6A-7

# ＩＳＶのバージョンアップの影響を回避し保守性を向上するアプリケーション構築基盤の提案

大江信宏† 渡辺 透† 小泉寿男†
特定非営利活動法人Ｍ２Ｍ研究会†

## １． はじめに

ハードウェアメーカと独立したソフトウェアベンダが提供するソフトウェアのことを総称してここでは ISV と呼ぶ。情報システムは，PC サーバと OS やミドルウェアなどの ISV の上に各種業務用のアプリケーションが稼動するという構造である。ISV はそれぞれ新しい技術や機能に対応するためにバージョンアップを繰り返し，またセキュリティ対策としてのアップデートも頻繁に行う。そのためにアプリケーションの互換性が失われたり，動作が変ったりといった問題が生じることがある。

このような ISV のバージョンアップの影響を可能な限り回避できると，それだけ情報システムの保守コストは削減でき，業務の改善や新しい機能の開発にコストをかけられるようになる。生産性向上ツールとしてアプリケーションソースコードを自動生成する製品[1][2][3]は多いが、ISV のバージョンアップを明示的に回避することはあまり考えられていない。本論文では，そのためのアプリケーション開発と実行の構築基盤を提案する。提案方式では，アプリケーションの開発を設計情報からのソース自動生成と，その実行基盤のミドルウェアを構築基盤として提案し，ISV のバージョンアップに対して，実行基盤のミドルウェアでまず吸収することで，アプリケーションの互換性を維持し，さらにそれだけでは対応不可の場合には，設計情報からのソースの再生成によって最新の ISV 環境へ対応するという手段によって，影響を回避することを提案する。

## ２． ISV バージョンアップによる課題

ISV はハードウェアの発展や新しい技術に対応した機能を開発・提供し，ユーザはそれを活用することで新しいニーズに対応したアプリケーションを開発する。しかし従来稼動していたアプリケーションの互換性に影響が出ることがある。ISV のバージョンアップは新しい機能を使いたい場合だけではなく，保守サポート上の理由やセキュリティ上の理由から，実施する必要があることも多い。また基幹業務系のシステムなどは寿命が長く，一般に 7 年から 10 年ほどは継続して使われることが多いが，一方ハードウェアの保守期限は 5,6 年というのが一般的であり，ハードウェアを置き換えた際に，同時に OS やミドルウエアを最新のものにする必要があるというケースも多い。しかしこれらのバージョンがアップされることによりアプリケーションが動作しなくなるとか動きが異なるなどの問題が発生する。これらの問題に対応するため，事前にテストを十分に行い，特にハードウェアをリプレースする際には入念にリプレース計画を立てて行うのが一般的である。しかしアプリケーションの作成方法によっては想定していない障害も起こる場合があり，ISV のバージョンアップには多くのコストがかかることが多い。

## ３． ISV バージョンアップの影響を回避

本論文では，ISV バージョンアップの影響を回避するためのアプリケーション開発と実行の基盤を提案する。提案方式は，図 1 に示すように設計機能とソース自動生成機能からなる開発環境と実行環境とから構成される。

### ３．１ 開発環境

開発環境は設計機能とソース自動生成機能から構成される。設計機能は，OS，データベース，言語に依存しない形とし、基幹系業務アプリケーションの特性から，項目設計，DB 設計，画面設計，帳票設計，ロジック設計という設計を行い，それぞれの設計情報をリポジトリに保存する。項目設

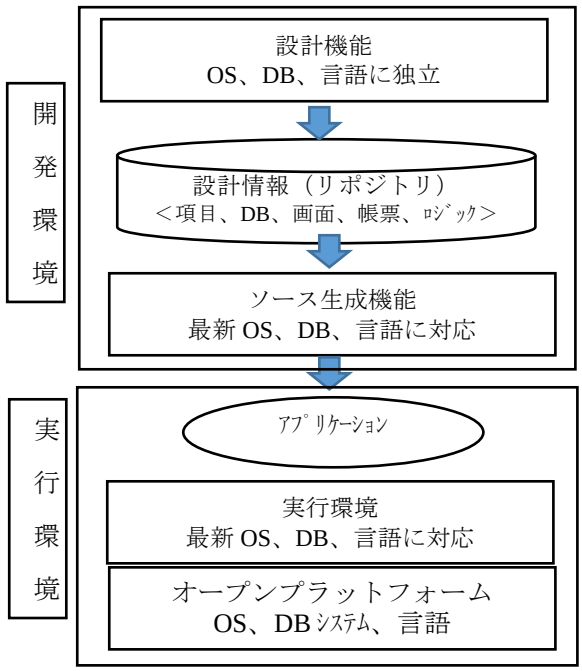


図１ 開発環境と実行環境

Application infrastructure to improve the maintainability to avoid the impact of the upgrade of ISV
†Nobuhiro Ohe†, Toru Watanabe†, Hisao Koizumi†
M2M Study Grouip（NPO）†

計は，他のすべての設計の元になる情報であり，これを変更した場合はその影響を受ける他の設計情報への反映の有無を確認できるようにする。ロジック設計は，主として画面の入出力処理を行うクライアントプログラムウィザードとﾋﾞｼﾞﾈｽﾛｼﾞｯｸを受け持つサーバプログラムウィザードとに分けて行い，サーバで動作するビジネスロジック部分は，指示書方式[4]の日本語スクリプトで設計する方式とした。設計情報からプロラムソースは自動生成する。生成するプログラムソースは，DB 設計についてはデータベースのテーブル生成ソース，画面設計は画面への表示，入力と入力形式やデータチェックロジックを含んだソースを生成する。ロジック設計情報は，DB，画面，帳票の設計情報を参照しており，これからアプリケーションのソースを生成する。それぞれの生成においては，データベースや言語の種類に応じたソースを生成する。設計情報は OS,DB，言語に依存しない情報であり，これに基づき実際に稼動する環境の OS,DB, 言語に応じたソースを生成するため，設計情報そのものは ISV の種類やバージョンへの依存がないという特徴を持つ。

### ３．２　実行環境

プログラムソースの生成と同時に，実行モジュールが生成され，OS や DB との間のミドルウェア層として実行環境があり，その上で実行する。この実行環境が，基幹業務アプリケーションに便利な機能や，DB や OS とのインタフェースを仲介するため DB の種類への対応やバージョンアップによる非互換部分を吸収するのに有効である。

## ４．　実装評価と考察

構築基盤の実装は，OS が Microsoft Windows server，DB が SQL Server および OracleDB，言語は VisualBasic を前提として実装した。Web アプリケーションを作成する場合の実装構成を図 2 に示す。この構築基盤を用いたアプリケーション・システムにおいて，発生した事例とその際の構築基盤としての対処を表 1 に示す。このような問題が発生した場合の原因究明から対応策をアプリケーションシステムごとに行った場合，会計システムの事例では，5〜6 人月の人工がかかったという例があ

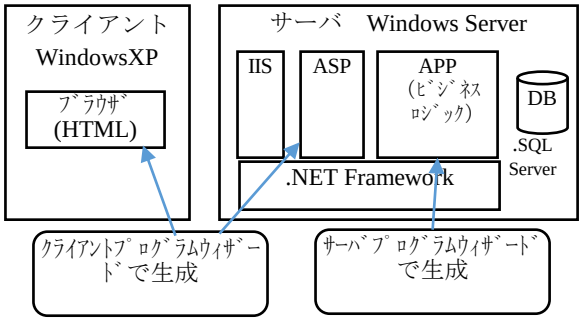


図２　Web アプリの実装構成

表１　ISV バージョンアップによる現象と回避の事例

| 現象・原因 | 本方式の対処 |
|---|---|
| ServicePack 適用<br>→動作していたﾌﾟﾛｸﾞﾗﾑがｱﾎﾞｰﾄ<br>原因：VB ｵﾌﾞｼﾞｪｸﾄの呼び出し処理における CreateObject 命令の非互換 | ソース自動生成機能でカバー（設計情報変更は不要） |
| VB バージョンアップ<br>→演算結果が異なる（int 関数）<br>原因：VB4 と VB5 で引数の型が非互換 | 実行環境でカバー |
| Oracle バージョンアップ<br>→アプリケーションエラーが発生<br>原因：SQL 文投入結果の解釈における仕様の非互換 | 実行環境でカバー |
| ODBC ドライバの版を変更<br>→動作していたﾌﾟﾛｸﾞﾗﾑが旧 OS バージョンの場合のみアプリケーションエラー | 実行環境でカバー |

る。本提案の構築基盤の上では，その基盤自身のバージョンアップ対応により問題を回避することができた。

表 1 に示すとおり，アプリケーション自身の修正を行うことなく，構築基盤の対処にて問題は解決した。通常は実行環境の対処によって問題が解決されることが多いが，それでもだめな場合にはソース生成をやり直すということでの対処が可能であり，ISV バージョンアップの影響を２段階で回避することができた。

長く使用する基幹業務システムはアプリケーション機能の保守改良，機能追加が頻繁に発生し，そのためのコストはやむを得ないが，ISV のバージョンアップに伴うコストはできるだけ抑えたい。そのための対策として本提案の構築基盤が役に立つと考える。

## ５．　まとめ

既存の情報システムの保守性を向上するために，ISV のバージョンアップの影響を回避するアプリケーション構築基盤を提案した。新しい技術を活用した新たな機能を追加していく上でも，有効であり、またスマートデバイスなどを端末とする新しい構築方式がオープンソースを含めて多く出てきているが，そうしたツールやフレームワークとも共存していけると考える。

## 文　献

(1) 金子数馬、添野元秀、名本聖矢：”訪問看護事業所支援システムの GeneXus による開発効果”， 情報処理学会全国大会講演論文集 2013(1), 341-343, 2013-03-06
(2) アプリケーションの自動生成に挑む 南米のツールを基幹系に活用, 日経コンピュータ (766), 84-91, 2010-09-29 日経 BP 社
(3) WEB PERFORMER,　HTTP://WWW.CANON-SOFT.CO.JP/PRODUCT/WEB_PERFORMER/ キャノンソフトウェア株式会社,　2011
(4) 三菱電機プログレスⅡ, http://museum.ipsj.or.jp/computer/ofos/mitsubishi/0005.html