Visual Basic プログラム資産の
マイグレーションの実践

（株）滋賀富士通ソフトウェア

■ 執筆者Ｐｒｏｆｉｌｅ ■

竹下　健一

1985 年　（株）滋賀富士通ソフトウェア入社
金融機関他のシステム開発に従事

現在　　マイグレーションサービス部所属

■ 論文要旨 ■

マイクロソフトは 2008 年４月に Visual Basic6.0（VB6）の延長サポートを終了した。VB６の後継言語が Micorsoft Visual Basic .NET であるが、この新しい言語は、その特性や言語仕様の面において VB６とは大きく異なり、VB から VB.NET へのマイグレーションは思うように進んでいない。

このような状況の中、当社では、VB 資産の.NET 移行として、「CoolCat for.NET VB マイグレーションサービス」を提供している。当論文では、VB 資産を.NET へ移行する際の課題を明らかにし、品質を損ねることなく、短期間でマイグレーションを実現した適用事例について紹介する。

■ 論文目次 ■

■ 図表一覧 ■

１．はじめに

１．１　VB 資産を取り巻く状況

1988 年にリリースされた Microsoft Visual Basic（以下 VB という）は、世界で最も使われたプログラム言語の一つであり、日本国内においても 90 年台前半からクライアント・サーバシステムの構築などの言語として使用されてきた。しかし、図１に示すようにマイクロソフトは 2005 年に VB６のメインストリーム・サポートを終了し、有料の延長サポートも 2008 年４月初に終了となり、今後の VB６のサポートはランタイムのみのサポートとなっている。

VB６の後継言語が Microsoft Visual Basic .NET（以下 VB.NET という）である。この新しい言語は、その特性や言語仕様の面において従来の言語とは大きく異なっている。VB.NET は VB６に較べ機能性が充実しており、アプリケーションの安定性も強化されているため、マイクロソフトのプラットフォーム上で、運用コストを抑えたビジネスアプリケーションを構築することができるメリットがある。

しかし、VB６と VB.NET の間には言語仕様に大きな隔たりがあり、VB.NET は発売以来、約７年近くが経過しているが、VB から VB.NET への移行は思うように進んでいない。VB で構築された多くのシステムは VB.NET へ移行されずに塩漬けされたまま稼働を続けている。2008 年４月初の VB６のサポート終了、或いは、これまで VB の動作を保証していた OS をサポートするハードの販売収束により、VB 資産を使い続けるリスクがますます顕在化する状況となってきている。

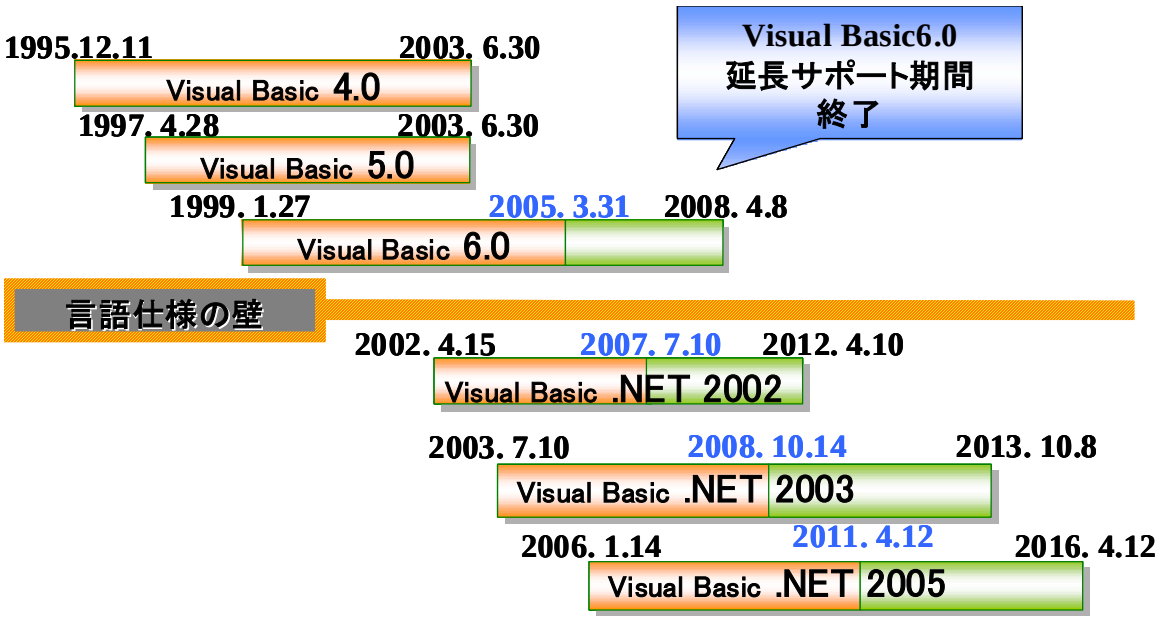


図１　VB と VB.NET のサポートライフサイクル

１．２　VB.NET への移行方法

こうした状況の中で、VB 資産を保有するユーザが既存の VB 資産を VB.NET へ移行する手法には二つの選択肢が挙げられる。一つは、システムの構築を要件定義から行い、既存システムを捨てて、新たに VB.NET でシステムを構築していく再構築手法である。もう一つは、業務仕様はそのまま継承し、新しい環境に合わせて非互換部分のソースコードを書き換えるマイグレーション手法である。

稼働中の VB システムが業務運用を行ううえで業務改善要求が多い場合は、再構築手法が一般的な移行方法である。一方、稼働中の VB システムの業務改善要求が少なく、できればこのまま使い続けたいと考える場合は、マイグレーション手法が最適な移行方法である。

１．３　当社の概要

当社は、1984 年に富士通株式会社と株式会社滋賀銀行の共同出資により設立されたシステムエンジニアリング会社である。現在では、コンピュータシステムに関わるサービスビジネスも手掛け、多様な業界各社との取引実績を積み重ねている。なかでもサービスの分野では、専用のツール「CoolCat® for .NET」（以下 CoolCat という）を用いて、企業内に蓄積された VB 資産を最新の VB.NET 資産に移行支援する「CoolCat® for .NET　VB マイグレーションサービス」（以下 CoolCat VB マイグレーション という）　を提供している。

今回当社は、製造業の A 社から同社の VB 資産を VB.NET へ移行する依頼を受け、CoolCat VB マイグレーションサービスを実施した。当論文では、このマイグレーションにおいて、私が実施したサービスの適用事例を紹介する。

１．４　Ａ社のシステム移行要件

A 社のマイグレーションプロジェクトは、他ベンダーが VB で構築した現行の受注・生産管理システムを VB.NET へマイグレーションするプロジェクトで、資産規模が約 1.5Mstep ある大規模システムのマイグレーションである。A 社では今回、現行システムへの業務改善要求が少ないことから、マイグレーション手法を選択した。業務仕様や操作性はそのままに、高い品質のまま新システムへ移行することが当プロジェクトの移行要件であった。また老朽化したハードウェアの保守サポート期限が 1 年後に迫っていたこともあり、移行期間は 1 年以内と、短期間での移行が計画された(図 2 に A 社移行スケジュールを示す)。このため、当マイグレーションプロジェクトにおいては、如何に移行リスクを抑え、品質の高いシステムを短期間で提供できるかが重要なポイントであった。

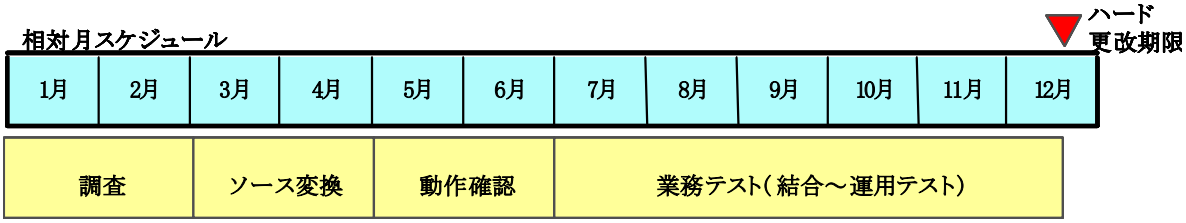


図 2　A 社 移行スケジュール

２．VB マイグレーションにおける従来の問題

２．１　手修正によるバグ混入

VB マイグレーションにおけるソース変換時のリスクは、一般的なマイグレーションツール（例えばマイクロソフトの標準アップグレードウイザード）の自動変換率が 60～70%と低く、ツールで自動変換できないソースコードに対して相当量の手修正が発生してしまう点である。手修正量の多さは開発コストに直接はね返るばかりでなく、修正を加えた箇所にバグの混入を誘発し、プログラムの品質を低下させてしまう。VB.NET 技術に熟練した技術者が修正を実施するのであればバグの混入をある程度防ぐことも可能である。しかし、現行システムの仕様を知らない経験の浅い VB.NET 技術者が手修正を行うと、ビジネスロジックにまでバグを混入させてしまう恐れがある。人手を介在したバグの混入はツールで機械的に混入したバグとは異なり、一様にバグが遍在する結果を生む。これは、後続のテスト工程におけるバグの検出が困難となるばかりでなく、マイグレーション後のシステムの品質に悪影響を与える原因となっていた。

A 社の受注・生産管理システムを変換率が 70%と仮定したマイグレーションツールで変換する場合に、必要となる作業期間を算出した結果が表１である。

	一般的なマイグレーションツール （変換率:70％）
調査	２．０ヶ月
ソース変換	８．３ヶ月
動作確認	５．５ヶ月
業務テスト	５．５ヶ月
総所要期間	２１．３ヶ月

・一般的なマイグレーションツールを用いた場合の生産性・品質モデル
　-変換できなかった箇所の修正時間を平均5分/Stepと仮定
　-テストケースの作成と消化の所要時間は平均10分/件と仮定
　-単体テストにおけるエラー修正時間は平均2時間/件と仮定
・変換できない部分のエラー発生数を富士通の大規模システムの再構築時のエラー発生数を用いて試算

表１　一般的なマイグレーションツール使用時の作業期間

この算出結果から、A 社の VB 資産の移行において、一般的なマイグレーションツールを使用すると、移行期間に 21.3 ケ月の期間が必要となり、A 社の移行要件を満たすことができないことが判る。

２．２　動作上の非互換の把握

VB と VB.NET の非互換には、コンパイル時に文法エラーとして検出できる非互換とプログラムを動かして初めて検出できる非互換の二種類がある。例えば、型の違いによる非互換はコンパイル時にエラーとなり検出することが可能であるが、画面の文字のフォントや太さの違いといった非互換はコンパイルエラーとはならず、実際に動かしてみないと検出することができない。

VB マイグレーションにおける動作確認時のリスクは、これら実際に動かしてみないと判らない非互換が、ソース変換時には見つけることができないところにある。

例えば、これら VB と VB.NET との間の動作上の非互換を把握しないままにプロジェクトの計画を作成すると、下流工程の動作確認時に思わぬ作業量が発生してしまい、稼働遅延や納期遅延を引き起こすことになる。プロジェクトの早い段階で動作上の非互換を把握しておくことが大切だといえる。

２．３　テスト工数の削減

一般的なマイグレーションツールを用いたマイグレーションにおいて、テスト時の工数が思うように削減できないことは大きな課題である。これは、ソースの変換率が低いために、変換できない箇所に余計な手修正が発生すること。加えて、手修正量が多くなることで、結果として、再構築と同等のテストが必要となるためである。例えば、ロジックを全網羅する検証、所謂「ホワイトボックステスト」を実施すると、再構築と同じコストが発生することになる。何不自由なく動作している既存システムをプラットフォームの異なる別システムへ移行するだけに、膨大なコストが発生することは企業の経営層からは理解が得られないことが多い。マイグレーションの多くの現場では、テスト工数の削減が大きな課題となっている。

３．課題解決へのアプローチ

そこで私は、前項で述べたマイグレーションの課題を今回のプロジェクトでは、以下の２つの方針を立て対策を実施することを A 社にアドバイスした。

・ツールの自動変換率を高め、手修正量を極力抑えること
・ソース変換前に動作上の非互換を把握し、変換時に非互換を解消させておくこと

上記の２つの方針のもと、それぞれの課題の解決に向け今回実施したアプローチを次に述べる。

３．１　手修正によるバグ混入への対処

手修正量はマイグレーションツールの自動変換率に大きく左右されるため、マイグレーションツールを正しく選択することが重要である。今回の A 社の移行プロジェクトにおいては、自動変換率の高さから当社が提供するサービスである CoolCat VB マイグレーションサービスを適用した。また、CoolCat が実装するオプション機能を活用した以下の２つの施策もあわせて実施し、更に変換率の向上に取り組んだ。

① A 社固有のコーディングのクセ（暗黙の型変換に依存するようなコーディングなど）に対する独自な変換ロジックの組み込み

② VB.NET ではサポートされなくなった関数やコントロールの非互換を吸収する VB 互換クラスの作成

この結果、表２に示すとおり、CoolCat VB マイグレーションサービスを適用した自動変換率は 99.9%となり、手修正量を僅か 1.5Ks に抑える成果を得た。

変換対象規模	1，500Ks
自動変換率	99．9%
手修正の割合	0．1%
手修正量	1．5Ks

表２ CoolCat VB マイグレーションでのソース変換率

３．２　動作上の非互換の把握への対処

動作させてみて初めて判る非互換を把握するためには、プロジェクトの初期の段階でフィージビリティースタディ（プロトの検証）を行い、動作時のエラー事象を把握することが大切である。A 社の移行プロジェクトにおいては、マイグレーション対象の資産から約 10%の代表的なプログラムソースを選出し、ソース変換から動作確認までを行うフィージビリティースタディを実施した。

フィージビリティースタディの内容は以下のとおりである。

- 変換対象のソースプログラムを参照し、ロジックや使用しているコントロール、特殊な処理の有無を確認する
- 次に、“どの部分をどのように変換するか”と“変換できない部分”を明確にする
- やむを得ず、手作業で変換元のソースプログラムに修正を加えることがこの段階で判明している場合は、その修正内容を明確にする
- 更に、動作確認環境で変換したプログラムを実際に動作させる。ここでは、表示上の非互換や動作上の非互換を実際に動かしてみることで確認し、これら

についてもその対応方法を明確にする。やむを得ず手修正が必要だと判断した箇所については、その追記内容を明確にする。

A 社において実施したフィージビリティースタディの結果を表３に示す。
文法上の非互換に加え、動作上の非互換を１８パターン２４３箇所で抽出することができた。

（総箇所数:6,758 箇所）

	パターン数	発生箇所
文法上の非互換	5パターン	17箇所
動作上の非互換	18パターン	243箇所

表３ フィージビリティースタディの結果

ここで検出した文法上の非互換や動作上の非互換は、ソース変換作業までに全て対処を完了させた。また、ソース変換作業後の動作確認でどの程度の非互換が発生するかを予測し、動作確認の作業量の把握を行った。更に、動作確認実施前に動作上の非互換の対処を施すことが可能となったことで、動作確認作業を効率良く進められる効果もあった。

３．３　テスト工数の削減への対処

A 社より以前に CoolCat VB マイグレーションサービスを適用し、テスト手法としてホワイトボックステストを選択した X 社のテスト実績値を表４に示す。X 社のテスト実績値は、再構時のそれに比べ、テスト密度が同じであるにもかかわらず、エラー検出率が低い値を示している。これは、現行システムをベースにするマイグレーションでは、ビジネスロジックの部分ではエラーの発生が少なく、既に単体テストレベルの品質が確保できているためだといえる。また、X 社の動作確認時に発生したエラーの内訳を表５に示す。画面の遷移や操作上におけるエラーである結合テストレベルでのエラーが多数を占め、ホワイトボックステストではテストが過剰に行われたと評価することができる。

そこで、今回の A 社のプロジェクトでは、X 社と同等のソース変換率による品質が確保できると仮定し、画面遷移や入出力項目に着目しテストケースを抽出する、所謂「ブラックボックステスト」による動作確認を実施することを考えた。これにより、実施するテストケース数を下げ、テスト工数を削減するアプローチを行った。

	X社実績 （ホワイトボックス）	再構築時の試算※ （ホワイトボックス）
テスト密度(ケース/ks)	１０４	７２～１０９
エラー検出率(件/ks)	０．１６	７．４～１１．１

※富士通における約30の大規模システム開発プロジェクトのデータから試算した値

表４ X 社のテスト実績と再構築時との比較

	件 数（％）
単 体 テ ストレ ベ ル （ホ ワ イトボ ッ ク ス テ ス ト で し か 見 つ か ら な い）	９件（２％）
結 合 テ ストレ ベ ル （ブ ラ ッ ク ボ ッ ク ス テ ス ト で 見 つ け ら れ る）	５６５件（９８％）

表５ X 社動作確認でのエラー内訳

４．効果

今回実施した課題へのアプローチにより、以下の効果を得ることができた。

４．１　期間短縮効果

表６および図３に示すようにソース変換および動作確認はそれぞれ２ケ月間で終了することができた。また、これら作業は並行で実施したことにより、ソース変換と動作確認はトータルで３ケ月間で完了した。この結果、当初の計画に比べ、移行期間を１カ月間短縮する効果があった。

また、A 社の実績を一般のマイグレーションツールを用いた場合と比較すると，約 10.8 ケ月、50%の期間短縮の成果を得た。CoolCat を用いたマイグレーションにおいては、ソース変換時の自動変換率が高く手修正量が低く抑えられたこと、変換後ソースの品質が高く、少ない工数で効率的に動作確認が実施できたことが期間を短縮できた要因である。結果的には、ハードのサポート切れの期限内で新システムへの移行が完了し、A 社のシステム移行要件を満たすことができた。

<table>
<tr><th></th><th>一般的なツールによる
マイグレーション
（変換率：７０％）</th><th colspan="2">ＣoolCatによる
マイグレーション
（変換率：99.9%%）</th><th>ＣoolCatの
優位性</th></tr>
<tr><td>調査</td><td>２．０ヶ月</td><td colspan="2">２．０ヶ月</td><td rowspan="5">約２倍</td></tr>
<tr><td>ソース変換</td><td>８．３ヶ月</td><td>２．０ヶ月</td><td rowspan="2">３．０ヶ月</td></tr>
<tr><td>動作確認</td><td>５．５ヶ月</td><td>２．０ヶ月</td></tr>
<tr><td>業務テスト</td><td>５．５ヶ月</td><td colspan="2">５．５ヶ月</td></tr>
<tr><td>総所要期間</td><td>２１．３ヶ月</td><td colspan="2">１０．５ヶ月</td></tr>
</table>

表６ テスト期間の短縮効果

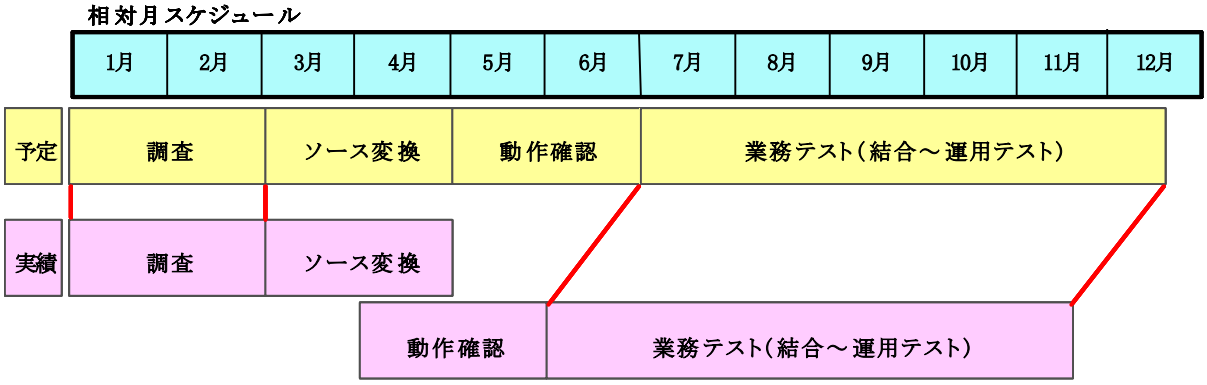


図３　A 社 移行スケジュール(実績)

４．２　品質の確保

ブラックボックステストを選択した A 社の動作確認とホワイトボックステストを選択した X 社の動作確認のテスト実績値を表７に示す。

A 社のテスト実績を分析するとホワイトボックステストを実施した X 社に比べ、テスト密度は小さくなっているが、エラー検出率は変わらないことが判る。これは、ブラックボックステストであっても、画面表示や画面操作上のエラーが X 社のテストと同じように検出できたことを示している。A 社の動作確認では効率的なテストが行え、少ない工数で、品質が確保されたことが確認できたといえる。

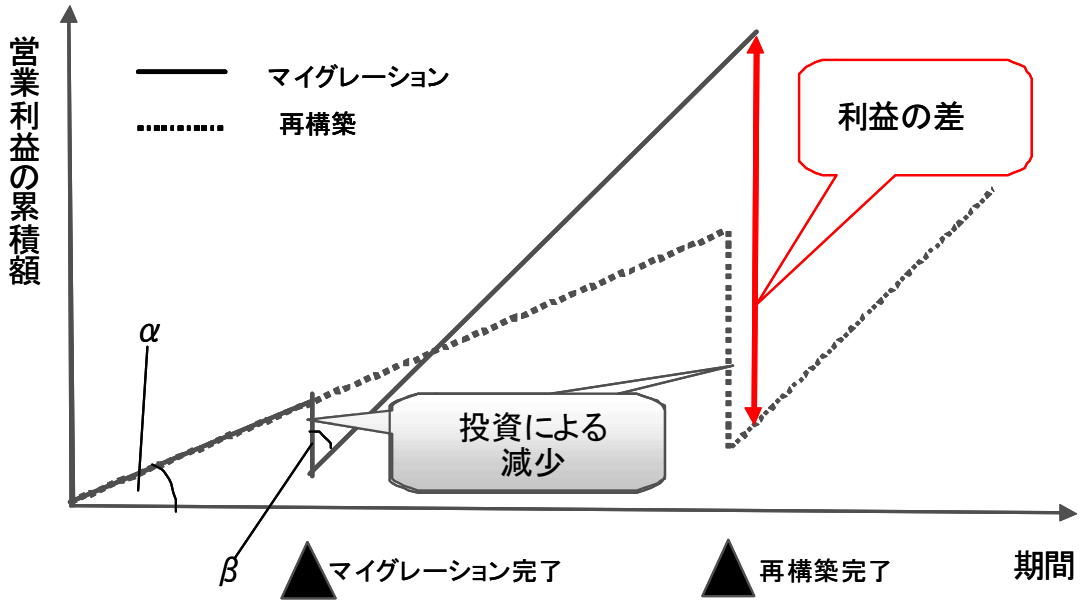
また、動作確認後に A 社が実施した業務テストにおいては、マイグレーションに関したエラーはほとんど発生することは無く、品質の高いシステムを提供することができた。後になって A 社からは、短期間でシステム移行ができたことおよび無事にシステムが安定稼働できたことについて高い評価を頂くことができた。

	A社実績 (ブラックボックス)	X社実績 (ホワイトボックス)
テスト密度(ケース/ks)	5．9	104
エラー検出率(件/ks)	0．12	0．16

表７　テスト実績値の比較

５．今後の展開

CoolCat VB マイグレーションサービスは、現行の VB 資産に満足しているお客様にとっては、コストと期間を大幅に削減し新しいプラットフォームへ移行できる最適なソリューションであるといえる。投資回収が早められるだけでなく、新システムでの利益創出が早期に行え、また、再構築手法に比べ、生涯の営業利益の累積額が高くなるメリットもある。お客様の既存資産を有効活用する戦略的サービスとして、今後幅広く展開していきたいと考えている。



∠αは現行システムにおける営業利益累積額の伸び率を示し、∠βはマイグレーション後の新システムにおける営業利益累積額の伸び率を示す。新システムにおいては、.NETへ移行することにより付加価値が生まれるため急傾斜となる。

図４　マイグレーション手法と再構築手法の投資回収比較

参考文献

なし

参考ＵＲＬ

［１］マイクロソフト株式会社　ホームページ
http://www.microsoft.com/ja/jp/default.aspx