メインフレームからオープン系3層CSSへの再構築開発事例

Case of System Reconstruction from Mainframe to Three-TearsCSS in Open Systems

村 上 真 二

要 約 A社にてY2K対応を機に行った生産管理システムの再構築は，メインフレームからオープン系CSSへのダウンサイジングとして開発された．このシステムの開発では，CSSの形態として，3層アーキテクチャ構造を採用し，データ処理やネットワーク上のデータ・トラフィック，アプリケーション保守などの効率を上げている．

また，基幹システムのため，障害対策としてNTクラスタリングという構成を採用し，高可用性を実現している．

開発体制としては，役割毎に六つのチームを編成し，その内の主力メンバをA社内に常駐させ，ユーザと密着した体制を取った．開発管理面では，進捗，コスト管理などでは，計画，実績（進捗率），見込みの情報を一目で把握できるようにし，課題の分析，早期対応に努めた．

メインフレームからオープン系CSSへのシステム移行という面では，ユーザに混乱を与えることなく，新システムの効果を早期に発揮させるため，段階的に本番化を行い，新旧システムが共存する形態でのシステム移行を実施した．

本稿では，その再構築開発において，システム概要と開発アプローチ，システム移行手順，システム基盤選定およびその実装について述べる．

Abstract The reconstruction of a production control system,which we implemented at the opportunity to respond to the year 2000 issues in the company ABC, was performed as the downsizing from the mainframe-based computer organization towards the open CSS. We adopted three-tears architecture as underlying CSS in the system development, and then the efficiency such as throughput in data processing and network traffic and maintenance of application systems are remarkably improved.

Also, we adopted the WindowsNT clustering technology to the nucleus system as the technological safeguards against the system failure, and the high availability has been maintained. As for the development system, we organized the six teams every role and took the organization arrangement that closed with customers by residing the major members of the teams with in the company ABC.

As for the system development management, we made an effort to grasp the information of the plan, actual outcome（proportion of work carried out）and expectation of the progress and the cost control with a single glance in order to analyze and respond to the problems in early stages.

As for the system change-over from the mainframe-organized system to the open CSS, we carried out the stepwise system migration in the environment where the new and the old systems coexist without giving confusion to the customers in order to demonstrate the effect of the new system early.

This paper describes the system overview, development approach, system change-over procedures, selection of the underlying technology of the system and its implementation method.

1. はじめに

A 社では，従来メインフレーム，オフコンをベースとした生産管理システムが稼働していた．しかし，初期開発から既に十数年たったシステムは，IT 技術面だけでなくシステム思想自体も古くなり，運用要件の変化に対応しきれなくなってきていた．さらに，A 社内部でも，競合他社よりもさらに競争力を高めるために，生産性向上に寄与するシステムが求められていた．そこで，Y 2 K 対応を機に，新たな新生産管理システムとして再構築することとなった．

再構築にあたり IT 技術やシステム思想の見直しだけではなく，ユーザ業務の仕組みの見直しも含めた業務改革から行った．

1996 年から，業務分析，基本構想のまとめを行い，1997 年 4 月から要求定義を着手．以後開発フェーズを 3 フェーズに分け，第 1 フェーズを 1998 年 5 月，第 2 フェーズを 1998 年 11 月に順次本番化し，最終第 3 フェーズの 1999 年 9 月に全面本番を迎えた．

新生産管理システムは，メインフレーム，オフコンなどで稼働していた，約 1200 本のプログラム，約 13 GB のデータベースを，WindowsNT サーバを基幹に据えたオープン系 CSS へと再構築したものである．特徴としては，3 層アーキテクチャ構造の採用，NT クラスタリングシステムの採用などが挙げられる．

本稿では，新生産管理システム開発について，システムの概要と開発アプローチ，システム基盤選定およびその実装，メインフレームからオープン系 CSS への順次ダウンサイジング方法を中心に報告する．

2. 新生産管理システム構築の目的

2.1 新システム構築の背景

旧生産管理システムは，十数年前に基本設計されたシステムであり，開発当時は革新的な思想のもとに構築されたシステムであるが，その後の生産技術の向上，経営戦略の変革などにより変化した業務運用には，段々とそぐわなくなってきていた．また，そのような変化に対応する形で様々な改善，拡張を行ってきたが，その結果，システム構成としては，メインフレームを基幹としてオフコン，UNIX など多種の機器を使用する複雑な仕組みとなっていった．

システムの機能は，定型業務のサポートが中心で，業務画面，帳票以外の情報を直接ユーザが見ることができる仕掛に乏しく，日々変化していくユーザの多種多様なニーズにタイムリーに情報提供することが困難となっていた．さらに，クライアント機は専用端末を使用しているため，一般 OA 業務との親和性がないものであった．

また，ユーザ側では，競合他社よりも競争力を高めるために，生産性向上に向けた経営戦略のもとに，生産リードタイム短縮，生産コスト削減を実現するための業務改革を検討をしており，それを支援できるコンピュータ・システムを求めていた．

このような背景のもとに，Y 2 K 問題対応を機に生産管理システムの再構築が進められることになった．

2.2 新システム構築の要件

新生産管理システムの構築にあたり，新しい業務の仕組みを支援し，旧システムの

課題の解決を図る新システムの要件は以下のものが挙げられた.

・新しい業務思想に基づいたシンプルなシステム
・メインフレーム，オフコンなどの分散された基幹業務の統合
・一般OA業務との親和性を含めた操作性の良いシステム
・業務の効率化・自動化の促進を支援するシステム
・基幹業務以外に自由に情報検索，分析ができるシステム
・24時間稼働を支援できる耐障害性の高いシステム
・保守効率の高いシステム
・業務改革の効果を早期に発揮し，Y2K問題に抵触しない移行開発

新システムは，ユーザ側で行う業務改革による新しい業務思想をベースとするとともに，統一されたアーキテクチャのもとに統合されたシステム構成が求めらた．具体的には，一般OA業務との親和性を考慮したWindowsベースのシステムの採用が求められた.

また，A社の工場は，昼夜交替勤務の形態をとっているため，システムは24時間稼働が前提となり，障害発生時には速やかに復旧される必要がある.

開発作業としては，業務改善の効果の早期発揮，Y2K問題の対応などにより，早期にシステム移行が実現できることが求められた.

3. 新生産管理システムの概要

3.1 新生産管理システムの構成

新生産管理システムの構成を図1に示す.

新生産管理システムは，8台のサーバ機と250台のPCクライアント，50台の工場現場専用クライアントなどから構成される．その概要は，生産管理の主要な業務を遂行する「基幹サーバ」を核に，生産計画系業務を処理する「生産計画サーバ」，工場での作業指示・実績収集のトランザクションを処理する「実績収集サーバ」，情報戦略業務を支援するEUCデータベースやOAドキュメントなどを保持する「EUCサーバ」などを中心におき，ドメインサーバ，メールサーバで構成されている.

「実績収集サーバ」を除いた全てのサーバは，WindowsNTを採用している.

以下に代表的なサーバの特徴を述べる.

1) 基幹サーバ（APサーバとDBサーバ）

生産管理業務において，受注管理，部品調達，作業指示票発行，原価管理など，生産計画業務，実績収集業務を除く全ての業務処理を行うメインサーバで，アプリケーション(AP)サーバ, データベース(DB)サーバの2台から構成されている.

アプリケーション構成は，VisualBasicを利用したActiveX形態の3層アーキテクチャ構造をしており，データベースにはOracleを採用している．障害対策としてクラスタリング構成を採用している点も特徴である.

また，他部門サーバとのデータ授受処理については，HULFTによりデータ交換を実現している.

2) 生産計画サーバ

生産計画業務を主に担当するサーバである．生産計画業務は，その計画レベル

によって，大きく 3 業務（中日程計画，工程計画，小日程計画）に分けられる．それぞれは，スケジューリング処理などを目的とした市販パッケージソフトウェアを利用しているため，基幹サーバとは独立した形態をとっている．

基幹サーバとのデータ授受は，Oracle のデータベース・リンク機能により，データの透過性を実現している．

3） 実績収集サーバ

工場での実際の生産作業に対する作業指示や実績情報などのトランザクションデータを処理するサーバである．作業指示情報は全て電子化しペーパーレスを実現している．工場に配置するクライアントは，一般の WindowsPC ではなく，防塵，耐熱対策を施された工場用の特殊機器を使用している．特殊機器のため，通信制御ソフトウェアに制限があり，このサーバのみ，WindowsNT ではなく，UNIX を使用している．

基幹サーバとのデータ授受は，Oracle のデータベース・リンク機能により，データの透過性を実現している．

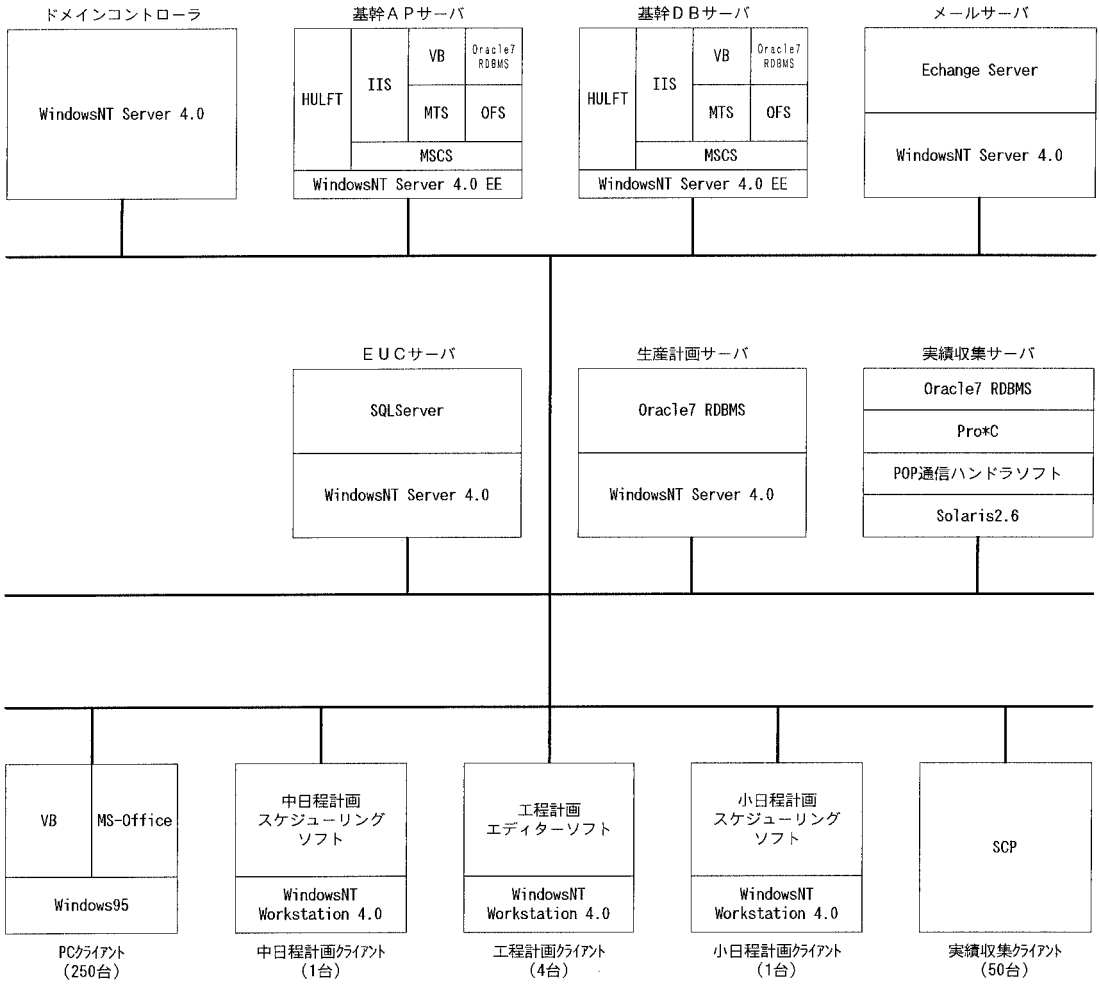

図 1　システム構成

4) EUCサーバ

生産計画情報や実績情報などのソース情報をユーザに開放するためのEUCデータベースを保持している．ユーザはこの情報を自由に検索，加工することにより，生産状況の分析などを行えるようになっている．また，各ユーザが作成したOAドキュメントなどを保管するファイルサーバとしての役割も持っている．ユーザに公開されるEUCデータは，夜間バッチによって基幹サーバからダウンロードされることにより更新される．

4. 新生産管理システムの開発手順

4.1 開発方針

今回のシステム開発は，ユーザ業務全般という広範囲に渡り，かつY2K問題への対応ということで遅延の許されないものであった．そのため，開発作業において問題を発生させない，あるいは発生する問題を最小限にとどめ，早期に対応することが要求された．

このような要件を踏まえて，開発手順として以下の方針を立てた．

・開発進捗会議を定例化し，プロジェクトメンバ全員が同じ状況認識，問題把握できるようにする

・当社開発メンバのうち要求定義・設計などの上流工程を担当するSEは，A社内に常駐し，日々ユーザ部門との情報交換を密に取ることにより，作業効率向上，開発品質確保につとめる

・当社担当のプログラム開発は，当社社内にて行い，設計担当の常駐SEが受入検証を実施した後，A社内開発環境にてテストを実施する．

・システム切替は，一斉切替を行わず，フェーズ分けをし段階的に本番化することにより，ユーザ部門の混乱を避け，Y2K問題抵触を段階的にクリアすると同時に，新システムへの業務変革を早い段階から推し進める

この様な方針のもと，以降で述べる開発体制，開発管理を実施した．

4.2 開発体制

開発体制は，次の6チーム編成にて行った．

A社側に

・システム開発・運用チーム

・ユーザ部門選任業務仕様検討チーム

当社側に

・設計担当常駐SEチーム

・プログラム開発チーム

・システム技術支援チーム

さらに，A社，当社双方の開発管理者により編成される

・マネジメントチーム

である．

それぞれの役割分担としては，まず，マネジメントチームは，開発作業を円滑にするための管理，指導などを行う．A社側のシステム開発・運用チームは，システム

化する業務の分析，要求定義，設計の上流工程からプログラム開発，テスト，導入および運用管理のシステム開発，運用の全工程を担当する．ユーザ部門選任業務仕様検討チームは，業務改革の中心的な存在で，ユーザ業務の仕組み見直し，新しい仕組みの立案，検証を行い，その結果をもとに要求定義を検討する．また，開発したプログラムの総合テストにも参加し，操作・運用手順書などの作成を行う．

当社側の設計担当常駐 SE チームは，A 社側のシステム開発・運用チームと協力，分担をして，業務分析，要求定義，設計の上流工程からテスト，導入を中心に行う．

プログラム開発チームは，設計担当常駐 SE チームの指示に従い，プログラム開発を担当する．システム技術支援チームは，新システムの基盤となる 3 層アプリケーション構造やクラスタリング・システムなどのシステム基盤の検証，導入，開発担当者への利用指導を担当する．

このような 6 チームが協力，分担を行ってシステム開発を進めた．

4.3 開 発 管 理

1） 開発スケジュール・作業進捗管理

開発スケジュール管理には，MS-Project を使用した．マスタスケジュールは，主にマネジメントチームが中心で作成し，これをもとに各チームのリーダが，詳細工程のスケジュール作成を行った．また，このスケジュールは，要求仕様などが完了し，開発範囲，仕様などが明確になった時点で，調整が行われ確定スケジュールとなる．

作業進捗管理としては，このスケジュール表上に，計画，実績（進捗率），完了見込の 3 種類の情報を記すことにより行った．また，詳細スケジュールでは，プログラム 1 本毎の工程別ガントチャート表現以外に，MS-Excel とデータ連携を行い，週別完了本数グラフを作成し，視覚的な状況表現も行った．

このような管理資料は，計画，実績（進捗率），完了見込の 3 本柱の情報を一目で把握できた点に大きな効果があった．

そしてこの情報をもとに，マネジメントチームの指示のもと，他の 5 チームのリーダーが集まり，週に 1 回の割合で定期進捗会議を実施した．また，この情報を開発プロジェクト室に掲示し，誰でも一目で状況を把握できるようにした．

この結果，現状把握，今後の作業状況推移予測が明らかになり，開発メンバ全体での状況認識の統一，課題への早期対策などが，はかられ，円滑な開発作業を実施することができた．

2） コスト管理

コスト管理は，主にマネジメントチームにて実施される．開発計画にそって，発生コストおよび，発生予測コストを管理し，月に 1 回の割合で行われるマネジメント会議にて，計画コストとの差異をもとに，課題の分析，対策立案を行う．

4.4 システム移行手順

今回の再構築作業では，システム移行は，全サブシステムの一斉切替ではなく，段階的に本番化する形態とした．これは，単に新システムの機能が段階的に拡張されていくというものでは無く，旧システムと新システムが，それぞれ役割分担をして，共存するもので，完全移行が完了するまでは，新旧のシステムが一つのシステムとして

稼働するものである．開発スケジュールとしては，大きく 3 フェーズに分かれ，そのフェーズ単位で本番化を行った．段階的に本番化することによって，旧システムの担当業務が，徐々に新システムへと移行され，最終の第 3 フェーズが完了した時点で，全システムの移行が完了となる．

旧システム全域を開発範囲として再構築したにも関わらず，この様な手順を取った理由は，

- 全システムの一斉切替によるユーザの混乱を避ける
- 段階的に本番化することにより，新業務体系への変換，効果を早期に推し進める
- 機能毎に異なる Y 2 K 問題抵触タイムリミットを段階的にクリアでき，開発にゆとりを持たせることができる
- ユーザ・インタフェースが異なるため，ユーザに早く慣れ親しんでもらうなどである．

しかし，共存運用により次のようなデメリットもあった．

- 旧システムとのデータ連携実現のため，使い捨ての機能を開発する必要があった
- 一部の業務担当者には，新旧 2 種類のシステムを使用しなければならなかった

データ連携は，基本的には自動化したが，旧システムの保持する情報では不足する情報は，新システム側で入力せざるを得ず，わずかではあるが，一部の業務担当者の業務負担増加はやむを得なかった．

結果としては，ユーザ側の理解もあり，共存運用させることで，大きな混乱を発生させることもなく，スムーズなシステム移行が実施できた．

4.5 システムテスト

システムテストは，開発用サーバ環境にて実施した．この開発用環境は，本番環境と同様の環境を備えており，本番使用時と同じ 3 層アーキテクチャ構造での実行が可能になっている．旧システムが稼働するメインフレームにおいても，本番環境と同じ構成の開発用データベース，アプリケーション環境を備えている．

この開発環境上にてシステムテストを実施することにより，新旧システムの共存を含めた形で，本番業務にまったく支障を与えることなく，本番業務を想定したテストを実施した．

5. 新生産管理システムの実装技術

5.1 システム実装方針

新生産管理システム構築にあたり，2 章で述べた要件を実現するために以下の方針を立てた．

- システム構成は，基幹業務と OA 業務を含めた統合環境として Windows ベースの CSS 形態とし，応答性が高く保守効率が高いシステムとする
- 24 時間稼働に対応するため，高可用性を実現できるクラスタリング・システムとする
- 新システムを段階的に本番化するため，新旧システムの共存・連携環境を提供

する

・開発言語，データベースなどは，マーケット・シェアが高い VisualBasic，Oracle などを採用する

・開発にあたり，設計・開発標準を規定し，成果物の品質を高いレベルで一定に保つ

・基幹業務以外に自由に情報検索，分析ができるシステムとして EUC データベースを構築する

・クライアント，アプリケーション管理を効率化するための運用支援機能を提供する

この様な方針のもと，以降で述べる実装技術を採用した．

5.2 3層アーキテクチャ構造の実装

新生産管理システムでは，Windows ベースの CSS 形態でのアプリケーション構築の採用と，応答性が高く保守効率が高いシステムが求められていた．

CSS においてレスポンスに関してボトルネックに陥りやすいのは，クライアントとサーバ間でのデータ転送である．このデータ転送量，回数を最小にできればレスポンス向上になるはずである．通常，クライアントとサーバ間でデータ転送量，回数が多い理由は，クライアント側にビジネスロジック制御があることが主原因と考えられる．そこで，このビジネスロジック制御の部分をクライアントでは無く，サーバ側へ移した場合を考えてみた．この場合，クライアントとサーバ間でのデータ転送は，理論的には，処理のキーになる情報と処理結果だけとなり，ネットワーク上のデータ転送量，回数は圧倒的に少なくすることができる．また CPU やメモリなどのマシンスペックについては，処理能力の高いサーバがあればよく，クライアント側の能力にはあまり影響されない．さらにクライアント側に DB アクセスのためのミドルウェアをインストールする必要はなくなる．また，アプリケーションの改修についても，ビジネスロジック制御部分はサーバ側に配置されているので，ほとんどの改修はサーバ側のみでクライアント側に再配布する頻度は少ない．

このような考えから，GUI 層はクライアント側，ビジネスロジック層とデータ層をサーバ側に配置する，3 層アーキテクチャ構造を採用した．

最終的に採用された 3 層アーキテクチャの構造が図 2 である．

開発言語としては，VisualBasic を採用した．VisualBasic は，CSS 形態におけるフロントエンドツールとしては，デファクトスタンダード的であり，ActiveX 技術が搭載されたことにより，クライアントとサーバとで同じ言語でリモートアプリケーションの開発が可能であった．

この 3 層アーキテクチャでの特徴は，サーバ側アプリケーションの実行管理に，MTS（Microsoft Transaction Server）を採用した点である．MTS を採用した理由は，サーバ側アプリケーションで消費されるメモリ量の低減である．

VisualBasic のリモートアプリケーションでの実行形態は, ActiveX の EXE 形式(以下 ActiveX EXE）と DLL 形式（以下 ActiveX DLL）の二つの形態がある．VisualBasic の標準機能で提供されるリモートマシン間での接続は，リモートオートメーションと呼ばれる手法で，この場合，クライアントから参照できるサーバ側アプリケーシ

ョンは，ActiveX EXEだけであった．しかし，ActiveX EXEの場合は，アウトプロセスとして実行されるため，メモリ消費量が多いという問題があった．単一マシンでの評価テストの結果，インプロセスとして実行されるActiveX　DLLの場合は，ActiveX　EXEに比べ，メモリ消費量を1/5程度に押さえられることが判っていた．そこで，Web形式での3層アーキテクチャの構成例をヒントに，サーバ側アプリケーションを，ActiveX DLLとして，MTS上のパッケージコンポーネントとして登録し，クライアントからDCOM（Distributed　COM）を利用して接続する形態をとることにより，消費メモリ低減を実現した．

また，UNIXサーバを利用している実績収集系業務についても，基盤ソフトウェア，プログラム言語などは異なるものの，同様に3層アーキテクチャ構造となっている．

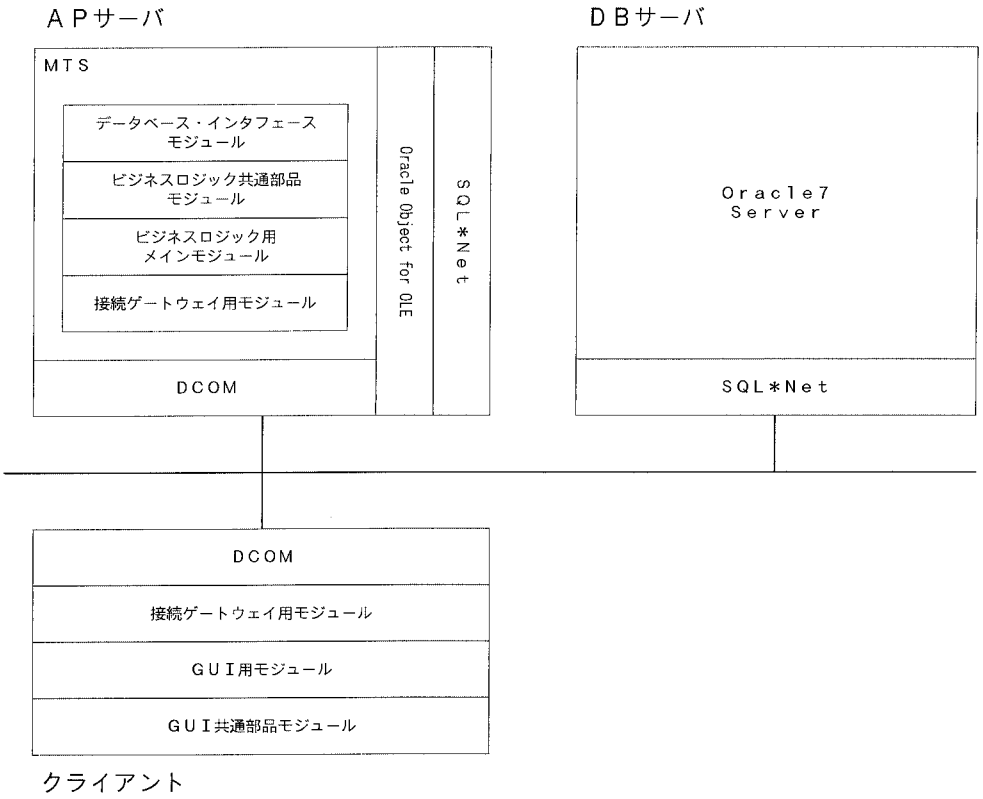

図 2　3層アーキテクチャ構造

5.3　高可用性の実装

新生産管理システムの要件の一つとして，24時間稼働がある．これは，障害発生時にシステムダウンしない，または，最小限のダウン時間で復旧できる事を要求している．

メインフレームで稼働していた旧生産管理システムでは，メインフレームのホットスタンバイ機能によって，高可用性を実現していた．新生産管理システムでも，その思想は継承される必要があり，NTクラスタリング技術を採用することとなった．しかしながらNT上でのクラスタリング技術は，開発当時は未知数の面があったため，

第 1 フェーズ本番（1998 年 5 月）での，適用は避け，評価テスト実施の後，第 2 フェーズ本番（1998 年 11 月）にあわせて適用することとした．

NT クラスタリングの構成としては，MSCS（Microsoft Cluster Server）をベースにおき，Oracle データベースのクラスタリング制御として，OFS（Oracle Fail Safe）を使用する構成とした．

検討の段階では，前出の構成の他に，MSCS と OPS（Oracle Parallel Server）の組み合わせと，OPS 単独で使用する場合の 3 種類の選択肢があった．この 3 種類の構成についての比較が表 1 である．

OFS は，MSCS 上で稼働する事を目的として開発されたツールで，OPS は，すでに UNIX に実装されているパラレルデータベース技術の NT 版である．

OFS は，単一ノード上で稼働するデータベースファイルと Oracle インスタンスを障害発生時に別ノードへ接続し直して処理を継続するもので，同一時点では，1 ノードからのみアクセスが可能である．OPS は，別ノード上で稼働する別々の Oracle インスタンスから共有データベースファイルを使用するもので，障害発生時に障害インスタンスを切り離し，正常インスタンスのみで処理を継続する．同一時点では，複数ノードからのアクセスが可能である．しかしながら，MSCS には，共有ディスクは同一時点では単一ノードからしかアクセスできないという制限がある．そのため，MSCS 上にて OPS を稼働させても，本来のパラレル処理による負荷分散機能を利用できないため，OFS に対する優位性は無いと判断できる．また，OPS 単独で構成する場合は，データベースのリカバリ機能しかないため，サーバ側アプリケーションや共有ディレクトリサービスなどデータベース以外の機能の障害対応ができない，という問題があった．比較検討の結果，全てを MSCS 上で管理でき，データベース以外のアプリケーションなどにも対応できるという点から，MSCS と OFS の組み合わせを基本構成として，クラスタリングの適用評価作業を行うこととした．

サーバ上での処理は，3 層アーキテクチャのうちのビジネスロジック層にあたるアプリケーション処理とデータベース処理と大きく二つに分けることができるが，結果的には，MSCS と OFS の組み合わせにしたことにより，各ノードの役割が明確になりシンプルな構成がとれる事になった．

評価作業は，約 3 ヶ月間かけて行ったが，評価ポイントは，

- 障害時の全体的なフェールオーバの動きの確認
- 更新データの状況
- 3 層アーキテクチャのフェールオーバ方法
- サードパーティソフトウェアのフェールオーバ方法
- クライアントからの再接続方法

などである．

このうち，3 層アーキテクチャとサードパーティソフトウェアのフェールオーバ対応を除く部分は，MSCS, OFS などで標準に提供される機能のための動き，確認が中心であった．実際のフェールオーバ処理も，障害検知後速やかに行われるため，本格的なフォールトトレラントシステムの様に無停止と言うわけには行かないが，最小時間のダウンタイムで業務復旧が可能となり, 当初要件をクリアできるレベルとなった．

問題は，MSCS 対応として開発されていない，3 層アーキテクチャ・アプリケーションやサードパーティソフトウェアの対応であった．これは，MSCS がノード情報の引継として，各ノードの仮想アドレス情報をフェールオーバする点に着目し，両ノードに 3 層アーキテクチャのベースとなる MTS やサードパーティソフトウェアをインストールしておき，クライアントから仮想アドレス接続する事により，正常時には，片ノードはバックアップ用にダミー稼働させ，障害時に仮想アドレス情報がフェールオーバされた時点で通常稼働とする方法で解決した．

この評価作業により，3 層アーキテクチャをベースとしたサーバ機能のクラスタリング機能は，第 2 フェーズ本番（1998 年 11 月）にあわせてリリースを行った．

5.4 新旧システムの共存運用環境

新システムを段階的に本番化を行い，旧システムと共存するためには，新システムと旧システムとの間で，データ交換が必要になる．このデータ交換には，UNIshuttle, HULFT などの分散環境支援ツールを組み合わせて実現した．

旧システムでは，メインフレーム，オフコン，UNIX など複数機種を利用していたが，UNIshuttle を利用して，異機種間でのデータ転送，リモートジョブ機能などによりデータ連携の自動化が行われていた. 新システムと旧システムとのデータ交換も，自動化する必要があった．しかし，それまで使用していた UNIshuttle は，新システムのサーバである WindowsNT 環境では稼働できなかった．この異機種間連携機能を独自開発するアイデアもあったが，この機能自体が完全移行が完了するまでの一時的なものであり，独自開発するには，開発期間，コストの面からあまりにもリスクが大きかった．

また，新システムでは，他部門サーバとのデータ交換のために，HULFT の使用が決定していた．HULFT は，UNIshuttle 同様に，異機種間でのデータ転送，ジョブ実行機能を持ったツールである．ただし，HULFT は，WindowsNT，UNIX では稼働できるが，メインフレームでは，稼働できなかった．そこで，メインフレームと WindowsNT サーバの間に UNIX を中継サーバとして，UNIshuttle と HULFT のコマンド変換を実施することにした．メインフレームから WindowsNT への指令は，メインフレームから UNIshuttle による指令を中継 UNIX サーバへ引き渡し，中継 UNIX サーバはこれを HULFT の指令に変換して，WindowsNT サーバへ引き渡し，データ転送，ジョブ実行を実現する形となる．WindowsNT サーバからメインフレームへの指令は，この逆である．

この仕組みにより，自動データ連携が可能になり，障害も発生することなく，新旧システムの共存運用を実現することができた．

5.5 設計・開発標準の規定

今回の再構築では，プログラム成果物の品質を高いレベルで一定に保つために，設計・開発標準の規定を行っている．この設計・開発標準は，3 層アーキテクチャ構造に基づいたものでり, コーディング規約など以外に, データベース・インタフェース・モジュールなど各種の共通部品モジュールを提供している．

このデータベース・インタフェース・モジュールは，データベースへの接続，切断をはじめ，一般的な，検索，追加，更新，削除から，ストアド・プログラムの実行，

表 1　クラスタリング・ツール比較

		MSCS+OPS	MSCS+OFS	OPS
クラスタ構成	最大ノード数	2	←	4
	共有ディスク	シェアードナッシング	←	シェードエブリシング
フェールオーバ可能なプロトコル		TPC/IP、NetBIOS	←	×
フェールオーバ可能なアプリケーション		Oracle データベース IIS MTS（MSDTC） 他　Cluster　API　が利用できるもの	←	Oracle データベース
フェールオーバ条件	ノード障害	○	←	○
	特定AP障害	○	←	△（Oracle のみ）
フェールオーバ単位		リソースグループ	←	Oracle インスタンス
データベース実行単位		アクティブ／スタンバイ （片側は待機）	アクティブ／アクティブ （複数インスタンスで独立） または，アクティブ／スタンバイ （片側は待機）	パラレル （複数インスタンスで共有）
フェールオーバ手順		再起動	←	インスタンスリカバリ
ﾌｪｰﾙｵｰﾊﾞ時のｸﾗｲｱﾝﾄ再接続		単純再接続	←	正常ノードを明示的に指定して再接続
長所		・MSCS の機能を利用するため DBMS だけでなく他の AP も含めｻｰﾊﾞﾄｰﾀﾙでｸﾗｽﾀﾘﾝｸﾞが実現できる ・ｱﾄﾞﾚｽ情報のﾌｪｰﾙｵｰﾊﾞが可能なためｸﾗｲｱﾝﾄ側で再接続の考慮不要	・MSCS の機能を利用するため DBMS だけでなく他の AP も含めｻｰﾊﾞﾄｰﾀﾙでｸﾗｽﾀﾘﾝｸﾞが実現できる ・MSCS で全て管理されるためｼﾝﾌﾟﾙ ・ｱﾄﾞﾚｽ情報のﾌｪｰﾙｵｰﾊﾞが可能なためｸﾗｲｱﾝﾄで再接続の考慮不要	・全ﾉｰﾄﾞで DB，AP が稼動するため負荷分散が可能
短所		・ClusterWare が必要な上、ｼﾝｸﾞﾙ DB のため OFS より複雑な割に機能が劣る ・MSCS の制約により，複数インスタンスでの共有 DB 化が出来なくなり OPS のメリットが無くなる	・ﾌｪｰﾙｵｰﾊﾞのみのｻﾎﾟｰﾄのため DBｻｰﾊﾞとしての負荷分散が出来ない	・Oracle 以外の AP のﾌｪｰﾙｵｰﾊﾞ不可 ・障害時ｸﾗｽﾀ再構築の間，一時的に全ｲﾝｽﾀﾝｽが停止する ・ｱﾄﾞﾚｽ情報のﾌｪｰﾙｵｰﾊﾞが不可のためｸﾗｲｱﾝﾄ側で再接続の考慮が必要 ・PCM ﾛｯｸ制御のｵｰﾊﾞﾍｯﾄﾞが高く，効率の良い形態に設計・ﾁｭｰﾆﾝｸﾞするのに高ｽｷﾙが必要
拡張性		・今後の MSCS の機能向上に依存	・今後の MSCS の機能向上に依存	・現時点では 4 ﾉｰﾄﾞまでだが UNIX と同様の進歩を遂げれば拡張性は高い

DDL 文を含めた任意の SQL 文操作が行えるメソッド群をパッケージしたものである．各ビジネスロジック層からのデータ操作は，すべてこのデータベース・インタフェース・モジュールを使用して行われる．他には，一般的なマスタ検索，エラーチェックやメッセージ送信等の共通部品がある．これらの共通部品モジュールは，全て ActiveX DLL として MTS 上に登録される．

実際の業務プログラムは，これらの共通部品モジュールを組み合わせることによって，統一された規定のもとに開発されることになる．

5.6 EUC データベースの構築

新システムの要件である基幹業務以外に，自由に情報検索・分析ができるシステムを実現するために，EUC 支援サブシステムを構築した．このサブシステムは，基幹系サーバとは，別の独立した EUC サーバにデータベースを構築している．この業務では，計画・実績などのソース情報を業務分析を主目的に情報検索するもので，一度に大量データのアクセスやソートが行われるため，基幹系業務への影響を与えず，基幹系サーバの処理能力を一定に保つために，独立構成を取っている．

また，このデータベースには SQLServer を採用している．基幹系データベースに Oracle を採用しているにも関わらず，SQLServer を採用した理由は，クライアント管理コストの軽減である．

EUC 業務では，各ユーザが，MS-Excel と MS-Query の組み合わせで自由にデータ検索，加工を行う仕組みになっている．データベースとして Oracle を利用した場合は，SQL*Net などのミドルウェアを各クライアントにインストールする必要がある．SQLServer の場合は，ODBC 環境があれば接続可能であり，ミドルウェアのインストールが必要な Oracle に比べ, クライアント管理コストを抑えることができる．

5.7 データベース開発支援

データベースには，基幹系には Oracle，EUC 系には SQLServer を採用しているが，その開発支援ツールとして ERwin を使用した．以前は，スキーマ構成やテーブル定義等のドキュメント情報とデータベース構築ツールなどが，それぞれ独立していたため，初期設計時の定義変更や運用開始後の維持管理などに煩わしさがあった．

今回，論理設計段階から ERwin を使用することにより，データベース・スキーマを一元管理でき，物理設計後に，データベース構築用スクリプトを生成すれば，すぐにデータベース構築が可能となった．

5.8 クライアント管理

クライアント管理は，主にインストール済プログラムの調査とプログラム配布が中心で，それぞれ自作のツールを使用し，ログオン・スクリプトとの組み合わせで実現している．インストール済プログラムのバージョン調査を行い，プログラム配布が必要なクライアントには自動コピーを行い，各クライアント毎のインストール済プログラムのバージョン情報を管理ファイルとして出力する仕組みになっている．

5.9 ジョブ実行制御

ジョブ実行制御としては，主にサーバ側でのバッチジョブの起動および次回の起動スケジュールの設定を行っている．当初は，前節のクライアント管理とともに，市販の統合運用管理ツールの適用も検討したが，バッチジョブの実行は，一般的なカレン

ダーではなく，工場カレンダーをもとにスケジュールされる必要があり，休日の扱いも機能毎に異なる．さらに日によって実行回数が異なるものもあり，市販ツールでは全要件を満たすことは困難でり，自作ツールにより行っている．

ジョブ実行制御機能は，次の 4 モジュールから構成されている．

・ジョブ登録モジュール（実行制御するジョブおよび実行条件を登録する）
・ジョブ起動監視モジュール（時刻監視を行い該当ジョブを起動する常駐モジュール）
・次回起動スケジュール算出モジュール（次回起動時刻を算出しジョブを再登録する）
・処理結果通知モジュール（ジョブの処理結果を業務担当者にメール通知する）

このジョブ実行制御機能も，クラスタリング対応アプリケーションとして開発されている．

6. お わ り に

今回の再構築では，メインフレームからオープン系 CSS へと大きな変革を行った．その中で，VisualBasic による 3 層構造や NT クラスタリングなど，当時としては実績の少なかった IT 技術の採用を行った．アーキテクチャも大幅に変更した広範囲の開発を順調に完了できた要因としては，主力メンバを常駐体制とし，常にユーザ部門との情報交換を密にとれる状況にしたことにより，素早いアクションがとれ，正確な作業の遂行ができたと考える．また，オープン系へ移行したことにより，OA 業務との親和性が高まり，ユーザ部門でのデータ活用率が飛躍的に向上した事は，大きな効果と考える．

最後に初期構想から 4 年余りもの期間，絶大なるご協力ご支援をいただいた A 社関係各位の皆様に心より感謝申し上げる．

執筆者紹介 村　上　真　二（Shinji Murakami）
1990 年名古屋市立大学卒業．同年，日本ユニシス(株)入社．製造業分野でのシステム開発に従事．現在，中部支社 I&C システム室に所属．