レガシー基幹システム移行
マイグレーションによる実例について

トナミ運輸株式会社

■ 執筆者Ｐｒｏｆｉｌｅ ■

横山　徳彦

2007 年　トナミ運輸株式会社入社
　　情報システム事業部配属
2014 年　基幹システム更新プロジェクト
　　参画（バッチ系リーダ）
2016 年　基幹システム更新プロジェクト
　　プロジェクトマネージャ
2017 年　現在 情報システム事業部
　　情報管理部　部長代理

■ 論文要旨 ■

当社は１９７２年以来４０余年にわたり改修を重ね、保守・運用を行ってきた基幹業務を担うメインフレームレガシーシステムを約３年半かけオープン系サーバシステムにマイグレーションした。マイグレーションの目的は「ＩＴコスト削減」「システム拡張性」であり、このマイグレーションを戦略的基幹システム構築のスタートと位置付けている。

マイグレーションを実施する上で通常発生する課題に加え、基幹システムや連携する周辺システムの固有要件に起因する多くの課題が発生した。

本論文ではマイグレーション決定に至る経緯とマイグレーション推進工程での発生した主要な課題と解決方法について述べる。本来、稼働後に不要となる「データ移行機能」「テストデータ生成系の開発」「オンライン機能マイグレーション」「周辺システムとの連携互換機能の実現」を重点的に述べる。

本事例がこれからマイグレーションを実施するプロジェクトの参考になれば幸いである。

■ 論文目次 ■

■ 図表一覧 ■

１．はじめに

１．１　当社の概要

当社はトナミホールディングスグループの中核会社であり、物流関連事業（貨物自動車運送事業、貨物利用運送事業、倉庫業及び港湾運送業）を主力としている。

北陸・関東・関西・山陽を結ぶ路線をグループで構築、他地域は地元運送業者と提携して全国ネットワークを構築している。

トナミホールディングスグループとして 51 拠点 664,661 ㎡の総保管面積を保有している。

【企業情報】

・創業　　1943 年 6 月
・資本金　141 億 82 百万円
・従業員数 6,494 名（2017 年 3 月末時点）
・事業セグメント
　貨物自動車運送事業・貨物利用運送事業、倉庫業・港湾運送事業・情報処理事業
　自動車修理業・物品販売並びに委託売買業、総合リース業･旅行業 他

【平成 29 年 3 月期実績】

・営業収益　125,509 百万
・営業利益　　5,118 百万
・経常利益　　5,383 百万

【情報システム部門概要】

・主要業務
　社内システム開発・保守・運用、インターネットデータセンター（ＩＤＣ）運営
　インターネットプロバイダ事業運営
・所属人数　５３名（ライン管理　３名、システム開発　２０名　ＩＤＣ運用　１０名
　　インフラ導入・保守　３名　社内向けカスタマサービス　９名
　　管理業務　４名）

１．２　基幹システムの歴史と移行方式の選定

当社の物流サービスをＩＣＴ面から支えている基幹システムは１９７２年（昭和４７年）にメインフレームによる業界初のオンラインシステムを先駆けて稼働させ、以来４０余年にわたり技術進歩に合わせながら機能の拡充を行い運用してきた。

将来性を考えた場合、メインフレームによる基幹システムはコスト面、機能拡充面の両面で限界があり、早急にオープン系システムへの移行が必須となっていた。

オープン系システムへ移行を行う場合、「業務パッケージ適用」「自社開発による再構築」「現状機能をそのままオープン系システムへ移行するマイグレーション」が代表的である。

業務パッケージ適用、自社開発による再構築は業務要件について、それぞれ「適用度合をみるための Fit&Gap とカスタマイズ」「業務要件定義とそれに基づいての基本設計～運用テスト」が必要であり特に初工程である「Fit&Gap」「業務要件定義」については難航が予測された。

この理由は業務部門が中心となる業務要件策定と実現機能決定において業務多忙によるリソースの捻出困難や業務要件スキル保持者の確保が困難であった。更に、いささか古典的であるが要件定義～運用テストまで１キロステップ／月の生産性で試算すると約 17,000 人月となり現実的に不可能であると判断した。（※資産規模を**表１**に示す。）

そのため現行の業務アプリケーションの機能を変更することなく業務要件を含めて、そのままオープン系システムへ移植するマイグレーションを選定した。

資産分類			本数	ステップ数（単位：キロ）
COBOL	プログラム	オンライン	386本	2,396
		バッチ	7,948本	14,014
		サブルーチン	686本	585
		プログラム計	9,020本	16,995
	COPY句		10,201本	1,351
JCL（ジョブ制御言語）			78,897本	6,387

※移行開始時点の資産規模。アセンブラ等の言語も少数存在するが記していない。

表１ 業務アプリケーション資産規模

１．３　基幹システム概要と移行範囲

基幹システム概要と移行範囲を**図１**に示す。

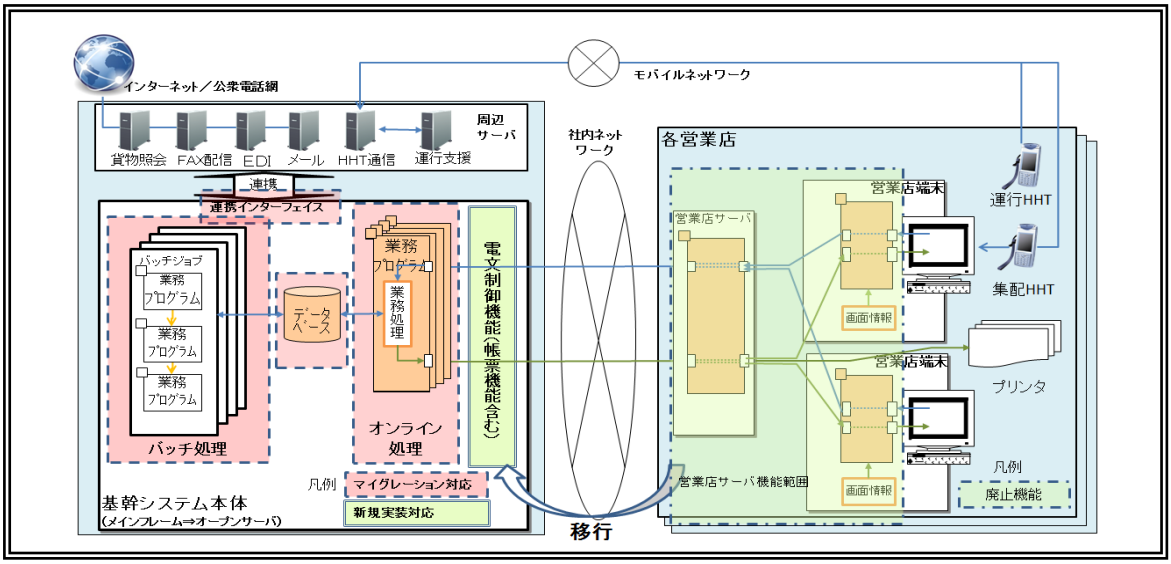

図１ 基幹システム概要（マイグレーション範囲）

マイグレーション対応としては「バッチ処理」「オンライン処理」「データベース」「連携インターフェイス」の範囲がある。

従来の「営業店サーバ」とそれに接続されていた「専用オンライン端末（営業店端

末）」は基幹システム側に「画面制御機能」「帳票印刷機能」を実装したことによって、廃止機能となった。この結果、マイグレーション後の基幹システム概要を図２に示す。

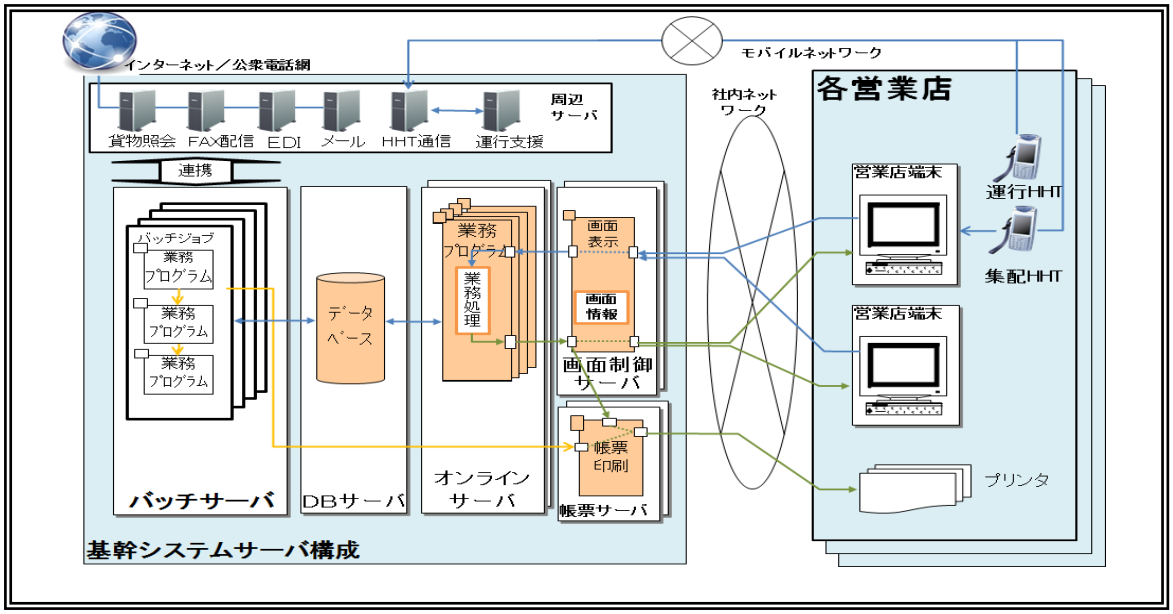

図２　基幹システム概要（マイグレーション後）

メインフレーム本体と営業店サーバ構成からバッチ、データベース、オンライン、画面制御、帳票印刷の各サーバの構成とした。バッチサーバは仮想環境上で稼働させ障害時には別仮想環境上で稼働させるよう障害対策を行っている。データベースサーバは障害発生時に予備システムへ切替を行う障害対策を行っている。オンライン、画面制御、帳票印刷の各サーバは負荷分散と障害対策の両方を目的とした仮想環境上で各複数環境の稼働を可能にしている。従来、各営業店に営業店サーバを配置し専用エミュレータを備えた営業店端末を接続し運用していた。今回「画面制御機能」を実装したことによって標準装備のブラウザ（インターネットエクスプローラ）を利用し、どのパソコンからでも入力を可能にした。

今回の移行スケジュールの概略を図３にプロジェクト体制を図４に示す。

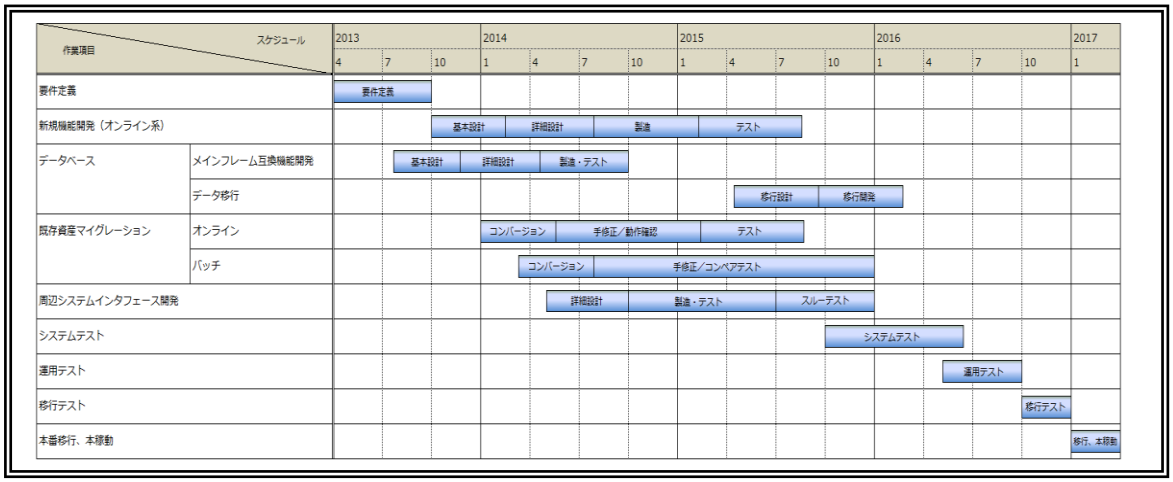

図３　概略の移行スケジュール

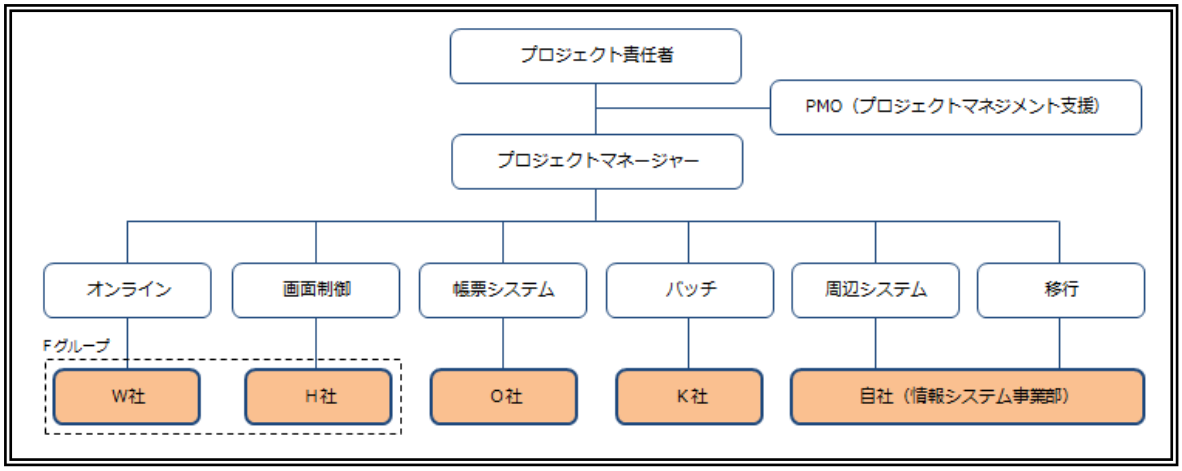

図４ プロジェクト体制

２．各機能の移行概要

各機能について移行の概要を示す。

２．１　バッチ処理

バッチ処理で実行する業務プログラム（以下、バッチプログラム）の大半がＣＯＢＯＬで開発されており、通常の手法であるソースコードの一括変換を行った後、非互換部分を手修正で対応した。

バッチ処理を行わせるためには一般的にメインフレームではジョブ制御言語（ＪＣＬ）によって一連のバッチプログラム処理を組合せジョブという単位で処理をさせる。オープン系サーバで同等機能はシェルと呼ばれるバッチプログラム処理を記述して、ジョブ相当の動作を行わせることが可能である。但し、ＪＣＬからシェルへの完全な変換は不可能であり、変換後に手修正を行う必要があった。更にＪＣＬはプログラム本数に対し１０倍の資産があり、すべてを手修正することは不可能であった。そのため、ＪＣＬの主要な機能である業務プログラムを実行させる構文、入出力ファイルを定義する構文などを単純な変換だけでメインフレームと同様に処理できる機能を持ったプログラムを開発し対応した。この対応から、シェルを実行することでメインフレームと同等の処理をさせることが可能となった。ジョブ間の実行関係を制御する機能はオープン系システムのジョブ実行管理用ミドルウェア(販売元：富士通株式会社　製品名：Systemwalker Operation Manager)を利用して実現している。

２．２　オンライン処理

オンライン処理の業務プログラム（以下、オンラインプログラム）もバッチ処理と同様にＣＯＢＯＬで開発されている。

メインフレームでのオンライン処理は営業店端末からの画面入力データが営業店サーバで処理され、社内ネットワークを経由し入力電文としてオンラインプログラムに渡される。その入力電文に基づきオンラインプログラムで処理された結果を出力電文として、営業店サーバで画面構成処理を行い営業店端末に表示される方式となっている。基本的に一問一答の形式であり、オンラインプログラムの開始時に入力電文を受取、処理終了時に出力電

文を発行して終了する。

そこで、オンラインプログラムの入力電文を受取る部分と出力電文を発行する部分のみを機械的にオープン系仕様に適合するよう修正を行った。それによって業務処理部分はそのままの機能にて利用することができた。

またオープン系システムのオンライン用ミドルウェア（Ｗｅｂ／ＡＰサーバ）においてＣＯＢＯＬプログラムの実行を可能とする機能が提供されていたこともマイグレーションを選択した大きな理由となっている。（販売元：富士通株式会社　製品名: Interstage Business Application Server）

画面を組み立てる機能、画面入力データを取込む機能は後述の「２．５　画面制御機能」にて説明する。

２．３　データベース

データベースについては、データベーステーブル構成移行とデータベースアクセス機能移行の二つがある。今回、データベースはＯｒａｃｌｅ社の Oracle Database 11g Release 2. Enterprise Edition を採用している。

(1).データベーステーブル構成移行

メインフレームのデータベースは階層構造をもつネットワークデータベースであったが、オープン系サーバではリレーショナルデータベースになる。

テーブルの列構成は基本的にメインフレームの列構成と全く同じ構成とした。

メインフレームの階層構造と同等構成を実現するため、リレーショナルデータベースに対して擬似的階層構造の構成をもたせて移行を行っている。

(2).データベースアクセス移行

メインフレームにおいてはデータベースアクセス用にＣＯＢＯＬの標準仕様にない独自構文を用意し処理する例が多いが、当該メインフレームにおいてはＤＢアクセス用のサブルーチンが提供され、そのサブルーチンを用いてアクセスするプログラム（バッチ、オンライン共）となっていた。

そこで、そのサブルーチン相当の機能を実装することによってデータベースアクセス部分のコードを書き換え不要にしている。

２．４　連携インターフェイス

図１にて連携する周辺サーバの主要なものを示しているが、実際に連携しなければならないサーバは３０システム以上に及んでいる。その連携方法の大部分はファイル転送（ＦＴＰ）によって実現している。メインフレームではＦＴＰによるファイル転送をバッチプログラムからＦＴＰサブルーチンを呼び出すことによって機能実装していた。しかしこのようなＦＴＰサブルーチンはオープン系においては標準提供されていない。

そこで、ＦＴＰサブルーチンと同様な機能を直接、ＦＴＰにてファイル転送せずファイル転送内容をほかのプロセスに間接的に依頼することで同等な機能を持つサブルーチンを実装することで実現した。

このことからバッチプログラム側の連携インターフェイスに対応する修正も必要なくなりかつ連携する周辺サーバ側の変更対応も最小限となっている。

２．５　画面制御機能

画面制御機能はメインフレーム時の営業店サーバに置換えのため実装した機能である。

営業店サーバは営業端末（パソコン）内の連携ソフトウェアと通信して基幹システムの表示要求に応じて画面を組み立て営業端末に画面を表示させ、画面入力データを取り出して基幹システムへ電文送信する機能を持つ。画面定義体と呼ぶ業務画面ごとの画面構成情報を元に基幹システムからの送信電文を含めて画面を構成し営業店端末に表示している。

各営業店に設置されている営業店サーバ機能は基幹システム側に実装した。画面定義体も一括変換を行うことでオンライン画面の定義再作成も不要としている。どうしても発生する位置ズレの軽微な修正に留めている。

この機能実装によって従来と同等の画面をパソコン標準ブラウザ（インターネットエクスプローラ）上での操作を実現している。

このことは営業店サーバにおいて接続営業店端末数の制限が撤廃され、どのパソコンからでも利用可能となり利便性の向上も実現している。

３．移行課題とその解決

３．１　主要課題一覧

移行課題はマイグレーション起因による課題、技術的な課題の二通りからなり、主要課題について**表２**に示す。これらの主要課題に対し、その概要と解決策を以降に説明する。

分類	課題
マイグレーション	(1)バイト構成するビット数が同一でないこと (2)２進表現バイト単位が任意のバイト数で構成可能なこと (3)１ビット単位での２進数構成が可能なこと (4)ＣＯＢＯＬ機能である領域再定義対応課題 (5)データ移行に関しての課題 (6)基幹システム改修内容の移行側システム反映課題
技術的	(1)オンライン用テストデータ作成課題 (2)オンラインテスト課題 (3)周辺システム連携テスト課題 (4)本番移行課題

表２　マイグレーション起因・技術的課題に対する主要課題

３．２　マイグレーション起因課題とその解決

マイグレーション起因による課題の多くはメインフレームとオープン系サーバ間のアーキテクチャが異なることから発生している。その主要な課題と解決方法について述べる。

３．２．１　バイト構成ビット数が同一でない

メインフレームにおいて１バイトが９ビットで構成されるアーキテクチャーであった。一般に１バイトは８ビットで構成されている。この違いによる影響は２進数項目を取り扱

う場面で発生する。２進数項目は２、４、８バイトで構成されビット数はそれぞれ 16, 32, 64 ビットである。一方メインフレームでは 18, 36, 72 ビットとなり２バイト構成で比較するとメインフレームでは１０進数表現で有効桁が５桁であるが８ビット構成では有効桁は４桁である。同様に４バイトでは１０桁と９桁、８バイトでは２１桁、１９桁となる。このことは２進数項目については単純なマイグレーションができないことを意味する。解決策については以降の２進数項目課題のまとめとして示す。

３．２．２　２進表現バイト単位が任意のバイト数で構成可能

直前に述べているように一般に２進数項目は２、４、８バイトであるがメインフレームでは１バイト単位で決定される仕様となっている。例えば数字２桁以下の項目と定義すると１バイト（９ビット）が割当てられ、数字８桁の項目を定義すると３バイト（２７ビット）が割当てられる。

３．２．３　１ビット単位での２進数構成が可能

２進数項目関連の課題として最後にメインフレームでは更に明示的にビット数を指定して１ビット単位で定義することができるようになっていた。これら一連の２進数項目課題に対しすべてマイグレーション先を外部１０進数項目と称する１桁を１バイト（文字）で表現する項目属性で置き換えることで解決策とした。しかし２進数項目の課題は解決するがデータ移行に２進数表現から１０進数表現に変換しなければならないことや単体テストの実施において課題が発生することが次項で説明する領域再定義課題につながっている。

３．２．４　ＣＯＢＯＬ機能である領域再定義対応

領域再定義とはプログラム内の記憶領域、ファイルシステム、メインフレームの階層データベースに適用されているものである。

この機能を利用する最大の目的は使用領域の節約である。過去においてメモリやディクス装置が非常に高価であった時代にコスト抑制のために利用された。

１９９０年代はじめ頃のメモリ価格は１メガバイト１万円であり現在の１ギガバイトは 1 千円である。実に 1/10, 000 の低価格になっており同一価格では 10, 000 倍の容量を用意することができる。その時代に作成されたプログラムがそのまま継続使用され、またファイルやデータベースについても同様に運用されていた。この負の遺産をそのまま引き継いでおり前項に述べた２進数項目のマイグレーション対応によって新たな課題となって発生している。本項の課題を図５に示す。

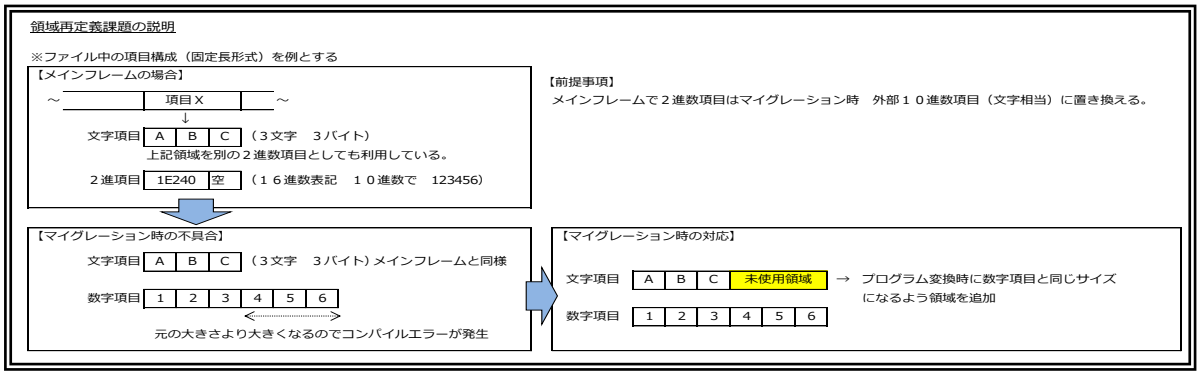

図５　ＣＯＢＯＬ領域再定義課題対応

ここでの問題点は再定義した領域長が再定義される領域より大きくなるケースが発生し文法上でエラーとなってしまうことである。２進数項目として領域を再定義する場合マイグレーションでは２進数項目は外部１０進数項目となるため再定義する項目は例えば２バイトの２進数項目は６桁（バイト）の外部１０進数項目となり、再定義側の領域が大きくなりエラーとなる。

これについては再定義されている場合マイグレーション後再定義される領域より再定義の領域が大きくなる場合は再定義される領域側に同一の大きさとなるように未使用領域を追加するようにコンパージョンするプログラムに機能追加し対応した。ただし、１００％変換はできずエラーとなった場合は手修正にて対応した。

３．２．５　データ移行に関しての課題

ここまでは１バイトの構成ビット数が異なることでの課題について述べてきたがこの課題はデータ移行に際しても大きな課題であった。メインフレーム側からファイルを取り出す場合にファイル転送（FTP）によってオープン系システムへ取込む場合を考えるとそのままでは文字データはよいが２進数項目のデータは１バイトが９ビットであるため最上位の１ビットが消失して転送されるため正しく移行できない。９ビットを含めた形式での転送も可能であったがオープン系システムで復元するためには内部構造を解析しなければならず現実的ではなかった。また、オープン系システムで使用するファイルのレイアウトに合わせて移行データをメインフレーム側で作成しファイル転送する方法も検討されたが、そのための移行用プログラムをファイル数分（8,742）作成する必要があり現実的でない。この解決方法の概要を図６に示す。

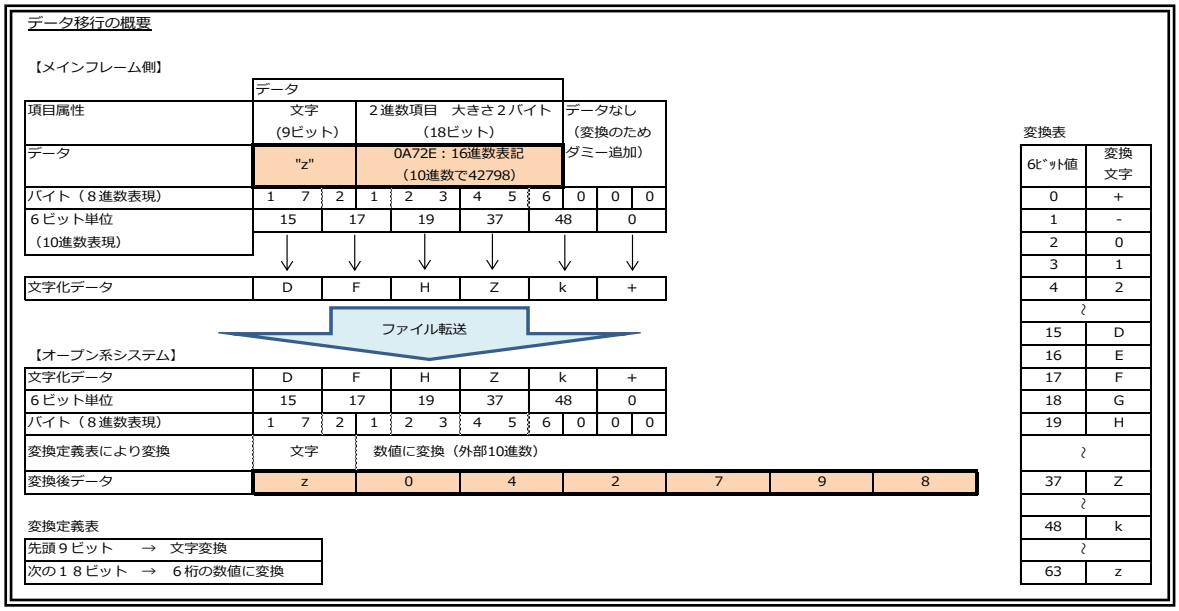

図６　データ移行の概要

ここでの工夫点はメインフレーム側で移行のためのプログラム作成を最小にすること、移行プログラム作成のための工数を抑えること、移行データが領域再定義などによる変換が対応できず変換について見直しが必要になった場合、移行プログラムの改修を最小かつ最短期間で行えることである。

メインフレーム側の移行用プログラムは単純なものとした。１バイト９ビット構成のデータを２バイト１８ビットを単位としてそれを６ビットで１文字データに置き換え３文字のデータとして変換（エンコード）しファイルを作成した。このファイルをオープン系システムへファイル転送しオープン系システム側で復元（デコード）、かつオープン系に合わせた形式に変換する表（以下、変換定義表）に基づく移行プログムを作成し、ファイルを生成している。また、この変換定義表はデータベース化して移行プログラムで都度参照し処理させることによって変更があっても基本的に変換定義表を変更することで対応できるようにして移行プログラムの修正を最小限にしている。さらにどうしても移行プログラムで吸収できないような例外処理に関してはデータベースに項目ごとに例外処理のサブプログラム名を登録できるようにしておき例外処理が必要であるときは対応するサブプログラムの作成と変換定義表への登録によって移行プログラム本体を改修することなく例外処理を実現させている。今回例外処理させるサブプログラムは１２本作成するに留まっている。このことによってデータ移行のための移行プログラムの開発工数を大幅に削減させた。

３．２．６　基幹システム改修内容の移行側システム反映課題

マイグレーションにおける大きな課題のひとつは逐次改修が運用中のシステムであるメインフレーム側に適用されることである。一方オープン系システムでは改修されることがわかっていてもテスト実施や要員確保の事情によって同時改修ができない場合がある。またこの変更情報を確実に管理し漏れがないようにする必要がある。
この対応策を図７に示す。

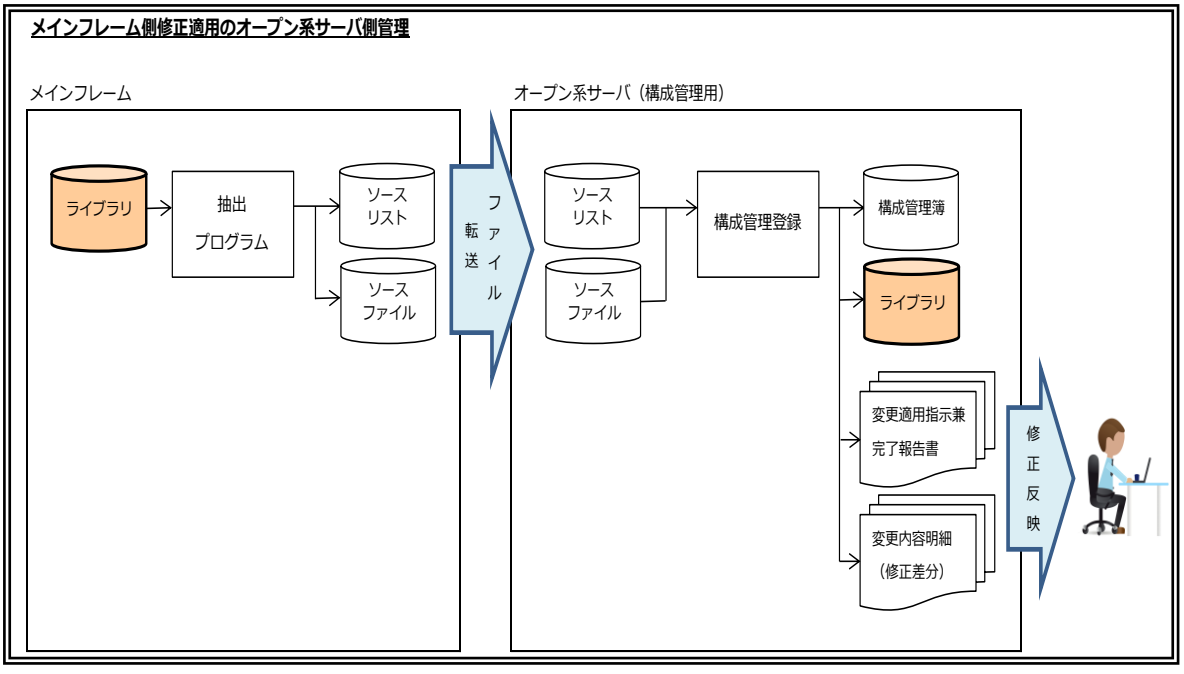

図７　基幹システム改修内容の移行側反映管理

ここではまずメインフレーム側にプログラム改修が適用されたことを検出するために定期的（１回／週）に最新のプログラムソースを最新更新日のリストとともにオープン系システムへファイル転送することとした。オープン系システムではプログラム名と最新更新

日などの情報を構成管理簿に登録、データベース化した。また、現在のオープン系システムがいつの時点のプログラムであるかも構成管理簿で管理している。これらの情報からメインフレームで修正された情報を、変更適用指示書（兼完了報告書）と新旧比較し差となっている内容とともに提供することから確実にオープン系システムへ反映させることができた。

更に、移行前のメインフレーム側の改修凍結期間を１ヶ月とすることができ利用部門への影響も最小にできたと評価している。

３．３　技術的課題とその解決

技術的課題の多くは運用中メインフレームのオンライン・周辺連携の実現方式をなるべく変更しないようにしたことから発生している。以降に具体的な課題内容と解決策を述べる。

３．３．１　オンライン用テストデータ作成課題

通常オンラインのテストは端末にオンライン画面を表示させデータを打鍵してその結果の画面表示内容を確認する。またはその打鍵内容をテストツールを使用し記憶させておきテストツールで打鍵を再現させることが一般的である。ここでの課題はどんな内容で打鍵すべきかは業務側でしか知らないということである。単体テストレベルであれば情報システム事業部員であれば何とか行うことができた。結合テスト、システムテストレベルでは業務に即した内容を打鍵であり実際の伝票などを見ながらの業務知識をもとにした打鍵が必要となる。

運用テスト前では業務サイドに多大な負担をかける支援はお願いできないため、なんらかの対策を講じることが必要であった。

この課題の解決策を図８に示す。

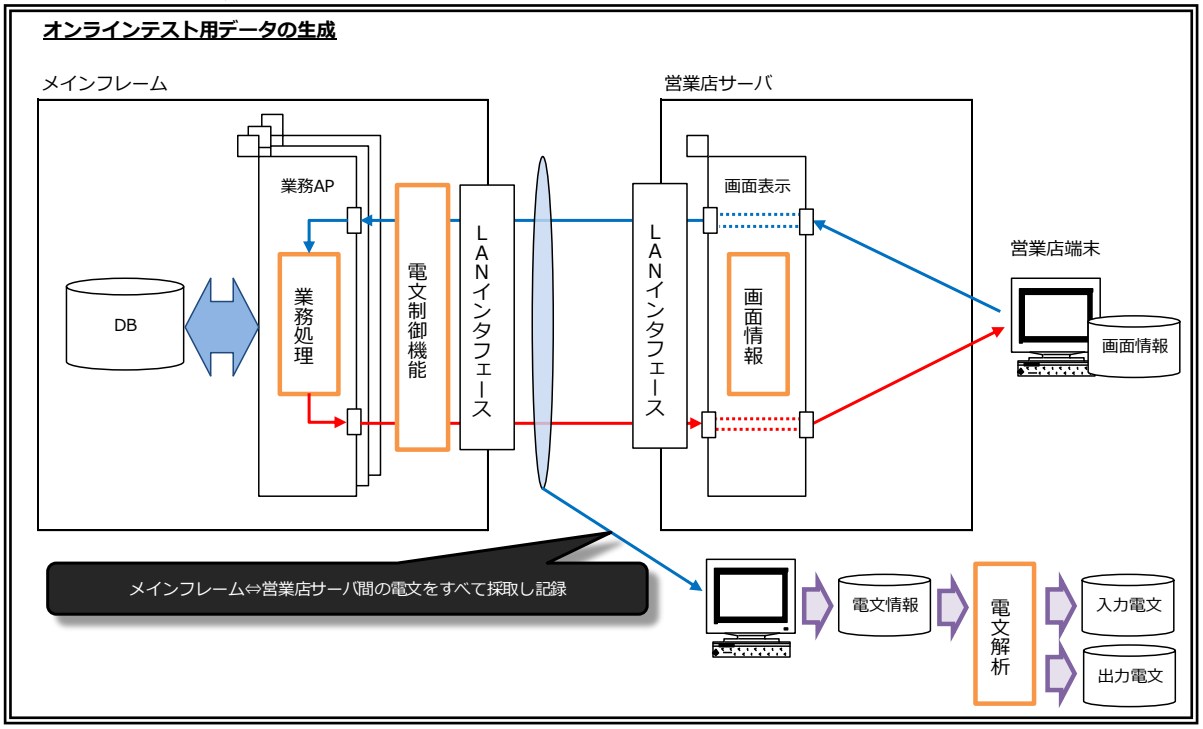

図８　オンライン用テストデータ生成

ここではメインフレームのオンラインシステムの仕組みに注目した。図８のようにメインフレームと営業店サーバ間では上り電文は入力されたデータが下り電文では固定表示項目以外の画面表示データを通信することで実現していた。また、その通信の仕様と通信データのフォーマットも公開されていた。そこでメインフレームの LAN インターフェイス部分に LAN アナライザを設置しそこを流れる営業店サーバとの電文を採取し、それを解析することによって打鍵データと表示データを生成した。

LAN アナライザ（フリーソフトウェア）においては採取したデータをいくつかの形式でファイル出力することが可能であった。そのファイルを入力し解析するアプリケーションを開発し入力電文、下り電文をデータベース化した。

この対応でほぼ１００％近い生の打鍵データとそのオンライン処理結果である画面表示データをペアで作成することを可能とした。

３．３．２　オンラインテスト課題

テストデータとして、実際にメインフレームを流れるオンライン用のデータを捕捉できたことから結合テスト、システムテストにおいてこのデータを最大限に活用している。

その活用方法を図９に示す。

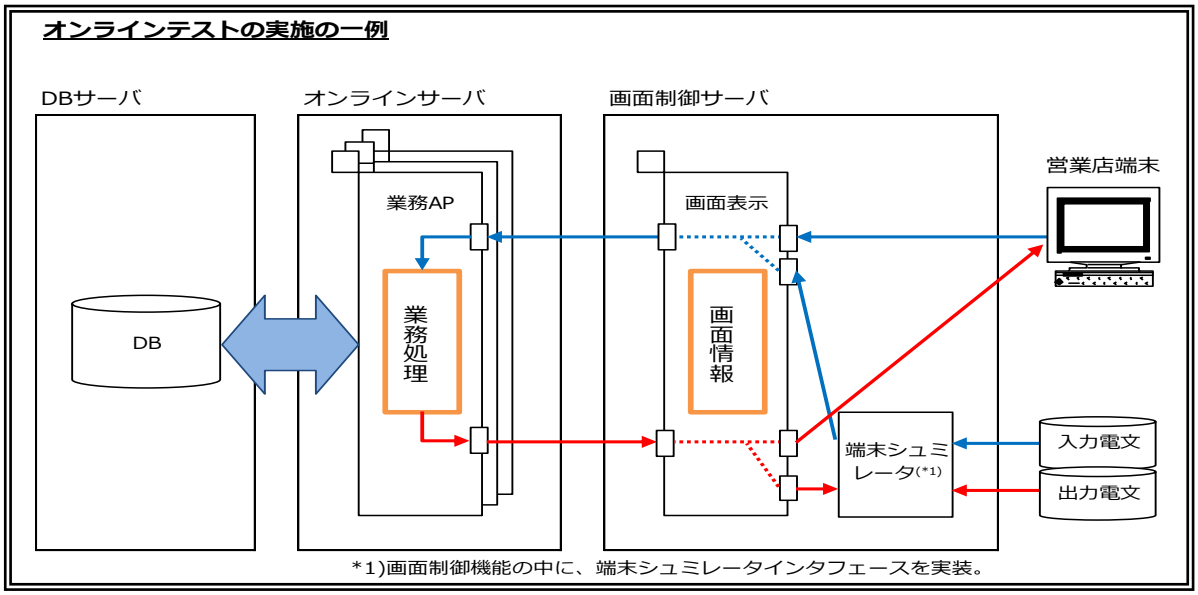

図９　オンラインテストの実施

今回、新規開発した画面制御サーバにテスト機能を実装している。オンラインテストデータをデータベースに登録しておき、それを取り出して順次オンライン電文としてオンラインサーバにあたかも打鍵データであるかのように送信する。オンラインサーバの処理結果である画面表示データ電文をメインフレームの画面表示データと同一かどうか判定することで実現している。

この評価は、ある程度の成果を得ているが１００％にはならないということである。

すべてのデータベースの更新がこの打鍵データによって更新されるわけではないということが理由である。データベースの更新はオンラインサービス時間中にオンライン中のバッチ処理によって更新される場合があり、その更新処理まで再現することが不可能なこと

が理由である。そのため適用範囲は結合テスト、システムテストとなり運用テストへの全面適用に至っていない。運用テストへの全面適用にはオンライン中のバッチ処理とも同期した仕組作りが必要である。

また、この機能は性能テストにも役立っている。メインフレームで実際に処理されたデータを時系列の順序を守ったままオープン系システムへ集中的にオンラインサーバに投入することで負荷集中時の性能情報を得ている。この情報からオンラインサーバ、画面制御サーバの負荷分散数を障害対策も考慮し算出している。

３．３．３　周辺システム連携機能代替課題

メインフレームと連携する周辺システムとのインターフェイスは「２．４　連携インターフェイス」で述べたようにファイル転送（FTP）を主として使用している。

ここでは代表例としてＥＤＩ装置との連携方法について図１０に示す。

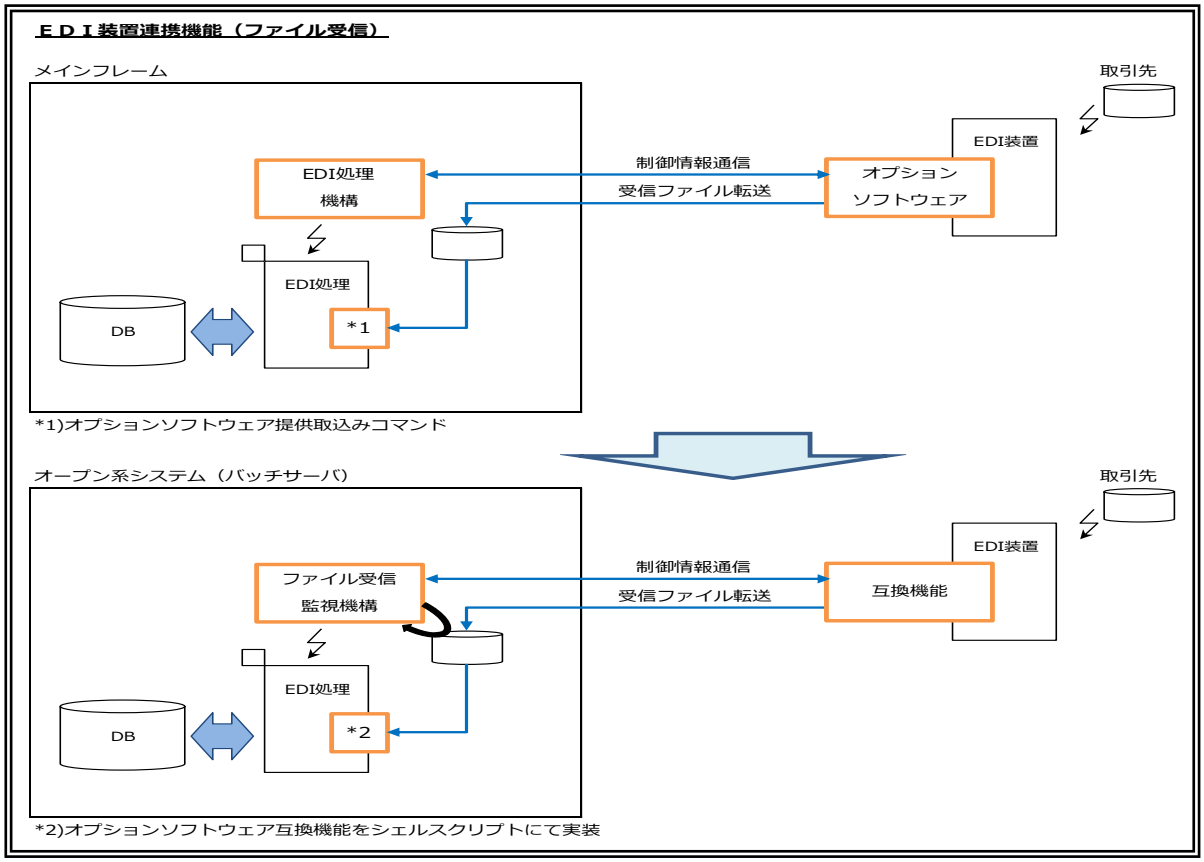

図１０　ＥＤＩシステムインターフェースの実装（受信機能の例）

メインフレームではメインフレームベンダからＥＤＩ装置と連携を行うオプションソフトウェアが提供され利用していた。ＥＤＩ装置にてデータ受信するとオプションソフトウェアの機能によってメインフレームと通信しファイルごとに定義されたメインフレーム側のＪＣＬを起動する。起動されるとそのＪＣＬ中でミドルウェアが提供するコマンドを実行させ受信データをファイルとして取込む。一方メインフレームからのデータ配信は同様にオプションソフトウェア提供コマンドで配信ファイルを指定するＥＤＩ装置にファイルが転送され定義された配信先へのデータ配信がＥＤＩ装置側で実行されるような仕組みとなっていた。

これをオープン系システムで同様な機能を実現する方法として以下が必要である。

・データ受信においてはＥＤＩ装置で受信したデータをメインフレームと同様に受信データをファイルとしてオープン系システムに転送する。
・オープン系システムでは受信ファイルを常時監視し受信ファイルを検知すると起動すべきオープン系システム側ＪＣＬ相当のシェルスクリプトを決定し起動する。
・シェルスクリプト実行時メインフレームと同等のコマンドを受信ファイルを取込む。
・データの配信においては同様にシェルスクリプト内で実行されるＥＤＩ送信コマンドによって配信ファイルをＥＤＩ装置にファイル転送する。
・ＥＤＩ装置側ではファイル転送されたファイルを指定の配信先に配信する。

これらの実装をプログラムを作成せずシェルスクリプトのみで行っている。シェルスクリプトのみで実現できた要因としてはメインフレームにおいてもＪＣＬ内でコマンド記述を行うことで運用を行っていたことと、豊富な機能を持つシェルスクリプトを有効に利用できたことによる。

３．３．４　連携する周辺システムとの検証課題

メインフレームと連携する各周辺システムの検証も大きな課題であった。上りの連携については各周辺システムからメインフレームに転送するデータをオープン系サーバへも同様に転送を行う機能を追加することで十分な検証を可能とした。多くの周辺システムでは開発環境をもっていなかった。そのため連携したとしてもファイル転送によって、その周辺システムへファイルが届いたことまでしか検証が行えなかった。

表３に今回対象となった連携すべき周辺システムの一覧と検証方法を示す。

項番	連携先周辺システム	手順(*1)	上り	下り	検証方法(*2)	備考
1	ＥＤＩシステム（１）	FTP	○	○	疑似環境	通信相手先：2000
2	ＥＤＩシステム（２）旧手順用	営業店	○	○	同時配信	通信相手先：200
3	インターネット貨物照会システム	FTP		○	ファイル照合	お客様向けサービス
4	運行車支援システム	FTP	○	○	ファイル照合	
5	運行車ダイヤ作成システム	FTP		○	ファイル照合	
6	運行車配車支援システム	FTP		○	ファイル照合	
7	ハンディターミナルデータ受信システム	営業店	○		同時配信	連携システム先：５箇所
8	ＦＡＸ配信システム	FTP		○	疑似環境	
9	e-mail配信システム	FTP		○	疑似環境	
10	基幹システムデータエントリ中継システム	FTP	○		同時配信	
11	貨物中継会社問合わせシステム	ソケット	○	○	疑似環境	
12	ＥＴＣ利用実績収集システム	FTP	○	○	同時配信	
13	データウェアハウスシステム	FTP		○	ファイル照合	社内情報提供用
14	各種Ｗｅｂ照会システム	FTP		○	ファイル照合	社内情報提供用
15	経営情報システム	FTP		○	開発機	
16	会計システム	FTP	○	○	開発機	
17	給与システム	FTP	○	○	開発機	

*1）手順の説明
「ＦＴＰ」･･･ファイル転送
「営業店」･･･メインフレーム時の営業店サーバ経由→画面制御機能で代替
「ソケット」･･･ＴＣＰ／ＩＰソケット通信

*2)検証方法の説明
「疑似環境」･･･疑似環境を構築し同一インターフェースにて検証（入力データは本番データを複写）
「同時配信」･･･相手先よりメインフレーム、オープン系システムへ同時配信を実施（下りも存在する場合はファイル照合対応）
「開発機」･･･連携先にも開発機が存在しているため開発機と連携

表３　連携するシステムの一覧と検証方法

この**表３**のように下り連携での検証は限定的な範囲でしか行えず本番移行時のリスクとなっていたが、できる限りのリスク低減させるためメインフレームから転送されるファイルとオープン系システムで生成されたファイルとの同一性の検証を行うことで担保している。

４．プロジェクト推進課題とその解決

プロジェクト推進の最大の課題はメインフレームサイドとオープン系システムサイド間のコミュニケーションをいかに円滑に行うかであった。一般にそれぞれのサイドの有識者は相手サイドを十分に理解できるスキルを保有していない。異国人同士で通訳を介さないで会話しているのに等しい。今回のケースでは特にメインフレーム側のミドルウェアの有識者の調達が困難であった。利用技術に精通する有識者は存在するものの機能面や構造面から対応可能な有識者は希少でありコスト面からも調達は困難であった。そのためメインフレーム経験者であり、かつオープン系システムをある程度経験している人材を調達し通訳とすることで解決策とした。

また図４のプロジェクト体制に示すようにマルチベンダ体制であったためベンダ間のコミュニケーションの課題が存在していた。

特に当情報システム事業部において今回のようなプロジェクトは２０００年問題対応、２００７年のメインフレーム更新時から経験したことがなかった。当該プロジェクトにおいてもメインフレームベンダのみの単一ベンダプロジェクトであった。

当時、中核となって推進した人材の多くは定年、配置転換などで情報システム事業部に在籍しておらず退職された方も多い。このような状況からメインベンダに支援をいただきプロジェクトマネジメントの強化を行った。

(1).プロジェクト管理ツールであるProjectWEB（富士通株式会社様提供）の利用

これはインターネットにアクセス可能なプロジェクト管理ツールであり TODO 管理、ライブラリ管理、イベント管理などが利用可能であり各ベンダの開発拠点が分散していても円滑なコミュニケーションを図ることが可能となった。

(2).インシデント管理ツールの利用

当時の情報システム事業部内でインシデント管理の仕組みはシステム化されておらず、手作業状態となっていた。これは基幹システムの保守・開発は１００％内製でありコールセンター機能も開発・保守要員が行っており、ほぼコール時に即時解決するケースが大半だったためである。

今回、各ベンダ・情報システム事業部の全体としてインシデント管理を行う必要がありメインベンダ様提供のツールを利用し対応を行った。

(3).構成管理機能の構築

メンフレームでは、開発系、本番系各ライブラリを設け更新管理と依存関係によ影響範囲を考慮した再コンパイル機能を実装し運用を行っていた。

マイグレーションにおいてはメインフレームの変更をオープン系システムでも把握し、マイグレーション完了後のプログラムにメインフレーム側で修正が発生した場合、再度マイグレーションを行うか、オープン系システム側で同様の修正を適用する必要がある。このような管理を手作業にて行うことはできず構成管理機能を実装し運用を行っている。

５．本稼働の状況と評価

このような経過をたどり２０１６年１２月３０日から２０１７年1月２日で移行を実施し２０１７年1月３日から本稼働を開始した。本稼働当初は想定以上のトラブルに遭遇した。その中で問合わせによる重大なトラブル報告は2か月間で約１００件となった。

この重大なトラブル報告に対して、状況に応じて暫定処置・恒久処理を順次実施し２月末までには、ほぼ収束させることができた。

このように収束が可能だった最大の要因は業務部門の「現場力」であったと判断している。「伝票が発行されない」「帳票の印刷内容がおかしい」「オンライン開始遅延」などの初期トラブルにおいて荷物の動きを止めることなく業務継続を行えた。この現場力には感謝の念に堪えません。現場力による強力な支援がなければ本稼働はとん挫していただろうと確信している。

この現場力に支えられ情報システム事業部員がトラブル解決に専念できたことから本稼働当初に発生したトラブルの早期解決が可能となった。

もうひとつの要因はこの基幹システムに長年携わってきていただいている１０名の開発・保守担当者の「経験力」である。トラブルが発生した際、どこがあやしいかを推測し的確に直接原因を突き止め最短での暫定処置をとり、まずは解決させる。この後、再発防止のための恒久対応を行う。この「経験力」がなかったら復旧させるまでに長時間を要していたであろう。

またトラブルの発生原因のひとつとしてメインフレームシステムの潜在バグの顕著化がある。本来はバグであったが、想定とおりの動作をしていたため、今回マイグレーションを行ったことにより明確なバグとなって発生しその対応も必要だったことを追加しておく。

本稼働当初はオープン系システムに対して不慣れな面もあり戸惑うこともあったが、マイグレーションにより機能を変えなかったことにより、長年の経験が有効に活かされ短期間にて安定運用が実現できている。

この対応期間の中でオープン系システムに対してのスキルも大幅に向上できている。

また、オープン系システムの持つ豊富な機能をフルに活用することによるトラブル解決時間の短縮も実現できている。その内容は以下である。

- デファクトスタンダードデータベース採用によって豊富なツール利用が可能となりデータの即時可視化ができる。
- 誤った処理によってデータベースの修復が必要となった場合もツール利用してプログラムレスで修復できる。
- システム標準のプログラム走行可視化機能利用による誤り箇所発見時間の大幅短縮の実現ができる。

・システム標準のプログラム異常終了時の内部状況可視化確認を利用した原因調査時間の大幅短縮ができる。

このような内容から、今回マイグレーションの選択は正しかったと評価している。特に、業務要件定義が不要であり、それに必要な期間のスケジュールを短縮できている。

画面構成もメインフレーム時と同様であり打鍵内容に違いがなく操作方法の違いのみ利用者に習得してもらうことだけで特別なトレーニング期間を設けることなく運用開始できたことが大きな効果である。

コスト負担面においても従来のメインフレームと営業店サーバを含めたランニングコストはオープン系システムに移行したことで約７割削減の実現をしている。

６．今後の取組

マイグレーションの結果、メインフレームのときと、ほぼ同じサービス内容であり、システム運用である。しかしオープン系システムであることの優位点を活用できているわけではない。現在はこの優位点を最大限享受しての戦略的システム構築へのスタートラインに立つことができただけである。

今後、データベース利活用によって即時にきめ細かな情報提供を行う。AI・GPS・地図情報などを有効に活用しＩＣＴによる強力な業務支援するシステムの構築を推進する。法改正、法規制の要請、経営環境に対応する要求に即座に対応できるシステムとする。そして、これらをオープン系システム上で短期間で構築を実現する。

従来のバッチ処理形態から順次データベース中心システムへ移行させてオンラインサービスの２４Ｈ３６５日にむけてシステムを変革させることも大きな目標と考えている。

そのためには、さらなるオープン系システム対応可能な人材育成も必要である。メインフレームを長年担当してきたメンバも、このプロジェクトを経験したことによってオープン系システムに対応できる基盤はできている。

いままでの経験を活かしながら、かつ新技術への対応力を養成できたこともこのプロジェクトの成果である。今後は更にオープン系システムへの理解を深めさせて貴重な戦力としていきたい。

情報システムへの益々の要請に対応していくためには自社戦力のみでは実現できない。外部への開発委託も必須となってくる。この面においてもマルチベンダによるプロジェクト参加経験は役立っている。技術基盤があっての開発委託であり要件のみ伝えるだけの委託となっていかないための人材育成を一層推進していきたい。

７．おわりに

今回のプロジェクトにおいては技術面で最新技術を適用するといったことは行っていない。従来技術の積み上げで実現している。このことは新技術に対しての正しい使用方法の習得や使用することによる想定できない落とし穴に入り込むリスクを回避できるという優位点を持っている。もちろん最新技術の動向を的確につかみ適用のため研鑽を積むことは必要である。

ただし今回のプロジェクトのように確実に課題解決しながら納期確保が最優先である場合、想定できない問題に対しての解決期間の長期化リスクを回避できたことは評価に値すると信じている。

基幹システムのマイグレーション担当中核メンバとそれを支援する周辺システム担当者、インフラ基盤構築・保守担当者の総勢２０名余りメンバにとっては非常に貴重な経験であったことを確信している。

ほとんどのメンバが今回のような基幹システム移行切り替えのプロジェクト経験はなく日常的な開発・保守の経験のみであった。プロジェクト推進にあたっては心の内外で戸惑い、疑問、反発、迷走が多くあったであろう。このメンバが、この経験を踏まえて更に成長し真のオープン系基幹システムへ発展させていけるものと信じている。

最後に本プロジェクトに参加いただいたベンダ各社の多大なるご支援に対して感謝いたします。

参考文献

なし