ＰＣアプリケーション開発に伴うソフトウェア資産の管理とその保守運用について

（株）第一システムプロダクト

■ 執筆者Ｐｒｏｆｉｌｅ ■

網　代　忠　明

1985 年　（株）第一システムプロダクト入社
システム業務担当

■ 論文要旨 ■

複数の開発者でアプリケーション開発を行う場合に共有のスクリプトを用いる事が一般的である．この場合の利点としては生産性の向上・メンテナンスの容易さなどが揚げられる．

旧来のメインフレームによるシステム開発においては非常に有効な手法として共通のアプリケーションを多人数で利用する手法がとられてきた．

近来のパソコンを開発用に利用する場合は各人のパソコン上にソーススクリプトや開発部品が存在する事で，システム全体の整合性が取れなくなってしまう．

それを回避してシステム全体の整合性を取りつつ開発運用を行う方法を論ずる．

■ 論文目次 ■

１．はじめに ……《 3》

２．開発中の資産管理の問題点 ……《 4》

３．資産管理の現状と課題 ……《 4》

４．資産管理の解決方法 ……《 6》

５．評価 ……《 9》

６．おわりに ……《 9》

１．はじめに

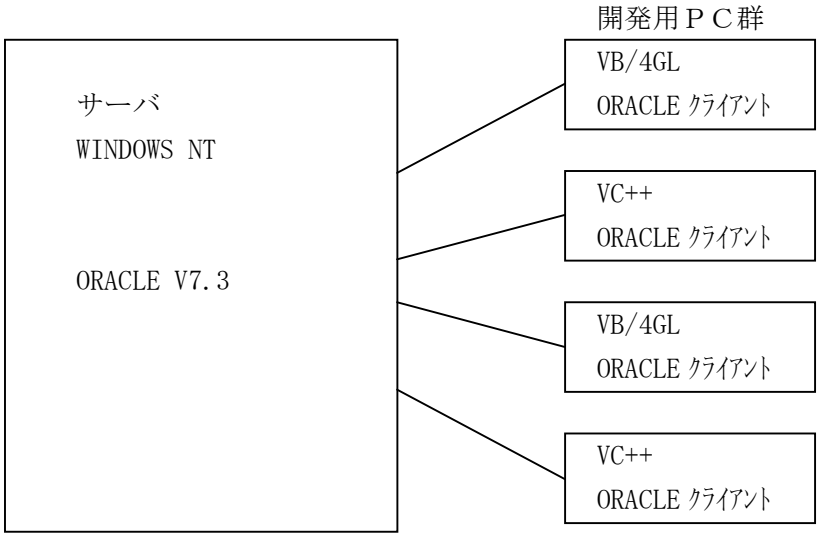
ユーザアプリケーションシステムの開発を業務としている当社としては，従来はメインフレームでの開発を主としてきた．一般的にメインフレームの場合は端末によってホストコンピュータ上のソースプログラム・画面定義体・帳票定義体を用いてシステム開発を行ってきた．

しかし，近年のパソコンでのシステム（クライアントサーバ）開発において共通部品を作成した後，多人数でそれを共有して開発を進めていった場合に，各人のパソコン上に開発環境を作成して開発を行ったために整合性が取れなくなってしまった．

そのため，顧客先に対してのメンテナンス作業でソフトの入替え時にアプリケーションソフトの不具合やバージョンの違いが多発してしまい，提供アプリケーションの品質を落とす結果をもたらした．

本論では，パソコンを使用したアプリケーション開発の開発面と運用面からの管理について論ずる．

開発環境について



２．開発中の資産管理の問題点

サーバとパソコンを使用してのシステム開発に伴う開発環境の不整合を解消して開発中のアプリケーションの品質管理を徹底するとともに提供ソフトのバージョン違いや不具合を起こさないようにしなければならない.

３．資産管理の現状と課題

一般的に品質管理を行う上での手法としては，ドキュメント類の整備・ソースコードの標準化などが挙げられるが実用な面で効果的とは言えない部分が多々ある.

これはパソコンが従来はその名の通りにパーソナルであるという性質からくる事で，各開発者自身が個々のパソコン上で資産を保持してしまう事に原因があると考えられる.

また，顧客に対してアプリケーションの配布，メンテナンス，新規パソコンの増設での開発ソフトウェアインストール時の組込みでのヒューマンエラーなどに伴う動作の不具合・製品の品質にばらつきが出てしまう事への配慮をしなければならない.

そこで開発面と運用面から問題点について考えてみる.

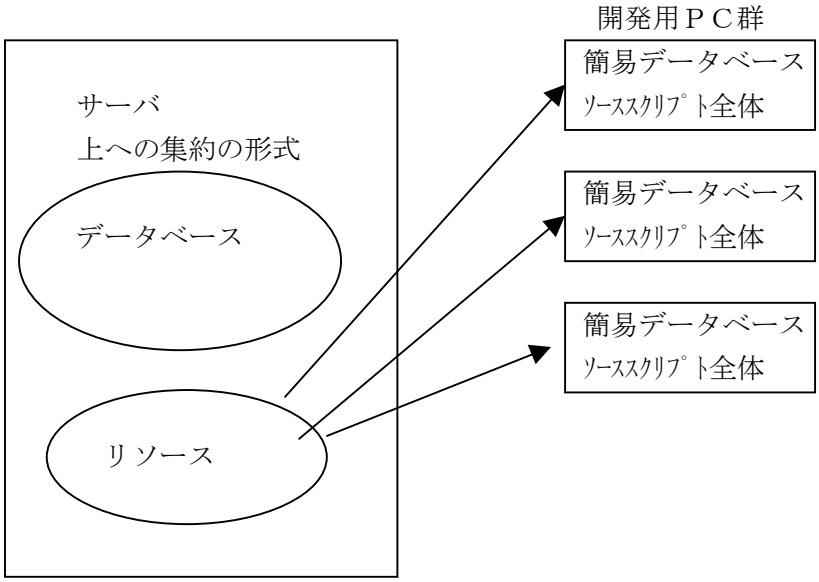
開発面からの問題点

- 開発者個々人のパソコン上に最新のソーススクリプトが散在する為に，資産の非互換が発生する

運用面からの問題点

- 弊社ではパッケージ形式でのソフト販売を行っているが，そのカスタマイズ要件によっての固有修正分が他の顧客のシステムに紛れこんでしまい安定稼動ができない

従来の管理方法について



手法　　個々のＰＣ上で単独での開発を行う

利点　　1.　他者の進捗状況に左右されずに開発が可能である為
開発者個人が自分の速度に合わせた形で開発ができた.

2.　顧客先にデモ等で出向く場合に個人のＰＣを切離して
単体での作業が可能となる.

欠点　　1.　開発者個人のＰＣ間での誤差が生じる為，成果物として
の品質に欠ける可能性がある.

４．資産管理の解決方法

１．開発面

① 資産管理

開発中のシステムソーススクリプトは必ずサーバ上で管理する．各開発者のクライアント上でソース管理を行う事はソース管理の上で危険度が大きい．
管理するべきソーススクリプトとしては，開発ソースコード・印刷レポート・データベース定義体などがあげられる．これらの資産を全てサーバ上に集約する事によってソーススクリプトの一元管理を行う．

実際の管理方法としては，各クライアント上にソーススクリプトの保存は厳禁とする．サーバ上で管理するソーススクリプトを保存用のフォルダと開発中のフォルダの２つのフォルダの配下に置き，開発者が手を加えるソーススクリプトは常に開発用のフォルダとする．開発が完了した時点で開発用のフォルダから保存用のフォルダに移行する事になるが，その際に移行前の状態の保存用フォルダのバックアップを取得しておく．このバックアップ処理によってソーススクリプトの世代管理を行い，一定期間の保存をしておく．これによって万一修正による不具合が生じても前の世代に戻る事により，最低でも修正前の状態までには戻す事ができるようになるので，顧客に対しての保全にもなる．

システムで使用する印刷用レポート定義体・データベース定義体においてもソーススクリプトと同様に保存用のフォルダと開発用のフォルダに別けて管理を行う．

パッケージ形式での販売でのカスタマイズ要件に伴うスクリプト・各種定義体の固有仕様分についても分離させて管理を行う必要がある．
フォルダ上での管理形態としては共通部分のフォルダと，各顧客固有仕様の部分とに分類してソーススクリプトの管理を行う．これにより固有仕様を含む顧客の資産を再現する場合には，まず共通部分のスクリプトを展開させた後，固有仕様の部分を上書きする事によって展開される．従って共通部分と固有仕様のファイル名は同一にしておく必要がある．

これにより顧客に提供済みのソースコードを完全に保護する事が出来る．

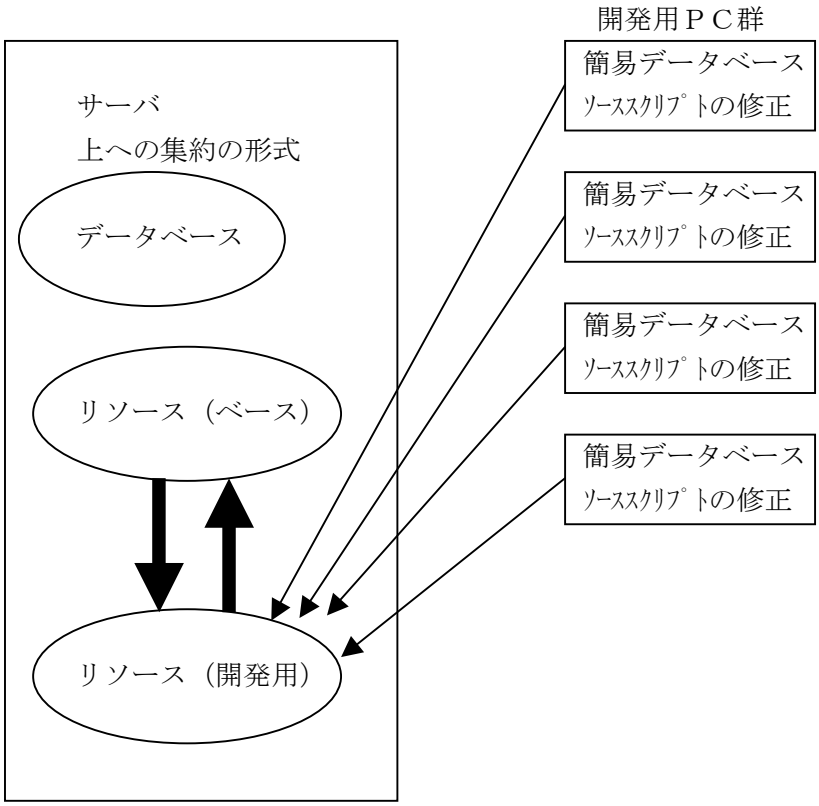
② 開発手法

さて，上記①の形式で管理された資産を用いての開発手法であるが，一番問題となってくるのが共通部分についての修正が発生した場合に多人数で同一のスクリ

プトを使用している事がよくある．このときに開発者同士での不整合が発生する可能性が高くなる．

開発方法としては，今誰がどのスクリプトを手にしているかを認識していく事で不整合を避ける．そのためには，ＥＸＣＥＬなどの表形式でそのソースの所有者を明らかにしておく．
当然ながら，その所有者以外はどの開発者も該当のスクリプトに変更を加える事は出来ない．

改良後の管理方法について



手法　　サーバ上にあるベースのリソースを開発用に COPY する．
開発用のソーススクリプトをクライアントで修正し内容確認の後，
ベースのリソースに COPY する．

利点　　1.　ソーススクリプトは常にサーバ上に存在しているので
サーバ上のスクリプトは正しい．

欠点　　1.　共有する形をとるので個人での自由度に欠ける．

２．運用面

①　考慮点

運用面での考慮点としては，リリースするべき EXE ファイルや，レポートの定義体などの世代管理を徹底する事である．
各顧客に対して配布した資源はその内容によりそれぞれの固有修正やバグ対応がなされている．しかも一時期のみではなく数度にわたる変更修正が発生するのが一般的だ．顧客側での世代管理の必要性はないが，開発元もしくは顧客のサーバ上に変更履歴と合わせて資産をその都度保存しておきいつでも元に戻せる状態にしておく．ＷＥＢ形式のシステム開発については各パソコン単位での不整合は発生しえない（但し，ブラウザのバージョン違いについては考慮が必要となる）がクライアントサーバ形式のシステムについては各パソコンごとに資産が存在するためにそこで不整合・バージョン誤りによる製品の不具合が起きてくる可能性が高い．

②　アプリケーションソフトの配布

各パソコンに対してアプリケーションソフトを設定する場合，ベースとなる物は全てサーバからの取込みをその手段として行う．
手法としては，タイムスタンプを見てサーバ上のアプリケーションが新しかった場合には自動コピーがＷＩＮＤＯＷＳの起動時に動くようにスタートアップへの組込みが一番簡単な方法と考えられる．
ただし，起動時の自動コピーがうまく機能しない場合を考慮して自己解凍形式の圧縮ファイルでもサーバ上に保存しておく．
これによって，何らかの事情によりメンテナンスのかからないパソコンについてはエクスプローラなどによる手動でのコピーを準備しておく事ができる．
顧客のスキルによってはコピー自体が得手としない場合もあるが，自己解凍にする事によってコピー先の指定を省略する事ができる．加えて前出のタイムスタンプでの判定とは違って無条件のコピーとなるのでファイルは存在するがその中が壊れていたなどの障害時の対応としても有効である．

５．評 価

開発当初は予期しない共通部分のソーススクリプトの変更，修正に伴うバージョン違いによるアプリケーションソフトの不具合，修正元となるスクリプトの消失などの問題が多々発生したが，フォルダ分けによる管理を徹底した事によってかなりの部分での不具合の発生が抑えられた.

但し，開発担当者の立場からした場合には各人の開発期間が自分の原因以外の要因で延びる可能性があるので評判としては今一歩である．しかしリソースの保全は重要な課題である事を認識してもらった.

６．終わりに

開発のためのソーススクリプト自体が完全にロックされる方式になれば完全と言えるのであるが実際の開発においては不可能な話である.

しょせんは決まり事で動いていく事であるので，開発担当者各人の努力が必要不可欠である.