ＩＳＶのバージョンアップに強い情報システム開発手法

大江信宏、　伊藤正裕、　上田尚純
三菱電機　情報通信システム開発センター

企業の基幹業務システムのアプリケーション・パッケージをオープンシステムで構築するケースが多くなっているが、変化への対応という観点から見た時に課題となるのは次の２つである。

・オープンシステムの市販ソフト（OS、データベース、言語など）のバージョンアップへの即時対応性

・業務システムの拡張・変更、カスタマイズへの対応性

前者は、バージョンアップによる動作評価・改修に時間を費やしている現状をどう解決するかという課題である。後者は、パッケージの適用におけるカスタマイズと以後の変更をいかに効率よく行うかという課題である。これらの課題に対して考察するとともに、基幹業務系パッケージを対象にＩＳＶ[1]のバージョンアップに強いソフトウェア開発概念とそのツールについて論ずる。

Development technique of information systems in third party software frequently updated

Nobuhiro Ohe,　Masahiro Itoh,　Takasumi Ueda
Mitsubishi Electric Corp. Information & Communication Systems Development Center

There are two problems to correspond change of business environment, when business applications are often built on open systems.

・ Can be easily upgrade and maintain of the application package, when the third-party software(OS, DB, Language, etc) is upgraded.

・ Can be correspond to customize, expand, or change function of business system

This paper describes how to solve these problems and introduces tools to give one of solutions.

[1] ISV: メーカに依存しない独立系ソフトウェア会社のことであるがここではそのソフトを指している。

１．はじめに

情報システムの導入は、財務・会計、人事・給与、生産管理、販売管理など定型業務での合理化・効率化目的のシステム構築に目処がつき、より困難な経営企画・戦略立案目的に移ってきている[1] [2]。情報処理プラットフォームのオープン化により、従来の会社毎の個別の手作りシステムから大小様々なアプリケーション・パッケージ（以降、パッケージと言う）を利用する傾向が増大し、合理化目的の情報システムはパッケージで構築することによってデータウェアハウス等の構築をも容易にし、より戦略的なアプリケーションへの連携を指向している。しかし一方現実を見ると、ハードウェアプラットフォームは PC サーバに代表されるように急速な性能向上、低価格化が進展し、それに伴うオペレーティングシステムやミドルウェアの頻繁なバージョンアップによって、その上に搭載されるパッケージの維持コストが増大するという状況が起こっている。すなわちパッケージを構築・提供するソフトウェア会社にとっては、対象領域でのノウハウの蓄積によって得た業務知識、アプリケーションを元に、さらに業務革新や、戦略的情報システムへと指向するが、一方で、ハードウェア、ソフトウェアの変化が激しいため、その対応、評価・改修に人手を取られているという現象が起こっていると考えられる。コンピュータメーカが OS、ミドルウェア、アプリケーションを提供していたプロプライエタリの世界では、あまり問題視されなかったことがオープンの世界では問題となっているのである。

本論文では、パッケージを構築・提供するソフトウェア会社にとってプラットフォームのバージョンの変化による影響を最小限に押さえ、さらにパッケージを使用する会社対応のカスタマイズをより効率的に行える情報システム構築手法について考察し、その１つの指針として CASE ツールを紹介する。

２．バージョンアップ問題とその対処

２．１　アプリケーション・ライフサイクルと OS ・ミドルウェアのバージョンアップ

パッケージが提供する定型的な業務は、その寿命が一般に長く通常でも５年以上は使い続けられる。ただしその間に発生する法改正、業務変更・拡張といった修正は行われていく。その一方、PC サーバは、ハードウェア性能が急速に伸びており、同時にオプレーティングシステム（OS）およびデータベース（DB）、言語などのミドルウェアも機能的にどんどん改善され、頻繁にバージョンアップされるという結果になっている。図１に主要 OS、ミドルウェアのバージョンアップ遷移を示す。これにさらにサービスパック（SP）の発行による変化も追加される。

WindowsNT のバージョンアップ

94/9	94/12	96/1	96/12	
3.1	3.5	3.51	4.0	2000

Oracle のバージョンアップ（WindowsNT 版）

94/6	95/9	96/1	96/7	97/9
7.0	7.1	7.2	7.3	8.0

Microsoft SQL Server のバージョンアップ

94/2	96/1	96/8	99/2
4.2	6.0	6.5	7.0

Visual Basic のバージョンアップ

93/3	95/12	97/5	98/10
2.0	4.0	5.0	6.0

図１　主要OS、ミドルウェアのバージョンアップ遷移

オープンなサーバ上では様々のミドルウェアやアプリケーションの最適なものを選択してシステムを構築できるためユーザにとって大きな魅力であるが、このようなマルチベンダシステムは、バージョンアップという側面からは弱点が多い。バージョンアップによって、組み合わせの問題、互換性欠如などの問題が生じるからである。たとえば新しい業務アプリケーションを追加するためにミドルウェアをバージョンアップすると従来動いていた業務アプリケーションが正しく動作しなくなるという現象が発生する。このため、必ず従来アプリケーションも評価・検証し、改修を行わねばならないということを繰り返すことになる。最近のコンピュータ系雑誌等にもそのような事例紹介がたくさんされているが、ここではまずこれらを整理し、その対策を考えてみたい。

２．２　バージョンアップが必要になる理由

ソフトウェア会社や利用者はなぜバージョンアップを行うのであろうか？　それは　①新しい機能を使いたい　②　性能向上を期待　　③　新たに導入するソフトが新版にしか対応していない　④古いバージョンがもはや入手できない　などの理由がある。

２．３　バージョンアップによる問題点

表１に示すような７つの問題点が指摘できるが、OS やミドルウェアの提供者側がバージョンアップすることに起因する問題と、利用者側で適用する時に起きる問題の２つに大別できる。

表１．バージョンアップに伴う問題

要因	No	パッケージ提供するソフトウェア会社および利用者の問題点
OS/ミドルウェア提供者側が製品をﾊﾞｰｼﾞｮﾝｱｯﾌﾟすることに起因する問題	①	旧版の製品でシステムを統一したいが製品が入手できない
	②	旧版の製品の保守サポートがなくなるため問題発生時の対処ができない、あるいはできなくなる恐れがある
利用者側がﾊﾞｰｼﾞｮﾝｱｯﾌﾟした製品を適用することに起因する問題	③	システムDLLの整合性（DLL：Dynamic Link Library）が取れなくなりシステムが不安定になる
	④	データベースの互換性、新たなバグ、バグの修正による動作の互換性がなく、正しく動作しない
	⑤	開発言語の言語仕様が変更され正しく動作しない
	⑥	バージョンアップにより、原因不明だが動作不正。試行錯誤的に修正。
	⑦	ミドルウェア間の相性あるいはバージョン不適合による動作不正。

これらの問題に対してパッケージを提供するソフトウェア会社や利用者はかなり困っているはずであり、実際 98 年 1 月の日経オープンシステム等の読者対象実態調査ではバージョンアップを行った企業の半数以上がなんらかのトラブルに遭遇しているという結果が出ている。

①の入手できない問題そのものは救えないが、新版での動作が保証されれば、問題は回避できる。②も①と同様である。しかし保証されない場合は、アプリケーションを修正することになる。③は、経験的にはミドルウェアインストール順の確認や整合性を検査するツールがあれば問題が大きくなる前にある程度回避できる。具体的にそういうツールを提供している製品もあり、トラブ

ル発生時の問題早期発見につながっている。④⑤⑥⑦は、アプリケーションをいろいろ調査して原因を追求し、ある場合には試行錯誤的に修正して対処するといったことが行われており、次の事例にも挙げているが、バージョンアップ問題として解決すべき対象といえる。

２．４　バージョンアップに伴うコスト

トラブルの調査やアプリケーションを修正にどのくらいの時間がかかるのかを事例で紹介する。

事例１：会計・給与パッケージ

Visual Basic＋Oracle7.2＋WindowsNT3.51 で動作していたが、WWW サーバに接続してシステムの一部の情報を利用者からも参照できるようにするために、OS を Windows4.0 に、同時に Oracle も WindowsNT4.0 では 7.2 が保証されていないためバージョン 7.3 に変更した。結果として、会計システムアプリケーションが動作しなくなった。この不具合の追求、改修、検証に５～６人月を要した。

事例２：原価管理システム

Visual Basic2.0 の時代に、データベースアクセスメソッドがなかったため、英語版の 3.0 を使用したが、今後のバージョンアップで VB5.0 にするにも移行方法が明確になっていないため、かなりの部分作りなおしになると考えられるが、見積もりも困難な状況になっている。

上記の事例は、事例１は該当アプリケーションそのものはなにもしなくても本来良かったが、別の理由でバージョンアップしたことが結果的に、トラブルを引き起こした。事例２はアプリケーションそのものを機能拡張などを行うときに、バージョンアップされたミドルウェアでは作れないため、いつまでも過去のバージョンをとっておく必要がありかつそれらの保守サポートがなくなってしまうため、どうにもならなくなることが目に見えている。

２．５　バージョンアップに強いソフトの構築法

ビジネスアプリケーションの場合、その構造を見ると、処理の中心にデータベースがあり、入出力としては、対話型処理を行う画面操作、帳票出力があり、ビジネスルールを実行するビジネス処理部分があるという構造を取る。画面操作特に最近はグラフィカルユーザインタフェース（GUI）を駆使したアプリケーションが増えているが、パソコンの Windows 環境では、OS のバージョンアップやビジュアル系の言語のバージョンアップによる影響を受けるケースが多いし、サーバ側と異なりクライアント端末となるパソコン側は、利用者が別のソフトウェアを導入するなどによってクライアント毎に異なる環境となるケースが多くこれもアプリケーションの動作保証という観点で見ると影響の出やすい部分である。これらはビジネスルールとは独立させておけばその影響を最小化できる。このようにこれらの処理を明確に分割した設計を行うことが、バージョンアップの影響を局所化するという点で重要なポイントである。この形態を実践するのが 3 階層クライアントサーバシステムである（図２）。初期の C/S システムは、クライアントにすべてアプリケーションを置き、データベースサーバと通信する２階層 C/S 構造でで構築されるケースが多かったが、ツールが整備されてきて作りやすくなったことから最近では３階層型が増えて

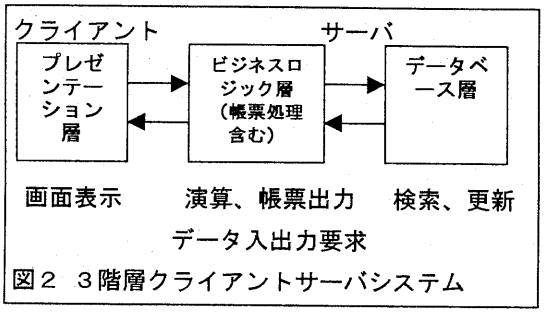

図２　３階層クライアントサーバシステム

きている。

最近では 3 階層型システムを効率良く開発するツールが多く出てきている。開発支援ツール（CASE ツール）は、言語やデータベースに依存しない設計情報から、実際に動くプラットフォームのプログラムソースを自動生成することができるので、これらを活用することによって、初期の生産性だけでなく、バージョンアップ問題にも柔軟に対処することが可能になる。

３．カスタマイズ問題とその対処

３．１　パッケージソフトのカスタマイズの状況

情報サービス産業の白書[3]によると、いわゆるパッケージソフトウェアと分類されるものには、ワープロソフト、表計算ソフトなども含まれこれらは通常カスタマイズは発生しないが、販売管理、会計、給与などの業務アプリケーションになると企業毎にカスタマイズが発生してくる。白書によれば、カスタマイズを事業者にしてもらう場合と、自社でカスタマイズをする場合を含めると、約４３%がカスタマイズを必要としている[3]。このアンケートには、上記のワープロ等のカスタマイズしないものも含まれていることから考えると、業務系アプリケーションパッケージの場合にはもっと高い確率でカスタマイズしているものととらえることができる。

３．２　業務系アプリケーションパッケージの種類と利用状況

業務系アプリケーションパッケージの種類には、財務会計、給与・人事、在庫管理、手形・売掛金管理、営業支援、経営支援、販売管理、業務統合などの種類がある。これらのユーザ利用状況を見ると、財務会計、給与・人事といったパッケージはよく使われており、逆に販売管理パッケージは、開発事業者側の投資が多い割には、ユーザの利用が少ないという結果が出ている[3]。これは、ユーザが販売管理業務を行っていないのではなく、パッケージを使わない手作りシステムで構築しているケースとパッケージとは呼んでいないが販売会社が保有する販売管理のベースシステムを元に構築をするケースが多いというようにとらえるべきである。実際にオフコン系システムを販売している販売会社での取り扱いシステムの半分以上が販売管理システムであることからそれが言える。しかしオープンの時代になってこのような手作りシステムではコスト的に見合わないため、パッケージをベースにカスタマイズするというビジネスが増えてきていることは事実で、販売管理パッケージも然りである。

３．３　カスタマイズの概要

業務アプリケーションの処理分類を、エントリ、計算処理、問い合わせ、データベース更新、帳票作成、データベースメンテナンスと分類すると、これは経験値であるが、帳票関連がもっともカスタマイズが多い。表２に代表的な業務パッケージでのカスタマイズ傾向を示す。

表２

パッケージ	カスタマイズが多いプログラム種別	理由
会計パッケージ	帳票作成：30～50% エントリ：30%	帳票：パソコンとの連携による分析、客先指定フォーム、部門別損益計算　など エントリ：各種自動仕訳、入力項目チェック機能など
給与パッケージ	計算処理、帳票作成： 各 30%程度	企業毎に支給算式が異なる 支給明細書、給与支払いからの自動仕訳など
販売管理パッケージ	帳票、特に管理帳票系多い。 エントリ系	ビジネスルールが複雑（企業によって異なるケースが多い）。小売り業、卸売り業等による違いは業態、顧客層（不特定多数と、特定小数）、支払い形態などの違いが大きい。同じ卸売り業でも取り扱い商品やその数え方などの違いなどで変更多い。その他販売統計などでは次元が増えるほど統計帳票が増える

３．４　カスタマイズのアプローチ

２つのアプローチがある。１つは、すべてのカスタマイズを想定し、すべてパラメータで設定可能とする。もう１つはシステムの細かなロジックを追わなくても簡単にカスタマイズできるパッケージである。パラメータ設定方式は、パラメータの個数が多くなく、パラメータ間の従属関係が少なければ売り切り販売に向いている。しかしすべてのカスタマイズには耐えられない。ＥＲＰパッケージには数千個のパラメータを設定するようなものもあるが、パッケージ構築にも相当の投資と期間がかかる。一方、簡単にロジックやデータベースをカスタマイズできるようにする方式は、パッケージ開発者にとっては比較的容易な方法であり、カスタマイズパターンを蓄積することでソフトウェア会社の事業基盤とできるという利点がある反面、いかにカスタマイズが簡単なパッケージにするかという点がポイントである。

３．５　カスタマイズを簡単にするための条件

以下に筆者が考えるカスタマイズを簡単にする条件を述べる。

① 作成されたパッケージが理解できるための仕様書が整備されていること
② プログラムそのものを変更することが容易にできるよう読みやすいこと
③ データ項目に対する定義・参照関係を管理できる辞書機能を保有すること
④ さらにビジネスルール、機能部品レベルの管理ができる辞書管理機能を有すること
⑤ カスタマイズする範囲が階層化できていること。これは例えば、まとまった業務ロジック（ビジネスルール）の単位で変更できることおよび、さらに下位の機能部品レベルで変更可能なこと

⑥ 機能の階層化とともに、バージョンアップのところでも取り上げた３階層構造は、カスタマイズの点でも有効

４．市販ソフトのバージョンアップに強く、カスタマイズを容易にするソフト開発の概念

上述してきたバージョンアップ問題、カスタマイズ問題の解決策として、以下のソフト開発の考え方を提案する。

すなわちアプリケーション設計情報を言語のバージョンやデータベースの種別に依存しない上位レベルの概念で保持することによってこれらの問題を解決する。このような開発支援ツール（CASE ツール）を利用することで、ソフトウェア開発会社の財産であるアプリケーションを資産として蓄積することが可能となる。資産継承の概念図を図３に示す。

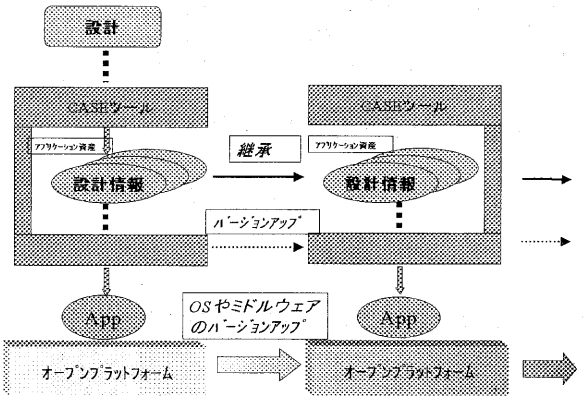

図３　CASE ツールによる資産継承の概念図

図３で、新しいプラットフォームに対応するには CASE ツール自身もバージョンアップに対応する必要があり、そこにはタイムラグは発生するが、アプリケーション・システム毎にかかるコストに比べれば CASE ツールがまとめて対処することでトータルのコストを削減することが可能となる。これはソフトウェア開発会社や利用者にとっては大きなメリットとなる。

またカスタマイズ問題への対処としても、CASE ツールは威力を発揮する。すなわち、プログラム機能の部品化、データ項目やプログラムの辞書管理機能、仕様書自動生成機能、プログラムの可読性向上機能などがその対象となる機能である。

最後に実際の製品例として、三菱電機では、ビジネスアプリケーション開発システム「HYPERPRODUCEⅡ」を製品化しており、上述したバージョンアップ問題やカスタマイズ問題への対処機能を実現している。

HYPERPRODUCEⅡで開発したシステムではどのような形で設計情報が使われ、どのようなアプリケーションが自動生成されるかを簡単に示したのが、図４である。

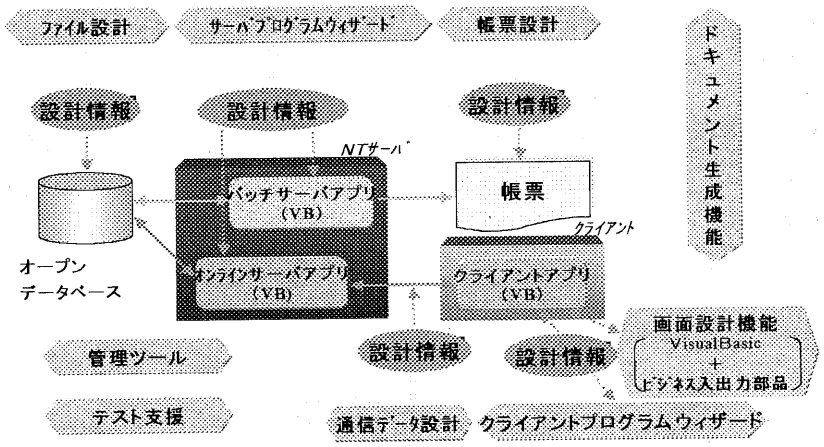

図４　HYPERPRODUCEⅡのシステムアーキテクチャ

５．むすび

CASE ツールを導入することで、開発言語そのものを用いてプログラムを記述するのとは異なり、バージョンアップによる問題発生時の対応が短期間で済む。CASE ツールを用いても最終的な確認試験は必要だが、自動生成はパターン化されたソースコードを生成するため信頼性が高く試験も容易に完了できる。カスタマイズの点でも、ドキュメント自動生成やデータ項目辞書を管理するリポジトリ管理機能により効率が上がる。これらの評価については、紹介した製品の例をもとに評価段階でありまた別の機会に述べたい。

CASE ツールの利用は、生産性、保守性を向上させることが目的だが、オープンシステムによる情報システム構築、パッケージベンダによるパッケージシステム構築には欠かせない道具であると考える。

参考文献

[1] 魚田勝臣『パッケージビジネスの確立と展開について』情報処理学会第５７回全国大会, 1998.10

[2] 『情報サービス産業白書 1998 年版　概要』, 社団法人　情報サービス産業協会, 1998 年

[3] 『情報サービス産業白書 1978 年版　図表』, 社団法人　情報サービス産業協会, 1998 年, 図表（図表 2-2-51、3-1-34）

[4] 伊藤正裕, 大江信宏　『バージョンアップに強いソフト開発』, 日経オープンシステム：オープンセミナー, 1998.8, 224-231 ページ