3.26 既存システムの流用開発に関する教訓（T26）

教訓 T26　既存システムの流用開発はその前提条件を十分把握し、そのまま利用可能な部分と変更する部分を調査して実施する

問題

A社のシステムは営業店に対してサービスを提供するオンラインシステムであり、夜間の業務終了から業務日付変更のバッチ処理が行われ翌日定刻にオンラインサービスが開始される（図3.26－1）。

ある日、定刻のオンライン業務が朝から開始されず取引が停止した。ログ調査の結果、通常は朝方に実施される業務日付変更処理がこの日は正常終了していないことが判明した。オンラインシステムが復旧し取引可能となるまで数時間を要した。

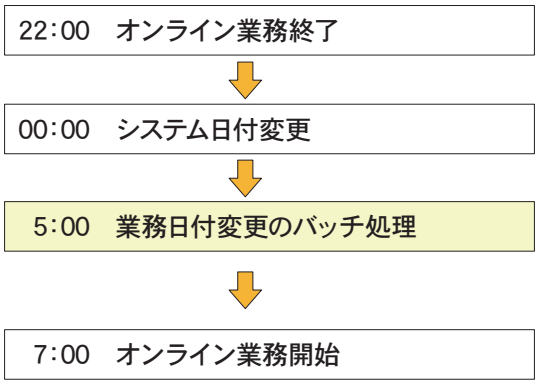

図3.26－1　オンラインシステムの日次起動スケジュール（自動運用）

原因

【直接原因】

原因は、業務日付変更のバッチ処理の不具合である。

当時、バッチの一つとして夜間に起動される業務日付変更処理は、通常時、次のように動作していた（図3.26－2）。

JOB管理機能から2つの運用ジョブXとYが順次起動され、並列に動作を開始する。各運用ジョブ内の先行サブ処理が完了するとJOB管理機能宛に完了メッセージを送る。JOB管理機能は、サブ処理X1からの完了メッセージを受領すると後続のサブ処理X2を起動する。同様に、サブ処理Y1からの完了メッセージ受領により、後続のサブ処理Y2を起動する。サブ処理X2、Y2がともに終了

すると、取引業務日付変更処理が完了する。先行する各サブ処理と後続の各サブ処理は完了メッセージを介して同期をとる仕組みであった。

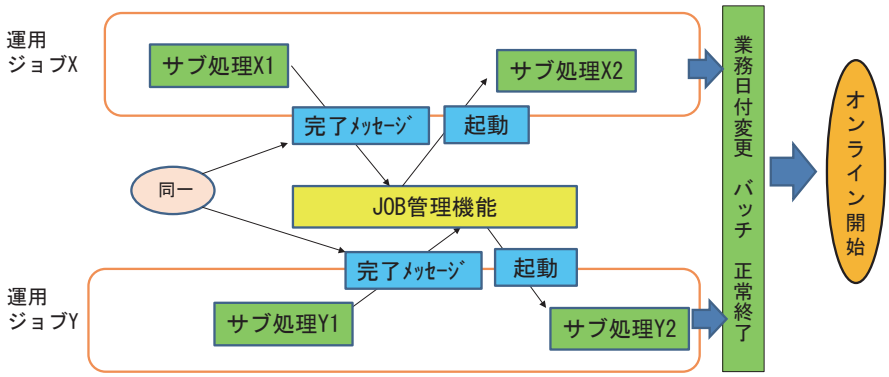

図 3.26－2　業務日付変更のバッチ処理（通常時）

ところが、当日は、サブ処理 X1 の処理が遅延し、サブ処理 Y1 がサブ処理 X1 よりかなり先に終了した。このとき、サブ処理 X1 からの JOB 管理機能宛の完了メッセージと，サブ処理 Y1 からの JOB 管理機能宛の完了メッセージとが同一内容であったため、JOB 管理機能が先にサブ処理 Y1 から受領した完了メッセージをサブ処理 X1 からの完了メッセージと誤認し、後続のサブ処理 X2 を起動した。サブ処理 X2 はサブ処理 X1 の終了が前提となっているが、この時点でサブ処理 X1 は終了していなかったため、処理に論理矛盾が発生しサブ処理 X2 が未処理となり業務日付変更のバッチ処理が正常に終了しなかった（図 3.26－3）。

過去のログを調査したところ、これまでにも同様の終了時刻の逆転はときどき発生していたが、極小時間の差でありサブ処理 X2 の開始までにはサブ処理 X1 が終了していたことから、JOB 管理機能による完了メッセージの誤認があっても問題は発生していなかったことが判明した。

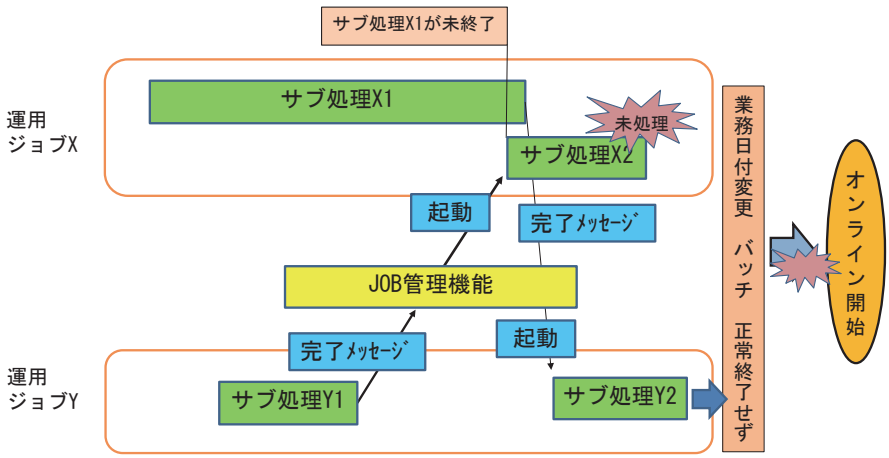

図 3.26－3　業務日付変更のバッチ処理（トラブル発生時）

【根本原因】

A 社のシステムは当初システムの開発から機能拡張－1、機能拡張－2 と追加改修が実施された。

表 3.26－1　システム開発の時期と運用ジョブの追加

システム開発の時期	当初システム	機能拡張－1	機能拡張－2
運用ジョブ	すべて単独ジョブ	運用ジョブ X 追加	運用ジョブ Y 追加
開発の形態	初期構築	流用開発	流用開発
完了メッセージ制御	なし	**あり**	**あり**
並行ジョブの有無	なし	なし	**あり**

機能拡張－1 から完了メッセージによる制御機能を入れたが、このときには運用ジョブ X は単独のジョブであったので問題にならなかった。機能拡張－2 から運用ジョブ Y が追加され、運用ジョブ X と並行して動作するようになった。追加した運用ジョブ Y は運用ジョブ X を流用して開発され、完了メッセージの内容がそのままコピーされていたため、2 つの完了メッセージが運用ジョブ X、Y どちらのものか識別できない（メッセージが一意でない）不具合が作りこまれた。

運用ジョブ X の処理詳細は仕様書に記載されていたが、運用ジョブ Y の設計者は運用ジョブ X の設計者とは異なっており、既存仕様書の調査分析が不十分であった。

また、サブ処理 X2 が未処理となっていることに気づかなかったため、原因究明に時間がかかったことも問題であった。

対策

問題の直接原因を取り除く措置として、実施した対策は以下の通りである。

- 運用ジョブ X と運用ジョブ Y で使用する完了メッセージをそれぞれ区別できるメッセージに修正し、メッセージの一意性を担保した。
- 念のため、運用ジョブ X と運用ジョブ Y を並行処理から逐次処理に変更した。

なお、全システムを調査して、同様のケースがあれば同じ対処も実施し、完了メッセージ一覧をドキュメントに追記した。

また、オペレータがトラブル発生を早期に検知するために、未処理となったジョブを監視するようにした。

再発防止策は以下の通りである。

- 設計でのレビューの強化

  流用開発における既存システムの前提条件の確認、追加部分と既存部分とが不整合や競合を起こすことがないか、システム全体を俯瞰するチェックを行う。

  特に、一部を流用している場合には、システム内に類似処理が存在することになり、それらが

コンフリクト（衝突）を起こすリスクがあることを認識する。（今回は、並行稼働する２つの運用ジョブの完了メッセージが同じものとなった。）

　このための手段としては、例えば、システム全体の運用スケジュールとプロセスやジョブの起動するシーケンス図を作り、ウォークスルーを実施する。特にメッセージやファイル名等の一意性が担保されているかチェックを行う。

- 既存仕様書への処理の前提事項の記述追加

　完了メッセージによる順序性の制御は単独処理が前提であり、並行処理する場合の留意事項の追加を記述する。

効果

　その後のシステム改修では再発防止策の観点で稼働中システムに対しても設計レビューや追加テストを実施するようになったため、以降は特に問題なく運用している。

教訓

　システムを流用開発する場合は、稼働済みの既存システムを信頼してまるごとコピーしがちであるが、メッセージやファイル名等が一意でなくなるなどの落とし穴があることが多い。既存システムでの前提条件を十分把握し、そのまま流用可能な部分と変更が必要な部分を調査分析し、可能であれば既存システムの開発者も交えてレビューすることが大切である。