水準ユーザ・インタフェースの提供については，集中スケジューリング，統合運行制御，およびマイクロスケジューラ/マイクロオペレータの機能として実現されている．また，連続運転モードの提供により，ホストマシンの24時間365日連続運転にも対応可能であり，今後のバッチ処理運用を担って行く上で必要となる種々の機能の提供が可能となっている．しかし，改善・機能追加を必要とする課題も多く残されており，まだまだ発展途上のシステムであるといえる．今後の主な課題としては，次のものが上げられる．

1) スケジュール作成における処理日や処理予定時刻の最適化の実現
2) 各ホストの負荷状況，リソースの使用状況に応じた起動制御の実現
3) システム負荷の軽減と確定関連処理における処理時間の短縮
4) マニュアル，メッセージ等の改善による使用容易性の一層の向上

IOF/WORKをより充実した運用管理システムにするためには，これらの課題を解決していくことはもちろん，常に導入ユーザにおける改善意見に耳を傾け，それらを反映したシステム改善を積極的に行っていくことが必要であると考える．

執筆者紹介　安坂敏秀（Toshihide Yasusaka）
昭和35年生．58年広島大学教育学部心理学科卒業．61年日本ユニシス（株）入社．運用管理システムの開発・保守に従事．現在，システム企画開発二部運用開発課に所属．

友枝　研（Ken Tomoeda）
昭和41年生．平成元年芝浦工業大学工学部工業化学科卒業．同年日本ユニシス（株）入社．運用管理システムの開発・保守に従事．現在，システム企画開発二部運用技術課に所属．

IOF/MONIによる集中監視制御

Centralized Monitoring and Control by IOF/MONI

荻 原 和 彦

要 約 コンピュータ・システムの利用形態はますます高度化・複雑化してきており，システム運用部門では，システム運用の安全性確保と経済性追求，高スキル要員の維持と運転要員の負荷軽減，などが重要課題となっている．拡張IOF（Integrated Operating Facility）はこのようなシステム運用部門の課題に対する支援ソフトウェアとして開発されたシステムであり，IOF/MONIはその中で集中監視制御に関する機能を担うシステムである．

本稿では，IOF/MONIの開発目的，機能構成と機能概要，運用方法とその適用事例を紹介する．

IOF/MONIは，ネットワーク化されたコンピュータ・システムを単一の制御点から監視・制御することを目的として開発されたシステムで，次の三つの機能から構成されている．

1) ホスト障害監視
2) ホスト稼働監視
3) 集中コンソール

また，マンマシン・インタフェースには，運転要員の負荷軽減を踏まえて，ワークステーションPW²のグラフィカル・ユーザ・インタフェース（GUI）を活用した集中監視制御機構（IOFワークステーション）を備えている．

Abstract As the modes of computer utilization are getting more diversified and complicated, so the problems that computer operating departments have to contend with are getting more varied, including assured safety in systems operation, pursuit of higher cost-effectiveness, the maintaining of highly skilled operators and their lessened workloads. IOF (Integrated Operating Facility) is a software tool developed to help solve those problems, and, particularly, IOF/MONI is a system which functions to provide centralized monitoring and control.

This paper discusses what IOF/MONI has been developed for, how its functionalities are configured, how they work and how to use them as well as how the system is actually being used. Developed to monitor and control networked computer systems from a single point, the IOF/MONI system provides three major functions: (1) host fault monitoring (2) host status monitoring, and (3) centralized console operation. In addition, its man-machine interface is equipped, for reduced workloads of systems operators, with a centralized monitoring and control feature (on the IOF workstation) for which the PW²'s graphical user interface (GUI) is used.

1. はじめに

近年，コンピュータ・システムの大規模化・分散化の進展に伴い，システムの利用形態はますます高度化・複雑化してきている．異なる機種のコンピュータ・システムがネットワークで結合され，バッチ処理やトランザクション処理など様々な処理形態が混在し，相互に関連し合いながら企業の基幹業務を実行している．このような状況

のもと，システム運用面での障害は企業の日常業務遂行に重大な影響を及ぼすばかりでなく，社会的にも深刻な事態を引き起こす可能性がある．ネットワーク化されたコンピュータ・システム全体を統制し，各システムが持つリソースを最大限に活用し，安全かつ効率的に業務を遂行していくことがシステム運用部門に求められた重要課題となっている．さらにシステム運用形態の複雑化は，運用要員の作業負荷増や高スキル化を要請しており，この面からもシステム運用の負荷軽減に関する方策が強く求められている．

統合運用システム（拡張IOF；Integrated Operating Facility）は，このような背景をもとに，これまで提供してきたコンピュータ運用総合自動化ファシリティ(UOF; Unattended Operating Facility）を始めとする各種運用支援システムの成果を活かし，かつ単一のコンピュータ・システムに留まらずネットワーク化されたコンピュータ・システム全体にわたる統合運用環境の整備と構築を目指して開発された統合運用支援ソフトウェアである．

拡張IOFのサブシステムの一つであるIOF/MONI[1]は，ネットワーク化されたコンピュータ・システムを単一の制御点から監視・制御する機構を提供する集中監視制御システムである．本稿では，IOF/MONIの開発目的，機能構成と機能概要および運用方法とその適用事例を紹介する．

2. IOF/MONIの開発目的

コンピュータ・システムの運用は，個々のシステム・コンソールにオペレータを配置して行われる．オペレータはシステム・コンソールに表示されるコンソール・メッセージを監視し，次のような障害または警告を認識する．

- オンラインプログラムのエラー
- バッチランのエラー
- 使用者プログラムから出力される応答型メッセージ
- 各種I/O機器のエラー・メッセージ，など

また，オペレータはシステム・コンソールを操作することにより，次に示すような項目の監視および表示されたメッセージに対する作業を実施している．

- トランザクション・キューの数
- テープ・ホールド/マスストレージ・ホールドの有無
- テープ装置に対するマウント要求
- プリンタ装置に対する用紙替え要求，など

このようにオペレータは発生が予知できない事象に対して，常にシステム・コンソールを監視すると共に，運用管理者への連絡，障害・警告に対するアクションなど，その作業負荷は大きい．一方，単一のコンピュータ・システムに留まらず，ネットワーク化された分散コンピュータ・システム環境，無停止連続処理が可能となる複数のホスト・システムを結合させたXTPA（eXtended Transaction Processing Architecture：拡張トランザクション処理アーキテクチャ）環境など，システムの利用形態も高度化してきている．

システム運用部門はこのような環境の中で以下に示すような課題を抱えている．

・システム・コンソールを中心とした運用に対する安全性の確保
・高スキルな運転要員の維持
・運転要員の削減

IOF/MONI は，システム運用部門の課題に対する支援機能を提供することを目的に開発された．イベント表示画面を中心とした集中監視制御の各種画面を利用することにより，運用管理者へのタイムリな通知，障害・警告などのオペレータが行動を起こすべき事象の通知が可能となる．また，IOF/MONI が提供する標準監視機能に対して，使用者の監視システム (IOF/MONI 機能外の監視システムを含む) と連携させたり，標準画面をカスタマイズすることが可能であり，使用者の環境に合わせた集中監視制御運用が構築できる．

3. IOF/MONI の概要

IOF/MONI は，複数ホスト，分散ホスト環境のコンピュータ・システムを単一の制御点から監視・制御する機構を提供し，個々のコンピュータ・システムを少人数で集中監視制御する運用を支援するソフトウェアである．ここでは，IOF/MONI のホスト構成の概念，仕組み，監視情報の概念，IOF ワークステーションの位置付け，機能概要を解説する．

3.1 監視制御対象ホストの構成

IOF/MONI で対象となるコンピュータ群は，IOF の共通概念であるエリアとして管理される．

1) IOF エリア……IOF エリアとは，拡張 IOF で管理されるコンピュータ・ネットワークのことで，マスタホストにより統合管理される．IOF/MONI では，このマスタホストが単一の制御点となる．分散ホスト環境で分散ホストの無人運転を行う場合には，マスタホストで集中監視制御を行う．
2) サブエリア……IOF エリアを分割し独立した制御を可能とする単位であり，サブマスタホストにより管理される．XTPA 環境下のホストはこのサブエリアを形成する．IOF/MONI では，分散ホスト環境で各分散ホストに運転要員が配置されている場合，マスタホストに代行してサブマスタホストでサブエリア内の集中監視制御を行うことができる．

複数のホスト構成で運用する場合には，必ず一つのマスタホストを設定する必要がある．各スレーブホストの監視情報がマスタホストに送られることにより，IOF エリア内の全ホストをマスタホストで集中監視することができる．システム構成の規模によっては，専用のマスタホストあるいはサブマスタホストを設置できないことがある．この場合には，仮想ホストを設定することにより IOF エリアまたはサブエリアを構成する．

マスタホストとサブマスタホストには，複数台の IOF ワークステーションを接続することができる．スレーブホストでは IOF ワークステーションを接続しない運用も可能である．IOF エリアの構成例を図 1 に示す．

3.2 IOF/MONI の仕組み

IOF/MONI は，監視機能，通報機能，検索・制御機能から構成されており，障害の

発生からその対処までの処理は各機能の連携によって行われる(図2)．ここではIOF/MONIの各機能の概要について解説する．

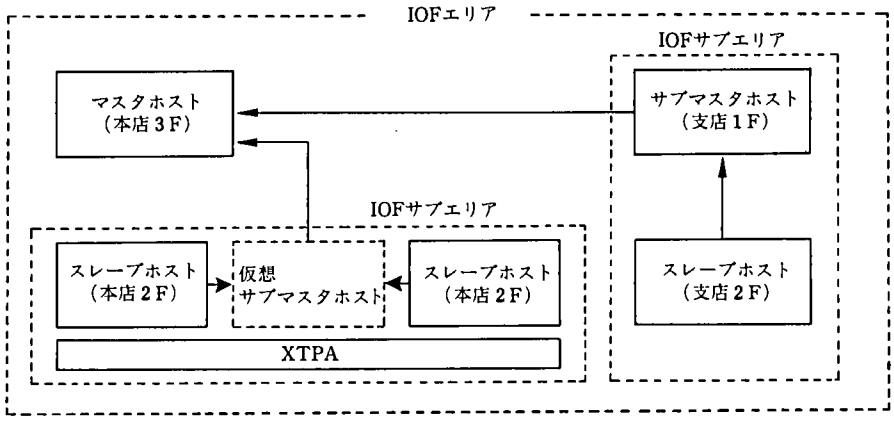


図1　IOFエリアの構成例

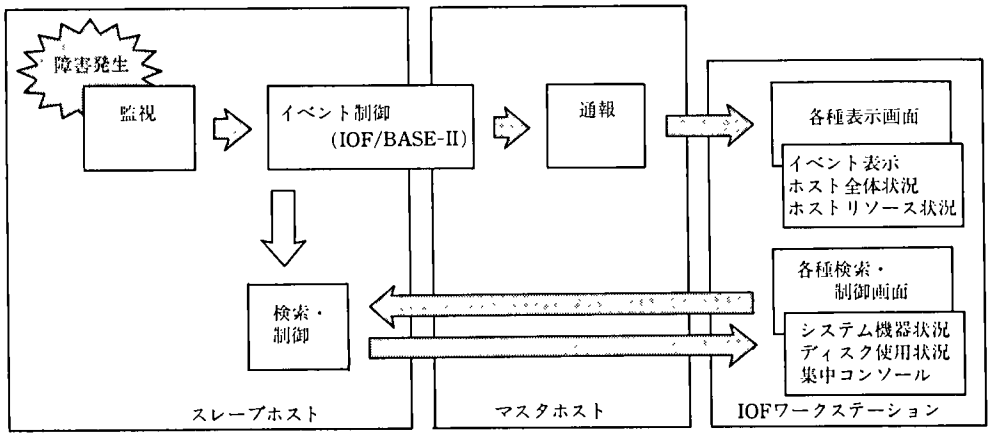


図2　IOF/MONIの機能構成

3.2.1 監　視　機　能

IOF/MONIの監視機能は，IOFエリア内の各々の監視対象ホスト（スレーブホスト/サブマスタホスト/マスタホスト）ごとに実行される．監視機能は，各ホストで発生した障害・警告の検知および定期的な稼働情報の採取を行い，監視情報を拡張IOFの基本機能であるイベント・メッセージの形式で作成する．IOF/MONIの監視機能は，次の二つの機能から構成される．

1)　ホスト障害監視……コンピュータ・システムの停止やシステム内の構成機器(CPU，メモリ，各制御装置，ディスク装置等)の状態異常の検出，プログラム障害等，オペレータの介入が必要な項目を監視する．

2)　ホスト稼働監視……コンピュータ・システム機器の使用状況（CPU利用率，メモリ利用率，ディスク装置のIO利用率，等)，トランザクション処理数や走行中のバッチラン数などのワークロードを監視する．

IOF/MONI は，標準の監視機能による監視情報の発生だけではなく，使用者独自の監視情報を受け付けるための使用者プログラム・インタフェースとユーティリティ・プログラムを用意している．

3.2.2 通 報 機 能

IOF/MONI の監視機能で作成される監視情報は，拡張 IOF の基本機能である IOF/BASE-II のイベント制御機能により上位ホストに通知され，さらに通報機能によりオペレータに通報される．

1) 上位ホスト通知……集中監視制御の運用は，IOF エリア内のマスタホストにオペレータを配置するのが通常である．イベント制御機能により，各スレーブで発生したイベントはマスタホストにイベント・メッセージとして通知される．上位ホストの立ち上がりが遅れている場合や，上位ホストで障害が発生し回復するまでの間は，下位ホストがその間に発生したイベントを蓄積する．通知可能な状態になると，イベント・リカバリ処理が起動し，蓄積されていたイベントが上位ホストに通知される．

2) 通報機能……監視機能により生成されたイベント・メッセージをオペレータに通報するための機能である．オペレータの監視業務を支援するための重要な機能であり，オペレータが行動すべきイベント・メッセージを日本語で通報し，障害等に対する対応漏れを防止する．また，IOF エリア内のホストの運転状況および稼働状況など，ホストの全体状況（障害・警告の有無）が把握できる．通報媒体としては IOF ワークステーションと音声出力装置があり，IOF ワークステーションを大型ビデオ・プロジェクタに接続して監視画面を拡大表示することも可能である．なお，音声出力装置はあらかじめ記憶されている文言の組み合わせにより，通報内容を音声で通報する装置のことである．

3.2.3 検索/制御機能

IOF/MONI の通報機能により通知された監視情報に従って，オペレータは各障害または警告に対して行動（アクション）を起こす必要がある．検索・制御機能はオペレータの障害または警告に対する行動を支援するための機能であり，検索機能と制御機能から構成される．

1) 検索機能……検索機能は，オペレータ通報機能で通知される障害または警告のイベントに対して，オペレータが行動を起こすために必要な詳細情報を検索するための機能である．IOF ワークステーションから操作する各種画面を提供している．また，集中監視制御の運用をより充実させるために，拡張 IOF の他サブシステムから提供されるマンマシン・インタフェースの一部の機能もこの検索機能に位置付けられる．

2) 制御機能……制御機能は，通報機能で通知される障害または警告のイベントに対して，オペレータが対応するための支援機能である．オペレータが各ホストのシステム・コンソールから対応するための操作が，IOF ワークステーションの集中コンソール機能で可能となる．また，検索機能と同様に，集中監視制御の運用をより充実させるために，拡張 IOF の他サブシステムから提供されるマンマシン・インタフェースの一部の機能もこの制御機能に位置付けられる．

3.3 マンマシン・インタフェース

IOF/MONIのオペレータ通報機能および検索・制御機能のマンマシン・インタフェースとして，拡張IOFの共通インタフェースであるIOFワークステーションの画面群が提供される．コンピュータ・システムを単一の制御点（通常はマスタホスト）から集中監視制御運用を行うため，IOFワークステーションの各種画面を利用することにより，オンサイトのオペレータと同様の監視・制御操作が可能になる．IOFワークステーションを利用した集中監視制御運用により，遠隔地・別フロアの無人運転，集中監視制御室の設置，機械室の無人化によるセキュリティ運用など，より良いコンピュータ・システム運用を実現できる．

3.3.1 IOFワークステーション

PW²上で稼働する運用コンソール・システムを使用して監視結果をグラフィカルに表示するとともに，マウスを使用した簡易なマンマシン・インタフェースを提供する．また，集中監視室などに大型ビデオ・プロジェクタを設置して画面を拡大表示することも可能である．IOF/MONIから提供される画面の特徴は以下のとおりである．

- 監視用の画面をグラフィカルに表示するため，状況が一目で把握できる．
- マウスを使用したメニュー中心の操作のため，コマンドなどの習熟を必要としない．
- 複数の画面をマルチ・ウィンドウ表示するため，監視と制御の操作が同一のIOFワークステーションにより可能である．
- 使用者による画面のカスタマイズ機能を使用して，IOF標準監視画面のカスタマイズ表示や使用者監視情報の組込み表示が可能である．

IOF/MONIで提供するウィンドウ群は一台のIOFワークステーションで表示可能である．使用者は監視対象ホストの数，オペレータの人数など監視業務の環境を考慮して，IOFワークステーションの台数および大型ビデオ・プロジェクタの採用を決定する．ホストグループ別またはウィンドウ機能別に複数のIOFワークステーションを設置することもある．また，IOFワークステーションはマスタホストだけではなくサブマスタホストやスレーブホストに接続することも可能であり，運用上必要な画面だけを表示することもできる．

3.3.2 IOF/MONIの画面体系

IOF/MONIは，集中監視制御運用を支援するためにIOFワークステーションから操作できる各種の標準画面を提供している．各画面は通報機能用と検索・制御機能用に大別される．画面体系は図3のとおりである．

1) イベント表示画面

オペレータに通報するイベントを日本語表示する．現在発生中の障害または警告が何か，対応状況はどうなっているかが一覧できる．

2) ホスト全体表示画面

IOFエリア内のホストの運転状況および稼働状況などを表示する．

3) ホストリソース詳細状況表示画面

特定ホストのシステム機器の利用状況，システム負荷状況などの詳細情報を表示する．

4) ホスト状況表示画面

システム機器状況およびディスク使用状況の詳細情報を表示する.

5) 集中コンソール

IOF エリア内のホストのコンソール操作やメッセージ検索を行うための画面群である.

6) 他サブシステム画面

拡張 IOF の各サブシステム (IOF/WORK, IOF/RLS など) が各システムの制御用に提供する画面である. イベント画面に表示された障害の原因解析後, 各サブシステムで提供される画面をを使用して, 再実行などの適切な対応指示ができる.

なお, IOF/WORK と IOF/RLS については, 本特集号別稿の『統合運用システム拡張 IOF の概要』を参照されたい.

7) オペレータ対応

イベント表示された障害または警告の内容によっては拡張 IOF が提供する IOF ワークステーションの機能で対応できない場合があり, その際のオペレータによる保守員のコール, 開発担当者への連絡などが必要なアクションを示している.

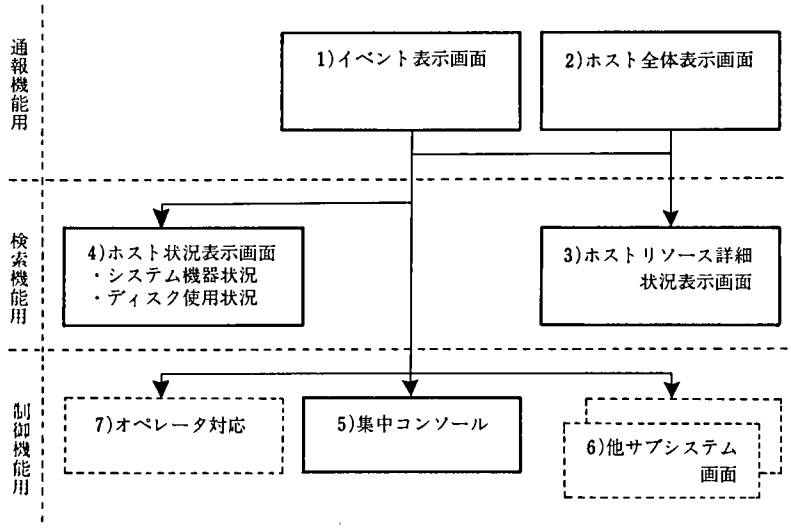


図 3 IOF/MONI の画面体系

4. IOF/MONI の監視機能

IOF/MONI の監視機能は, ホスト障害監視とホスト稼働監視の二つの機能から構成される. 各監視機能のために監視プログラムが IOF エリア内の各々の監視対象ホストごとに実行され, 監視インターバル, 監視対象項目については標準値を使用者の環境に合わせて変更できる. ここでは IOF/MONI の各監視機能について, その目的および内容を解説する.

4.1 ホスト障害監視

ホスト障害監視とは，コンピュータ・システムの停止，システム内の構成機器(CPU，メモリ，各制御装置，ディスク装置，等）の通常運用時の状態異常の検出，オンライン・プログラムやバッチラン等のオペレータ介入を必要とするホスト障害について監視する機能である．ホスト障害監視は障害を検出すると，監視情報（イベント）を生成し，イベント制御機能により障害発生ホストから上位ホストに通知する．ホスト障害監視の監視主要項目は表1のとおりである．

表1　ホスト障害監視の監視内容

分　類	主な監視内容
システム構成監視	・ホストダウン等のシステム障害の監視 ・CPU/メモリ/制御装置/ディスク装置/テープ装置のシステム機器の監視(モジュールまたは装置が予定通りの状態であるかを検査)
I/O障害監視	・ディスク装置のI/O障害(ファイル障害)，CTL障害等の監視
プログラム障害監視	・オンライン・プログラム(TPS)障害の監視
システム運用	・オペレータがシステム・コンソール・メッセージを監視し，対応が必要な事象の検出(応答型メッセージの発生，マスストレージ/テープ・ホールドの発生，テープのマウント要求等)
重要ラン監視	・スケジュール・ラン以外の重要ラン障害の監視(監視対象の常駐ラン等の重要ランは監視通告用プログラムを使用した事前登録が必要)
メッセージ監視	・使用者が事前に監視対象として登録したシステム・コンソール・メッセージの監視 ・IOF/MONIで特定した形式のメッセージ発行によるイベント通知(監視対象メッセージの事前登録は不要)

4.2 ホスト稼働監視

ホスト稼働監視とは，コンピュータ・システム内の構成機器の利用状況（CPU利用率，メモリ利用率，ディスク制御装置の利用率，等)，トランザクション処理数，走行中のバッチ・ラン数などのシステム負荷状況を一定時間ごとに監視する機能である．

ホスト稼働監視は一定時間ごとにデータ採取を行い，稼働データとして監視情報(イベント）を生成し，イベント制御機能により上位ホストに通知する．構成機器の利用状況監視にはSIP (Software Instrumentation Package)を使用している．PAR (Performance Analysis Routine) がシステムの詳細な稼働状況の調査に利用される

表2　ホスト稼働監視の監視内容

分　類	主な監視内容
システム機器の利用状況	・システム本体のSPU利用率(全体，詳細) ・システム本体のメモリ利用率(全体，詳細) ・ディスク制御装置利用率(ビジー率，I/O回数) ・ディスク装置利用率(ビジー率，I/O回数，I/O時間)
システム負荷状況	・単位時間当たりのトランザクション処理数 ・トランザクションのキュー数 ・走行中のバッチ/デマンドラン数 ・スワップ発生率 ・応答型メッセージ数 ・テープ/マスストレージのホールドラン数 ・その他システム・コンソールからの任意型キーインであるシステム状況表示(SSキーイン)で表示される内容

のに対して，IOF/MONI のホスト稼働監視は定常的な利用状況の把握に利用する．採取された各項目の現在値に対し，使用者が設定した基準値との検査を行い，警告状況を検出すると警告用の監視情報（イベント）を生成する．また，採取された監視情報はログ・ファイルとして保存され，標準提供されるログ・ファイル入力ルーチンを使用することにより，使用者独自の帳票を作成することも可能である．ホスト稼働監視の監視主要項目は表2のとおりである．

5. IOF/MONI の通報機能

IOF/MONI の通報機能は，監視機能で作成された監視情報(イベント・メッセージ)をオペレータに通報するための機能である．オペレータの監視業務を支援するための重要な機能であり，オペレータが行動すべきイベント・メッセージを日本語で表示するイベント表示画面と，IOF エリア内のホストの全体状況（障害・警告の有無）を表示するホスト全体表示画面を IOF ワークステーションの画面として提供している．オペレータは通報機能により，障害・警告に対する適切な処置や保守員への連絡など，速やかな行動が可能となる．その他の通報媒体として IOF ワークステーションの画面を拡大表示する大型ビデオ・プロジェクタ，あらかじめ記憶されている文言の組み合わせにより通報内容を音声で通知する音声出力装置がある．通報媒体の種類・台数については，監視業務の環境により決定する．次に IOF/MONI の通報機能が提供する画面について解説する．

5.1 イベント表示画面

オペレータの監視業務を支援するための最も重要な画面で，現在，オペレータが行動すべきイベント内容を日本語表示する（図4）．この画面により，現在発生中の障害または警告が何か，対応状況はどうなっているかが一覧でき，対応漏れまたは監視業務の引継漏れを防止することができる．オペレータはイベント内容に従って，検索用画面により詳細状況を把握すると共に，目的別の制御用画面を使用することにより障害または警告に対する処置が IOF ワークステーションから可能になる．イベント表示画面の日本語表示の文言はカスタマイズが可能であり，使用者の慣用語で表示するこ

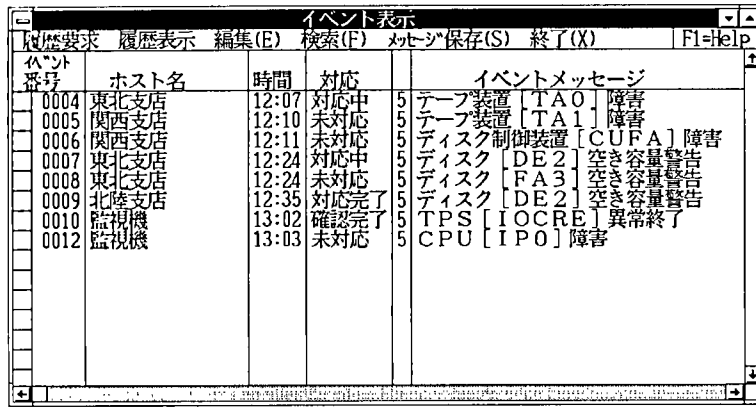


図 4 イベント表示画面の例

ともできる．表示されたイベントは，一部の障害を除いて，オペレータの対応が完了し該項目が正常状態に復帰すると自動的に消去される．

5.2 ホスト全体表示画面

IOFエリア内のホストの運転状況および稼働状況などを表示する画面で，ホストの全体状況（障害・警告の有無）が把握できる（図5）．監視対象ホストの数が多い場合，運用管理者にとって，とくに有効な画面である．ホスト全体表示画面は，標準提供された画面を使用者独自の形式（レイアウト，監視項目の削除・追加）にカスタマイズすることができる．

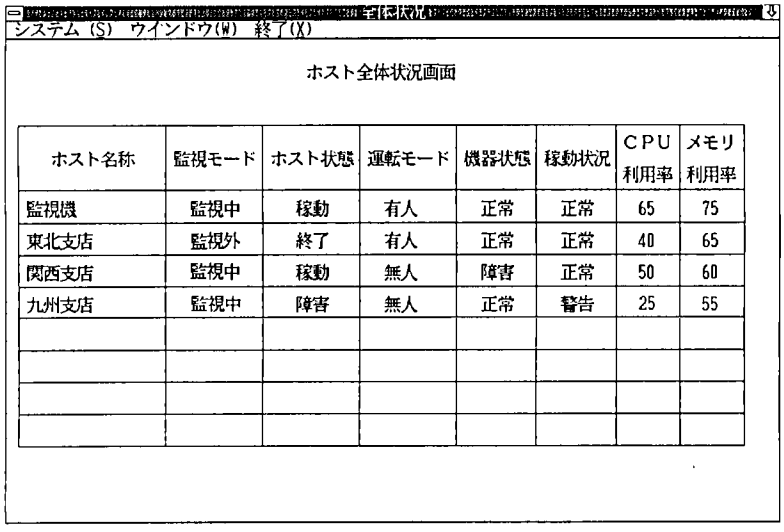


図5　ホスト全体表示画面の例

6. IOF/MONIの検索・制御機能

IOF/MONIの通報機能により通知された監視情報に従って，オペレータは各障害または警告に対して行動（アクション）を起こす必要がある．検索・制御機能はオペレータの障害または警告に対する行動を支援するための機能であり，検索機能用の画面群と制御機能用の画面群を提供する．各画面はマウスを使用したメニュ中心の操作が可能であり，コマンドなどの習熟を必要としない．

また，IOF/MONIの制御機能として，通報機能によるオペレータ通知だけではなく，イベント制御機能の使用者通知機能（アクション機能）を利用することにより特定の監視情報（イベント）に対する自動処理も可能である．たとえば，障害の種別に対して使用者が事前に準備した手続きに従って，障害に対する自動リカバリ処理を実施することが可能である．

6.1 検索機能用画面

検索機能はオペレータが必要な詳細情報を検索するためのホストリソース詳細状況表示画面とホスト状況表示画面をIOFワークステーションの画面として提供している．集中監視制御の運用をより充実させるために，拡張IOFのサブシステムである

IOF/WORK が提供するマイクロオペレータの一部の機能や IOF/RLS が提供する状況検索用のウィンドウもこの検索機能に位置付けられる．

6.1.1 ホストリソース詳細状況表示画面

ホスト全体表示画面のメニュから，特定ホストを指定してシステム機器の利用状況，システム負荷状況などの詳細情報を表示する詳細検索用画面である．ホスト全体表示画面やイベント表示画面で表示された警告に対して，該当の監視項目名および現在値の検索，または特定ホストの詳細な稼働状況を把握するために使用する．ホスト全体表示画面と同様に，標準提供された画面を使用者独自の形式（レイアウト，監視項目の削除・追加）にカスタマイズすることができる．

6.1.2 ホスト状況表示画面

独立したメニュから起動し，システム機器状況およびディスク使用状況の詳細情報を表示する問い合わせ型の詳細検索用画面である．ホスト全体表示画面やイベント表示画面で表示された障害・警告に対して，該当の監視項目名および現在値を検索するために使用する．システム機器状況は指定したホストの監視対象システム機器の状態を，ディスク使用状況は指定したホストの監視対象ディスク装置の使用状況（空き容量）を表示する．

6.2 制御機能用画面

制御機能は，オペレータが各ホストのシステム・コンソールから対応するのと同様な操作が可能である集中コンソール画面を IOF ワークステーションの画面として提供している．また，検索機能と同様に，集中監視制御の運用をより充実させるために，拡張 IOF のサブシステムである IOF/WORK が提供するマイクロオペレータの一部の機能や IOF/RLS が提供する指示用の画面もこの制御機能に位置付けられる．

6.2.1 集中コンソールの機能

集中コンソールは，複数のコンピュータ・システムのコンソール・メッセージを収集し，単一の制御点からのコンソール操作やメッセージ検索を可能にする機能であり，IOF ワークステーションからのマウス操作またはキーボード操作でコンソール・メッセージの表示，応答，任意型キーイン入力等を可能にしている．

監視対象ホストで発生するコンソール・メッセージの取得とホスト間の通知はオペレーションの集中化とシステムの無人運転を支援するソフトウェアである統合オペレーション・システムを使用している．集中コンソール画面の操作方法はメニュ方式を採用しておりコマンドを習熟する必要はない．集中コンソールの機能と主な内容は表3のとおりである．

6.2.2 集中コンソール画面群

IOF/MONI の通報機能で通知された障害または警告の内容に従って，該当ホストのシステム・コンソールと同等の操作を IOF ワークステーションから行うための制御用画面である（図6）．応答型メッセージに対する応答，バッチランがエラー時に表示するエラー・メッセージの検索が必要な場合等に使用する．

集中コンソールの各画面は，IOF ワークステーション上にマルチ・ウィンドウ（複数の画面が重複して）表示される．該当の操作を実施するために，操作目的に応じて次の画面が提供される．

表3　集中コンソール機能の内容

機　能	主 な 内 容
メッセージ表示	・表示対象ホストごとの画面割り当て(各ホストのコンソール・メッセージを各画面に振り分けて表示) ・各ホストに対する任意型キーインの入力(各ホスト用画面の入力フィールドから操作) ・表示メッセージの検索(ホスト画面ごとに最新300メッセージがスクロール操作で検索可能) ・表示メッセージの印書
メッセージ応答	・表示対象ホストにて発生した応答型のメッセージの表示(応答型メッセージ発生時，応答用画面をトップ・ウィンドウとして表示) ・応答型メッセージに対する応答(応答用画面に表示されている応答型メッセージを選択し，入力フィールドから応答内容を入力)
メッセージ再表示	・過去に表示されたコンソール・メッセージの再表示(一回の再表示操作で300メッセージまでがスクロール操作で検索可能) ・ホスト指定や各種条件の指定(時間指定，文字指定が可能) ・再表示メッセージの印書
運用設定変更	・表示対象ホストの変更 ・メッセージ応答権の設定(獲得，放棄)変更(IOFエリア内の複数のIOFワークステーション間の応答権の変更が可能)

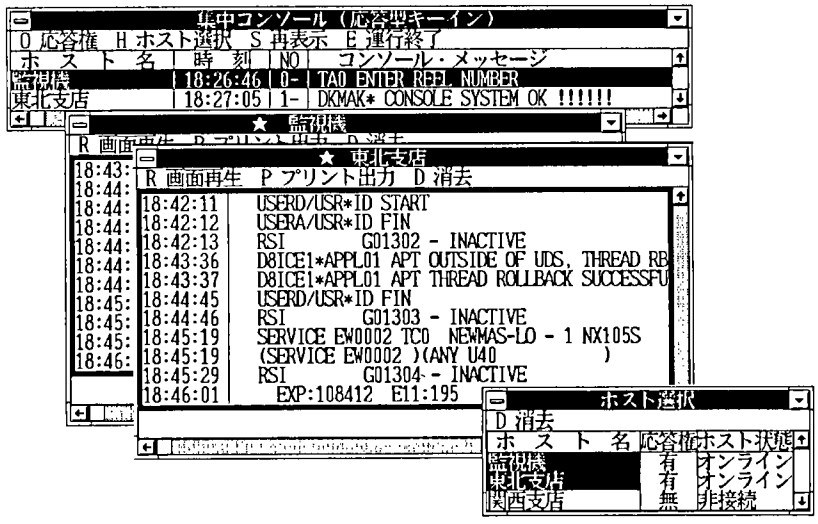


図6　集中コンソール画面の例

1)　ホスト表示画面……メッセージ表示の機能を提供する．表示対象ホストごとにそれぞれの画面が割り当てられ，各ホストで発生したシステム・コンソール・メッセージの検索および任意型キーインの入力を行うために使用する．

2)　集中コンソール画面……メッセージ応答の機能を提供する．応答型メッセージを表示する画面であり，主に応答メッセージを入力するために使用する．また，この画面は各種操作のメニュを兼ねている．

3)　メッセージ再表示画面……メッセージ再表示の機能を提供する．集中コンソール画面のメニュから呼び出され，特定ホストの過去のシステム・コンソール・メッセージを使用者の指定条件で検索するために使用する．

4) ホスト選択画面……運用設定変更用の機能を提供する．集中コンソール画面のメニュから呼び出され，表示対象ホストの変更を行う場合に使用する．

7. IOF/MONIの適用例

IOF/MONIで提供される機能およびIOFワークステーションで提供している標準画面で集中監視制御の運用は十分可能であるが，IOFワークステーションの画面のカスタマイズ，使用者独自のイベント追加等を行うことにより，ユーザの環境に適したより良い集中監視制御の運用システムが構築できる．

本章ではIOF/MONIの適用方法を事例をおりまぜて解説する．

7.1 ホスト全体表示画面のカスタマイズ

IOF標準のホスト全体状況画面およびホストリソース詳細画面に対して，使用者独自の監視情報またはIOF標準の監視情報を使用者定義の画面として，IOFワークステーションに表示できる．使用者はホスト側の監視情報をIOFワークステーションの監視画面に表示するために，次の2通りのカスタマイズが可能である(図7)．

1) 使用者監視画面……ホスト側はIOF/MONIの標準監視機能で行い，ホスト全体表示画面のみを使用者独自の監視画面としてカスタマイズする．

2) 使用者監視システムと使用者監視画面……IOF/MONIの標準監視機能以外に使用者の監視システムで使用者イベントを作成し，使用者独自の監視画面としてカスタマイズする．

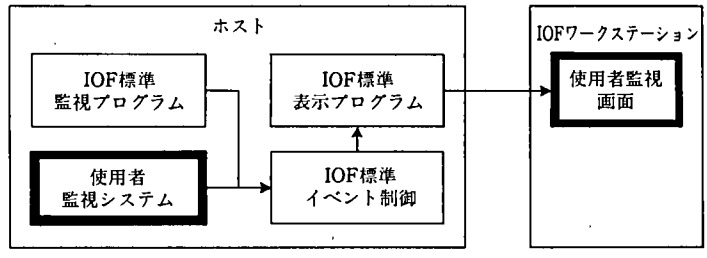


図 7 使用者監視システムと監視画面のカスタマイズ

■事　例

1) IOF/MONIの機能範囲外であるネットワーク監視システムと連携しネットワーク監視の情報を使用者のデザインしたネットワーク監視画面として表示する(図8)．

2) ホスト全体表示画面の標準表示項目を変更して，使用者の運用に合わせた他の標準項目を表示する(例：CPU利用率，メモリ使用率の代わりに未応答メッセージ数，ホールド・ラン数を表示)．

7.2 使用者イベントの追加

IOF/MONIの標準イベントに対して，使用者独自の監視項目をイベントとして追加することにより監視業務の充実を計ることができる．また，イベントの文言については，IOF/MONIの標準イベントの編集用パラメータを変更することにより使用者の慣用語でイベント表示することが可能である．使用者イベントの発生方法は次の2

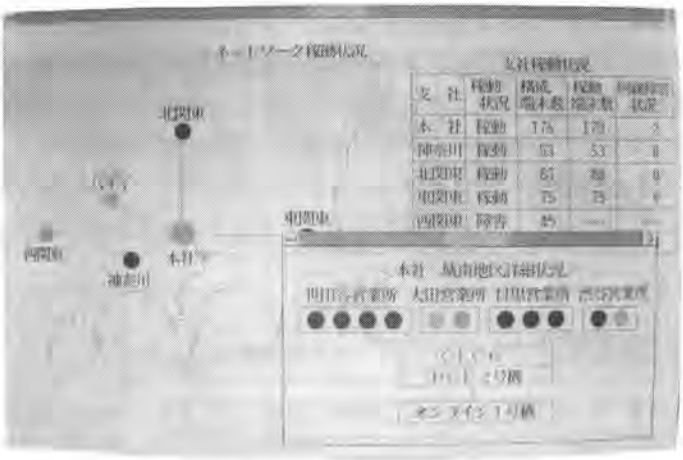


図 8　使用者監視画面の例

通りがある.

1)　使用者メッセージ監視機能の利用……既存の使用者プログラムから表示するメッセージを検査しイベント表示する.または,OSが表示するシステム・コンソール・メッセージを利用してイベント表示する（例：テープのマウント要求のメッセージをイベントとしてリール識別名/ファイル名等を表示する).

2)　使用者プログラムからのイベント発行……使用者プログラムから拡張IOFが提供するプログラム・インタフェースを使用して,または使用者JCLからユーティリティ・プログラムを使用して監視情報を拡張IOFのイベント制御機能に引き渡す.なお,この機能は拡張IOFの基本機能として提供される.

7.3　イベント制御機能のアクション機能

IOF/MONIの標準イベントおよび使用者独自のイベントに対して,使用者のアクションを事前に登録することにより,発生した障害,警告に対して自動的に使用者プログラムまたは使用者バッチランを起動することが可能である.

■ XIS連携

ディスクのI/O障害（ファイル障害）またはCMS等の重要ラン・エラーをIOF/MONIで監視し,障害を検知すると,該当イベントをXISに通知することにより,自動的なファイルの回復処理,ホストの切り替え処理を可能にしている.

7.4　異機種監視

IOF/MONIの標準イベント形式および拡張IOFの基本機能であるコマンド制御の形式を開示することにより,シリーズ2200以外のホスト障害監視またはホスト稼働監視が可能である.

C社では,この方式を採用して他メーカのホスト監視(ハートビートによるホスト停止の監視,システム負荷状況の表示,バッチランの障害等）を行い,イベント表示画面,ホスト全体状況画面に監視情報を表示して異機種を含めた集中監視制御運用を実現している（図9).

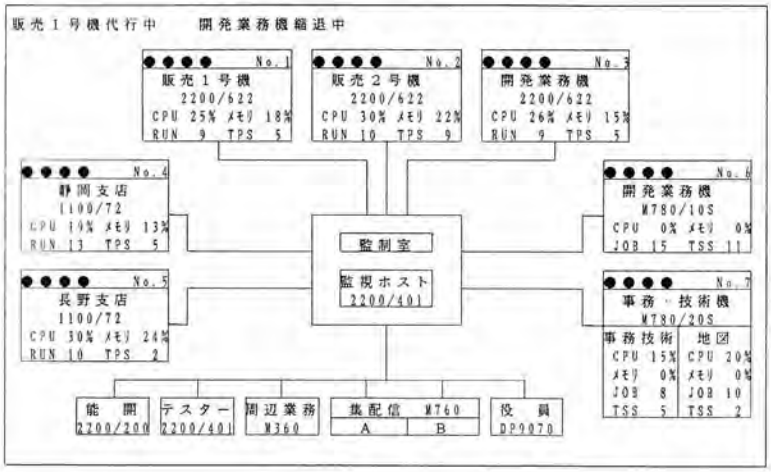


図 9　異機種監視画面の例

7.5　集中監視室の設置

IOF/MONI のマンマシン・インタフェースである IOF ワークステーションおよび大型ビデオ・プロジェクタ，音声出力装置を集中監視室に集約し，機械室の無人運用を実現できる．また，すべての監視情報を集約することにより複数のコンピュータ・システムの少人数運転がさらに容易となる（図 10）．

図 10　C 社集中監視室

8. お わ り に

コンピュータ・システムの運用形態として，UNIX* を中心とした分散システム運用が急速に展開されつつある現在，IOF/MONI では分散システム運用との連携が当面の課題である．集中監視制御運用を構築する上で，マンマシン・インタフェースは必要不可欠であり，統合された機能提供が必要となる．つまり，シリーズ 2200 側の監視運用と分散システム側の監視運用が相互に監視/制御情報を授受できる連携機能が，使

* 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．

用者の環境に合わせた柔軟な集中制御運用を可能にする．

また，集中監視制御の運用を実施するにあたっては，システム運用部門だけでシステム構築ができるわけではない．サービス体制に対するシステム利用部門の理解，オペレータが的確に行動するために必要なメッセージ表示，不要な応答型メッセージの削除等，使用者アプリケーションを担当するシステム開発部門の協力も，より良い集中監視制御の運用を構築するために必要である．

参考文献　［1］　シリーズ2200・1100統合運用システム(IOF)/MONI解説書，日本ユニシス(株)，1993.

執筆者紹介　荻　原　和　彦 (Kazuhiko Ogihara)

昭和26年生．50年東京理科大学工学部経営工学科卒業．同年日本ユニシス(株)入社．電力関連のSEサービスに従事した後，平成元年よりIOF開発を担当し現在に至る．システム技術第一本部システム企画開発二部所属．

分散システムの統合運用管理ツール——SPO

SPO——an Integrated Systems Management Tool for Distributed, Open Client/Server Environments

入 貝 健 介

要 約 企業情報システムの分散化，マルチ・ベンダ化が浸透するにつれ，分散されたホスト・システム上の情報やコンピュータ・リソースをLANなどのネットワークで結ぶことによって，より有効に活用する必要に迫られてきている．しかし，異機種分散システム間では，アプリケーションの相互運用性に問題があり，実行環境のみならず，システム全体の統合運用管理にも大きな障害となっている．

U 6000シリーズ上で稼働するSPOは，このような異機種分散システム環境の統合管理を可能とする全く新しいタイプのシステム運用管理ツールである．

SPOは，操作員インタフェースとしてMotif*のマルチ・ウィンドウ画面を用い，シリーズ2200，Aシリーズ（予定），UNIX**(予定)，IBM***(予定)などのマルチ・ベンダ環境の複数ホストを統合して1台のX端末から監視/操作/制御することができる．また，多くの自動化機能と使い勝手の良さを提供することにより，操作員は余分な負担を感じることなく複数のホストを統合運用できるようになる．

SPOはUAのSMSに準拠したプロダクトであるため，将来的にはCORBAに準拠した『オブジェクト指向管理基盤』をプラットフォームとして，異機種分散システム環境を統合的に運用管理すると思われる．本稿では，汎用性のあるシステム運用管理ツールを目指すSPOの現状と，今後どのように発展していくかを述べる．

Abstract Because of the ongoing decentralization of corporate information systems and the growing acceptance by computer users of more multivendor-based installations, arising from among users are the needs to make more effective use of information on distributed host systems as well as computer resources by means of networking——through local area networking, for instance. However, interoperability of applications in distributed heterogeneous computing has been a great barrier not only in application processing environments but also in integrated operation management of the whole system. SPO (Single Point of Operations), operating on the U6000, is a totally new type of systems management tool that allows integrated computer operation in such open, multivendor client/server environments.

Using Motif multi-window screens as its interface with operators, SPO enables one U 6000 terminal unit to monitor/operate/control, in an integrated manner, such heterogeneous computing as is made up of the Series 2200, the A Series (being planned), UNIX products (now on the drawing board) and IBM computers (in the scheduled list). Also by providing a wide range of unattended operation functionalities with greater ease of use, SPO helps operators execute integrated multi-host operation management without making them feel that they are doing extra loads of work.

Being a UA/SMS-based tool, SPO is expected to grow into an enhanced version with its platform

* Motifは Open Software Foundation の登録商標である．
** 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．
*** IBMは米国 International Business Machines 社の登録商標である．

based on CORBA-compliant "object-oriented management infrastructure." This paper describes the current position of SPO which is aimed to be an established general-purpose systems management tool, and how it is going to develop further in the future.

1. はじめに

企業内の組織の分権化にともない，80年代以降，企業内の情報システムもかつての中央集権的なシステムから部門ごとに必要な情報を処理する分散型のシステムへとシフトしていった．

それと同時に，分散したメインフレーム上の企業情報の一部は，サーバ/クライアント・モデルやユーザ・フレンドリなGUIなど，新しいソフトウェア・パラダイムのもとで急速に成長してきたエントリー・コストの安いワークステーションやPCへと分散シフトしていった．

このような状況の中で，異なったハードウェアやOSが混在するいわゆるマルチ・ベンダ環境が徐々に形成されていった．

しかし，LANなどのネットワーク技術が普及するにつれ，部門ごとに分散されたシステム環境を互いに接続することで企業内の情報およびコンピュータ・リソースをもっと有効に利用しようという気運が高まった．ところが，このようなマルチ・ベンダ環境では，アプリケーション間の相互運用性（協調動作）に問題があり，このことがシステム・ネットワーク全体の運用効率を大きく低下させる原因になっている．

この問題には二つの重要な側面がある．エンドユーザにとっては，この問題が企業内情報システムのオープン化の大きな障害になっている点であり，メインフレーマにとっては，これがメインフレームと分散システム環境との連携の大きな障害になっている点である．

アプリケーション間の相互運用性を高め，この分散システム・ネットワーク全体の効率を高めることを目的としたソフトウェア・アーキテクチャがUA(Unisys Architecture)である．(UAでは，このようなネットワークのことをインフォメーション・ネットワークと呼ぶ)

本稿では，UAのアーキテクチャの中でもとくにインフォメーション・ネットワークの運用管理を目的としたフレームワークSMS(Systems Management Services)に準拠したSPO(Single Point of Operations)の概要と汎用性のあるシステム運用管理ツールを目指すSPOが今後どのように発展していくかについて紹介する．

2. SPOが目指すもの[4][5][6]

企業情報システムの分散化，マルチ・ベンダ化が浸透するにつれ，しばしば指摘されるこのアプリケーション間の相互運用性の欠如の問題は，アプリケーションの実行環境だけの問題にとどまらず，システム全体の統合運用管理の分野にもそのまま当てはまる．すなわち，これは従来のシングル・ベンダ用のシステム管理手法や運用ツールでは，このような複雑な分散処理環境に対処できないことを意味しており，まったく新しい発想の統合運用管理ツールが必要になってきていることを示している．

SPOは，このような分散システム環境の統合管理を目的としたU 6000シリーズ上

で稼働するシステム運用管理ツールである．

当社では，フレームワークに準拠したシステム運用管理プロダクト群をその管理目的に応じて四つの管理分野（プラットフォーム管理，分散システム管理，アプリケーション管理，ネットワーク管理）に分け，さらにそれらの統合的なインタフェース（統合管理 View）を加えた合計五つの分野で具体的なプロダクト群を ASMF（Advanced Systems Management Framework）として体系化し 94 年 2 月に発表している．

ASMF の概要説明に関しては本稿の目的ではないので他稿に譲るとして，ここでは分散システム環境におけるシステム運用管理ツールに求められる一般的要件について考えてみよう．

日本では，オブジェクト指向技術（以下 OO 技術と略す）に関しては，つい最近まで，いつの時代にもある『次世代の有望技術』の一つに過ぎないと考えられていたが，いまや，（とくにソフトウェア・ベンダやメイン・フレーマの間では）緊急に対処が必要な重要課題という認識が一般的になりつつある．

まずその一般的な理由としては，OO 技術を利用すれば現実の世界のモデリングが比較的容易になり，オブジェクト指向方法論や CASE ツールと組み合わせて開発生産性の向上，メンテナンス・コストの改善に大いに効果があるという評価が浸透してきたことである．

そして，もう一つの（さらに本質的な）理由として，OO 技術の本質が『物の複雑さを外から隠す』技術であり，そういう意味で，異機種分散ネットワーク環境における AP 間の協調動作を保証するという観点から考えると，リクエストする側の AP（オブジェクト）は，相手 AP（オブジェクト）がどこで，どのような環境（ハードウェア，OS，言語）で稼働しているか，などをいっさい気にする必要のないシンプルなインタフェースを構築するには OO 技術をおいては考えられないという意見が，とくにメインフレーマを中心に定着してきたことである．

これが分散システム環境でのシステム運用管理とどう関係するかというと，まず管理する側（管理マネージャ）と管理される側（エージェント）をすべてオブジェクトとして考えると，管理マネージャは，各管理対象（エージェント）がそれぞれどのような環境で稼働しているかをいちいち考慮しなくても，管理対象であるエージェントへメッセージを送ることができ，マネージャ/エージェント間の通信が成立する点である．さらに，管理するエージェントをクラスとして定義しておくと，新たにシステム構成に追加/変更するエージェントが生じても，OO 技術の一つである『継承』を使えば楽に定義ができる．また同じく OO 技術の一つである『多態性』により，管理マネージャは管理対象の種類の多少の違いを意識せずに同じ管理コマンド（メッセージ）を各エージェントに送付することができ，マネージャの構造をシンプルにすることができるという利点もある．

このように分散システム環境でのシステム運用管理にとって，OO 技術は必須の技術なのである．

分散システム環境でのシステム運用管理ツールを目指す SPO が，どのような形で OO 技術を取り込んでいくかについては 4 章で説明する．

3. SPOの機能概要

本章では，現時点でSPOが提供している機能を中心に説明する．

3.1 SPOの特徴

SPOはイーサネット*LANで接続されたシリーズ2200をはじめ，Aシリーズ，UNIX, IBMなどのマルチ・ベンダの分散システムを一か所から統合運用管理することを目的としている．

SPOの特徴を以下に示す．

1) ASMFに準拠した分散システム環境でのシステム運用管理ツール
   - 当社のSMSフレームワークにおける中核的プロダクトである．
   - シリーズ2200，Aシリーズ(予定)，UNIX(予定),IBM(予定)などマルチベンダ環境を統合して管理する．
2) システム全体の信頼性の向上
   - SPO障害時も管理対象には一切影響を与えないフォールト・トレラントな設計である．
   - 熟練したシステム管理者がインフォメーション・ネットワーク全体を統合管理可能である．
   - 監視対象ホスト・システムの障害発生時は，SPOからの障害回復およびSPOのコンソールログ監視機能による障害追求など速やかな対応が可能である．
3) システム運用コストを大幅に削減
   - 省力/無人運転中のマルチ・システムの集中監視/操作/制御によるオペレーションコストの削減ができる．
4) 扱い易い操作員インタフェース
   - 業界標準GUIを使用した大画面マルチ・ウィンドウを使用する(マウスによる操作)．
   - 個々のシステム操作は従来と互換性を保ちながら，可能な限り対象機種の操作性の違いを意識させない統一された操作員インタフェースを提供する．
   - 複数ホスト・システムを効率良く監視/操作/制御できるように工夫された操作員インタフェースの提供を行う．

3.2 SPOの機能

3.2.1 SPOネットワーク

SPOが支援する典型的なSPOネットワーク構成を図1に示す．

3.2.2 SPOの各種ウィンドウが提供する機能

SPOは，日本語MotifをGUIとして採用し，マルチ・ウィンドウ上で各種機能を並行して実行する．代表的なウィンドウの種類とその機能は以下の通りである．また，画面の表示例を図2に示す．

① SPOシステム・ウィンドウ：SPOを制御するためのコマンド・メニュと警告アイコンから構成される．

② 警告ウィンドウ：警告アイコンをクリックすると表示される詳細ウィンドウ．

* イーサネット(Ethernet)は米国Xerox社の登録商標である．

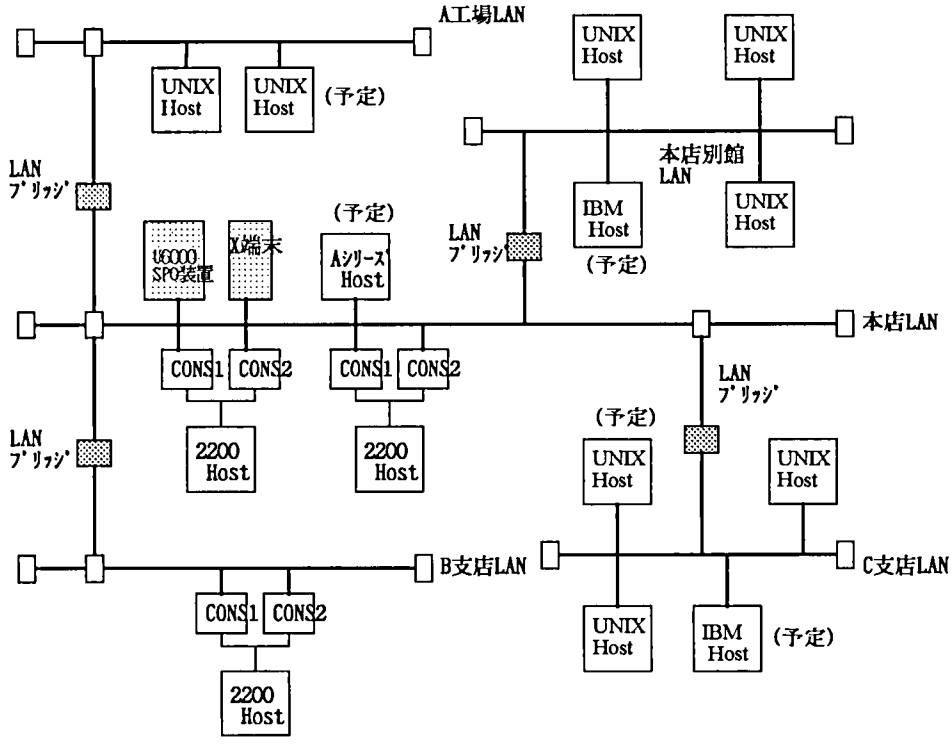


図 1　典型的なSPOネットワーク例

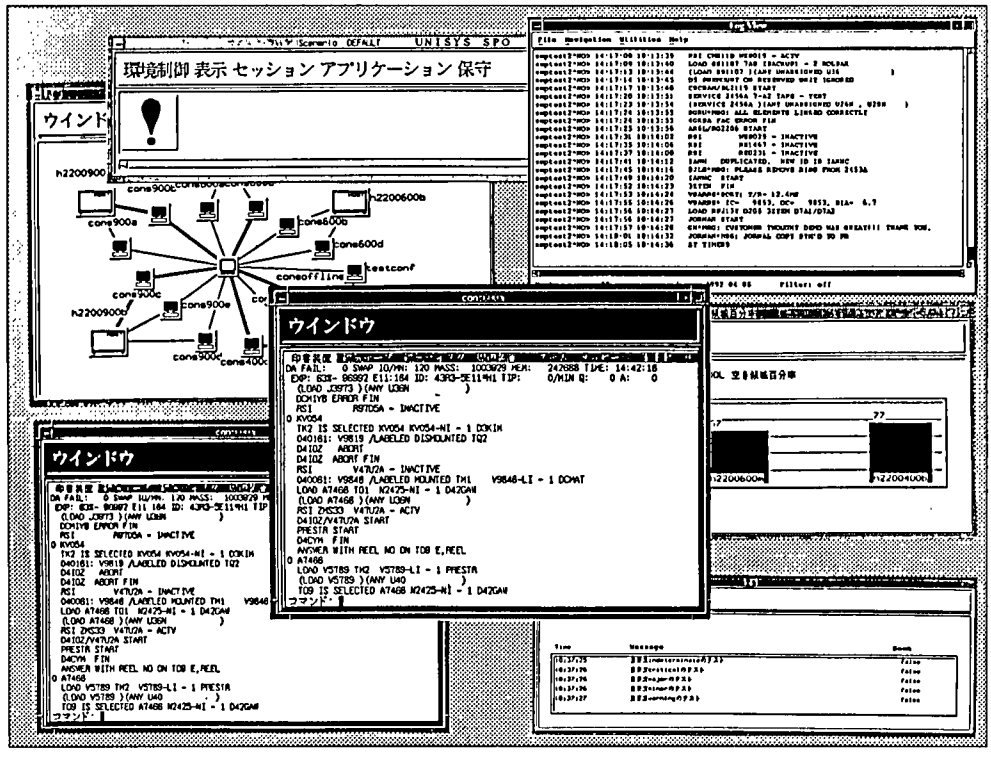


図 2　SPOの代表的な表示画面の例

③ サイト構成図：監視対象のホスト・システムとコンソールのアイコンおよびその間の回線の構成が図で表現され，それぞれの色はその状況を表す．

④ ターミナル・ウィンドウ：U 6000 上の IS 6000 の会話型サービスを利用して，SPO で UTS エミュレータ・ウィンドウをオープンし，対象ホストの端末として利用できる．

⑤ コンソール・ウィンドウ：システム・コンソールと 1 対 1 で対応し，これとまったく同じ表示と機能を持つ複写ウィンドウである．これを使って SPO よりシステムの監視/操作および制御を行う．シリーズ 2200，A シリーズのホスト・システムとの接続は，イーサネット LAN の TCP/IP が使用される．UNIX ホストとの接続は，Async 端末コントローラを通して行われ，U 6000 シリアル・ポートまたは LAN に接続される．

⑥ グラフィック・ウィンドウ：監視対象のホスト・システムの状況は，バーチャートや表の形でこのウィンドウに表示される．これらの状況は，あらかじめ決められたしきい値と常に比較され，異常があれば操作員に警告が通知される．

⑦ ログ・アプリケーション・ウィンドウ：各システムのコンソール・ログ，警告メッセージのログが SPO 装置上に 1 ファイル/1 日，最大 1 か月分保存される．このウィンドウにはフィルタリング機能があり，時間，文字列などで自由にログを検索，編集，印書および保存ができる．

⑧ ステータス・アプリケーション・ウィンドウ：これは現在開発中の機能であり，監視対象のマルチ・ホスト・システム環境のハードウェア・コンポーネント（たとえばテープ装置，ディスク装置，プリンタ，CU，メモリなど）やソフトウェア・コンポーネント（ユーザ・ランなど）の状況を監視し，これらのコンポーネントの状況を操作員にわかり易い一覧表やコンポーネント・アイコンなどにまとめて表示する．またこれらのコンポーネントに異常が発生した場合は，SPO の警告アイコンが赤くなり，操作員に通知される．

3.2.3 SPO の監視サービス

SPO は以下の条件を常時チェックしており，異常があれば警告アイコンを赤く点灯し，警告ウィンドウに警告内容を表示する．また，SPO のログ・ファイルにその情報を残すことができる．

① AMS データベース*に登録済みの文字列と一致するメッセージをホストから受け取った結果，それに対するアクションとして警告イベントが発生した．

② 監視ホストのシステム状況がしきい値を超えた．

③ SPIP(Single Point Interface Pipe)経由で外部アプリケーションから警告メッセージを受け取った．

④ SPO が内部エラーを受け取った．

しきい値の値と AMS データベースの内容はあらかじめシステム管理者が設定しておき，必要があれば，SPO 実行中に操作員がしきい値の変更やデータベースの切り換えを行う．

---

* 詳細は，4.1.2 項参照のこと．また，シリーズ 2200 以外のホスト用データベースとして，SP-AMS データベースが将来提供される．詳細は，4.2.1 項参照のこと．

3.2.4 外部アプリケーション・インタフェース(SPIP)

外部アプリケーションは，SPO装置であるU 6000上のUNIX Pipeファイルに対して警告メッセージを書き込むことで，SPOの警告アイコンを赤くし，操作員に警告を発することができる．また，この警告メッセージはSPOのログファイルにも同時に書き込むことができる．

3.2.5 SPOのコンソール・メッセージ監視機能

2200システム・コンソール上のAMSデータベースは，受け取ったコンソール・メッセージと登録済みのパターンを比較して一致すると，あらかじめ決められたアクションを実行することができる．

このアクションには，監視ホストへコマンドを送付したり，応答型メッセージに応答したりする以外に，SPOに対してイベント・レポートを送る機能が追加された．詳細は，4.2.1項を参照のこと．

3.2.6 フォールト・トレランスの提供

SPOはX端末およびSPO装置(U 6000シリーズ)を2重化し，それらに障害が発生した時は自動的に切り替わり処理を続行するホットスタンバイ機能を提供している．

4. SPOの内部構造と開発コンセプト[1][2][3]

SPOのプロダクトとしての発展過程は，大きく以下の三つの段階に分けられる．

1) レベル1……複数の2200ホストの統合管理
2) レベル2……多岐にわたるオブジェクトの統合管理
3) レベル3……汎用性のある分散システム管理ツールとしてのSPOの実現

4.1 レベル1：複数の2200ホストの統合管理

前章で説明した機能範囲を有し，SPOレベル1R3(92年10月ジェネラル・リリース済み)以降で提供されている．

特徴は，以下のとおりである．SPOのシステム構造概念図は図3に示す通りである．(図3の中の数字は各ウィンドウへの出力パスを示しており，本文とはとくに関係ない)

① 複数の2200ホスト管理
② オブジェクト指向データモデル
③ グラフィック・ユーザ・インタフェース
④ TCP/IP，イーサネットLAN

4.1.1 レベル1でのSPOの内部構造

レベル1でのSPOの構造概念図(図3)において，U 6000ボックス内の四角は，プロセスまたはシステム起動時に起動されるデーモン・プロセスを表す．

グラフィカル・ユーザ・インタフェースには，日本語Motifを使用する．また，SPOはオブジェクト指向型のデータモデル(DM)を包含している．これは，イベント・ハンドラを含んでおり，マウスやキーボードからの入力イベントやコンソールからのイベントなどを受け取り，関連するコールバック関数*を実行する．それにより，データモ

* コールバック関数は，イベントの種類ごとにあらかじめ用意されているC言語インタフェースのファンクション・コールである．

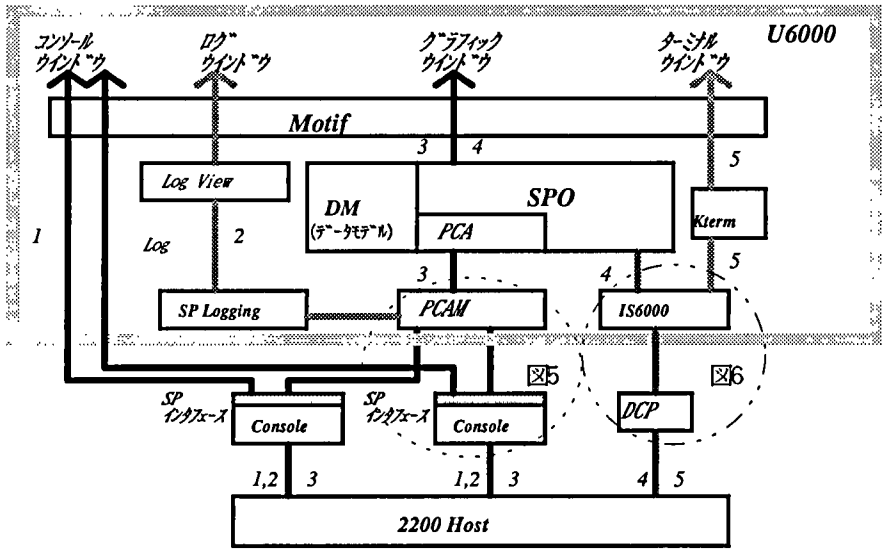


図 3 レベル1: SPO システム構造概念図

デル内のそのイベントに関連する属性値が更新され，データモデルの内部デーモンがそれを感知して関連するメソッドを実行するいわゆるイベント駆動型のデータモデルであり，SPO の中核となる推論エンジンである．

実際の SPO は，SNAP(Strategic Networked Applications Platform)*といわれる開発環境上で構築される．その上で構築された SPO のデータモデル(DM)からは，C 言語インタフェースの外部関数を呼び出すことができ，その関数を経由して，GUI (Motif), TCP/IP, 外部データベース, 外部アプリケーションなどにアクセスするための SNAP 関数(ライブラリ群)を簡単に利用することができる．これらの関連図を図 4 に示す．

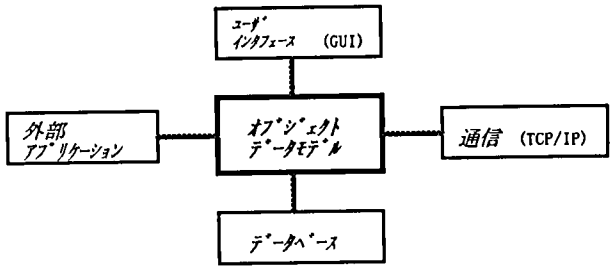


図 4 SNAP 関連図

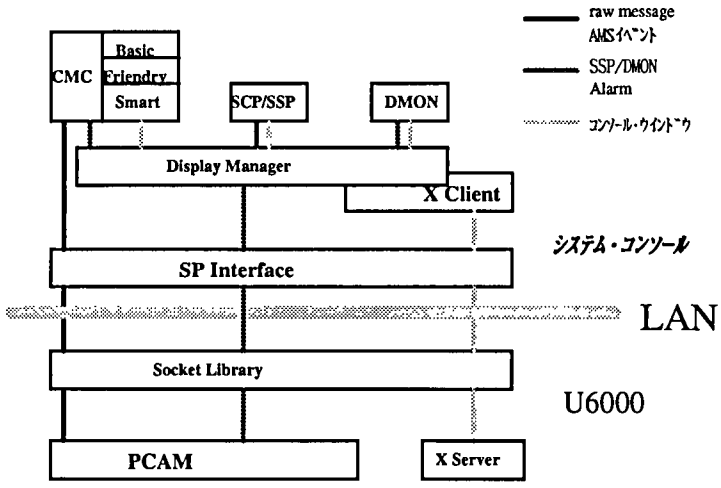
このデータモデルには，管理対象(2200 システム・コンソール，2200 ホスト・システムなど)がオブジェクトとしてクラス構造で定義されており，各オブジェクト間の通信は，メッセージの送付によって行われる．

そういう意味でこの SNAP は，ORB(4.3.2 項を参照のこと)の一種と考えることが

* SNAP は，Template Software, Inc. の登録商標である．

できる．

次に，2200 システム・コンソールと U 6000 との接続を図 5 に示す．



PCAM：　SP インタフェースと通信を確立した後，コンソールからのデータ(raw message，イベント)を受け取り SPO へ渡す SPO 装置上のデーモン・プロセスである．また，SPO からの入力もコンソールへ渡す．

SP インタフェース：2200 システム・コンソールの LAN インタフェースとして，TCP/IP プロトコルで PCAM とやりとりをする外部アプリケーションである．2200 システム・コンソール・ソフトウェアの一部としてインストールされる．

Display Manager：システム制御画面(SCP/SSP)，DMON セッション(PC や Host のフォールト状況を表示するコンソール・セッションである)，コンソール・アラームの各データを PCAM へ送ると同時に，X クライアントとして SPO(X サーバ)へコンソール・ウィンドウを出力する 2200 システム・コンソール上のコンポーネントである．あくまで出力のみ．

CMC：　2200 システム・コンソール上のコンポーネントで，Host と SPO の双方向のコンソール・メッセージのデータフロー(raw message)の制御をする．また，AMS イベントも SPO に送る．

図 5　2200 システム・コンソールと U6000 との接続図

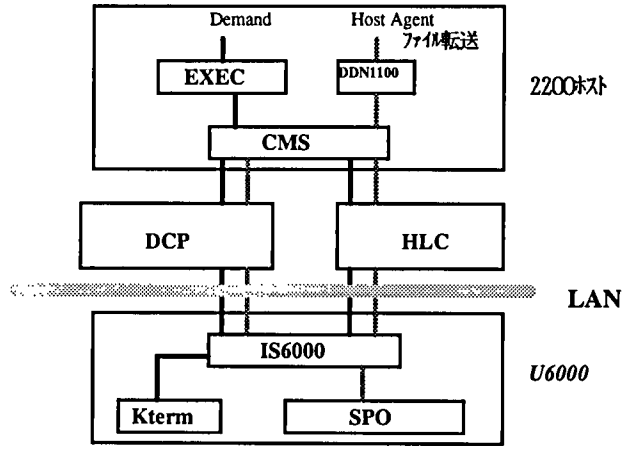


図 6　U6000 と 2200 ホスト・インタフェースとの接続図

また，U 6000 の 2200 ホスト・インタフェースを図 6 に示す．これによって SPO のターミナル・ウィンドウおよびホスト・エージェント・ソフトウェアと SPO 間のファイル転送(SPIP)が支援される．

4.1.2 AMS(Autoaction Message System)データベース

AMS データベース(AMS ランタイム・システムを含む)は，2200 システムからのコンソール・メッセージ・トラフィックを監視し，あらかじめ AMS データベース上に登録してあるメッセージ・パターンと一致するメッセージがあれば，所定のアクションを実行するいわゆるメッセージ駆動型のデータベースである．

アクションには，この例のように 2200 システムに対するコマンド送付の他に，コンソール・ベルの鳴動，AMS データベースのアクティベイト/ディクティベイト，また外部アプリケーション・インタフェース(SPIP)を利用するランの起動による SPO への警告の送付や取り消し，SPO ログ・ファイルへのメッセージの書き込みなどがある．

AMS データベースの記述には独自の仕様の言語が使われる．以下はその簡単な例である．

例) メッセージに自動応答する．

```
DEFINE  "BOOTMSG" 10
   MESSAGE "SHOULD BOOTSTRAP TRY AGAIN TO CHANGE MSR (Y, N)"
   TYPE      READ-AND-REPLY PRIVILEGED-EXEC
   INSTANCE PRIMARY
   PRIORITY 128
   TOKEN     KEYWORD  2
   TOKEN     FIXED      1
   TOKEN     FIXED      3
   ACTION    ALL  RESPONSE "Y"
END
```

4.2 レベル 2: 多岐にわたるオブジェクトの統合管理

SPO レベル 2 R 3(94 年度下期)以降で提供される機能であり，特徴は以下の通りである．

① 異機種ホスト・システムの監視/操作/制御だけでなく，システムを構成するコンポーネントの監視をもターゲットに入れる．

② マネージャ/エージェント・モデルでのシステム管理

③ SPO 追加アプリケーション

シリーズ 2200 用の AMS データベースと同等の機能をもったデータベースが，シリーズ 2200 以外のホスト・システム（たとえば A シリーズや UNIX など）用に，SP-AMS データベースとして提供される．

ただしシリーズ 2200 の AMS データベースを除くこれらのデータベースは，SPO 装置上にインストールされる．

また，SP-AMS データベース(または，AMS データベース)には，イベント・レポートと呼ばれる新機能が追加され，この機能を利用していくつかの新たな SPO アプリケーションが提供される．

一つは SPO テープ・モニタ・アプリケーションで，テープの Load メッセージや SERVICE メッセージがあらかじめデータベースに登録されており，該当するメッセージを検出したら，SP-AMS データベース(または AMS データベース)は SPO テープ・モニタ・アプリケーションに対してイベント・レポートを送付する．

このイベント・レポートは，ある決まったフォーマットの文字列で，SPO に対してファイル転送される．

さらに同じイベント・レポートを利用したアプリケーションにステータス・アプリケーションがある．これは，システムを構成する各コンポーネント(ハードウェア，ソフトウェア)の状況を表すコンソール・メッセージ(たとえば，ABC ABORT FIN など)をデータベースに登録しておき，該当する状況が発生したら同様にステータス・アプリケーションにイベント・レポートを送付する．

このように SP-AMS データベース(または AMS データベース)は，今後ともインフォメーション・エージェントとしての役割を果たすだろう．

ステータス・アプリケーションやテープ・モニタ・アプリケーションなどは，SPO Addin Application(追加アプリケーション)と呼ばれる．

SPO の追加アプリケーションを作成するということは，このデータ・モデルのサブセット(Local DM)を中核とした新規アプリケーションを作成することである．それを組み込んだ概念図を図 7 に示す．(図 7 の中の数字は，各ウィンドウへの出力パスを示しており，本文とはとくに関係ない．)

また，このレベルで提供される管理マネージャとエージェント間のプラットフォームは，レベル 1 と同様に SNAP である．

4.3 レベル 3：汎用性のあるシステム運用管理ツールとしての SPO の実現[6]

4.3.1 CORBA について

2 章で述べたように，汎用性のある分散システム環境の管理メカニズムを OO 技術をうまく使って実現するためには，世界的な規模での OO 技術の標準規約が必要になってくる．OO 技術をキー・テクノロジとして，異機種分散環境における AP 間の協調動作(相互運用性)を実現しようとする仕掛けは一般的に ORB(Object Request

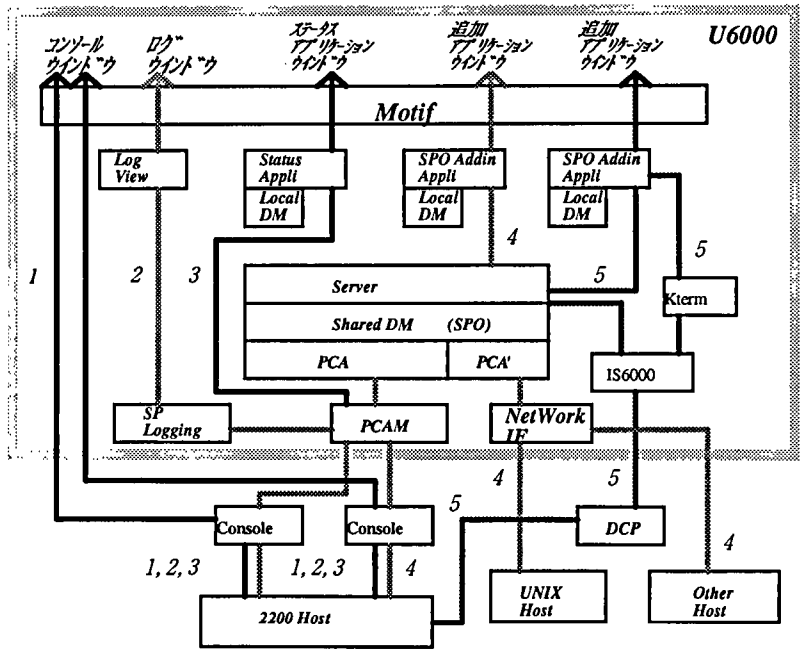


図 7 レベル 2：SPO のシステム構造概念図

Broker)*と言われ，これまで統一された規約はなく，各社独自で仕様を決め，製品化していた．前述したSNAPもこの中の一つと考えることができる．

しかし，その中でもOMG(Object Management Group)**が1992年に提唱したCORBA(Common Object Request Broker Architecture)は，業界標準規約になるだろうと言われている．

4.3.2 CORBAの仕組み(概略)

CORBAの通信メカニズムの基本はRPC(リモート・プロシジャ・コール)である．RPCは別ホストへのプロシジャ・コールであり，ローカル・コールと大きく違う点はマーシャリングを必要とすることである．マーシャリングとは，呼び出しや応答の際の引数や結果を取り出し，ネットワークの通信用にデータ変換(パック/アンパック)し，到着後相手マシン環境で処理できるデータ形式に戻すことをいう．

そのためには，クライアント側，サーバ側との引数や結果に関する参照名や値を対応付ける『手続きの定義情報』が必要になる．また，通信相手がどのマシンのどこにいるかという情報も必要である．

分散オブジェクト環境では，当然マルチ・クライアント，マルチ・サーバの形態になると思われ，これらの情報は，非常に複雑になってしまう．

CORBAでは，このRPCライブラリ(OSIでいうところのプレゼンテーション・レイア)の部分にオブジェクト指向のアプローチを使用する．RPCの概略図を図8に示す．

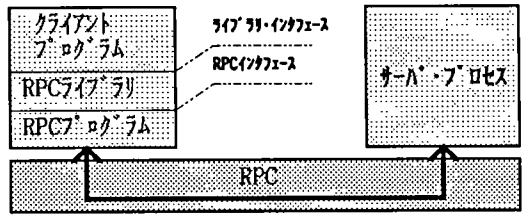


図8 RPCの概略図

CORBAでは，RPCに必要な情報ばかりでなく，クライアント，サーバAPが扱うすべてのもの(データ，ファイル，ハードウェアなど)をオブジェクトとして定義している．

それらのオブジェクトは，IDL(Interface Definition Language)と言われるC++の仕様に近いCORBA独自のオブジェクト指向言語によって記述され，インタフェース・リポジトリと言われるデータベースに格納され，ORBから動的に参照される．

IDLという独自言語を採用しているのは，ORBがクライアントやサーバAPを記述している言語から独立するためで，この独自言語とクライアントやサーバAPの言語とのマッピングはIDL側で提供される．(言語別のIDLコンパイラが用意され，IDLで定義したオブジェクト情報は，目的の言語にマッピングされる)

---

* ORBについては，4.3.2項を参照のこと．

** OMGはOO技術の標準化と普及を目的に1989年に設立された国際的非営利団体である．

また，CORBA の大きな特徴は，既存の非 CORBA アプリケーションや非 OO アプリケーションに対しても ORB へのインタフェースが提供される点である．これは，上記インタフェースを利用している．

オブジェクトを扱うことによるメリットは，前述した物の複雑さを外から隠すことだけでなく，オブジェクト指向技術の持つ多くの利点(たとえば，モデリングの容易さ，継承，カプセル化，多態性など)をそのまま享受し，新規オブジェクトの定義やメンテナンスの容易さに結びつけられる点である．

実際のオブジェクトは，クラス定義で構造化され，属性が付加され，そのオペレーションはクラス内にメソッドとして記述される．

クライアント(オブジェクト)が ORB を経由して相手のオブジェクトにリクエストを出す場合，ORB に対するインタフェース(API に相当する)が定められており，以下のようなパラメタが要求される．

- 相手オブジェクトを特定するためのオブジェクト・リファレンスという識別子
- オペレーションの種類
- オペレーションへのパラメタリスト
- リクエストが実行される環境に関する情報を収めたコンテキスト・オブジェクト

これらのリクエストは，メッセージとして ORB に渡され，目的のオブジェクトに送付される．

ただし，ここで重要なポイントは，AP(クライアント)から見ると，このリクエストは，あくまでファンクション・コールにすぎない点である．ORB インタフェースの概念図を図 9 に示す．

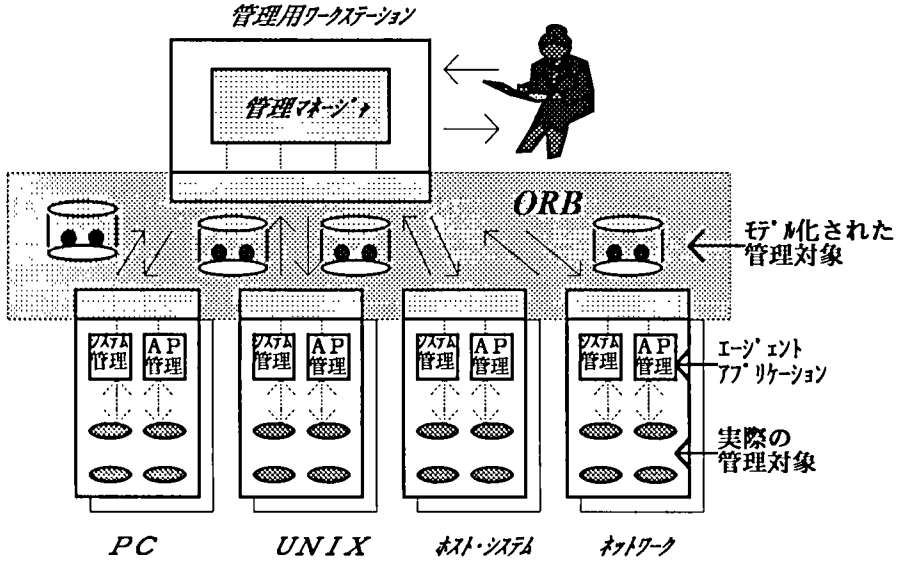


図 9　ORB インタフェースの概念図

4.3.3 ASMF と SPO

ASMF の最大の特徴は,『オブジェクト指向管理基盤』を支援することである. この管理基盤は, CORBA に準拠しており, DSmgr/フレームワークというプロダクトによってオブジェクト・フレームワークという名称で提供される.

そのため, このレベル 3 の SPO は, 正式にオブジェクト・フレームワークを支援した分散システム環境でのシステム運用管理ツールとして提供され, 分散されたマルチ・ベンダ・ホストのシステムを統合して管理すると思われる.

レベル 3 での SPO のシステム構造概念図を図 10 に示す. (図 10 の中の数字は各ウィンドウへの出力パスを示しており, 本文とはとくに関係ない)

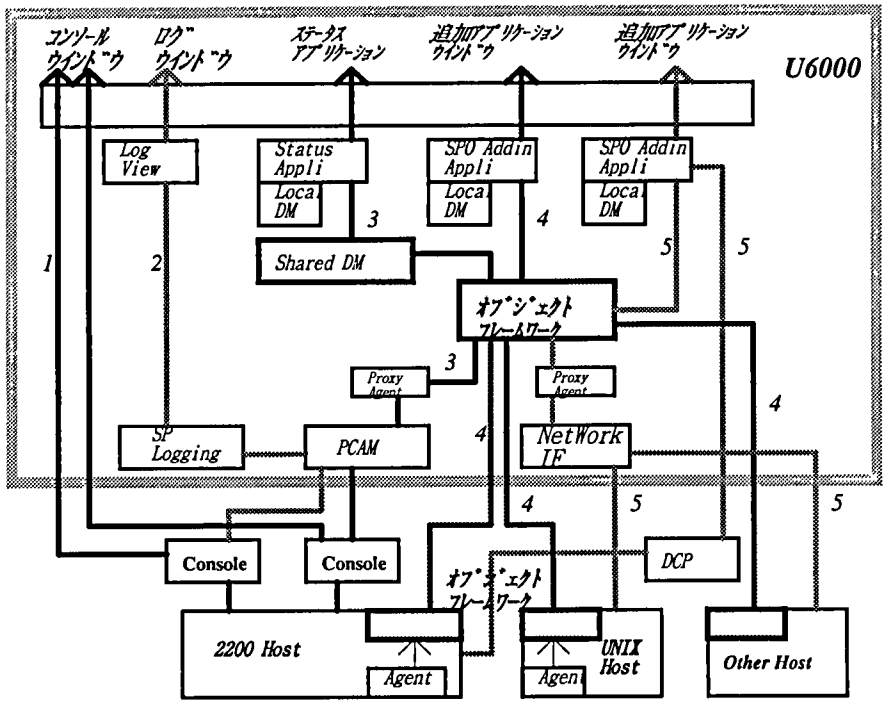


図 10 レベル 3: SPO のシステム構造概念図

ここで注目すべきポイントは, SPO(Shared DM や Local DM) は, CORBA 準拠のオブジェクト・フレームワークと直接インタフェースを持っている点と, 既存のプログラムである Network IF や PCAM などは, Proxy Agent*を仲介してインタフェースを持っている点である.

5. おわりに

異機種分散システム環境における汎用性のあるシステム運用管理ツールを目指す SPO が, 今後どのように発展していくのかを中心に紹介してきた.

本稿の中でも紹介したが, SMS フレームワーク(当社では ASMF)の中核は『オブジ

* Proxy Agent は, 非 ORB アプリケーションに対して ORB のインタフェースを提供するための特別なエージェントである.

ェクト指向管理基盤』であり，CORBAの規約に基づいてインプリメントされる．

1994年にOMGは，CORBA 2.0を発表する．これによりORBの標準化は本格化し，95年以降これに準拠した製品が各社から発表されると思われる．SPOを含めこれらの動向に注目していくつもりである．

参考文献 [1] Ian. R. Hepburn, "Single Point of Control Functional Specification", Unisys, Roseville, March, 12, 1990.
[2] "SNAP Concepts Document", Software Architecture & Engineering, Inc.
[3] James. E. Walster, "Operations Automation Using Single Point AMS and Smart Console", Unisys, Roseville, 1993 Fall USE Conference.
[4] James. Rumbaugh, Michael. Blaha,.., "Object-Oriented Modeling and Design", 1991, Prentice Hall, Inc.
[5] "オブジェクト・レポート"(The Newsletter of Object Oriented Technology and Market), ㈱創研プラニング, ㈳日本オフィス・オートメーション協会/オープン・システム環境研究委員会, 5/15/92, 10/15/92, 1/15/93, 9/15/93.
[6] "共通オブジェクトリクエスト・ブローカ～構造と仕様", Object Management Group, ㈳日本オフィス・オートメーション協会.

執筆者紹介 入 貝 健 介 (Kensuke Irigai)
昭和55年同志社大学工学部機械工学科卒業．同年日本ユニシス㈱入社．2200 OSの開発/保守，日本語関連アプリケーション開発/保守を経て平成2年よりUnisys製システム運用管理関連のアプリケーションの開発/受け入れ/評価に従事．現在システム・プロダクト本部サーバソフトウェア開発部に所属．

Aシリーズ統合運用管理システムによる運用の自動化

Automated Operation by A Series Integrated Operating Management Systems

山口和男, 丸岡茂敏

要 約 システム運用には,「システムの初期設定と終了処理」,「操作員による操作」,「業務システムの稼働」の局面がある. 本稿ではAシリーズにおける運用の自動化のために, 自動運転支援システムSCJ II(System Control for Japan II), システム操作支援ソフトウェアASSISTANTおよび統合運用システムIOF(Integrated Operating Facility)に関する仕組みとその適用について紹介する. オペレーティングシステムMCP(Master Control Program)の技術を利用したこれらのソフトウェアの提供によって, 運用の自動化を可能にする.

運用の自動化対応として, SCJ IIによるシステムの運転開始/運転終了とオペレーティングシステムとの連動, 操作員の定常作業と監視のために実行運用環境の設定や事象の採取・制御および作業の流れをASSISTANT言語で記述する. その運用環境の基でIOFデータ登録によって業務ジョブの運行日時・条件などを計画的にスケジュールし, 単一視点でジョブの実行起動, 稼働監視, 実績管理をする. また, IOFは運行時に必要な資源(ディスク, テープ)の使用条件を制御し, システムの障害の監視や障害に対する回復支援, 外部通報を行う.

Abstract In systems operation, there are three phases: "initial system setup and shutdown," "operation by system operators" and "the running of applications." This paper is intended to discuss the design framework and implementation of the SCJ II (System Control for Japan II) automatic control system, ASSISTANT (as system operation assistant software) and the IOF (Integrated Operating Facility) tool that all help support automated systems operation for A Series computers. Those software items into which technologies of the MCP (Master Control Program) operating system are incorporated bring the enhanced automation of systems operation into existence.

For automated operation, system power-on/off is taken care of by SCJ II, and descriptions in the ASSISTANT language make possible linkage with the operating system, the setting-up of job/task processing environments, and the collection and control of various events as well as work flows required by operators for their daily routine. In ASSISTANT-created operating environments, users can define processing schedules/conditions, start/monitor the jobs and obtain the results from a single point more effectively by using IOF-registered data and IOF's job management functions. In addition to controlling/monitoring the usage of resources (disks and tapes) required for job processing, IOF also monitors systems failures, provides recovery support and issues warnings if jobs being processed are detected in an abnormal state.

1. はじめに

Aシリーズにおけるシステム運用の自動化は, ジョブの実行環境と実行手順を定義するワークフロー言語WFL(Work Flow Language)と複数ジョブの実行に伴うシステム負荷の制御機能などを中心としたワークフロー管理システムWFMS(Work Flow Management System)に始まった. 1970年に提供されたこのシステムの狙い

は，業務遂行に必要なジョブやタスク，それらの扱うファイルというオペレーティングシステムの制御対象を単位に，その動作をスケジュール化させ，スケジュール通りに実行させることにあった．また，1985年にコンピュータシステムのスケジュールによる自動電源オン/オフ機能が提供された．しかしながら，操作員作業の自動化や運用管理から見たジョブスケジュールの自動化，実績履歴管理の自動化など業務システムの運用そのものの自動化に課題を残してきた．

システム運用を，1)電源装置や外部通報装置などと連動したシステムの運転開始/運転終了，外部通報といったシステムの運転管理での運用の局面，2)予定されたあるいは偶発的に発生するシステム事象に対する操作員対応の局面，3)ワークフロー管理システムが対象としているジョブの実行の局面，の3局面に分ける．それぞれの局面において，自動運転支援システムSCJ II，操作記述言語を提供するシステム操作支援ソフトウェアASSISTANT，ワークフロー管理システムを強化した統合運用システムIOFが運用の自動化を支援する．

本稿は，システム運用の各局面での自動化に焦点を当て，その現状とそれを支える技術や仕組みを紹介する．また，これらのソフトウェアの導入先のシステム運用モデルにそって適用上の課題を明らかにし，それに対する自動化運用の解決策を検討する．最後に，今後の自動化運用を取りまく環境の背景と要求について簡単に述べる．

2. システム運用の範囲

本章では，従来多くの部分を人手に頼りながら実施しているシステム運用において，どのような局面に自動化の導入が可能であるかを見極めるために，システム運用の項目を整理し，対象としているシステム運用の範囲を明確にする．

2.1 システム運用の各局面

システム運用は，コンピュータシステムへの電源投入から始まり，システムの初期設定，業務システムの稼働，システムの終了，電源切断という過程を踏む．

業務システムの稼働に関しては，コンピュータシステム内部での稼働という分野だけでなく，日・週・月・半期・年という単位で運行業務を計画したり，稼働実績に基づいて予実管理する業務も含まれる．また，システムの初期設定から終了処理の過程では，繰り返し可能な形で手順化できる作業ばかりでなく，状況に応じて操作員が判断しながらコンピュータシステムに指示する作業もある．

したがって，システム運用は，業務システムの運行を計画し，コンピュータシステムの資源を使いながら実行し，その結果を評価するという「業務システム稼働」の局面，業務システムの実行運用環境の「初期設定や終了処理」の局面，さらに，何らかの業務的な要件やシステム事象の発生に伴う「操作員による操作」の局面に分けることができる．

2.2 システムの初期設定と終了処理

この局面は，業務システムが稼働する実行運用環境の設定と終了を扱う．環境の設定と終了にはコンピュータシステムへの電源投入と電源切断も含む．コンピュータシステムそのものを制御の対象としていることから，業務システムの全面停止となる事態の外部通報機能も備えられている必要があり，したがってその機能を用いた運用も

対象になる.

2.3 操作員による操作

コンピュータシステムが稼働し，業務システムが稼働すると，操作員による指示・応答が必要になる場合がある.一つは,オペレーティングシステムMCPを介して通知されるRSVP*(操作員応答待ち)メッセージへの応答である．このメッセージは，主にプログラムからの操作員入力待ち，資源の容量不足や障害により業務処理の続行ができなくなった時，操作員に指示を求めて出力される．第二は，システム内で発生する事象を監視しながら操作員が指示する場合である．事前に規定できる業務処理固有の事象に関しては以前からワークフロー言語WFLを用いて自動化されているが，システムレベルの事象に関しては操作員に頼る場合がまだ多い．第三は，業務上の要件からシステム構成やネットワーク構成を変えたり，ファイルの配置を変える指示である.

操作員による操作という局面では，業務システム向けの実行運用環境の設定・終了や業務システムの稼働中におけるシステムレベルの監視と対応が操作の対象となる．それらの操作に関しては，システムでどういう事態が発生したかを検出・確認し，それに基づいて指示し，指示の結果を確認する，という一連の動作を記述できる操作員の操作記述言語が威力を発揮する.

2.4 業務システムの稼働

業務システム稼働の局面は，ジョブレベルの運用管理と，ジョブが稼働するシステムレベルの運用管理の組み合わせとして捉える.

ジョブレベルの運用管理では，ジョブの運行計画，すなわち，業務処理を行うジョブのネットワークを定義し各ジョブの起動条件を設定することから始まり，そのジョブの扱う入出力データの管理，実行操作，監視，実績管理が運用項目となる．ワークフロー言語WFLは,ジョブの実行開始から終了までを自動化する.ジョブ固有の局所的な障害への対応では，WFLで記述し自動化することは一般的に行われていることであり，また，このレベルの障害対応を標準化して専用のジョブを設け，IOFのジョブ管理サブシステムに運用を委ねることができる.

システムレベルの運用管理には，ファイルの保全やディスク容量監視を行うリソース管理，システム資源の利用状況を監視しながら実行効率低下の要因を抑止したり，稼働実績の蓄積情報に基づくキャパシティ計画を行う性能・パフォーマンス管理，システム障害を監視し早期発見と迅速な対応を行う障害管理，ソフトウェアが正しいバージョンのもので確実に実行されていることを保証するリリース管理がある.

3. 運用の自動化ツールの仕組み

運用の三つの局面での運用の自動化ツールの検討は，すでに提供されている運用ツールとそれを補完する自動化ツールの適用範囲を明らかにし，機能分担することから開始した.

自動化ツールには，運用担当者が該当の業務処理を計画・実行し，都度の処理要求に対して判断・対処するという運用作業の特性と振る舞いから，二通りの方式が求め

* RSVPは，Reply, if you pleaseの意味である.

られる．運用作業をある条件の基で計画的にスケジュールでき処理する場合には，それらに必要な入力データと作業の内容や順序を対応付けしながら簡単な操作で定義できるデータ中心型の処理登録方式を適用する．また，その都度に発生する操作員への指示・応答の動作を支援するには事象駆動型，または時間制御型の操作記述方式が有効である．

以下に各自動化ツールの仕組みやそれを支える技術を紹介する．

3.1 自動運転支援システム（SCJ II）

SCJ IIは，ハードウェアとしてのSCJ II装置とホスト上で稼働する支援ソフトウェアから成り，運用スケジュールに従ってメインフレーム，周辺機器および空調装置の電源を制御する基本機能を含め，以下の運用局面を支援する機能を兼ね備えている．

1) 電源のオン/オフ機能……SCJ II装置は，カレンダ・時間でのスケジュール登録，操作パネル上のプッシュボタン，遠隔端末などの操作によってコンピュータシステムの電源投入・切断を制御する．またホスト内のSCJ II支援ソフトウェアからの指示によって電源切断ができる．

2) ソフトウェアの起動・終了……コンピュータシステムの立ち上がり直後に，ホスト内のSCJ II支援ソフトウェアのシステム初期化用プログラム(SCJ II/INITIATOR) は，SCJ II装置とのハートビートによる状態監視を開始し，システム実行運用環境の設定動作を自動的に実行するシステム操作支援ソフトウェアASSISTANT を起動する．

この初期化用プログラムはオペレーティングシステムMCPが立ち上がった後，M CPから自動起動されるスーパバイザプログラムであり，ホスト内の他のシステムとはポートファイルによるメッセージインタフェースを提供している．

終了処理では，ホストのシステム終了から電源切断までの一連の流れを実行させるために，システム停止用プログラム(SCJ II/KILLER)は，動作パラメタ設定に従ってシステム操作支援ソフトウェアASSISTANT を終了させた後に，SCJ II装置本体に対するコンピュータシステムの電源切断を指示する．

3) 外部通報……SCJ II装置には接続された外部通報装置に対して信号を送り出す機能がある．これを利用してシステム自動運行中の障害発生を通報できる．通報対象は地震・火災の事実，ホスト停止，およびホスト側のハードウェア・ソフトウェア障害などである．

図1にその概念を示す．

3.2 システム操作支援ソフトウェア（SYSTEM/ASSISTANT）

ASSISTANT は，システム操作向けの操作記述言語を提供する事象駆動型・時間制御型の支援ユーティリティである．このソフトウェアは，「何が」起こった場合には「どうする」という形式の事象別のフロー定義を事前に設定することで，その事象の発生の都度に作動する．たとえば，定められた時刻，オペレーティングシステムの起動やデータ通信の初期化の完了，プログラムの起動や終了，出力メッセージなど，操作員が注意すべき局面を的確にとらえ，操作員指令入力やワークフロージョブ起動をプログラム的に実行することができる．そのシステム操作記述言語ASSISTANT による記述例を図2に示す．

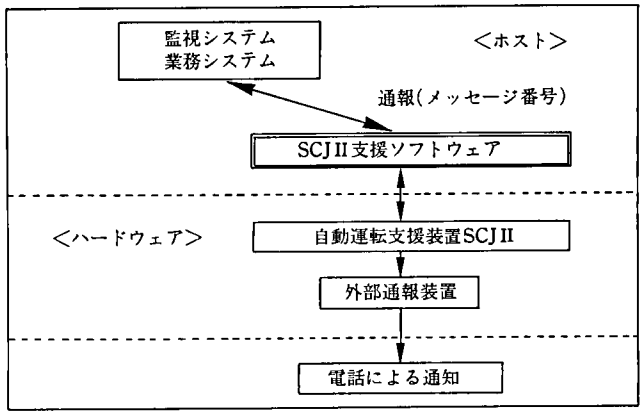


図 1　SCJ IIによる外部通報

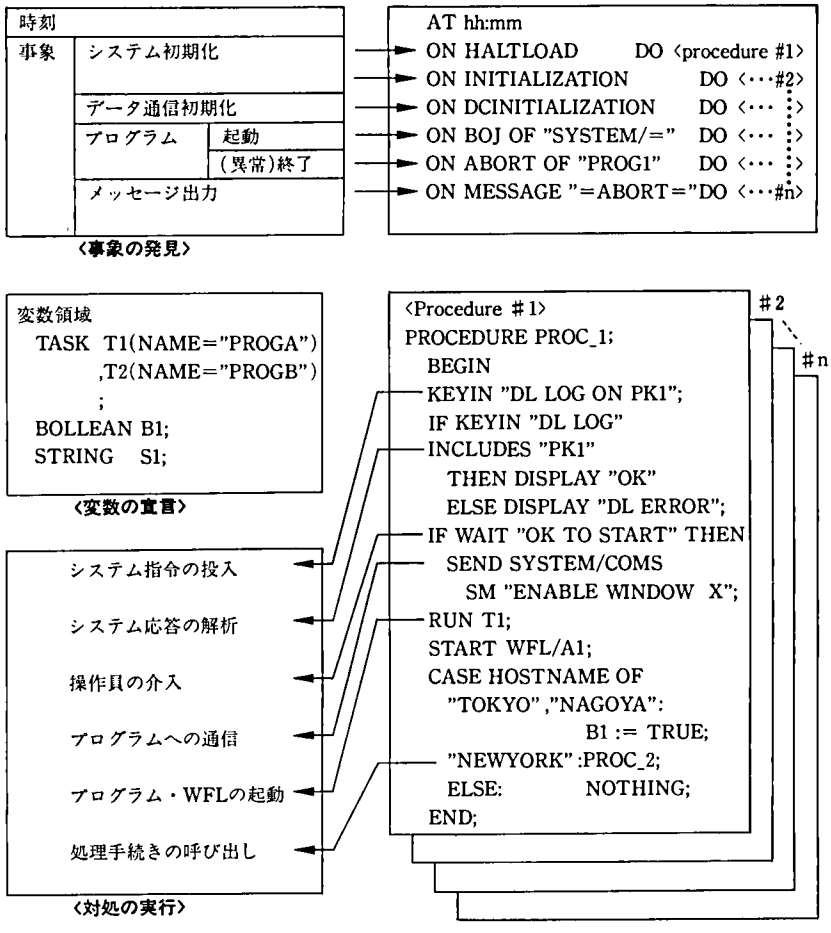


図 2　システム操作記述言語 ASSISTANT の記述例

ハードウェアの自動電源オン/オフ機能に伴って反復されるべき操作員の操作を自動化するために，この言語の適用は非常に有効である．ASSISTANT は事象検知の機能を基本に設計されており，自ら発行した指令やジョブの結果を確認できる．これは迅速性および確実性において，操作員の手作業や従来ながらの MCP の操作指令入力関数 DCKEYIN を使用した簡易プログラムに勝っている．

このシステム操作記述言語 ASSISTANT は意図する処理(操作)を構造化して記述する上で ALGOL に似た手続き型言語を採用しており，処理相互間の独立性を高めている．また，動作定義（環境変数，状態検知，動作指示など）は，操作員の感覚で表現できるレベルで具体化されており，記述が容易である．システム事象を検出する管理モジュールの部分(ON・・・文)では，特定のプログラムや出力メッセージに関して，名前や文字列をワイルドカード方式で動的な識別・検出ができ，操作員に代わって指示できる．

この操作記述言語の適用により，オペレーティングシステムの動作パラメータの設定，オンライン・トランザクション処理を含め業務システムの稼働する実行運用環境の設定など，指示すべき動作が決められているものを，逐次的あるいは並列的に記述させて動作を自動化することができる．

業務システム運用での監視・対処では，定型化した業務の実行が主であり，種々のシステム資源への要求も定常的で比較的小規模なシステムでは，都度に発生する事象に対してこの操作記述言語は適用できる．しかしながら，複数の業務システムが稼働し，適時に多くのシステム資源と操作員の判断を業務的な関連から求められる中規模以上のシステムでは，この操作記述言語で事前に稼働システムのすべての動作要件を定義したり，稼働環境の変化に対しても再定義・保守することは容易でない．したがって，業務処理のスケジュールや実行の条件を設定したり，システム障害や業務処理の異常の監視とその回復処理を一定の規則で対処し，都度に業務処理の進捗状況を把握するには，データ中心型の処理登録方式に基づく統合運用システム (IOF) の適用を勧める．

3.3 A シリーズ統合運用システム (IOF)

IOF は，業務システム稼働とシステムレベルの運用管理の局面を支援すべく設計されている．IOF は図 3 に示すサブシステムから構成されており，各サブシステムは共通基盤サービスを核として連携し，個別の導入・適用が可能である．

各サブシステムは，運用の処理内容に合わせ機能を分担し，運用管理の目的別に独立している．共通基盤サービスは，DMS II (Data Management System II)統合データベース機構による運用情報の一元管理，ライブラリ機構による共有手続きの提供，メールテキスト交換のための待ち行列によるインタフェースの提供，外部通信メッセージサーバのための MCS*(Message Control System:メッセージ制御システム）機能の提供を行う．これらオペレーティングシステム MCP の技術を利用した共通基盤サービスにより，各サブシステム相互間の処理の独立性を高めている．また，指示・統制の共通化のためにグラフィカル・ユーザ・インタフェース GUI を利用し，単一視

* メッセージ制御システム MCS は端末やプログラム間のメッセージの経路づけ，通信を行うソフトウェアで，リアルタイム型，対話型，診断型の MCS などがある．

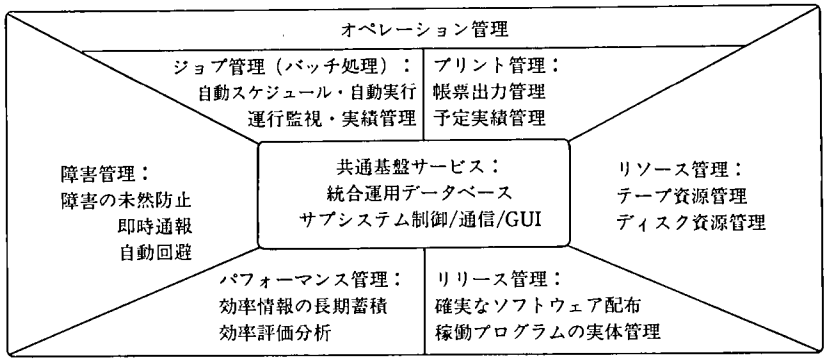


図 3　統合運用システム（IOF）のサブシステム体系

点での操作性向上を図っている．

3.3.1　ジョブ管理サブシステム

ジョブ管理サブシステムは，業務システムの稼働を遂行するジョブの計画から実行・評価までの範囲を自動化する．運用資産である従来のワークフロージョブを活用し，データ登録を通して処理のスケジュールや実行に関する属性を定義することにより，長期間を対象としたジョブの自動実行を可能にする．

ジョブの実行実績はデータベース上に一元管理され，画面照会および帳票の出力を可能にする．また，連続的にあるいは並列的に実行させるジョブを組み合わせ，作業を意味づける一連の処理の固まりをジョブグループとして定義することにより，業務処理の作業単位での実績把握を容易にする．

ジョブに対する処理の運行状況は逐次監視端末に伝送され，処理の進捗を運用担当者に伝達する．処理に異常が発生した場合には，監視端末に異常通知がブザー音と共に報告され，端末を通じてその対応処置が可能である．また，異常発生は自動運転支援システム（SCJ II）を通じ外部に通報することができる．

このサブシステムの対象は業務システムが抱えるバッチ処理であり，それらの様々な実行形態を自動化の対象とする．日次・週次・月次などの異なった周期のジョブを組み合わせて順序正しく処理する作業を機械的に実行する．また，起動するための条件に時刻指定・先行後続の関連付け・ファイル存在・操作員の指示・プログラム間通信での指示などを定義でき，各々のジョブに関する運用担当者向け管理表や日別・月別の運行スケジュール表を出力する(図4参照)．ジョブを構成するタスクの実行環境や実行手順に関する動作定義は，ワークフロー言語 WFL によって記述できる．ジョブ管理サブシステムでの最小の管理単位は，ジョブである．

運用管理者による，データ登録方式によってジョブの流れを自動制御し管理する仕組みを実現するために，オペレーティングシステム MCP の技術を利用している．該当ジョブはスケジュールに従ってジョブ開始関数 CONTROLCARD によって起動される．ワークフロー管理システム WFMS が一括管理しているジョブ・タスクの状態やメッセージを MCS 間メッセージインタフェース機構の INTERCOMQUEUE 待ち行列から受理し，その内容の判断で実行管理を行う．また，メッセージの内容が停止

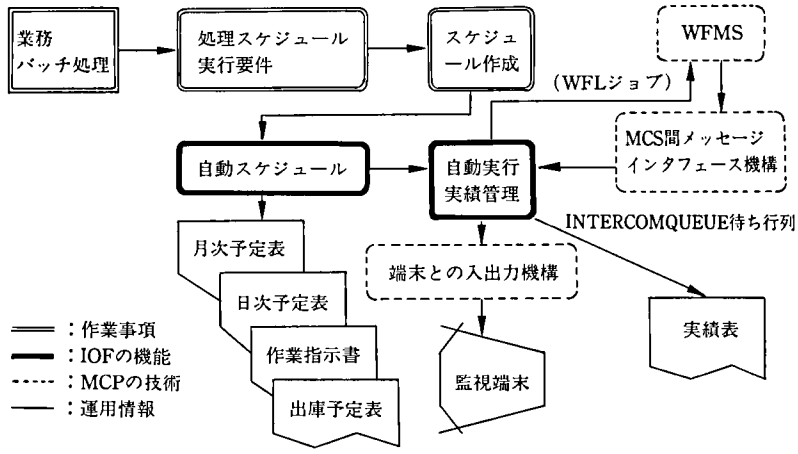


図 4　ジョブ管理サブシステム処理の流れ

(WAITING) や中断 (ABORT) の場合は，異常として監視端末に通知する．運用情報を必要時に登録・更新・検索できる様に DMS II統合データベースを用いている．

3.3.2　プリント管理サブシステム

プリント管理サブシステムは，帳票の計画的な出力，指示・操作の支援，帳票出力の予実績管理を支援する．かつ，MCP のプリントシステム (PrintS/ReprintS) を制御機構としてそのまま用い，それを補完するソフトウェアとして位置づけた．MCP のプリントシステムは，ジョブの終了やタスクの終了，あるいはプリンタファイルの閉塞などを契機にセンタプリンタやネットワーク上の遠隔プリンタに自動出力する機構を持つが，どのプリンタにどのように出力するかは，WFL レベルであらかじめ定義しておく必要があり，それを変更する場合には，プリントシステムへの指示命令を必要とする．

補完した主たる機能は，遠隔プリンタへの出力で，画面メニュー方式やプログラムからの操作支援ライブラリの呼び出しによって，利用者が簡便な出力指示を可能にすること，業務プログラムが作成したプリンタバックアップファィルのプリント出力時に，プリンタ装置のハードウェア属性（主に日本語対応）に合うように書式変換する TRANSFORM ライブラリの提供，出力制御のための動作パラメタのデータ登録方式の提供，帳票出力の実績管理機能の提供などである．

DMS II統合データベースによってプリントデータの情報管理をしているので，業務システムの利用者は操作インタフェース機能を利用して，帳票出力の実績を照会したり，予定帳票に対する作業の経過を管理したり，あるいは帳票出力の際の障害に対する回復処置に対処できる．また，帳票グループを設定することで，ジョブの実行順序と無関係に帳票出力の単位や順序を設定でき，大量の帳票出力に伴う仕分け作業を容易にしている．

3.3.3　障害管理サブシステム

このサブシステムは，システムが発する事象からシステム全体の運用や業務運用の続行に影響を与えそうな障害要因を抽出し，該当プログラムの停止，指定ジョブの起

動，強制的なスケジュール待ちなどの回避処置を自動的に実行し，障害の未然防止および拡大防止を支援する．

障害管理サブシステムでは，障害をホスト障害，データベース障害，データ通信障害，ネットワーク障害に分類し監視する．ホスト障害での監視項目には，プロセッサとメモリのハードウェアエラー，ディスクやテープなどのチャネル・周辺装置エラー，プロセッサとメモリの利用状況，プロセッサとメモリの過多使用プログラム，システム資源ファイル配置のディスクファミリの残容量，指定プログラムの中断などがある．データベース障害での監視項目には，データベースのバッファ使用のための許容メモリ量，データベース制御プログラムの中断，データベースのサービスの停止，指定データセットのファイル容量などがある．データ通信障害での監視項目には，通信管理システム COMS (COmmunication Management System) の未処理メッセージのための許容メモリ量，COMS の中断，COMS のサービスの停止，トランザクション待ち行列の深さなどがある．ネットワーク障害での監視項目には，CP 2000 (Communication Processor 2000) 通信装置のエラー，ホスト通信インタフェースのエラー，回線エラー，端末エラーなどがある．

障害管理に求められる障害監視・検知・回復，記録・報告などの処理の流れ，利用者の作業事項，MCP の技術の関連を図 5 に示す．

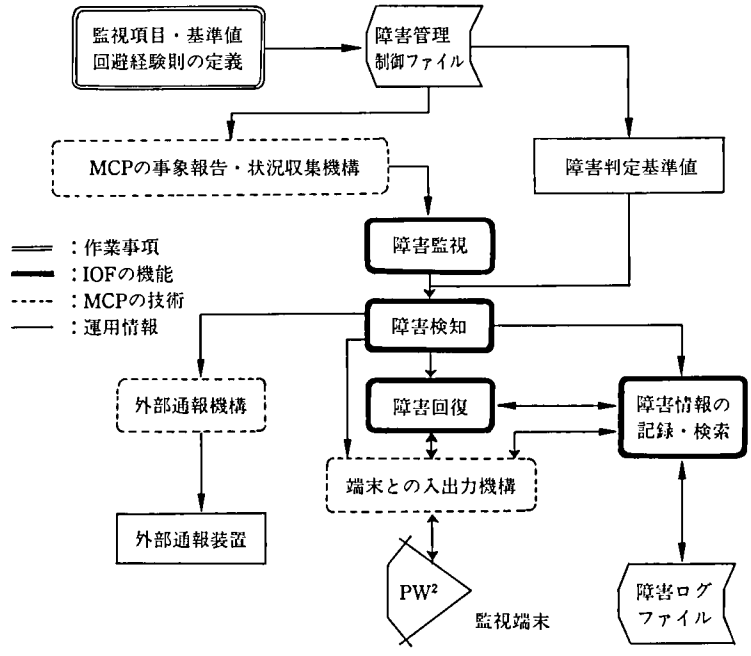


図 5　障害管理サブシステム処理の流れ

障害監視・判定の仕組みは，オペレーティングシステム MCP の事象・状況収集機構と IOF の障害処理機構とに分けられる．システム内で発生した事象は，システムログにすべて記録されるが，その記録されるログ記述項のすべて，あるいは選択的に事象

の発生と同時にプログラム的にアクセスすることができる．これは，ログエントリ報告関数 REPORT_LOG_ENTRIES の MCP 手続きを呼び出すことで行う．このログ記述項は，システム内で発生した事象を大分類・小分類しており，それに基づいて障害発生の事象を分析し判定できる．また，メインフレーム機器や周辺機器の状態と利用状況を示す情報は，資源状態関数 GETSTATUS/システム状況関数 SYSTEM-STATUS の MCP 手続きを介して入手できるので，観測値の偏差をみながら状況を分析し判断する．

制御ファイルには，対象とする監視項目，警告判定の時間間隔やしきい値および回避方法を定義する．実際の運用では，障害検知とその報告機能だけを働かせ，運用管理者がそれを分析・評価する段階と，それに基づいて回復動作を定型化し，その動作を実行させる段階を繰り返し行うことになる．それらの段階を通じて障害対処への自動的な回復処置を図ることができる．また，障害データの情報管理としては，「障害の検知」，「回復動作の実行」，「障害状況からの回復」のデータを分類・記録しており，照会を可能にしている．

3.3.4 リソース管理サブシステム

このサブシステムは処理作業で使用するディスクファミリ容量，テープリール・ファイルに関するデータを一元管理する．利用者の作業事項，IOF の機能，オペレーティングシステム MCP の技術，運用情報の関連を図 6 に示す．

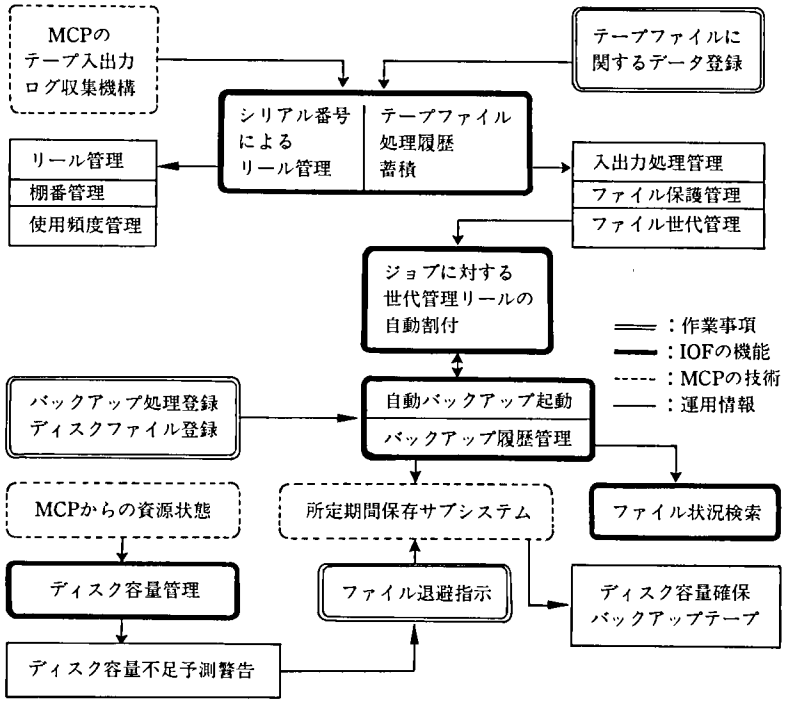


図 6　リソース管理サブシステムの機能関連付け

ディスクファミリの残容量管理としては，指定したディスクファミリの使用領域と未使用領域の容量を資源状態関数の MCP 手続きで一定の時間間隔で観測し，その現

在値および偏差を収集・管理する．また，しきい値との比較により容量不足を警告しMCPの所定期間保存サブシステム（アーカイブ機能）と連携し，最近使用されていないファイルを指示によってテープへ退避させ，使用可能な容量を確保する．

ディスクファイルの保全では，データ保全の実行を外部機構である所定期間保存サブシステムに委ね，バックアップテープへのファイルのコピー，ディスクへのファイルの復元，バックアップ記録の追跡などが提供されている．リソース管理は，所定期間保存サブシステムを補完するソフトウェアとして，GUI画面での操作の容易性や作業の履歴管理に重点を置き，該当ディスクファミリから指定したファイルグループごとのバックアップ採取を所定期間保存サブシステムに対して定期的に自動指示する．

テープリールの管理では，DMS II統合データベースに，ログエントリ報告関数のMCP手続きを呼び出して，テープに関する入出力操作のログ記述項から指定シリアル番号に基づく使用頻度(物理入出力の回数，エラー回数)，使用ファイル名を収集し，そのテープリールの棚番（所在，目的）と関連付けて一元的に管理する．テープファイルの入出力操作としては，収集したテープファイルの入出力ログ記述項に従ってテープファイル名，使用テープのシリアル番号，使用したタスクとジョブ名，入出力区分と操作日時などのデータを記録し，ファイルの世代と実績履歴を管理する．さらに，ワークフロー管理システムのジョブ開始関数による起動時に，そのワークフロージョブ内のファイル属性値SERIALNOに対して，ファイル世代によるシリアル番号を自動的に割り付けて，テープの論理ファイルと目的のテープ媒体との関係を保証している．また，業務ジョブに関する実行スケジュールの確定時には，テープファイルの使用予定データを作業指図書に出力する．

4. 自動化ツールの組み合わせによる自動運用の解決策

以上紹介した自動化ツールを組み合わせて，統合化することにより自動運用が実現できる．本章では，コンピュータシステムにおける日次周期の運用の各局面におけるシステム運用事例を紹介する．システム運用の自動化に必要な機器構成を図7に示す．

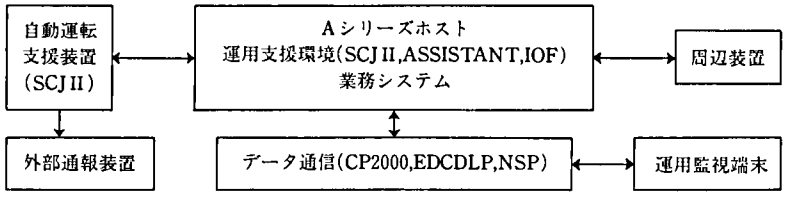


図7　自動運用の機器構成

4.1 システムの初期設定

Aシリーズシステムの電源投入から業務システムの処理開始に至るまでの一連の操作では，自動運転支援システム（SCJ II）およびシステム操作支援ソフトウェア（ASSISTANT）を適用して自動化する．システムの初期設定の流れを図8に示す．

電源投入はあらかじめ登録されたシステム立ち上げスケジュールに従い，SCJ II装置が実施する①．ホストシステムは起動直後にMCPのスーパバイザ・プログラムとして，システム初期化用プログラムのSCJ II/INITIATORを起動する．これを契機に

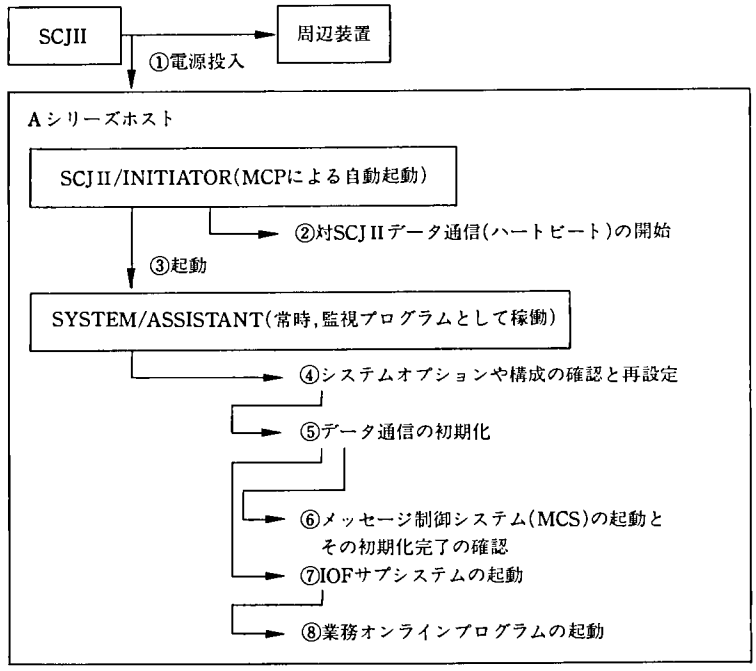


図 8　システムの初期設定

ホストシステムに接続されているSCJ IIとデータ通信（ハートビート）を開始し，ホストとSCJ II装置間のインタフェース経路を確保する②.

実行運用支援環境ソフトウェアの初期化のために，SCJ II／INITIATORからの利用者指定ジョブの起動によって監視プログラムとしてのASSISTANTに引き継がれる③. この機能は個別の利用者指定のジョブでも一部動作が代替可能ではあるが，動作結果の確実な保証，処理順序の確保，および操作記述の容易性などの観点からASSISTANTを適用する.

ASSISTANTは，システム操作記述の定義に従って，まずオペレーティングシステムMCPの動作オプションパラメタの設定やシステム構成の内容が前回の運用時と相違のないことを確認し，異なれば設定値をシステム指令の投入で修正する④. 次にデータ通信を全て起動し⑤，その初期化の完了を待って，実行環境であるオンライン・トランザクション処理支援環境のメッセージ制御システムとしてのCOMSや他のMCS, および運用支援環境のIOFを起動する⑥⑦. このMCSの初期化完了は配下のサブプログラムの開始によって判断し，COMSの配下で稼働する業務オンラインプログラムを起動する段階へ移行する⑧.

データ通信の確立後は，システム運用，操作に関する障害・異常（データ通信障害，ディスク装置障害，資源利用過負荷異常，プログラムループ異常，業務システム異常など）はSCJ IIを通じての外部通報が可能である. 実行運用支援環境の障害管理としてはASSISTANTが，業務システム稼働時の障害管理としては運用環境の設定に基づいたIOF障害管理サブシステムが，監視と通報の役割を受け持ちホストシステムの環境全体を監視し，業務システムの稼働環境の運用管理を支援する.

4.2 業務システムの稼働

業務システムを安全，的確に稼働させるために，実行運用環境での業務処理と運行監視の運用関連を図9に示す．

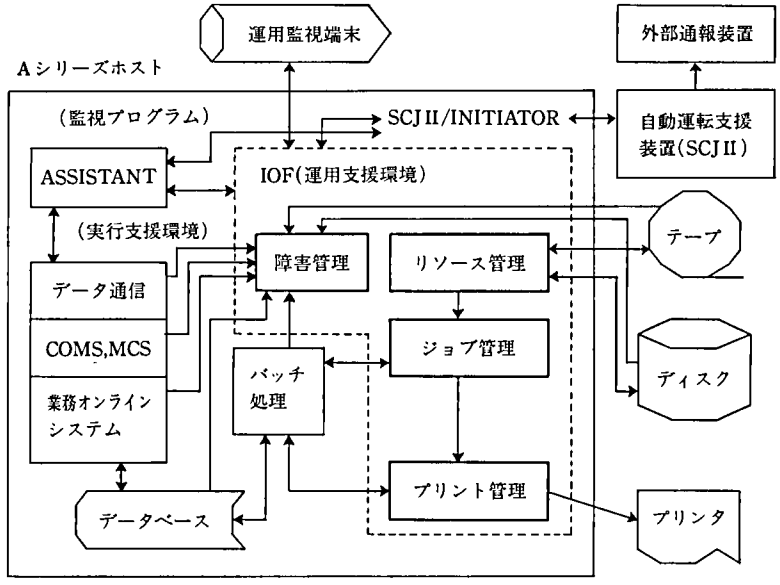


図 9　業務処理と運行監視の運用

IOF ジョブ管理サブシステムを利用して，業務システム運行の日程計画，業務バッチ処理の実行操作，監視，実績履歴管理を行う．このために，操作員の介入を排除しタスク制御と障害対応（タスク異常終了処理など）をバッチジョブ内で完結させる必要がある．業務処理として，事前にデータ登録済みの業務ジョブ間のネットワークフロー，日時設定・起動条件などの処理特性に従って，該当日のジョブ運行スケジュールは常にその実行経過と共に運用監視端末に表示される．また，個々のジョブは起動条件を満たすと自動的に実行する．このことは，処理漏れや起動指示の間違いなどの安全な運行を妨げる人的要因を除去でき，操作員の作業負荷の軽減に役立つ．

業務バッチ処理が作成する帳票ファイルの出力は，入出力装置の登録と帳票の定義に基づき，IOF プリント管理サブシステムを通じて処理される．運用端末からの検索により，当日作成予定の帳票とその現在の状況がわかる．操作員は，帳票グループ単位の一括出力を指示することにより出力後の仕分け作業を容易にし，従来の作業時間を軽減できる．出力時に用紙ジャムや位置ずれなどが発生した場合には，該当の帳票を検索して個別に再出力を操作・指示できる．

日中・夜間を通じて，IOF 障害管理サブシステムはシステムの動きを監視し，ハードウェアの異常，処理運用に影響のある状況(資源利用の過負荷，プログラムループ，データベース稼働異常，COMS 稼働異常など) を発見しだい，また，業務処理のバッチジョブで停止（操作待ち）および実行時エラーが発生すれば，それらを監視端末にブザー音と共に通知する．業務用ディスクファミリの空き容量が不足しそうな場合には IOF リソース管理サブシステムによって監視端末に警告する．操作員は，必要に応

じて自動的にテープへ退避させるべく，監視端末に対してデイスクファミリの選択と必要な容量を指示し，当該ディスクファミリの空き容量を確保する．

これらの運用は，障害の発生確認と関連作業に関する一定基準を設定することで，経験と勘に依らず運用者の個人差や作業負担を軽減し，業務処理のサービス継続を可能にする．

テープ装置を使用した入出力情報は，リソース管理サブシステムのデータベースに管理されており，テープ操作時のファイル世代を確定・制御する．テープ入出力を伴うバッチジョブは実行時点で必要なテープファイルの世代情報（シリアル番号）を受け取りテープリールの入出力操作を保証する．

夜間やセンタ運用が無人運転体制の場合には，運用管理者はIOF外部通報機能を設定することで，IOFにおける業務ジョブ運用の異常およびシステム障害は，電話回線を通じて外部の運用担当者に通知できる．

4.3 システムの終了処理

業務システムの処理終了を契機に，システムの実行運用支援環境を正常に終了させ，コンピュータシステムの自動終了（システム電源切断）までの流れを図10に示す．

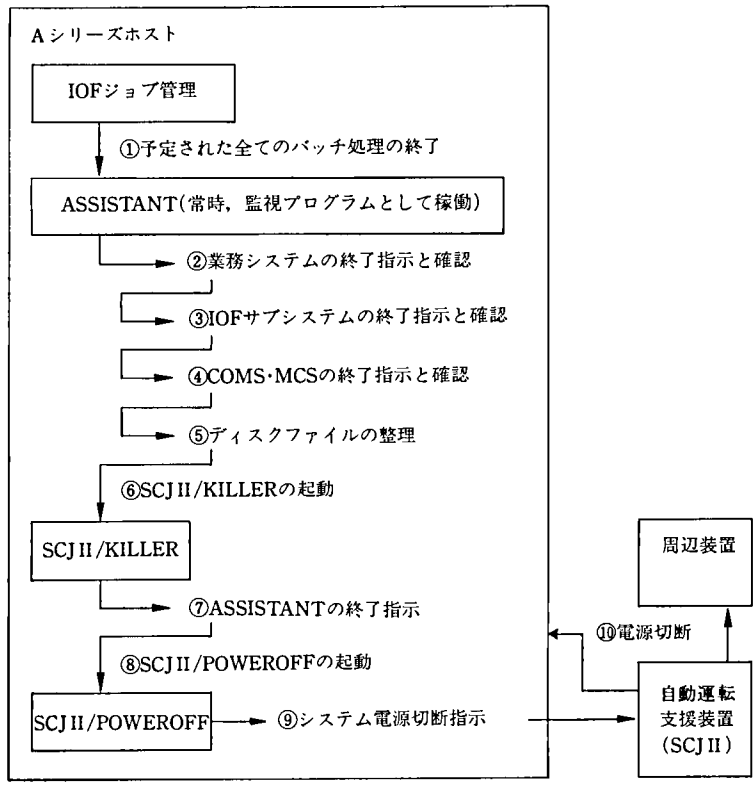


図 10　システムの終了処理

ジョブ管理サブシステムによる予定した全ての業務バッチジョブの終了の連絡を受け，ASSISTANTは，システム操作記述の定義に従って，終了判定用ジョブを起動してその判定結果によって業務終了と判断する①．その後，第一段階として継続実行中

の業務オンラインプログラムを終了させるために，実行環境であるオンライン・トランザクション処理支援環境のCOMSに対して指示する②．第二段階として業務オンラインシステム終了後に，運用支援環境のIOFに対する終了を指示し確認後に③，メッセージ制御システムとしてのCOMSや他のMCSなどに終了指示する④．

最後に，ディスクファミリ上のワークファイルなど不要な資源を整理するワークフロージョブを開始し終了後に⑤，第三段階としてシステム停止用プログラムのSCJ II/KILLERに引き継ぐ⑥．SCJ II/KILLERはシステム操作支援ソフトウェアASSISTANTを終了させる⑦．その後，SCJ II向け動作パラメタの内容に従い，システム上に他のプログラムが実行中でないことを確認した後に，システム電源切断用プログラムのSCJ II/POWEROFFを起動する⑧．SCJ IIに対して電源切断指示が通知され，ホスト，周辺装置および空調装置の電源が切断され，一日のシステム運用処理が終了する⑨⑩．

4.4 24時間運転への応用

24時間連続運転を行う場合にはSCJ IIによるシステムの運転開始/運転終了の操作は不要である．この場合には，システム初期設定の再設定が必要な時に，ASSISTANTによってシステム構成や動作パラメタの確認と設定を行うことができる．任意の時間に，ASSISTANTに対して，業務システムの稼働開始・終了の時間帯やシステム環境の変更条件を判断する手続き，および時刻指定または操作員入力によるシステム指令の投入を記述することで，連続運転時に必要となる動作環境を再設定できる．

5. 自動運転機能の今後の展開

センタ中心のシステム運用の自動化は単一ホストでの構築にあるが，今後の自動運転の拡張としては，以下に記述する分散処理の運用強化のために，単一視点での遠隔地ホストの集中監視・制御・管理，障害時の内外部通報に関する視覚・音声の対処ガイダンス，障害時の回復処置の自動化などの機能拡充が求められている．

5.1 遠隔地ホストの集中監視

複数ホストによる分散処理体制をとっている場合，各地に配置された遠隔ホストに対する運用では，夜間の人員配置や障害時の処置などの問題を抱えている．この対応としては，ASSISTANTやIOFを複数ホスト向け監視機能に拡張し，全ホストの事象・状況をセンタホスト上の運用監視端末で集中監視する．そのためには，BNAネットワーク経由での遠隔システム間通信を利用して，センタ側の集中監視操作卓による単一視点での各遠隔地ホストのシステム制御や操作指示，運用情報の一元管理を可能にすべく，操作性向上の機能拡充が必要となる．

この集中監視操作卓には，システム環境，各ホスト稼働状況，業務システムの稼働状況，ネットワーク稼働状況および端末稼働状況を常に監視して状況変化に対処可能な管理機能を持たせる．

5.2 障害時の機能拡充

複数ホストで構成されている分散処理環境では障害発生時に内部通報する場合，多種多様の障害やシステム状況を運用担当者に正確に伝達し画一な対処を促すため，到達メッセージに基づき各ホスト別に障害の内容と稼働状況，障害回復の対処ガイダン

スを集中監視操作卓にGUI視覚や音声で，わかり易く告知する必要がある．

このことは標準の運用支援ガイダンスに従って，障害発生後の操作員の即時な行動を促す．また，無人運転時の外部通報では，在宅運用担当者への電話呼出による音声ガイダンス，自動FAX発信での障害状況や対処方法についての報告手段も必要となる．

障害管理の回復処置の方法では，対処作業の自動化を狙うには，単純な対応から運用担当者の経験則によるヒューリステイックな規則を作業実体に等値するエキスパートシステムの採用が必要となる．

6. おわりに

Aシリーズシステムの運用を自動化するためには，オペレーティングシステムMCPの技術を活用した運用支援環境用のツールを導入し，単一システムでのホスト運用における通常時の操作や管理をいかに自動化して，少ない作業量で特殊な知識を必要とはせず，一定の標準的な規則によって運用を行うかにある．実行運用環境においての業務処理の運行・実績管理，障害対処，性能・効率管理，ソフトウェア管理など，より的確な運用推進には，運用担当者の役割は大きい．また，近年，オープンシステムを含めた分散処理環境の集中管理の課題がクローズアップしている．

オープンシステムの分散処理環境では，各システム単位の自動運用の支援のために，特定の所に操作員やシステム運用の知識を集約し，単一の場所から各システムの集中操作，監視・作業管理などを単一視点によって運用できる仕組みを計画している．

執筆者紹介　山口和男 (Kazuo Yamaguchi)

1970年茨城大学文理学部理学科卒業．同年日本ユニシス(株)入社．主に流通・製造業関連のシステムサービス業務に従事，現在，システム技術第二本部 利用技術二部 運用システム開発課課長．情報処理学会会員．

丸岡茂敏 (Shigetoshi Maruoka)

1984年学習院大学文学部卒業．同年日本ユニシス(株)入社．金融・保険業関連のシステムサービス業務に従事後，運用管理ソフトウェアの開発に参画．現在，システム技術第二本部利用技術二部運用システム開発課所属．

オブジェクト指向による分散システム運用

An Operation of Distributed Data Processing Systems Through the Use of Object-Oriented Design

小 林 良 弘

要 約 多くのユーザがUNIX*やPCを中心とした分散処理の導入を進めている．分散システムにおいては，コンピュータがネットワーク上の広範囲に配置される．ユーザはこれらの分散されたコンピュータに管理者を置くことはできない．このため遠隔地からこれらのコンピュータを管理する仕組み等が必要とされる．

最近，分散システム管理ソフトウェアの製品化が活発になっている．管理の面からの分散システムの特徴に，異機種が混在する，ネットワークの変更が頻繁に行われる，などが挙げられる．これらの要件はシステム管理ソフトウェアにとって非常に難しい．近年この分散システム管理にオブジェクト指向技術を適用することが進められている．オブジェクト指向は，利点に拡張性，保守性の良さが挙げられる．この利点を生かして，拡張や変更に耐えられる柔軟なシステム管理ソフトウェアを開発することが，その目的である．

現在，弊社で開発中のDSmgrは，そのオブジェクト指向を適用した分散システム管理ソフトウェアである．

Abstract Nowadays, a large number of computer users are working on UNIX- and PC-based distributed data processing. In the system of this type, computers are so widely deployed throughout a network that users cannot afford to assign a system manager or an operator to each of them. That is why users want new ways to make it possible for those computers to be remotely controlled.

More and more software items which serve to control and manage distributed systems have been put on the market. Typically, distributed systems generally co-exist with a mixture of varying types of machines provided by different vendors and with frequent changes in the network configuration. Since those factors are headaches to systems management software, the current trend sees object-oriented design technology being used for the development of such software because of its superior expandability and maintainability. The objective of using the technology is to exploit its advantages to develop software flexible enough to be easily extended and modified.

DSmgr, now being developed at Nihon Unisys, is an software item so dedicated, for which object-oriented design is adopted.

1. はじめに

汎用機での集中処理から，UNIXやPCを中心とした分散処理の本格的な導入が進行しつつある．近年の企業システム全般のUNIX分散化や，企業の中枢的な基幹業務への適用がその流れを裏付けている．優れたコストパフォーマンスを持つサーバ/ワークステーションや高速で接続性の高いLANがこの流れを支え加速している．この分散処理により，クライアント・サーバシステム等の新しい形態のシステムが登場した．

* 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．

しかし，システムの自由度が増加した反面，管理する面では難しさが増している．その結果，分散システムの開発と維持のコストの8割を，運用管理が占めるとも言われている[1]．このような状況を反映して，分散システムの運用管理を支援するシステム管理ソフトウェアの製品化が活発になってきている．

2. 分散システムにおけるシステム管理の要件

システム管理ソフトウェアは，主に汎用機においてそのコンピュータ運転の自動化を目標に発展してきた．汎用機に対応するシステム管理は，管理技術を持つ専任のシステム管理者やオペレータが，集中的に設置された1台あるいは複数のコンピュータの管理を行う作業のことである．

しかし，分散システムではLANなどのネットワークを介して複数のコンピュータが広い範囲に分散されるので，それぞれに汎用機と同じように専任の管理者やオペレータを，設置することは不可能である．しかもUNIXなどの分散システムのOSは，システム管理の機能面で汎用機のそれと比べて大きく改善されているわけではない．例をあげれば，業務上重要なファイルのバックアップ作業は依然必要であり，誰がいつどのように行うかが問題となる．ここで分散システム管理の機能要件を整理すると以下のようになる．

1) 管理対象の分散への対応……分散システムでは，管理者が管理対象（たとえばサーバ）の近くに常駐できない．したがって以下の要件が要求される．
   - 管理作業を自動化してシステムの無人運転の実現
   - 離れた場所にいる管理者からの管理作業の遠隔操作化
   - エンドユーザなどの未熟な操作者でも操作可能なわかりやすいインタフェースの提供

2) オープン・ネットワーク対応……分散システムは，UNIXやLANと言った標準化された実行環境の中で異なったベンダのハードウェアが混在することが一般的である．この異機種を含めた管理の必要性から，どのハードウェアでも稼働するシステム管理ソフトウェアとしての移植性の考慮と，他の管理システムとの協調管理の実現のための国際/業界標準への準拠が要求される．

3) システム管理とネットワーク管理の統合……分散処理は，ネットワークを介した複数のマシンで協調処理される．このようなシステムの正常稼働を監視するには，システムとネットワークの両面からのチェックが必要となる．クライアント・サーバシステムにおけるワークステーションへの未応答の原因として，処理データ転送の経路となるLAN通信機器の不良の場合もあれば，サーバ上のプログラムの異常終了の場合もある．これらの障害を発見するためにはシステムとネットワークを同一の視点から扱う管理機能が求められる．

4) 構成変更やシステム拡張に対応……LANの接続性の良さにより，機器の追加接続や移動が簡単に行えることや，高性能で低価格な機器の短いサイクルでの市場投入により，システムやネットワークの構成の変更が頻繁に行われることが予想される．これに対応するため，管理システムとしては管理対象の追加/変更/削除やその関係の変更に対して，システム稼働中に動的に行える機能が要求される．

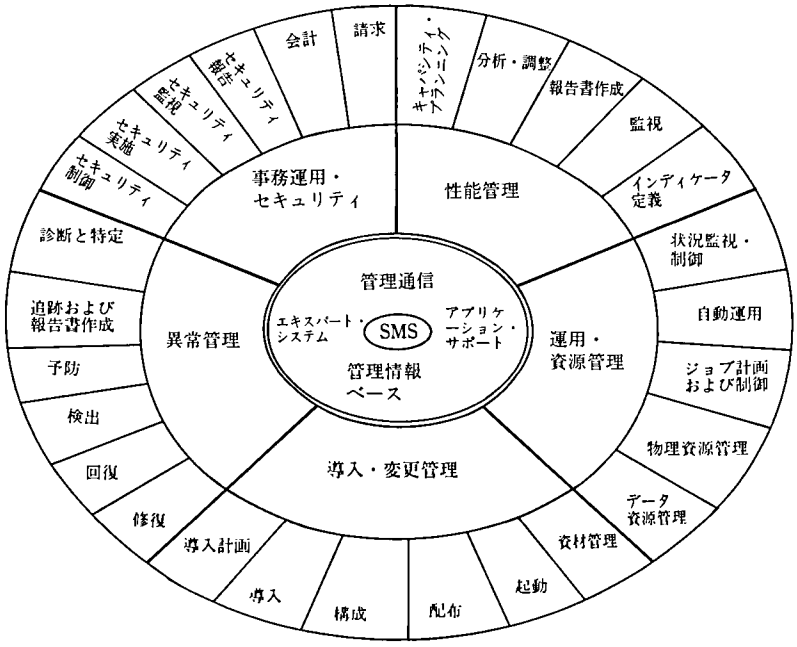


図 1　SMS

さらに，従来からのシステム管理ソフトウェアの技術的な流れとして，統合化がある．

5)　管理機能の統合……システム管理の機能範囲は広く，システム内で発生した障害の検知，ファイルの管理，バッチジョブの起動など多岐にわたっている．これらの機能が，体系化されていないバラバラのシステム管理ソフトウェアで実装されていると，ソフトウェアごとのオペレーションの修得や，同一管理データの重複した登録や不一致が発生する可能性がある．これではシステムそのものに問題があるばかりでなく，効率的な管理作業を支援するとは言えない．機能ごとにシステム管理ソフトウェアを統合化して，管理者インタフェースの統一化によるわかりやすい管理操作環境の提供や，管理データの共有による登録作業の効率化と内容の一貫性の維持が要求される．

なお，管理機能の体系化については，米 UNISYS 社のユニシス・アーキテクチャ (UA) の中のシステム管理サービス (System Management Service) が行っているので参考としてあげる (図 1)[2]．

さて，今まで述べた分散システムの管理に対する要件を標準化や実際のシステム管理ソフトウェア製品がどのように対応しているかを次章で説明する．

3. 分散システム管理の技術動向

従来システム管理技術は，ソフトウェア工学的アプローチがあまりなかった分野であり，管理現場の経験則から機能が磨かれてきた在野の技術と言える．しかし，分散システムにおける運用の重要性の高まりとともに，その標準化の推進や管理ソフトウェア製品化が活発になってきた．ここでは，視点を変えて標準化の動きと製品化動向

を探る．

1) UNIX のシステム管理……UNIX の標準化団体である UI や OSF は，自らの分散システムアーキテクチャの中でシステム管理を高い位置付けで取り扱っており，これらの団体はシステム管理の標準化のための規約を決めるのではなく，参加企業の UNIX 機で稼働する共通のシステム管理プロダクトを開発し提供することにより，参加企業内の相互運用性を実現しようとしている．システム管理の標準化は，標準化項目が多種多様になることから，実装に向けての調整を含めると，その作業量は膨大なものになる．その意味でこの共通システム管理ソフトウェアの供給案は，非常に現実的な解と言える．

しかし，現段階ではこれらのソフトウェアは機能ごとに開発企業が異なることから，使用者インタフェースの不揃い等の寄せ集め的な観が強く，またメンバ自身が持つ製品との関係など問題含みでもある．両者の共通した特徴として，システム管理ソフトウェアのシステム基盤に Tivoli 社の WizDOM を採用し，オブジェクト指向の管理システムになっている点が挙げられる[3]~[7]．

2) ISO……国際標準化団体である ISO は，ネットワーク管理の標準化として OSI 管理の標準化を進めている．この OSI 管理も管理データの持ち方を規定する管理情報モデルに，システムの柔軟性や拡張性を理由にオブジェクト指向を取り入れている．OSI 管理の標準化のスケジュールを添付した（表 1）．先にも述べたように，システム管理の標準化は決める項目が多く，ベンダの利害も絡むことから，標準化に長く時間がかかっている．すでに CMIP/CMIS (Common Management Information Protocol/Common Management Information Service) の管理プロトコルについては標準化が終わっているが，障害管理などの管理機能に関しては標準化された範囲も狭く，かつ抽象度が高いことから，実装にはまだ労力と時間がかかると思われる[8]~[13]．

3) システム管理プロダクト……UNIX 分散システムに対応したプロダクトが出揃い始めているが，これらはいくつかに分類することができ，それぞれに長所短所を持っている．

① LAN 管理拡張型

すでに業界標準となった TCP/IP の管理プロトコルである SNMP に対応した LAN 管理システムは，多く市場に出回っている．これらの LAN 管理システムは，UNIX サーバ/ワークステーションの生死や，CPU 使用率などの性能データが採取可能なことから，システム管理的機能をも持ち合わせることができる．LAN 管理拡張型は，これら LAN 管理にソフトウェア配布機能などを追加して拡張したものである．製品も多く選択肢は広いが，システム管理機能は付け足し的で機能が弱い．

② メインフレーム運用管理移行型

メインフレームで実績のあるシステム管理ソフトウェアを，UNIX へ移植したシステム管理製品である．メインフレーム運用管理での実績ある管理技術が反映されており，管理システムとしての完成度は高い．ただし，メインフレームのシステム管理ゆえにスタンドアローン型であり，複数コンピュータが連携

表 1　OSI 管理の標準化状況（1993.12 現在）

文書番号	表　　題	CD	DIS	IS
7498-4	管理の枠組み			89/1
10040	システム管理概要（SMO）			91/8
	管理情報構造（SMI）			
10165-1	管理情報モデル（MIM）			91/8
10165-2	管理情報定義（DMI）			91/8
10165-4	管理オブジェクト定義ガイドライン（GDMO）			91/8
10165-5	一般管理情報定義（GMI）			93/2
10165-6	管理情報実装適性宣言様式（ICS）			93/6
10165-7	一般関係モデル（GRM）		94/3	95/3
	共通管理情報サービス（CMISE）			
9595	共通管理情報サービス定義（CMIS）			90/11
9596-1	共通管理情報プロトコル仕様（CMIP）			90/11
9596-2	CMIPのプロトコル適合性宣言様式（PICS）			92/6
	システム管理機能（SMF）			
10164-1	オブジェクト管理機能			91/8
10164-2	状態管理機能			91/8
10164-3	関係表示属性			91/8
10164-4	警報報告機能			91/8
10164-5	事象報告管理機能			91/8
10164-6	ログ制御機能			91/8
10164-7	安全保護警報報告機能			91/8
10164-8	安全保護監査証跡機能			92/6
10164-9	アクセス制御のためのオブジェクトと属性			94/3
10164-10	利用計測機能			94/5
10164-11	計測オブジェクトと属性			93/3
10164-12	試験管理機能			92/10
10164-13	集計機能			93/6
10164-14	信頼性と診断テストの分類			TBD
10164-15	スケジュール管理			94/3
10164-16	管理知識管理機能		94/2	94/10
10164-17	切替え機能		95/6	96/6
10164-18	ソフトウェア管理機能		95/6	96/12
10164-19	管理ドメインと管理ポリシー管理機能		95/6	96/6
10164-xx	拡張事象制御機能	94+7	95/6	96/6
10164-xx	時間管理機能	94/7	96/12	97/12
10164-xx	一般関係管理機能	94/12	96/6	97/6
10164-xx	コマンド配列機能	95/6	96/12	97/12
10164-xx	応答時間監視機能	95/12	97/6	98/6

して管理する分散システムに対応した機能が弱い．

③　分散対応管理ツール型

小規模の分散システムの管理を目的としたシステム管理プロダクトで，分散システムで必要となるプログラム配布やバックアップ等の単機能運用管理ツール群である．軽微な機能を提供した簡易ツールであり，履歴やそれぞれの機能の進捗監視などの管理者支援機能が弱い．

④　オブジェクト指向型

管理システムの基盤として，オブジェクト指向技術を取り入れたシステム管理製品である．ISO や X/Open*，OSF と UI がオブジェクト指向を表明していることから今後のシステム管理の本流となる可能性がある．新しい技術ゆえ，標準化の行方などの解決すべき問題を抱えていることも事実である．

このように分散システム管理は様々な動きを見せているが，標準化ではオブジェク

* X/Open は X/Open 社の登録商標である．

ト指向を取り入れた分散システム管理が主流になりつつある．オブジェクト指向の適用は，システム管理ソフトウェアにとって，機能の拡充や新プロトコルの対応などの今までの発展とは違った，ソフトウェアとしてのシステム構造を見直す新たなアプローチと言える．次にシステム管理へのオブジェクト指向の適用についてふれる[14][15]．

4. オブジェクト指向のシステム管理への適用

オブジェクト指向は，ここ数年脚光を浴びているシステム技術で，オブジェクト指向言語，オブジェクト指向設計，オブジェクト指向データベースなどの広がりを見せている．本章ではオブジェクト指向技術のシステム管理における適用を説明したいと思う．OSI 管理を始めとして，システム管理のオブジェクト指向の適用についてはその理由としてシステムの拡張性を挙げている．ここではオブジェクト指向の特徴と照らし合せて，もう少し詳細にその適用について検証してみる．

オブジェクト指向技術を簡単に説明すると，システム構築にあたってそのシステム化の対象となる実世界の様々な実体（たとえば請求書，製品）を抽象化し，それをオブジェクトとして計算機内に写像して実装することである．オブジェクトは，システム化の対象となる実体が持つ性質としてのデータ（属性値）と，仕事としての処理（メソッド）が一体化したものとして実装される．

システム管理ソフトウェアの開発をオブジェクト指向で考えると，システム管理ソフトウェアにとってのオブジェクトは管理対象そのものとなる．システム管理ソフトウェアは，管理対象(Managed Object)である実際のハードウェアやソフトウェアに対して管理操作である処理を実行する．また，これらの管理対象の状態変化を通知する．つまり，オブジェクト指向分析の最初の課題とされている「何をオブジェクトとするか」は管理対象そのものをオブジェクトとして定義すればよいことになる（図 2）．

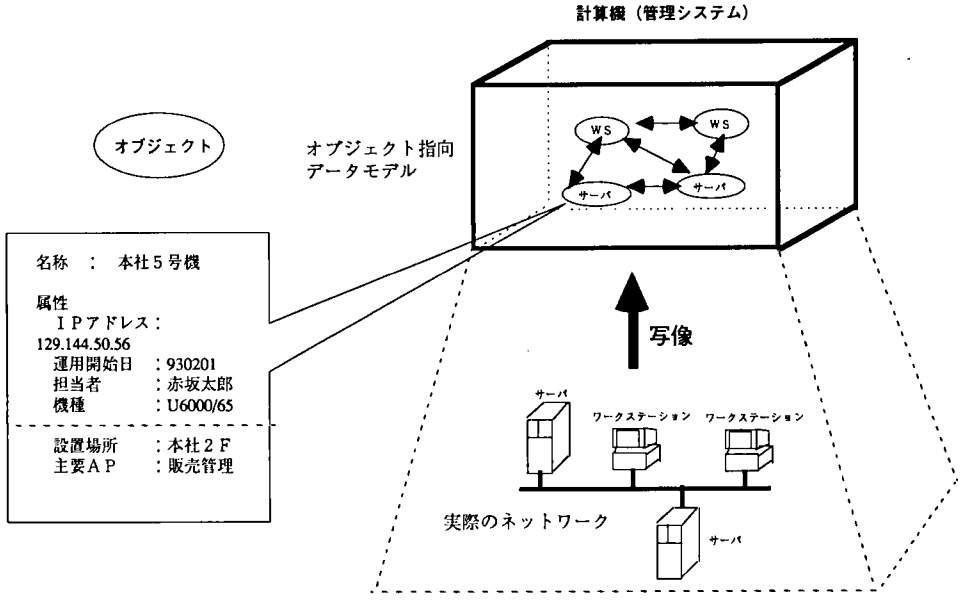


図 2 システム管理における管理対象のオブジェクト化

この説明のみでもシステム管理へのオブジェクト指向の適用のしやすさがわかると思うが，その他にもオブジェクト指向によるシステム管理ソフトウェア構築の利点がいくつか挙げられる[16]~[19]．

1)　クラスとインスタンス……オブジェクト指向では，システム化の対象となる実体をオブジェクトとして定義するが，共通の属性（性質）を持ったものをグループ化して一つの型（タイプ）として認識する．それをクラスと呼び，そのグループを構成するメンバをインスタンスと呼んでいる．これをシステム管理の実装で言い換えると，管理対象としてのワークステーションと，実際に設置されているワークステーションの本社販売システム用1号機や人事システム用2号機がそれにあたる．

このように，管理対象が持つ属性のデータ形式や構造と，それぞれ実在する管理対象が持つ値を別々に管理することにより，管理システムとして何を管理するかが明示的に示され，外から解かりやすいシステムになると共に，拡張性のあるシステムとなる（図3）．

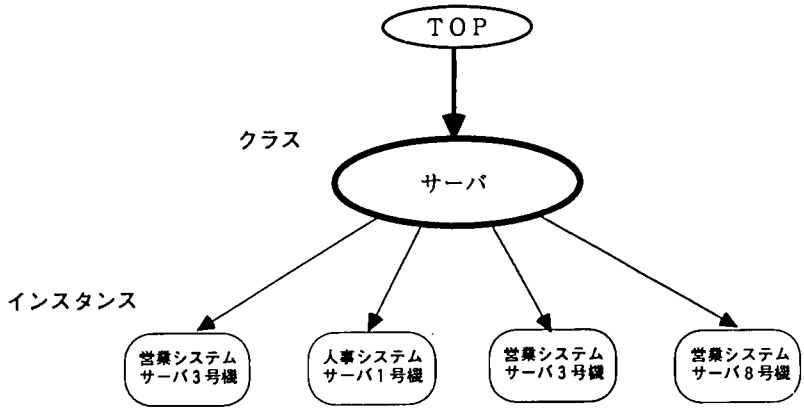


図 3　管理対象におけるクラスとインスタンス

2)　継承(インヘリタンス)……オブジェクト指向で定義したオブジェクトのいくつかを比較してみると，一方が他方の性質を含んで，より広い概念を表しているものがある．たとえば動物と人間である．人間が動物としての性質を持ち合わせていることは明白である．動物と人間はそれぞれ先に述べたクラスに当たるから，これらのクラスを階層化して上位のクラスの性質を下位のクラスが引き継ぐ形で表すことをインヘリタンス（継承）と呼ぶ．オブジェクト指向では動物に当たるクラスをスーパークラス，人間に当たるクラスをサブクラスと相対的に呼んでいる．

システム管理の管理対象の例をとると（一般的な）コンピュータとサーバが挙げられる．これらを比較してみると，コンピュータにはサーバの他に，汎用機やワークステーションやPCも含まれ，サーバより一般的で広い実体を指すクラスであることが理解できる．コンピュータとサーバは，共にメモリを持つため，メモリ容量と言う属性を持つことができるが，この属性をコンピュータの属性とし

て定義し，サーバはそれを継承する形にする．この継承の導入により，管理対象の属性を整理して持つことができるとともに，管理対象をコンピュータとして，ワークステーションとして見るなど管理の観点から複数の見せ方が可能になる．

また，拡張性についても，たとえば携帯用ワークステーションのような新しい管理対象を登録する場合でも，新たに追加された属性とメソッドを追加するだけで，その機器を管理対象として取り込むことができる（図4）．

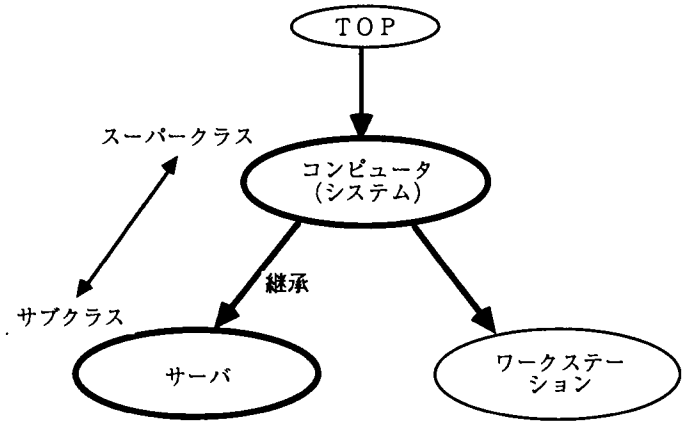


図 4　管理対象オブジェクトの継承関係

3)　カプセル化……カプセル化とは，オブジェクトが持つデータへのアクセスを外部から遮断することである．これをシステム管理で適用すると，管理対象のデータへのアクセスや管理操作を行える権限の制御が可能となる．言い換えると誰もが管理作業を行えるのでは問題が生じる．そこで管理対象ごとの属性や処理を，オブジェクトとして封じ込める（カプセル化）ことにより，それを扱うプログラムから情報を隠蔽し，管理対象の不必要なアクセスから保護する（図5）．

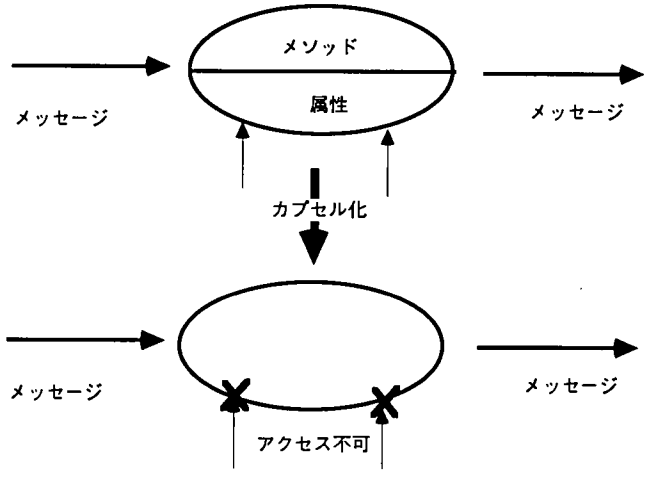


図 5　管理対象オブジェクトのカプセル化

4)　多態(ポルモルフィズム)……オブジェクト指向でいう多態とは，オブジェクトの持つメソッドを抽象化して，同一のインタフェースで複数の対象に行う処理を可能にするため，オブジェクトが最適なメソッドを選択して処理を実行することである．

システム管理の処理で説明すると，電源オフという処理依頼を，対象がワークステーションでも，ルータでも，同じインタフェースからの指示で可能にすることである．実際の電源オフ処理は管理対象ごとに違うが，その選択をオブジェクトが行うことにより，指示する側が意識しないで指示することができる．この多態を利用することにより，管理対象の追加などでも管理操作の指示側は変更なしに拡張できる（図6)．

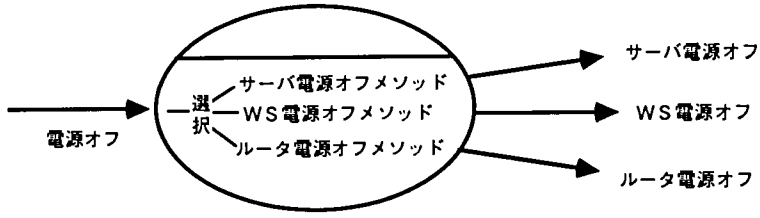


図 6　管理対象オブジェクトの多態（ポルモルフィズム）

ここで述べたように，オブジェクト指向の特徴が分散システム管理のソフトウェア構築にうまく適用できることが理解できると思う．システム管理におけるオブジェクト指向適用の最大の利点は，そのシステムの柔軟性にある．システム管理ソフトウェアは本来，先ず管理すべき業務システムやハードウェアが初めにあり，それをあるがままに管理するという受動的な面を持つ．

先にも述べたように，分散システムの登場とともにその管理対象とその関係のバリエーションの拡がりと，その急激な進化の速度への追従がシステム管理ソフトウェアを非常に難しいものにしている．その柔軟性ゆえに保守性/拡張性が優れているといわれているオブジェクト指向の適用がここにある．

次に，現在弊社で開発中のオブジェクト指向の分散システム管理ソフトウェアであるDSmgrについて説明する．

5. DSmgrの概要

DSmgrは，UNIXを中心とした分散システム環境のためのシステム管理ソフトウェアとして，現在開発中である．オブジェクト指向技術を導入している．ここでは，DSmgrのシステム管理ソフトウェアとしての機能や構造を説明しながら，分散システム管理のオブジェクト指向適用を説明する．

1)　管理モデル……DSmgrの管理モデルは，管理範囲として，小規模のLANではなく企業規模のネットワークとした．このネットワークをマネジャエージェントによる1地点からの集中管理とした．その理由として，分散処理によりシステムの自由度が増大しても，システムの管理的側面には拘束力のある統一化されたルール（規律）が必要だからである．たとえば，セキュリティは複数の脈絡のない

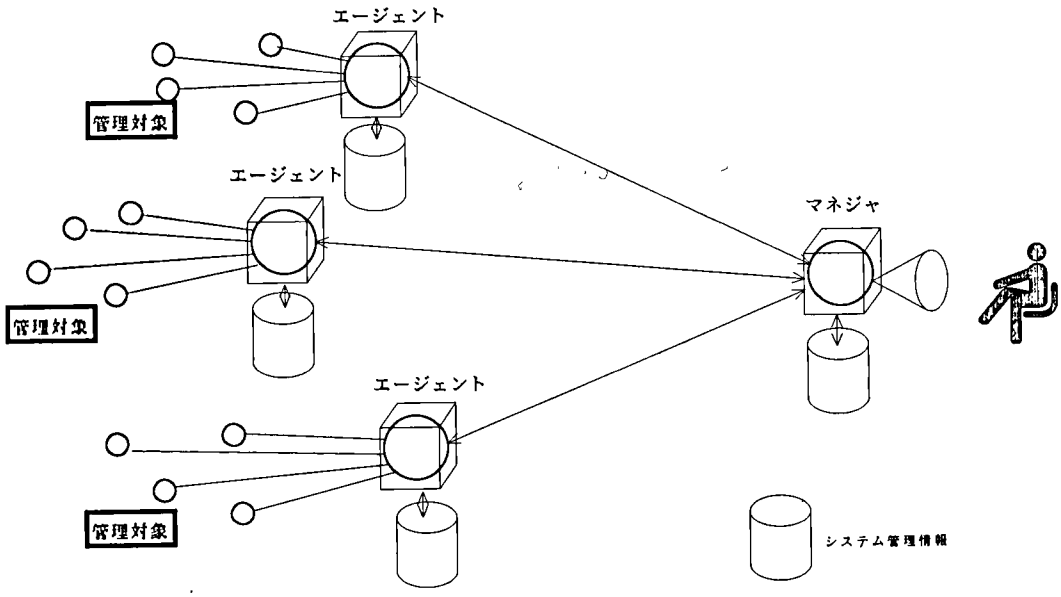


図 7　マネジャ/エージェントモデル

運用を行えば，意味と機能を失う．分散システムであっても，システム管理は限定された専任の管理者による集中した管理が必要であると考えた．また，管理者の育成は難しく費用と時間が必要なことから，分散システムの管理者は，既存汎用機のシステム管理部門の要員がシフトすることが予想される．

このような観点から，DSmgr は既存の管理技術のある管理者が 1 地点から分散処理全体を管理することを前提とし，OSI 管理のマネジャ/エージェントモデルを採用した (図 7)．このモデルは，管理操作は管理する側のシステムのマネジャから発行され，管理される側のエージェントで実行される．管理対象の状況変化を伝えるイベントはエージェントから発行され，マネジャに通知される．

2)　ネットワークモデル……ネットワークモデルは，管理対象が散在するネットワークを管理システムがどう捉えるかである．先にも述べたように DSmgr はその管理範囲を企業規模のネットワークに設定している．したがって，管理対象の数が非常に大きくなること，それらが広範囲に分散されることが想定される．このような広域ネットワークでは管理作業の分業も必要とされると考えた．DSmgr では分散システムにおける管理域をリージョンと呼んでいる．そして，1 リージョン内に管理域内の管理対象を監視・制御するマネジャを設定している．さらにこのマネジャの上位に位置して複数のリージョンを監視・制御するマネジャオブマネジャを DSmgr では設定した．このマネジャオブマネジャとマネジャにより，企業規模のネットワークを 2 階層の階層化で管理する仕組みを提供する (図 8)．

3)　システム構成……DSmgr のシステム構成は，障害やプログラム配布などの管理のための処理を行う管理アプリケーションと，管理通信機能と管理データを格納する管理データベースの役割を担う管理基盤の，二つに大きく分かれる (図 9)．

また，管理者と管理情報の通知報告や，管理操作の実施などを行う画面インタ

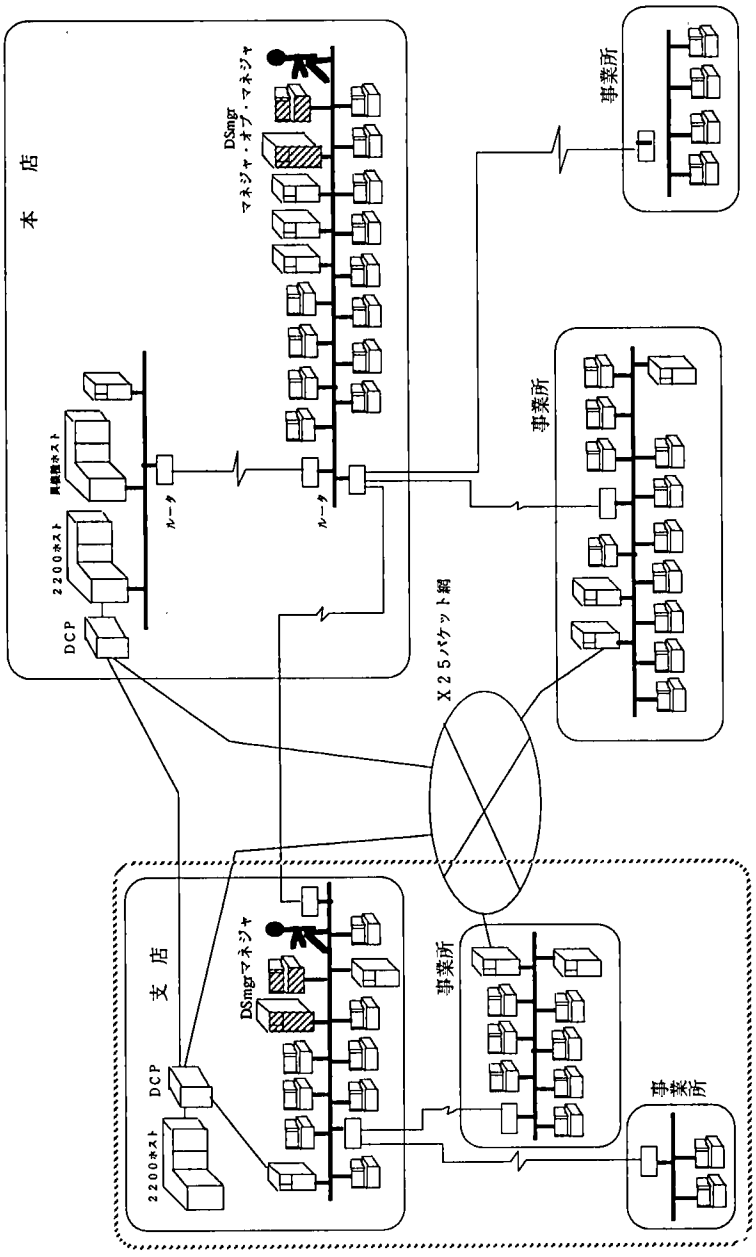


図 8　DSmgr の適用例

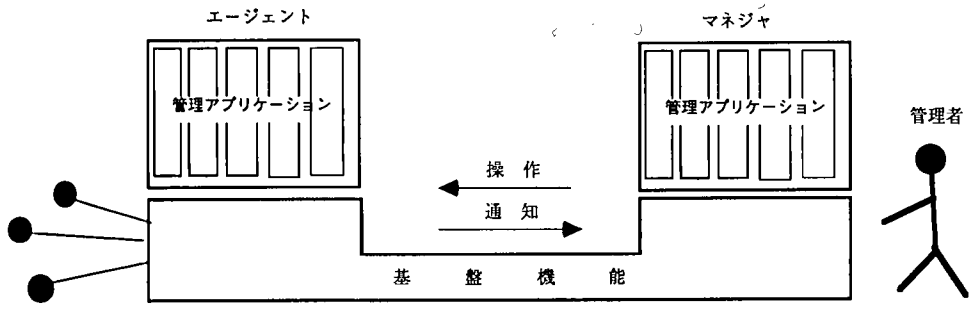


図 9 DSmgr のシステム構成

フェースを，SMView として分離した．管理アプリケーションは，複数の管理機能を統合化する方向ではあるが，多様化するユーザニーズに対応するため，ユーザが管理アプリケーション単位に選択できるように独立させた．併せて，管理アプリケーション共通の管理基盤の導入により，従来管理アプリケーションごとに実装されていた管理データベースを統合した．これにより，管理システム内の管理対象情報を一本化して，登録作業の効率化と情報の一貫性の維持を狙った．

5.1 基 盤 機 能

基盤機能は USL(Unix System Laboratory)が管理システムの基盤機能として採用している DMF(Distributed Management Framework)を採用した．この DMF は Tivoli 社の TME(Tivoli Management Framework)を SVR 4 化したものである．DMF はオブジェクト指向の基盤機能で OMG(Object Management Group)の CORBA(Common Object Request Broker Architecture)のオブジェクト構造と一致する．このオブジェクト指向基盤機能により，DSmgr の管理アプリケーションは全てオブジェクト指向のシステムとして実装される．DMF を基盤機能として採用した理由として以下の項目が挙げられる．

- システム管理ソフトウェアの基盤機能として十分な機能を持っていた．
- すでに UI/OSF によって採用されており，この DMF が UNIX を中心とした分散システム管理の業界標準ソフトウェアとしてもっとも近い位置付けにあった．
- 基盤機能として既存のソフトウェアを利用することにより，DSmgr の管理アプリケーションの開発に注力できる．

DMF はマネジャとエージェント間の処理の依頼を仲介する RPC(Remorte Procedure Call)として機能する．ここでは DMF の機能を簡単に紹介する．DMF は次の三つの部分に分類される（図 10）．

- コアオブジェクトサービス
- 共通ファシリティ
- アプリケーション開発環境

1) コアオブジェクトサービス……DMF のオブジェクトに対する，実行を支援する機能である．オブジェクトのライフサイクル，継続性，インタフェースメソッドの実行，名前付け，配置，継承，トランザクション，同時処理，安全性を支援

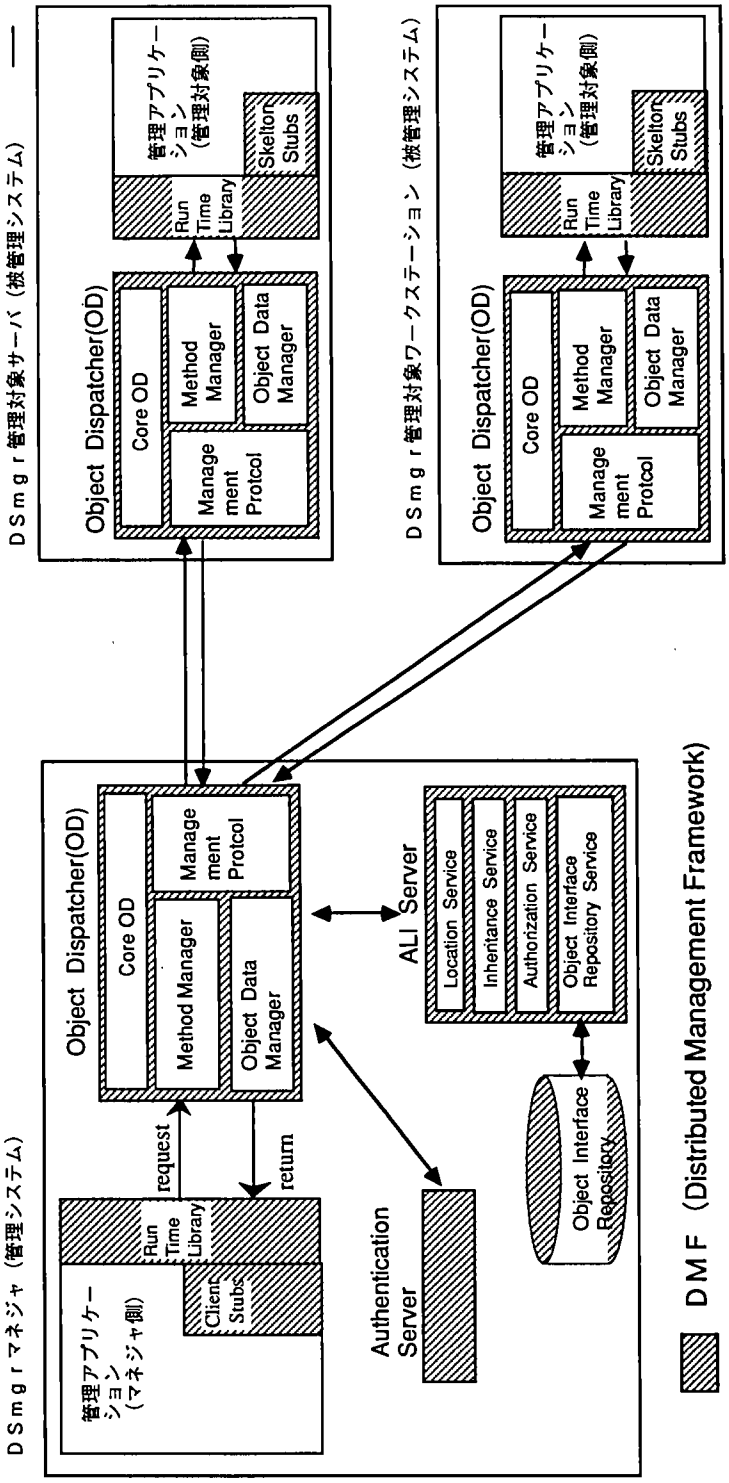


図 10　基盤機能(Distributed Management Framework)の概要

する．主な機能としては，オブジェクトディスパッチャ(OD)，ALIサーバ，オブジェクトデータマネジャがある．

ODは各コンピュータに実装され，オブジェクトに対する処理要求を，依頼されたオブジェクト（ターゲットオブジェクト）へ配送し，その処理結果を要求元へ送り返すことを行うオブジェクトリクエストブローカとして機能する．

ALIサーバはリージョン内に一つだけ存在し，クライアントからの処理要求に対してターゲットのアドレスを教えるオブジェクトの配置サービスと，実行されるオブジェクトの継承されたメソッドを決定する継承サービスと，クライアントがオブジェクトに対する処理依頼の特権をチェックする権限サービスなどがある．

オブジェクトデータマネジャはオブジェクトの状態を保存するオブジェクトデータベースの役割を担う．

2) 共通ファシリティ……共通ファシリティはクラスオブジェクトの作成や保持を支援するクラスオブジェクトサービス，オブジェクトの異常終了やエラー状態を通知するオブジェクト通知サービス，管理者との画面インタフェースとして機能するユーザインタフェースサービスなどから構成される．

3) アプリケーション開発環境……アプリケーション開発環境は，二つの開発言語コンパイラからなる．一つはDSLコンパイラ(Dialog Specification Language Compiler)で，DMFのオブジェクトへ処理依頼するための画面を介した会話処理をこのDSLで記述し，コンパイルすることにより，画面インタフェースを作成する．DSmgrの実装には後述するSMViewを使用し，このDSLは採用しなかった．理由は，当時日本語化が未スケジュールであったこと，ネットワーク図表示に必要なビットマップ表示機能がなかったことである．もう一つはIDL(Interface Definition Language)コンパイラで，これはOMGが定義するIDLと同じもので，二つのオブジェクト間の通信のインタフェースをこのIDLで記述することにより，両方のオブジェクト（プログラム）にリンクされるスタブプログラムや，交換される情報のデータ構造や定義を表すヘッダーファイルを作成する[20]~[22]．

5.2 SMView

SMViewは，管理者と管理アプリケーションとの画面インタフェースを実行する機能である．システム管理で使用者の管理に対する流儀が明確にでるのが画面と管理帳票である．汎用機のシステム管理ソフトウェアにおいても，そのためのカスタマイズが高い頻度で発生していた．管理画面については，表示される項目やその順位/位置について企業や管理者の考え方や視点が異なることから，画面変更の要求も強かった．今までは既存の画面をそのまま使うか，新たに別の画面を開発するしかこれらに対処する方法がなかった．SMViewは各管理アプリケーションの画面表示機能以外に，画面作成や変更を支援するGUI構築ツールとしての機能も備えている．そのためSMViewはDSmgrが提供する画面の項目変更等をプログラムレスの会話形式で行う．

SMViewは開発環境と実行環境の二つに大別される．SMViewは，オブジェクト指向GUI構築ツールで，Motif*のウィジェットと画面から入出力されるデータの処理

* Motifは Open Software Foundationの登録商標である．

プログラムをオブジェクトとして定義し，相互のオブジェクトを画面からグラフィカルに接続ができる機能を持っている．この機能により入力項目から選択ボタンへの変更などが，画面から簡単に行うことができる．DMF と SMView は，それぞれオブジェクト指向実行環境であり，オブジェクトの実装も違いがある．このため相互のオブジェクトの通信を実行する接続機能を開発し，連携を実現した．

6. 管理アプリケーション

管理アプリケーションは，DMF 上にオブジェクト指向システムとして実装される．ここでは，管理アプリケーションのコンフィグ，フォールト，リリースの概要を説明する．

6.1 コンフィグ

コンフィグは，ネットワーク上に分散されるハードウェアやソフトウェアの管理対象をオブジェクトとして格納し，その情報を管理する．これらのオブジェクトは，次の四つの木構造の関係によって構造化され管理される（図 11）．

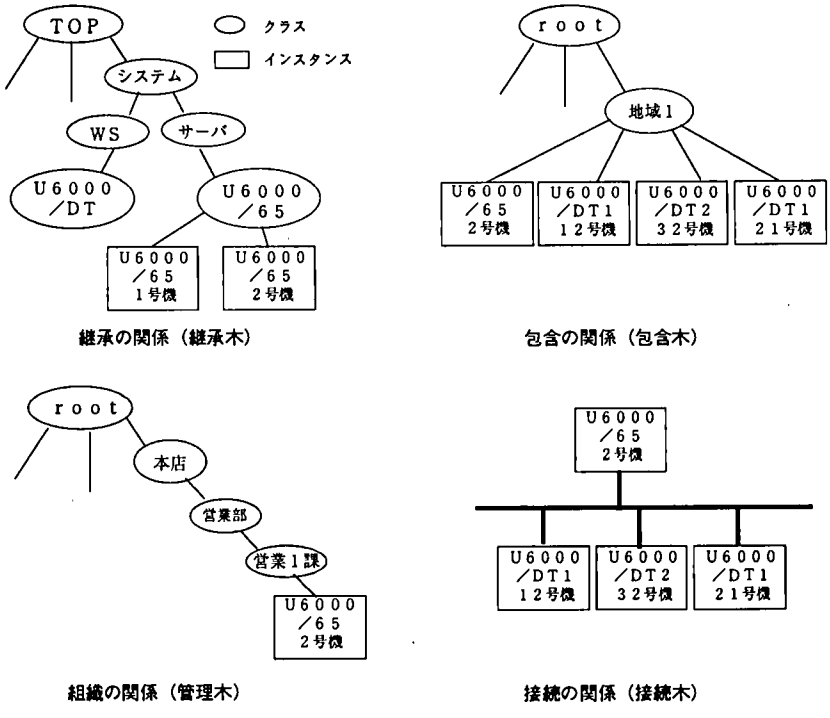


図 11　コンフィグにおける管理対象間の関係

・継承の関係（継承木）

管理対象の性質（属性と管理操作のメソッド）を整理して階層化した関係．これにより，新機種の登録も追加された性質のみの登録で済ませることができる．

・包含の関係（包含木）

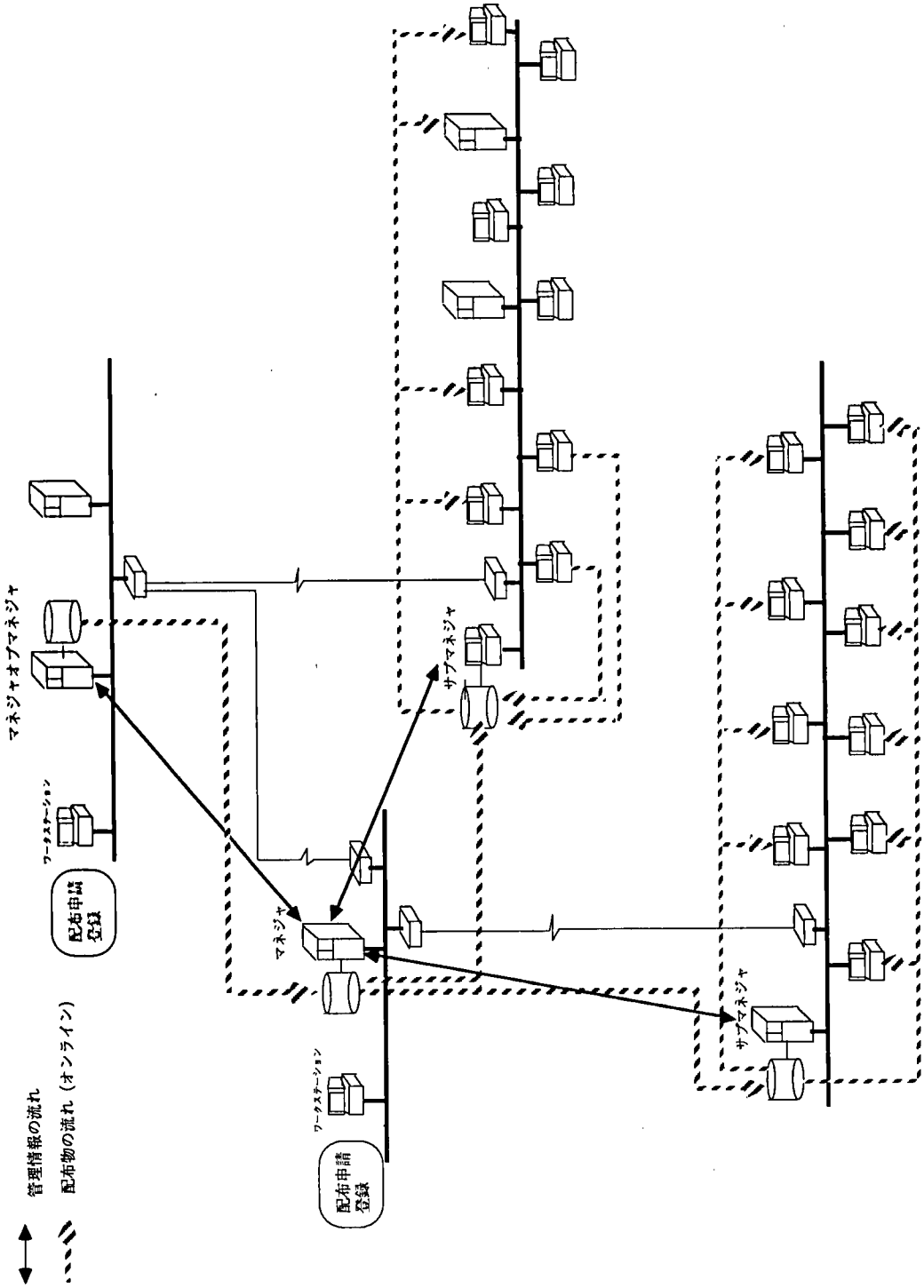


図 12 DSmgr/リリース

管理対象であるオブジェクト（インスタンス）の管理関係を表す．これにより分散システム上の管理対象が，どのマネジャに管理されるかが表現される．

・所属組織の関係（管理木）

管理対象を所有する企業等の組織を表し，管理対象がどの組織に所有されているかなどのアプリケーション上の関係が表現される．

・ネットワーク上の接続関係（接続木）

管理対象が，LAN 上でどのように接続されているかを表す．

コンフィグの機能としては，管理対象であるオブジェクトの登録，変更，削除検索である．この機能は，LAN 管理システムの構成管理機能と同等の機能である．DSmgr はこの構成情報を格納する管理データベースに先に述べた DM フレームワークを採用しており，オブジェクトデータモデルになっている．このモデルの採用により，新規クラスの追加や既存クラスの属性追加を容易に行うことができる．この特性を生かして，使用者が DSmgr の標準管理対象以外のもの，たとえば異機種汎用機を管理対象として追加登録できる機能を提供する．また，標準管理対象に属性を追加削除する機能も提供することにより，ユーザ独自の社内管理番号を属性として追加することや，ユーザにとって不必要な属性を削除を可能にする．

6.2 フォールト

フォールトは管理対象に発生した障害を検知してイベントとして通知し，管理者へ報告する機能である．これらの障害はオブジェクトとして生成・保存され，後で検索することができる．検知することのできる障害は，サーバなどのハードウェアの生死，デーモンプロセスの生死，設定されたリソース利用状況のしきい値超過，API によって通知されるユーザエラーである．また，障害の内容に対応したリカバリープロセスを登録/実行する機能を提供している．

6.3 リリース

リリースは，分散された複数のサーバやワークステーションへプログラムやデータを自動的に配布し，設定されたタイミングで本番使用のための入替を行う．機能は，配布情報の登録申請を行う配布計画機能，申請されたリリース対象物をその配布先に自動的に配布する配布機能，配布された配布物を実行環境に設定する入替機能などからなる．配布申請によって生成され，配布に関する情報を持つ配布申請書は，オブジェクトとして管理される（図 12）．

7. おわりに

本稿では，分散システム管理が必要とされる背景とその機能要件を述べ，そこにおける技術動向を標準化や製品系列について説明し，それを支える技術として，現在主流になりつつあるオブジェクト指向の適用とその有効性を，現在開発中の DSmgr の概要を交えて紹介した．一時進んでいた分散システム管理の標準化は，UNIX のベンダ団体である UI の解散や OSF の再編などによる停滞と，Windows/NT*の登場による PC の地位向上で，その行方は見えにくくなっている．しかし，ユーザの分散システ

* Windows/NT は米国 Microsoft 社の登録商標である．

ム化は着実に進行しており，分散システム管理ソフトウェアの必要性は高まるばかりである．

我々はこのような状況下でDSmgrの開発に着手したが，分散システムのシステム管理は新しい技術分野であり，そのオブジェクト指向技術の適用も始まったばかりである．新しさゆえの技術的な課題もまだ多く存在する．本稿執筆時はDSmgr開発の最盛期であり，分散システム管理としての機能の洗練化とオブジェクト指向の実装への取り組みの途上にある．また，個々の管理アプリケーションもそれぞれの特色や実装も違っており，それぞれが1テーマとして十分な内容がある．開発が一段落した時点でそれらについても是非御報告したい．

参考文献

[1] 中島丈夫, 情報処理 Vol. 34 No. 10.
[2] Unisys, Unisys Architecture Functional Overview Version 1.3 July 23, 1991.
[3] Open Software Foundation, Distributes Management Environment September 1991.
[4] Open Software Foundation, Distributes Management Environment An OverView September 1991.
[5] Open Software Foundation, Distributes Management Environment Rationale September 1991.
[6] Peggy Quinn , George Preoteasa, Reconciling Object Model for Systems and Network Management 1992 UniForum.
[7] Lisa Campbell, An Object-Oriented System Management Framework for SVR 4 1992 UniForum.
[8] ISO/IEC 10040: 1992 Information technology-Open Systems, Interconnection - Systems Management Overview.
[9] ISO/IEC 10165-1: 1992 Information technology-Open Systems Interconnection - Structure of Management Information-Part 1: Management Information Model.
[10] ISO/IEC 10165-4: 1992 Information technology-Open Systems Interconnection - Structure of Management Information-Part 4: Guideline for the Definition of Managed Objects.
[11] ISO/IEC 10165-5: 1992 Information technology-Open Systems Interconnection - Structure of Management Information-Part 5: Event Report Management Function.
[12] ISO/IEC 9595: 1991 Information technology-Open Systems Interconnection - Common Management information service definition.
[13] ISO/IEC 9596-1: 1991 Information technology-Open Systems Interconnection - Common Management information protocol-Part 1: Specification.
[14] 日経コンピュータ, UNIXの運用基盤, 1994.1.24.
[15] 日経オープンシステム, 新たな運用手法・形態の模索が始まる, 1993.10.
[16] 米沢明憲, 柴山悦哉, 岩波講座ソフトウェア科学17, モデルと表現, 岩波書店.
[17] J. ランボー他, オブジェクト指向方法論OMT, トッパン.
[18] 宮木淳, オブジェクト指向システム分析入門, SRCハンドブック.
[19] OMG監訳: 相磯秀夫, 共通オブジェクトリクエスト・ブローカ, 構造と仕様, 日本オフィスオートメーション協会.
[20] DM/Framework Administration, UNIX System Laboratories May 1993.
[21] DM/Framework Reference, UNIX System Laboratories June 1993.
[22] DM/Framework Application Programming, UNIX System Laboratories June 1993.

執筆者紹介　小　林　良　弘 (Yoshihiro Kobayashi)

昭和27年生，51年埼玉大学理工学部物理学科卒業．同年日本ユニシス(株)入社．製造系SEサービス，ネットワーク利用技術サービス，2200/IOF開発をへて，現在DSmgrの主管であるシステム企画開発一部分散システム二課課長．

UNISYS オープンフレーム A18 シリーズ

1. はじめに

日本ユニシスは企業情報システムの中枢を担うエンタプライズ・サーバとしてAシリーズ大型クラスの「オープンフレームA18シリーズ」全16モデルを発表した.

「A18シリーズ」は22倍の性能幅をカバーする1〜6プロセッサ構成で設置場所でのアップグレードが可能である.

ユーザは,これら16モデルから業務量に合った処理能力を持つシステムを選択することができ,ライトサイジングに最適なシステムといえる(写真1).

「A18シリーズ」は従来のメインフレームの利点に加えて,伝統的なサーバの特性を活かしたエンタプライズ・サーバで,UA(Unisys Architecture)で定義した情報中枢に求められる次の八つの属性を持っている.

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) 無停止連続処理
4) オープンな相互接続性/相互運用性
5) 先進的なデータベースシステム
6) 高生産性アプリケーション開発/実行環境
7) システムの無人運転/統合管理
8) 高度なセキュリティ

写真1 A18シリーズ

本稿に記載の会社名,商品名は,一般に各社の商標または登録商標である.

2. A18シリーズの特徴

「A18シリーズ」は最新のテクノロジを採用し,情報中枢機能を支える以下の特徴を有している.

2.1 高コスト・パフォーマンスの実現

「A18シリーズ」は設置面積わずか0.94 m²,高さ1.75 m²のキャビネットで6プロセッサまでアップグレード可能であり,大幅に設置面積を小さくしている.また,A19シリーズで実現した空冷技術を採用し,設備,運用面でも高いコスト・パフォーマンスを実現している.

2.2 オープン性の強化

「A18シリーズ」ではUAに基づき国際標準,業界標準であるTCP/IP,OSI,SNA対応の機能強化を図り,他システムとの連携を可能とした.加えて,セッション数の拡大やレスポンス速度のパフォーマンス向上を図り,相互接続性を強化している.

さらに高速なネットワークを直接Aシリーズに接続するために,FDDIやIEEE 802.3チャネルの提供を計画中である.

ソフトウェアの移植性では,C言語の強化を図ると共に,POSIXの実装を順次行い,移植性の向上を図っている.

相互運用性では,異機種との分散トランザクション処理環境を実現するために,X/OpenのDTPモデルに準拠したOpen/OLTPの提供を計画中である.

2.3 トランザクション処理能力の強化

従来より提供していたグローバル・ディスク・キャッシュ・システム(キャッシュ無しのディスク装置に対して主記憶をキャッシュ領域として利用するシステム)を強化,キャッシュ専用の低価

表1　A18シリーズシステム構成

モデル名	相対性能比	中央処理装置数	メモリ(MB)		入出力処理装置数	チャネル数			コンソール	
						基本		最大	基本	最大
			基本	最大		MLI	SCSI			
211	1.0	1	96	1152	1	2	2	32	1	1
311	1.3	1	96	1152	1	2	2	32	1	1
411	1.9	1	96	1152	1	2	2	32	1	1
511	2.5	1	96	1152	1	2	2	32	1	1
611	3.4	1	96	1152	1	2	2	32	1	1
711	4.4	1	96	1152	1	2	2	32	1	1
222	1.9	2	192	2304	2	4	4	64	2	2
322	2.4	2	192	2304	2	4	4	64	2	2
422	3.5	2	192	2304	2	4	4	64	2	2
522	4.7	2	192	2304	2	4	4	64	2	2
622	6.4	2	192	2304	2	4	4	64	2	2
722	8.3	2	192	2304	2	4	4	64	2	2
732	12.0	3	288	2303	2	4	4	96	2	2
742	15.7	4	384	2304	2	4	4	96	2	2
752	18.7	5	480	2304	2	4	4	96	2	2
762	21.8	6	576	2304	2	4	4	96	2	2

格メモリ「DCM(Dedicated Cache Memory)」(24 MB～480 MB) と共に新キャッシュ・システム「MACS(Memory Accelerated Cache System)」を提供する.

また，高速データ転送を支援するSCSI-IIチャネル (10 MB/秒) の提供を行う.

2.4　高生産性アプリケーション開発環境の強化

多くの実績を持つ統合CASE「LINC」やLINC設計支援ツール「LINC-DA」をさらに強化し，提供する.

「LINC-DA」はシステムの設計，開発，デバッグをPCのGUI環境で実現でき，大幅な生産性向上を図っている.

2.5　無停止・連続処理機能の強化

一つのキャビネット内に，電源装置，冷却装置，中央処理装置，メモリ，I/Oを含む全ての機能を二系列配置することによって，耐障害性の向上を図る「ドメイン・コンセプト」を採用している.

また，総合的にシステムの可用性を高める各種機能を「ASAP(A Series Availability Plus)」コンセプトに基づき順次提供していく予定である.

「A 18シリーズ」ではハードウェアの障害回復の自動化を一層強化する「ECR(Enhanced Console Recovery)」を標準装備するほか，アプリケーション障害の自動検知と自動回復を支援する「AHB(Application HeartBeat)」の機能を提供する.

2.6　無人運転・統合管理機能の強化

「A 18シリーズ」では「ASMF(Advanced Systems Management Framework)」に基づき，統合運用管理システム「IOF(Integrated Operating Facility)」を強化し，すでに提供済みのジョブ管理，プリント管理，リソース管理，障害管理の各モジュールに加え，新たにパフォーマンス管理，リリース管理を行う.

さらに，一つのコンソールで複数システムを集中監視，制御ができる「SPO(Single Point Operations)」の提供を計画中である.

また，ユーザのシステム稼働状況を当社の監視センタより集中監視するリモート監視サービスを提供する.

3.　商品概要

「A 18シリーズ」の販売開始により，デスクサイド型のA 7シリーズから超大型のA 19シリーズまで，同一のOS「MCP/AS」により完全互換を保証し，300倍の拡張性をもつUNISYS Aシリーズは，さらに充実した製品ラインナップとなった.表1に「A 18シリーズ」のシステム構成を示す.

情報の共有活用・協創 コラボレーション・コンピューティングを実現するOpenMAPPER

1. はじめに

日本ユニシスはUA(Unisys Architecture)に基づき，PCからメインフレームまでオープンな商品を提供してきているが，今回，長年お客様に親しまれてきたMAPPERをこれからのマルチベンダ，マルチプラットフォーム環境に対応した情報活用の有効なツールとして，より広く使っていただくためにOpenMAPPERとして，発表した．

2. 背　景

情報技術，すなわちIT (Information Technology)が発達し，これらの適用分野が大きく変わりつつある．小型化・低価格化は，利用者にとってより身近に，容易にITが使える環境を生み出してきている．OpenMAPPERは，これらを可能とし，支援・促進する環境を提供するが，機能を述べる前に，求められるITの課題を考察する．

2.1 コンピュータ利用分野の変化

コンピュータのハードウェア・コストが下がることにより，企業における求められる適応分野や機能が，定型的な大容量の計算処理から，非定型な情報活用へと広がりつつある．とくにITを使って企業競争力を強化するとか，ITを中核として企業構造や，仕事のやり方を改善するという分野が注目をあびており，これらの分野で活用できる道具が求められている．

2.2 エンドユーザ・コンピューティング

エンドユーザ・コンピューティング(以下EUC)が世の中で言われて久しいが，日本におけるEUCは，PC，ワープロを中心とするパーソナルなレベルで展開されてきたといえる．

80年代に起こったOAのブームはFAX，ワープロ，PCを残しただけと言われるが，結果として，個人レベルのキーボード・アレルギーをかなり解消したと言える．過去にEUCといった場合，一番の阻害要因はこのキーボード・アレルギーであった．

しかし，家庭にもワープロやPCが入り込む時代となり，またWindowsに代表されるGUI (Graphical User Interface)の発達は，マウス，アイコンの操作による使いやすいマンマシン・インタフェース環境を提供しはじめた．その結果，ITはより身近なものとなり，利用する能力は一般的に向上した．

企業においては個人レベルの情報活用は実用の段階に入っている．しかしながら，グループによる情報共有・情報活用はいまだ未成熟な段階である．これは日本企業において，LAN等のインフラの発達が米国に比べて遅く，情報の共有できる環境が少なかったことが要因の一つと考えられる．しかし各企業においてLANの設置が推進されてきた今，情報共有の条件は整ってきた．

2.3 IT活用による生産性向上

今日のような経済環境においては企業の競合力を高めるためにITを活用することが必須である．ホワイトカラーの生産性向上が求められる環境において，ITによる情報の共有・活用こそが企業改革の鍵と言われている．

現在，ホワイトカラーの日常業務においては，伝票処理のような定型業務が大半を占めているが，これからは非定型な創造的作業を行っていくことが多くなる．非定型な作業は試行錯誤を繰り返しながら考えをまとめていくことが多いが，このような企業内活動・作業を効率よく支援してくれるITが求められている．

2.4 コラボレーション・コンピューティング

日本のOAは先に述べたように，現在は個人レベルのOAであるが，これからはITを中核としたグループの情報活用の形態をとるものと考えられる．

コラボレーションという言葉は『協創』とか『共創』と訳されるが，個人個人が互いに啓発しながら，より創造的な活動を行うことを指している．

キーボード・アレルギーがGUI等により解消し，情報の読み書き能力，利用能力が一般的に向上してきた次の段階こそ，グループの情報共有，このコラボレーション・コンピューティングが求められる（図1）．

2.5 分散システムによるコスト構造の変化

ダウンサイジング，オープン化の流れは分散化

本稿に記載の会社名，商品名は，一般に各社の商標または登録商標である．

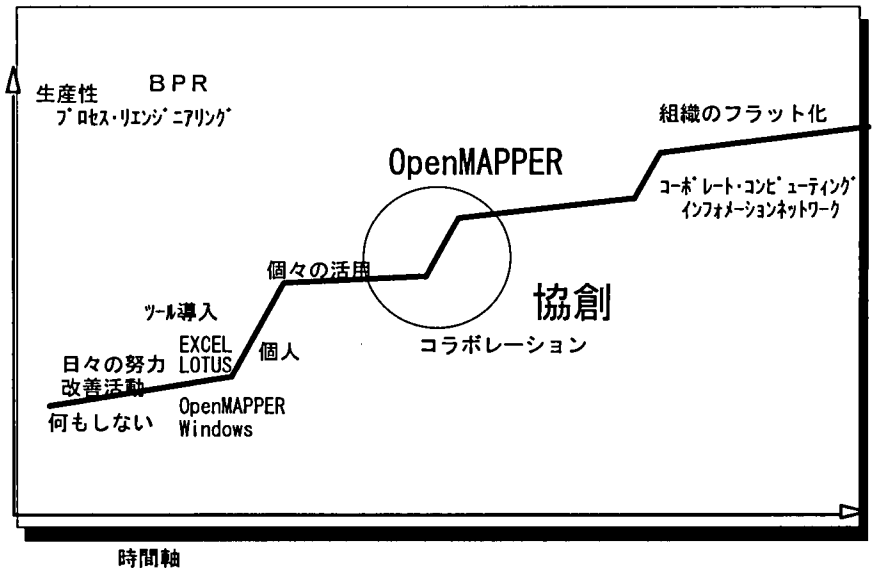


図 1　Open MAPPER の位置づけ

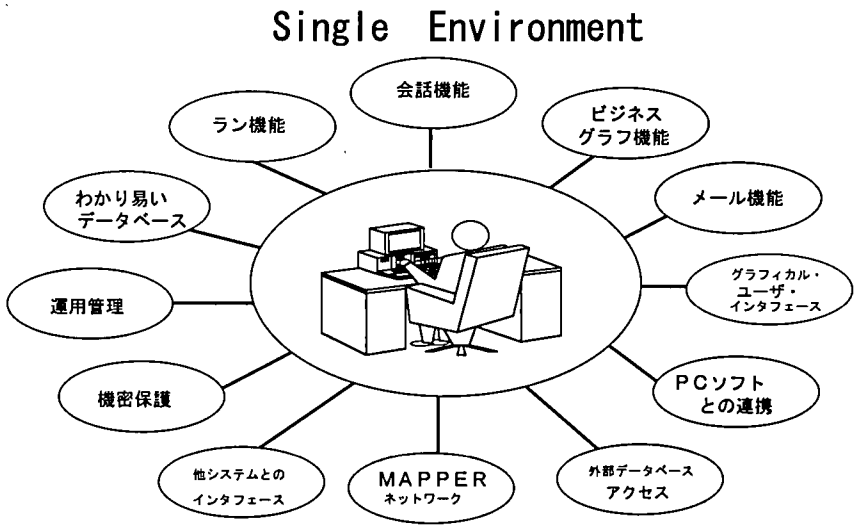


図 2　MAPPER の機能

を進めている．しかしながら，分散システムは地域的にも離れた場所に，各種のハードウェアやソフトウェアを扱うことにより，運用管理コスト，教育コストの増大といったコスト構造の変化を招いている．

3. MAPPER の機能

MAPPER は，非定型な情報活用のための EUC ツールとしての側面と，マクロ機能であるランによる 4 GL (第四世代言語) としてのシステム構築言語の二つの側面を持つ．

MAPPER はキャビネット，ドロワー，レポートという利用者にとって，わかり易いデータベース構造を持ち，これを扱う会話機能という非定型な対話処理や簡単に描けるビジネスグラフ機能を持っている．また，他部署へレポートを送付したりする電子メール機能や非定型処理を定型化するためのマクロ機能 (ラン機能) を持っている (図 2)．

MAPPER はそういう意味で単なる表計算ソフトでなく，定型処理と非定型処理の情報活用双方

の機能を持っている『環境』である.

また，非定型な処理の試行錯誤は，その課程において保存する必要のない中間成果物を作り出し，展開していくが，MAPPERによる処理の進め方はこれに近い.

この他にMAPPERは色々な機能を持っているが，外部データベースへのアクセスやOSの制御文(JCL)との連携など他システムとのインタフェース機能も持っている．これにより広く企業内の情報を活用できる.

また，異なるプラットフォーム上のMAPPER同士が連携できるネットワーク機能もある．これにより企業内の様々な情報に対し協調連携処理を行い，利用者にとって最適なMAPPER環境で処理を行ったり，クライアント/サーバ処理をMAPPERだけで構築するということも可能である(図3).

OpenMAPPERにより既存のハードウェア資産の有効活用やデータが活用でき，分散システムを構築するにあたってもプラットフォームによらない同一環境により，コストを圧縮できる.

MAPPERは長年使い込まれてきたが，それと同時に，機能強化を年々行ってきた進化するソフトウェアであると言える (図4).

社会のニーズの変化や情報技術の発達により，MAPPERの機能強化は常に行われてきた.OpenMAPPERも最新の技術を使ったものであり，上述の従来機能に加え，下記の機能強化がなされている.

1) GUI機能

すでにDW(Designer Workbench)で提供されていた機能であるが，今回Windowsで稼働するMAPPERにもマウス，アイコンを使えるGUIを採用した．これにより，キーボードを使わなくても，Windowsのメニューやボタン，リストボックスを使ったインタフェースで操作できる.

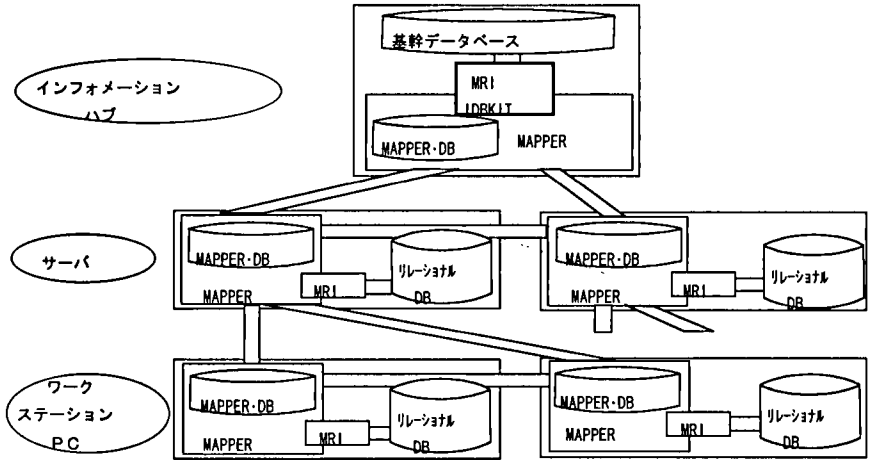
2) MAPPERと他システムとの連携

MQL(MAPPER Query Language)というインタフェースにより，SQLを知っているユーザがMAPPERのデータベースをSQLでアクセスできる機能も提供する.

また，分散環境における協調処理を構築する分散トランザクション・モニタへのインタフェースを持つことにより，クライアント/サーバ・コンピューティングのアプリケーションをMAPPERだけで構築することもできる．そして，これらは単に機能強化による利点が加わっただけでなく，既存のシステムやMAPPER以外のシステムにある情報を連携して活用する際にも同様のインタフェースが使えるといった点で極めて有効なものである.

3) データベース容量の拡大

レポート数やライン数等，データベース容量の拡大が行われた．これにより開発の柔軟



IDBKIT: Integrated DataBase Kit
(データベース検索用プログラム)

図3 MAPPERによるコペラティブ処理

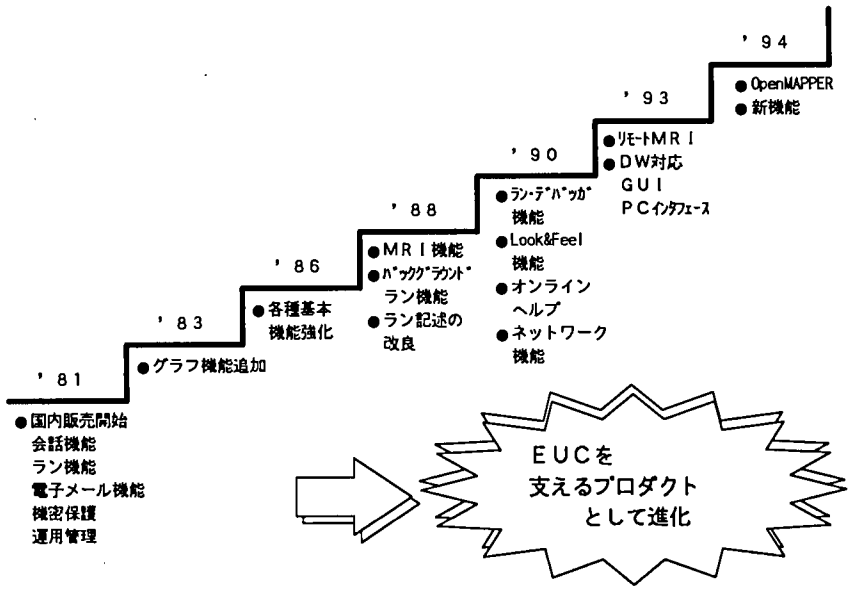


図 4　機能の強化

性，移植性の向上がはかられた．

4)　データの保全性の向上

リカバリ処理等のデータ保全性の向上も計られた．

5)　エンドユーザによる開発支援環境

エンドユーザがアプリケーション開発を行う時の支援機能も強化されている．

6)　きめの細かい運用管理機能

MAPPERが他のソフトウェアと較べて大いに異なる点は，もともと汎用機の世界から生まれ，多くの人達が共通して使うといったことが前提であったため，パーソナルなデータ処理から発生したソフトウェアと較べると，運用管理機能や機密保護機能がそのレベルやきめの細かさや障害対策において格段に充実しているところである．

また，PCで稼働するソフトウェアでは，利用者はデータを自由に活用できる権利を享受できるとともに，そのデータを管理するといった義務も負担せねばならない．その点MAPPERは運用管理者が障害対策を含め，そのような作業を行うため，利用者は権利のみを活用できる．

とくにこれからコラボレーション・コンピューティングといったグループによるデータ共有が企業において必須となる環境では個人任せのデータ管理だけでは運用できないといえる．

7)　Windows対応ソフトウェアとの連携

Windows対応のソフトウェアとDDE (Dynamic Data Exchange)やカット・アンド・ペーストを使って連携処理を行うことができる．

これらの機能充実により，今までの処理形態だけでなく，MAPPERという一つの世界の中だけでも，もしくは他のシステムとの組み合わせでもクライアント/サーバ処理や分散協調処理が可能となる．

4.　MAPPERとWindows対応他ソフトウェアとの連携

最近のパーソナルなデータ処理ソフトウェアはさまざまな機能を持ち，美麗なアウトプットを出力するには便利な機能を数多く持っている．一方そのために処理の大半を占める基本的な単純機能を行うにも操作が煩雑で，時間がかかるといった問題も出てきている．MAPPERはマルチプラットフォームで稼働する．それを生かして，処理の種類やデータの特性に合わせて最適な環境で処理ができ，試行錯誤的な処理もしやすい．

そして最終的に美麗なアウトプットが必要な時などにWindows対応他ソフトウェアにデータを

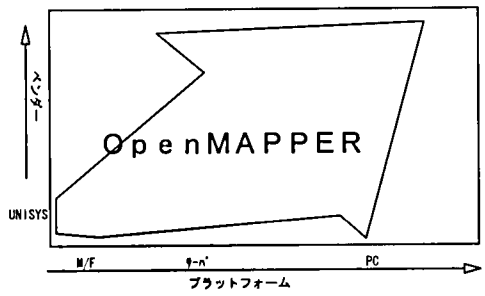


図 5 Open MAPPER の展開

渡すといった組み合わせで，効率よく仕事が推進できると考えられる．

5. 搭載機種

今回 OpenMAPPER 搭載可能機種は UNIX 版は U 6000，US ファミリー，サンマイクロシステムズ社の SunSPARC 機，ヒューレットパッカード社の HP 9000 シリーズ 700/800 機である．また Windows 版は当初は DOS/V 機から登載可能となる．

6. コンサルティング・サービス

OpenMAPPER により，ソフトウェアのみの販売を開始したが，コラボレーション・コンピューティングを推進して行くには，もはや利用者個人の努力に頼るだけでなく，企業全体に亘る意識改革や組織づくり，推進体制が必要となることも多い．日本ユニシスはそのようなコンサルテーション・サービスも提供する．

7. まとめ

MAPPER のオープン化により，プラットフォームの選択枝が増えた（図 5）．

これは，既存の資産を生かすなど，各企業の状況に合った情報技術基盤内でのデータ共有の領域が広がることになる．ユニシスは今後もお客様の視点に立って情報活用の支援を行っていきたいと考えている．

掲載論文梗概

コンピュータシステムの利用形態の高度化・複雑化にともない，システム運用部門では，システム運用の安全性確保と経済性追及，高スキル要員の維持と運転要員の負荷軽減等が重要課題となっている．荻原和彦は**IOF/MONIによる集中監視制御**の中で，IOF/MONIの開発目的，機能構成と機能概要，運用方法とその適用事例を紹介している．

U 6000シリーズ上で稼働するSPOは，異機種分散システム環境の統合管理を可能とする新しいタイプのシステム運用管理ツールである．入貝健介の**分散システムの統合運用管理ツール——SPO**は，UAのアーキテクチャの中でも特にインフォメーション・ネットワークの運用管理を目的としたフレームワークSMSに準拠したSPOの概要と汎用性のあるシステム運用管理ツールを目指すSPOの今後について紹介している．

システム運用には，「システムの初期設定と終了処理」，「操作員による操作」，「業務システムの稼働」の局面がある．山口和男・丸岡茂敏の**Aシリーズ統合運用管理システムによる運用の自動化**は，Aシリーズにおける運用の自動化のための，自動運転支援システムSCJ II，システム操作支援ソフトウェアASSISTANT，統合運用システムIOFに関する仕組みとその適用について紹介している．

多くのユーザがUNIX*やPCを中心とした分散処理の導入を進めている．分散システムではコンピュータがネットワーク上に配置され，このため遠隔地からコンピュータを管理する仕組み等が必要とされる．小林良弘は**オブジェクト指向による分散システム運用**の中で，分散システム管理が必要とされる背景とその機能要件を述べ，そこにおける技術動向を標準化や製品系列について説明し，それを支える技術としてオブジェクト指向の適用とその有効性を，現在開発中のDSmgrの概要を交えて紹介している．

* 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．

掲載論文梗概

ISSN 0914-9996

技　報
UNISYS TECHNOLOGY REVIEW
Vol. 14　No. 2　(No. 42)

発　行　日　平成6年8月31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