オープン勘定系システム稼働までの軌跡

A Trace to Banking System based on Windows' Open Architecture Operation

向　井　　丞

要　約　2007年5月，株式会社百五銀行にてWindowsベースのオープン勘定系システムが稼働した．2003年12月に，このシステムの開発開始を発表した時の一般的な反応は，「(無謀な)チャレンジ」というものであった．それまで，オープン化は難しいと言われていた銀行勘定系システムを，Windows上に構築すると発表したのであるから，この反応は無理もないものである．しかしながらこの発表は，当然の事ながら，オープン化に関し多くの段階を経て一歩ずつ着実に実績を積み重ねた上でのものであった．

銀行勘定系システムのオープン化に対しては，オープン系のH/W，S/Wの進展に合わせ，段階的に行うという方針が，非常に早い段階から打ち出されていた．その方針に則り，開発環境/ネットワークのオープン化から開始し，勘定系の対外接続部分のオープン化，勘定系システムの実機検証，それらの経験/成果をベースにしたオープン系ミドルウェアの開発，そのミドルウェアで稼働するソリューションの開発と，段階的に実績を積み重ね，最終的に勘定系システム全体のオープン化に着手した．

本稿は，非常にミッションクリティカル性の高い，超大規模システムである銀行勘定系システムを，オープン環境で稼働させる際に，どのようなプロセスを経て実現させたのか，その軌跡を辿ったものである．

Abstract　The first banking system based on Windows' open architecture was established in May, 2005 at the Hyakugo Bank, Ltd. At the time of public announcement about the launch of our development, general consensus was "This might be a death-or-glory challenge". This reaction was quite understandable according to the commonsense at that time, i.e. "the implementation of banking system on top of Windows operating system is extremely difficult". However, we were very confident of our development plan based on our achievements steadily accumulated during our past step by step approaches towards the banking solution on top of open infrastructure.

In the very early stage of our planning of our banking solution based on open architecture, our essential policy was focused on the step by step approaches following to the evolution of open H/W and S/W. According to that policy, we have started from the migration of development environment and network technology to open architecture. The development of open online middle-ware followed inheriting the outcomes of the open migration of interface system between banking system and surrounding systems, and proto-typing analysis of banking system on top of open servers. Then the application development methodology was accomplished on top of the open online middle-ware. Then we finally put our hands to the implementation of open total banking system according to all the achievements accumulated during the preceding steps.

In this paper, we are going to describe the trajectory of the processes towards the banking system based on open architecture, which was highly mission critical and extremely large scale.

1. は じ め に

2003年12月，日本ユニシス株式会社（以下，日本ユニシス）は，マイクロソフト株式会社（以下，マイクロソフト社），株式会社百五銀行とWindows® ベースのオープン勘定系システム開発に着手することを発表した．

発表当時の一般的な反応は，「（無謀な）チャレンジ」というものであった．しかしながら，オープン勘定系システム開発着手の発表に至るまでには，その遥か以前から，綿密に実績を積み重ね，多くの段階を経てきており，そのプロセスに携ってきた開発者たちにとって「チャレンジ」という言葉は正しくなく，今まで積み重ねてきた実績の集大成という意味合いの方が強かったのである．

本稿では，オープン勘定系システム「BankVision」が2007年5月に本番稼働するに至るまでの軌跡を辿る．

2. オープン勘定系システムに向けてのスタート

「銀行勘定系システムをオープン環境上に構築する」，この試みのスタート地点は，1996年4月に遡る．

1996年4月，信用金庫業界向け勘定系システムパッケージ「SBI21」の開発に着手したが，その際，このシステムに求められた要件の一つは，将来のオープン環境上への移植性を確保するということであった．

当時，勘定系システムを構築するには，オープン環境はH/W，S/Wとも，性能，機能，安定性にまだ課題がある状態であり，実行環境，運用環境に関してはメインフレームに一日の長があった．しかしながら，今後の技術動向を踏まえた場合，いずれは勘定系システムもオープン環境で稼働させることになると予測された．従って，勘定系システムのオープン化に際しては，以下の二つを基本方針とした．

- オープン系のH/W，S/Wの進展に合わせ，段階的にオープン化する．
- 実行環境/運用環境がオープン化された際，アプリケーションをオープン環境向けに一から開発し直すのでは，コスト/期間を要し，リスクも大きいため，オープン環境になってもアプリケーションの全面的な更改の必要がない構造とする．

特に後者に関しては，アプリケーションの安定性という側面から考えても，重要なポイントであった．

SBI21の開発に際し，上記方針に則ったオープン環境への移植性の確保のための具体的な考慮点は以下のとおりである．

1) アプリケーション構造にオブジェクト指向技術のカプセル化の考え方を取り入れる
2) 開発環境をWindows上に構築する
3) 勘定系端末と勘定系システムとのネットワーク接続のプロトコルはTCP/IPとする
4) バッチ処理による各種作表処理に関しては，オープン系のサーバ上に顧客/口座の情報を展開し，オープン系のツールで作成できるようにする

2.1 SBI21で実現したアプリケーション構造

SBI21のアプリケーション構造には，オブジェクト指向技術のカプセル化の考え方を取り入れている（図1参照）．

図 1　SBI21 のアプリケーション構造

この構造を採った目的は，当然のことながら，アプリケーションの保守性の向上にあった．ただし，その詳細に関して記載することは本稿の趣旨とは若干異なるため，ユニシス技報通巻 61 号「特集：金融ビックバンと情報技術」(1999 年 5 月刊行）に掲載の論文「オブジェクト指向技術による勘定系システムのリエンジニアリング」を参照願いたい．

このアプリケーション構造は，将来のオープン化を見据え，データベースと処理をサブシステムという単位に予めカプセル化しておくことによって，オープン化に際してデータベース構造等を変更せざるを得ない状況になっても，勘定系システム全体を見直す必要はなく，個々のサブシステム単位に見直しを行えばよいという構造にしたことがポイントである．

BankVision のアプリケーションは，SBI21 をベースにオープン化し，地方銀行の要件及び S-BITS コンソーシアム[*1] から提示された要件を加味するという手順で開発したが，このアプリケーション構造は，SBI21 のオープン化作業の省力化に非常に大きな効果をもたらした．

2. 2　Windows 上に構築した開発環境

開発環境を Windows 上に構築する際は，「メインフレーム用のアプリケーションの開発環境を Windows 上に構築する」という考え方ではなく，「Windows 上の開発環境で作成したアプリケーションをメインフレーム上で稼働させる」という考え方で取り組んだ．「SBI21/DE」と呼ばれるこの開発環境は，データベースを使用した単体テストを行うことが可能であり，データベースのコミット/ロールバック処理も実装している．従って，プログラム単体レベルではあるが，この時点で，Windows 上でアプリケーションは稼働していたのである．

このことは後に，BankVision 開発のために SBI21 のオープン化作業を行った際，前述したアプリケーション構造との相乗効果により，その作業の省力化とオブジェクト内の安定性確保に効果を発揮した．加えて，BankVision 開発者にとっては，実行環境が Windows に変わるということを意識することなく，SBI21 開発と同じ感覚で開発作業を行うことができるという効

果をももたらした.

2.3 勘定系端末と勘定系システムとの接続のオープン化

SBI21 以前の勘定系システムでは，勘定系端末と勘定系システムとの間は，端末ベンダと取り決めた固有のプロトコルで接続されていた．SBI21 では，これを TCP/IP で接続することとした．これは，勘定系システムのオープン化に先駆けて，勘定系端末側のオープン化が進展することを予想してのことでもある.

TCP/IP 化に関しては，Inquiry/Answer 型の電文処理の実装は容易であったが，送達確認機能や為替処理のような一方送信型の電文処理に工夫を要した.

2.4 各種作表処理のオープン化

SBI21 では，勘定系システムのデータベースを切り出してオープン環境に展開し，バッチ処理で行う各種帳票作成をオープン環境で行えるようした.

1996 年当時のネットワークスピード/コストでは，顧客/口座単位のデータベースの切り出しは，処理時間とデータ鮮度との関係で，月次処理向きのみ提供していた．BankVision では，このバッチ処理向けの顧客/口座単位のデータベースを日次単位で提供し，各種帳票作成，他サブシステムへのデータ切り出し等は，このデータベースを中心に処理している．BankVision では，この仕組みを基幹系 DWH と呼んでいる.

3. オープン勘定系システムに向けての加速

オープン勘定系システム構築のトリガとなったのは，ES7000 の出現である．日本ユニシスは，2000 年 3 月，最大 32 個の Intel プロセッサを搭載し，Windows ベースのデータセンターを実現する「Unisys e-@ction Enterprise Server ES7000（以下，ES7000)」を発表した．これにより，オープン勘定系システム実現の障壁の一つであった H/W 性能が一気に解決したのである.

また，同年 9 月，ES7000 に搭載するオペレーティングシステムとして Microsoft® Windows® 2000 Datacenter Server を 10 月から出荷すると発表した．これにより，勘定系システムに求められる重要な要件であるシステムの高可用性が少なくとも OS レベルでは確保できたのである.

3.1 オープン勘定系システム稼働までの軌跡

SBI21 の開発に端を発したオープン勘定系までの軌跡を図示すると，図 2 のようになる.

ES7000 の販売開始発表と時を同じくして，SBI21 のアプリケーションの一部をオープン化し，ES7000 上で稼働させるという実機検証を開始した．それと並行して，三井住友銀行（当時，さくら銀行）向け新 BANCS プロジェクトが発足していた．これは，統合 ATM[*2] センタとの接続システムを ES7000 上に構築するというプロジェクトであった.

その後，上述した二つのプロジェクトの成果をベースにオープンミドルウェア「MIDMOST」が開発され，MIDMOST 上で稼働する国際勘定系システム・パッケージ「BankForce-NE」，総合資金証券管理システム・パッケージ「Siatol-NE」，対外系システム・パッケージ「ETAIGAI」の開発が開始された.

図 2　BankVision 本番稼働までの軌跡

1章で，オープン勘定系システムの開発はチャレンジではなく集大成の意味合いが強かったと述べたが，日本ユニシスではBankVision開発に先立ち，社内のソリューション開発において既にWindowsベース，MIDMOSTベースのシステムが次々と開発されており，オープン勘定系システムの実現には十分な手ごたえを感じていたからなのである．

3.2 MIDMOSTの誕生

オープン勘定系システムの実現には，MIDMOSTは欠かすことのできない存在であった．

オープン勘定系構築に際し，ES7000を使用することで，OSはWindows，データベースソフトウエアはSQL Server®という方針は決定していたが，課題だったのは，ミドルウェアである．

日本ユニシスのメインフレームには，XISという勘定系システムを支えるミドルウェアがあるが，オープン環境上には，そのようなミドルウェアは存在していない．オープン環境上で稼働するミドルウェアを用意するために，以下の二つのアプローチを取った．

- 新BANCSプロジェクトでは，勘定系システムほどはミドルウェアに要求される機能は多くはないため，ミドルウェア機能ごと開発し，その後，それを発展させて勘定系でも使用できるミドルウェアに持ち上げる．
- それと並行して，オープン環境向けに販売されているTPモニタ[*3]を使用し，それに適合するようにSBI21のアプリケーションの一部を書き換え，実際にES7000上で稼働させて実機検証するというプロジェクトを立ち上げる．

後者の実機検証プロジェクトでは，TPモニタとして3社の製品を使用して行ったが，いずれも基本的な機能においては問題はないものの，勘定系システムで使用するとなると，相当の機能を作りこまなければならないことが判明した．

既存のTPモニタに対し，不足する機能を補完機能として作りこむには，以下の2点の懸念があった．

- 万一，障害が発生した場合に，OS，TPモニタ，補完機能のいずれに問題があるのかの判別に時間を要す可能性がある．
- アプリケーションプログラムに不具合があった際のプログラムの動的な緊急入れ替えやプログラムの動的な流量制御等々，TPモニタの外側に補完機能を作りこむというやり方では対応が難しい機能要求が存在する．

そのため，ひとつめのアプローチ，即ち新BANCSプロジェクトの開発完了を待ち，そのミドルウェア機能部分を活用し，勘定系システムでも使用できるミドルウェアに機能拡張することを選択した．それがオープン系ミドルソフトウェア「MIDMOST」であった．

MIDMOSTの開発体制は，新BANCSプロジェクトと実機検証プロジェクトの両方から技術者を拠出して編成した．そして，その技術陣に対して，メインフレームで勘定系システムが使用しているXISが保有する機能のうち，オープン環境でも必要な機能を要件として提示していくという形態で開発を行った．従って，MIDMOSTは勘定系システムでの稼働実績はないものの，完成した時点で勘定系システムのミドルウェアとして十分な機能を満たしていた．

ただし，勘定系システムは数千のアプリケーションを稼働させる大規模システムであるため，実際に勘定系アプリケーションを実装するにあたっては，MIDMOSTにCPUやメモリといったリソースを最適利用するための機能を追加実装し，チューニングを施した．

3.3　アーキテクチャの統一

MIDMOST と並行して開発の検討が行われていた次世代国際系システム・パッケージ，次世代総合資金管理システム・パッケージはアーキテクチャが統一され，オープン環境上で稼働する次世代金融ソリューション体系「UNIFINE」として整理された.

アーキテクチャの統一とは，主に以下の部分の統一による.

- ミドルウェアとして MIDMOST を使用
- アプリケーション構造は，基本的には SBI21 のアプリケーション構造を踏襲し，ES7000 での実機検証で得られた経験値をベースに手直しをした構造を採用
- 開発環境は，SBI21/DE をベースに MIDMOST 上で稼働させるために，機能の追加/変更を加えた MIDMOST/DE を採用
- 運用環境は日立 JP1 を中心に，不足機能は MIDMOST が運用補完機能として提供

これにより，アプリケーション構造，開発手順，開発環境，実行環境，運用環境とも統一することができ，BankVision 開発に先駆けて，アプリケーション構造面の正当性や開発手順の妥当性，開発環境/実行環境/運用環境の機能性等々が，既に社内の開発において検証済みとなっていたのである.

また，無店舗型の銀行向け勘定系システム「BANKSTAR」を同一のアーキテクチャで開発していたが，このシステムが，BankVision の本番稼働に先駆けて，2006 年 1 月に，株式会社セブン銀行にて本番稼働できたことも，BankVision 開発者達にとっては大きな出来事であった．通常の銀行とは業務形態が少し異なるが，勘定系システムとしての稼働実績ができあがったのである.

4.　マイクロソフト社の技術

先に記述した新 BANCS プロジェクトにおいて，マイクロソフト社との連携は既に実績があり，また，ES7000 を使用した実機検証においても，良好な協力関係にあったため，開発の連携という側面に不安要素はなかった.

技術的側面に関して言えば，BankVision の開発と前後して提供された以下の二つの技術が特に大きな効果をもたらした.

1)　Windows Server® 2003 Datacenter Edition 64bit バージョンの提供
　・Windows Server 2003 x64 Edition 版
　・Windows Server 2003 for Itanium-based Systems 版
2)　データベースミラーリングの提供

4.1　Windows Server 2003 Datacenter Edition 64bit バージョンの提供

2000 年 3 月からの実機検証において，懸念事項として挙がっていたのは，デスクトップヒープ領域[*4] のサイズであった.

2000 年 3 月当時，実機検証に使用していた OS は，Windows2000 であり，Windows2000 が持っているデスクトップヒープ領域のサイズ（48MB 固定）では，同時稼働可能なプロセス数が少なすぎ，勘定系システムを稼働させることは，非常に困難であった.

Windows Server 2003 においては，デスクトップヒープ領域が拡張可能（デフォルト 48MB, 最大 512MB）となり，Windows2000 の約 10 倍程度に拡大されたが，机上計算では問

題はないものの，決して余裕のある状況ではなかった．これがWindows OSの64Bit化によって，事実上，制約がないと言える容量（デフォルト104MB, 最大8GB）に拡大されたのである．

4.2 データベースミラーリングの提供

メインフレームではXTPA[*5]という仕組みによって，メインフレーム1台に障害が発生しても，数十秒で他のメインフレームに切り替わり，仕掛かり中のトランザクションを除けば，利用者は障害の発生による停止を体感することなく，取引を継続して行うことができる．

BankVisionにおいても，APサーバ障害の場合は，MIDMOSTの機能により他のAPサーバへ処理が自動的に振り分けられるため，XTPAと同様，利用者がAPサーバ障害発生による停止を体感することなく，取引を継続できる．

問題は，DBサーバ障害発生時の切り替え時間にあった．DBサーバは，Active-passiveのクラスタ構成をとっており，障害発生時には，自動的にpassive側に切り替わる仕組みであるが，この切り替えには，最短でも2分を要した．従って，XTPAの仕組みを使用した勘定系システムを稼働させているユーザからは，機能低下と見られてしまう．

この問題を解決したのが，SQL Server 2005のデータベースミラーリングである．これはDBサーバとして2台のサーバにそれぞれデータベースを保有する構成を持つものであり，データベースの更新は，プリンシパルと呼ばれるDBサーバ側を更新した後，その更新情報をミラーと呼ぶもう一方のDBサーバに伝達し，常に2台のDBサーバが同期をとって更新されているという仕組みである．従って，プリンシパル側のDBサーバに障害が発生しても，ミラー側のDBサーバが自動的にプリンシパルに切り替わるだけでよく，DBサーバ障害時の切り替え時間は大幅に短縮され，メインフレームのXTPAとほぼ同等の時間で切り替えを行うことができるシステムを実現することが可能となった．

5. BankVisionのシステム構成概要

本章では，BankVisionのシステムの概要を記載する．個々のシステムに関しては，本特集号に掲載した他の稿を参照願いたい．

システムの全体構成の概要は，図3のようになっている．中核となっているのは，BankVision APサーバ（平日用），DBサーバ（平日用）と記載されているサーバ群である．これらのサーバで平日の勘定系オンラインシステムが稼働しており，金曜日から土曜日に日付を切り替えるタイミングで，APサーバ（休日用），DBサーバ（休日用）に切り替わる仕組みとなっている．これにより，24時間365日稼働しながらも，平日用サーバ群は休日に，休日用サーバ群は平日に各種メンテナンス，環境変更等が行えるようになっている．また，平日用DBサーバ，休日用DBサーバとも各々前述したミラー構成を取っており，これにより，DBサーバ障害時の復旧時間の短縮を図っている．

BankVision基幹系DWHのAPサーバとDBサーバは，各種バッチ処理を行うためのサーバである．勘定系オンラインサーバの日付繰越のタイミングで，その時点の静的データベースがこの基幹系DWHサーバに供給される仕組みとなっている．これにより，オンラインは24時間365日稼働し続けていながら，バッチ処理側では，その日終了時点での静的なデータベースを入力して処理ができるようになっている．

図 3　BankVision システム全体構成図

勘定系システムをオープン環境上で稼働させる際の課題の一つがリソースマネジメントであった．メインフレームの OS は，一つのハードウェア上でオンライン処理とバッチ処理という異なる稼働形態のプログラム群をプライオリティ制御しながらリソースを割り当てていくための豊富な機能を有しているが，オープン系の OS はメインフレームほどの機能を有していない．従って，その観点からも，オンライン処理用のサーバとバッチ処理用のサーバを分けるということは重要なポイントであった．

6. お わ り に

オープン勘定系システムの本番稼働までの軌跡を辿ってみると，多くのお客様にご協力を賜ってのことであった．

BankVision の開発を共同で行っていただいた株式会社百五銀行様は言うまでもない．SBI21 開発でご協力を賜った朝日信用金庫様，統合 ATM の領域とはいえミッションクリティカルな勘定系システムを構成するサブシステムに初めて Windows/ES7000 を採用いただいた株式会社三井住友銀行様，BankVision 稼働に先駆けて Windows でバンキングシステムを稼働させていただいた株式会社セブン銀行様．こうしてみると，業態的にも都市銀行，地方銀行，無店舗型の銀行，信用金庫と非常に幅広い．その他にも多くのお客様にお世話になった．本稿を借りて深く感謝申し上げたい．

＊ 1　S-BITS（= Succeeding Banking Information Technology for Success consortium，エスビッツ）コンソーシアム
新たなバンキングシステムを研究し，その実現のため，対応策の策定と実証モデル（ひな型）

開発を行い，参加行に提供することを目的とした，次期バンキングシステム検討・検証コンソーシアム．日本ユニシスと日本ユニシス地方銀行勘定系ユーザ7行で構成されている．2000年11月15日に設立．

＊2　統合ATM
都市銀行や地方銀行がそれぞれ独自に保有する現金自動預払機（ATM）や現金自動支払機（CD）を相互接続するための中継ネットワークシステム．

＊3　TPモニタ
Transaction Processing Monitor．オンライントランザクション処理を実現するソフトウェア．

＊4　デスクトップヒープ領域
Windows ベースのプログラムが使用するOS管理領域で，実行されるすべてのプロセスが関連付けられる．ヒープを使用してウィンドウ管理情報やメニュー，フック，文字列などが作業用として格納される．このヒープ領域のサイズは，WindowsOSの起動時に決定され，不足時にもダイナミックに増やされないため，利用できる物理メモリとページファイル メモリが残っているにもかかわらず，“メモリ不足”となることがある．ハイボリューム・ハイトラフィック処理時のチューニング項目のひとつ．

＊5　XTPA（eXtended Transaction Processing Architecture）
日本ユニシスのメインフレームにおいて，大規模無停止トランザクション・システムを実現する拡張トランザクション処理アーキテクチャ．

執筆者紹介　向　井　　丞（Susumu Mukai）
1976年4月日本ユニシス株式会社（当時，日本ユニバック株式会社）入社．勘定系システム・パッケージ「FAST1100」の開発，勘定系システムユーザSEサービス等を担当．1996年4月より，信用金庫業界向け勘定系システム・パッケージ「SBI21」の開発に従事．その後，SBI21オープン化のES7000での実証実験プロジェクト等を経て，2004年4月より，オープン勘定系システム・パッケージ「BankVision」開発に従事．