APP-08-22
J. Duggan
(G00151340)

**Research Note**
February 25, 2008

# ソフトウェアの言語とツールにおける経年劣化の評価

旧来のコンピュータ言語やツール・ソフトウェアの成熟段階は、各種の指標によって判断することができる。標準的な言語も、その発展段階に照らして評価されなければならない。ここでは、ソフトウェア資産について、そのリスクおよびライフサイクルにかかわるコスト軽減に役立つ、評価と管理のための戦術と戦略を提案する。

# 要約

ソフトウェア言語やツールの衰退は、本リサーチノートで挙げた8つの基準に照らして判断できる。状況に応じてリスクを許容範囲にとどめるには、アプリケーション、ツールおよびインフラストラクチャに関するポートフォリオをモニタリングする必要がある。

## 主要な所見

- 言語またはツールの継続的存続力は、ベンダーのサポート状況、あるいはトレーニングやスキルの有無などの外部的要因によって大きく影響を受ける。
- 本リサーチノートの8つの基準により、特定の言語がどの段階にあるかを判定できる。
- 言語とツールの経年劣化要因は、アプリケーション・ポートフォリオのリスクに大きな影響を及ぼす。

## 推奨事項

- 使用中の言語およびツールのライフサイクル・リスクに対する自社の許容度に合わせて、技術的アーキテクチャのガイドラインを作成する。
- ビジネスおよびアーキテクチャに関するイニシアティブに沿った、レガシー・アプリケーションおよびインフラストラクチャを分類するため、アプリケーション・ポートフォリオ管理プロセスを作成し維持する。新たな開発、パッケージ購入、大幅な機能向上によって置換、強化され得る製品と、従来の形式で存続すべき製品とを分類する。
- 新たに提案されるプロジェクトについて、従来型のメソッド、ツール、スキルによって新規開発や大幅な強化を図るべきか、あるいは新たなソリューションを採用すべきかをメリット、コスト、リスクの比較に基づいて判断する。

## 要旨

8つの指標に照らし、自社のポートフォリオを形成している言語とツールを評価する。経年劣化の兆候が明らかであれば、適切なリスク管理ポリシーを採用する。

## 分析

最新のテクノロジを積極的に試そうとする人はどこにでもいるものである。一方、同じような熱意で旧来のソフトウェア言語やツールを扱う人はあまりいない。しかし、ビジネスの中心にあるのはレガシー・アプリケーション・ポートフォリオであり、既存の言語やツールを大幅に入れ替えてしまうことは、現実的な措置ではない場合が多い。新たなサービス・アーキテクチャに移行するためには説得力のある理由が求められ、これにより、確立された従来型のアーキテクチャでは得られない俊敏性が必要となるビジネス・プロセス改善イニシアティブの実現を支援する。アプリケーション・ポートフォリオ、言語、ツール、スタッフを次世代のソリューションに移行させるためのコストとリスクの両面で評価し、移行戦略を構築する必要がある。

ビジネス・ニーズに対応する言語ポートフォリオ戦略の評価および管理は、アプリケーション・ポートフォリオの定期的な評価の一環として行うのが理想である。こうした定期的な評価では、トレーニング、開発、サポート、および実行環境に関する問題を含めた評価を行うべきである。評価の形式としては、正式なアプリケーション・ポートフォリオ・イニシアティブ(「Meshing Architecture, Project and Application Portfolio Processes for Effectiveness」参照)、対象を限定したアプリケーションのヘルス・チェック(「Use a Health Check to Determine Your Application's 'Fitness for Duty'」参照)、あるいは単なるプロジェクトの一環としての評価などがある。しかしいずれの場合も、アーキテクチャやイニシアティブで旧来の言語およびツールの使用を継続することに伴う、リスクとコストを評価しなければならない。言語またはツールの存続力は、スキル、トレーニング、そしてベンダーによる継続的なサポートの有無によって大きく影響される。本リサーチノートで提示される基準と一般的な評価方法は、特定のプロジェクトまたはポートフォリオ評価の基盤になり得るものである。その上で組織がそれぞれの判断で、独自にリスク要因を追加することが望ましい。

旧来の言語やアプリケーションについて検討する場合には、得てして運用上の明らかになっている問題を解決するための短期的な修復や、ソフトウェア資産の価値とは関係のないテクノロジの進歩に意識が向きやすい。長期的な観点から見れば、COBOLなど旧来の言語も、中堅・中小企業にとってはいくつかの現実的な問題が予想されるとはいえ、少なくとも今後10年は有効に存続すると考えられる。しかし、Programming Language/One (PL/1) やRPGなどの言語は、Smalltalk、Advanced Programming Language (APL)、FORTRANに続いて衰退に向かいつつある。確かに旧来の言語の一部には、ニッチ・アプリケーションで継続的に利用されているものもある。例えばFORTRANは、超並列処理アプリケーションでは広範に使用されている。しかし、旧来の言語とツールは、人材の確保やテクノロジに関するさまざまなリスクが伴うため、格段の注意が求められる。

同じ言語でも、バージョンの違いや特定のプラットフォーム上で導入されているケースでは、経年劣化の段階が異なる場合がある。あるプロバイダーが積極的に「成年期」の製品を維持する一方で、旧来の、または普及率の低いプロバイダーが、同じ言語の「老年期」または「老朽化」のバージョンを提供していることもある。ポートフォリオ評価は、新たなテクノロジを導入し続ける強迫観念や、新たな言語の拙速なマイグレーションによるコストとリスクを回避し、有益なソフトウェア投資を保護し維持することを目的としなければならない。

Visual Basic (VB) の状況を考えてみる。MicrosoftのVB6などは、言語の導入が進み、安定した基盤ができているが、いずれはVB.NETなど新たなバージョンに取って代わると考えられる。VBやJavaのように恒常的に発展する言語の場合は、各リリースのバージョンや特徴に注意しなければならない。それ以前のバージョンで開発されたコードでは、旧式のアプリケーション・プログラミング・インタフェース (API) またはアーキテクチャが使用される。また、最新の状態を維持しないと、ベンダーが供給するアップグレード・ユーティリティを使用した場合に問題が発生する可能性がある。アップグレードを通じて自動的に移行されるコードが40%のみであるなら、資産を廃棄して最初から構築してもコストは同等になる。

パッケージで購入するのが通例である企業においても、基盤となる言語に関し、年数の経過とサポート状況に注意を払うべきである。そのパッケージがどのような言語、データベース、そしてミドルウェア・テクノロジによって構成されているかを把握しなければならない。ビジネス・ニーズを満たすパッケージ・ベンダーの能力は、基盤の言語のサポートが十分である場合に最大になる。より重要なことは、ベンダーとユーザーとで優先順位がどのように異なるかを認識することである。例えば、旧来のプラットフォームにおけるパッケージ・サプライヤーは、基盤となるテクノロジのリスクを最小にする必要がある。こうしたサプライヤーは、主に新たなテクノロジへのマイグレーションに要する資金を確保するために、保守サービスによる収益の流れを維持しなければならない。新しいプラットフォームにおけるベンダーは、アップグレードやマイグレーションを駆り立てるために、旧来のプラットフォームのリスクを誇張する。古いプラットフォームに固執することも、新しいプラットフォームに慌てて移行することも、結果的に大きな損失につながる可能性がある。

## 言語やアプリケーション開発ツールはどの時点で老朽化するか

生物と同様、ソフトウェア・システムにもライフサイクルがある。ガートナーではこのライフサイクルという概念を利用して、テクノロジやアプリケーションが「幼年期」にあるという表現をすることがある。こうした考えに基づき、言語、ソフトウェア・ツール、およびそのほかのソフトウェア・システムが廃棄に至るライフサイクルに沿って、各段階を表す8つの指標が定義されている。これらの指標を認識、管理して、経年劣化のプロセスに伴う問題に対処すれば、クリティカルなソフトウェアのサポートが突然中止されるような緊急事態が発生しても、そのリスクを軽減することができる。

備考 : テクノロジのライフサイクル (「ゆりかごから墓場まで」) に伴うリスクに関するガートナーの評価の詳細については「Managing Technology Life Cycle Risks」を参照されたい。言語とツールの使用について、ライフサイクルの概念を適用した例を紹介している。

表1に、発展や劣化を表す4つの段階と、あらゆるテクノロジの発展段階を評価できる8つの指標を示す。表の最後には、管理ポリシーとして各段階で実行可能な短期的および長期的な措置を示している。短期的および長期的な管理戦略は、その言語の成熟度マトリクスにおける位置によって調整する必要がある。

表1　ソフトウェアの経年劣化の4つの段階

| 定義 | 成年期 | 成熟期 | 老年期 | 老朽化 |
|---|---|---|---|---|
| | 言語またはツールが活発に発展している。ユーザーとサードパーティのコミュニティが活況を呈し、主要なベンダーを補完している。ユーザー・ベースが拡大している。 | 言語またはツールが広範に使用されている。コミュニティは活発で、サードパーティも関与している。既存の導入環境での利用が拡大する傾向にある。ユーザー・ベースは拡大または縮小のいずれもあり得るが、比較的安定している。 | 言語またはツールが、主に既存の導入環境で利用されている。コミュニティの活動は衰退している。ベンダーがコミュニティを後援するケースがある。ニッチなアプリケーションは依然として成功する余地がある。 | スキル・ベースとコミュニティのサポートが衰退し、小規模になる。サードパーティのサポートはほとんど／まったくなくなっている。 |
| **基準** | | | | |
| 1. リリース頻度 | 定期的であり、機能が向上している。 | 定期的であり、プラットフォームとパッチのサポートが主体である。 | 不定期／ほとんど行われない。 | リリース中止 |
| 2. サードパーティによるトレーニングの利用可能性 | 高 | 減退 | 低 | なし |
| 3. 熟練したスキルの利用可能性(組織内外を含む) | 高 | 安定的 | 減退、地域により不足 | 減退、低 |
| 4. 新規プロジェクトでの利用度(各地域) | 頻繁 | 安定的 | まれ | なし |
| 5. アプリケーション導入の成長率 | 急速な拡大 | 一部分野の拡大 | 安定的 | 低下 (減退) |
| 6. プログラミング構成 | 安定的、頻繁な拡張 | 安定的 | 安定的であるが最新の標準に比べ劣る。 | 安定的であるが不十分 |
| 7. サードパーティおよびコミュニティによるツールのサポート | 拡大 | 安定 | 減退、不完全 | 最小限 |
| 8. ハードウェアのサポート | 十分 | 十分であるが、遅れている場合がある。 | 固定的かつ不完全 | 縮小 |
| **管理ポリシー** | | | | |
| 年1回の存続力評価の内容 | 現行バージョン、スキル計画、ベンダー・サポート、リリース・ロードマップ<br>イベント評価：M&A、新規リリースの発売 | 現行バージョン、スキル計画、ベンダー・サポート、リリース・ロードマップ<br>イベント評価：M&A、新規リリースの発売 | スキル、テクノロジに伴うリスク、運用上のリスク、ベンダーのサポート体制<br>イベント評価：M&A、代替製品の発売 | マイグレーションおよび廃止計画の検討<br>イベント評価：M&A、代替製品の発売 |
| 短期的な戦略 | アーキテクチャの適切性を評価する。 | アーキテクチャの適切性を評価する。 | 廃止を見据えた措置を取る。ベンダーの取り組みとロードマップを精査する。 | 拡張を禁止する。 |
| 長期的な戦略 | 継続的な使用を可能にする設計主導型のプロセスを推進する。 | 最新のフォームに移行する。ツールの利用を拡張する。 | 新規の利用を避ける。 | 代替製品導入の優先度を高くする。 |

出典：ガートナー (2007年12月)

## 言語／開発ツールの成熟度判断のための質問

1. リリース頻度：開発は継続されているか。新しい機能が定期的に追加されているか。特定のバージョンについてベンダーがサポートを中止していないか。
2. サードパーティによるトレーニングの利用可能性：トレーニングに利用できるソースはどれだけか。コース・カタログの広範さ、トレーニング・コースの頻度はどの程度行われているか。トレーニング教材やコースが提供される頻度は、1年でどのように変わっているか。
3. 熟練したスキルの利用可能性 (組織内外を含む)：スタッフには、予想される作業負荷に対応できるだけのスキルがあるか。熟練したスタッフを確保、保持するのはどの程度困難であるか。これらのスキルを保持するために、割増賃金は必要か。スキルをアウトソーシングによって確保できるか。
4. 新規プロジェクトでの利用度 (各地域)：製品が、(社内および同地域の他社の) 新しいプロジェクトや作業で使用されている頻度はどの程度か。その頻度は1年前に比べてどのように変わっているか。
5. アプリケーション導入の成長率：その言語またはフレームワークは市販の製品で使用されているか。過去1年で導入率は向上しているか。
6. プログラミング構成：そのソリューションは完全、かつ関連するシステムの現行のレベルと互換性があるか。古いAPI／言語機能がマイグレーションまたは拡張の妨げにならないか。
7. サードパーティおよびコミュニティによるツールのサポート：システムをサポートするサービスはどの程度提供されているか。サービスを提供するサプライヤーの財務状況は良好か。ツールの新しいリリースにはシステムのサポートも含まれているか。
8. ハードウェアのサポート：ハードウェアに関する改善は、製品に完全かつ迅速に反映されているか。

これらの質問をより一般的な言語環境に適用することで、表2に示すITポートフォリオにおける最も一般的な言語に対して高度な評価を行える。これらは、2007年第4四半期の事例証拠に基づく、定性的な評価である。場合によっては、すべての評価対象が同じ段階に位置することもあるが、より広範に利用されている言語 (COBOLなど) の場合は、ベンダーと言語の改良型を別個に評価する必要がある。1つの企業における各言語の戦略には、ビジネス、スキル、投資に関するさまざまな問題を考慮しなければならない。

表2　言語評価の例

| 種類 | 名前 | 段階 | コメント |
|---|---|---|---|
| 言語 | APL | 老年期 | 特殊なマトリクス・アプリケーション以外は老朽化の段階にある。スキルは大幅に制約されている。いくつかの小規模なベンダーだけがサポートを提供している。 |
| 言語 | BASIC | 老朽化 | スクリプト言語としてニッチ・アプリケーションで利用されている。 |
| 言語 | C | 老年期 | 航空宇宙産業、自動車、組み込みシステムなどにおけるニッチ・アプリケーションで利用されている。 |
| ベンダー独自の言語環境 | C# | 成年期 | ECMA標準アクティビティ |
| 言語 | C++ | 成熟期 | 多くの場合、コンパイラはCコードをサポートし、Cに取って代わるものである。スキルを備えたリソース、トレーニング、およびサードパーティによるサポートは減少している。独立系ソフトウェア・ベンダー (ISV) および組み込み型の垂直市場には大きな需要がある。 |
| 言語 | COBOL：陳腐化した改良型。(DEC COBOL、IBM OS/VS COBOL、IBM COBOL II) | 老朽化 | ツール、スキル、サポート上の問題がある。 |
| 言語 | COBOL：現行の改良型 (通常はCOBOL 88またはCOBOL 2002標準に対する改良型)。IBMメインフレームをベースにしたLE | 成熟期 | ほとんどのベンダーが現行および前バージョンのリリースをサポートしている。2007年にMicro FocusがAccuCobolを買収し、分散型COBOLのサポートを統合している。一部の特殊な用途でスキルの不足が発生している。 |
| ベンダー独自の言語環境 | Delphi (CodeGearによるWin32および.NET用Object Pascalの改良型) | 成熟期 | 現行のリリースについてはサポートが充実し、拡張性も高い。古いバージョンは老年期に向かっている。スキル・ベースは北米よりも欧州の方が優れている。 |
| 言語 | FORTRAN | 老朽化 | 超並列プログラミング用のニッチ製品は成年期にある。そのほかのニッチ製品は成熟期である。スキル、ツール、機能上の問題がある。 |
| 言語 | Java | 成年期 | 古いバージョンは成熟期から老年期に向かっている。 |
| 言語 | JavaScript | 成年期 | |
| ベンダー独自の言語環境 | Natural：改良型 (Software AG) | 成熟期 | Mainframe Natural V4以前、およびDistributed Natural V5以前は老年期に分類される。 |
| 言語 | Pascal | 老年期 | スキル、互換性、サポート上の問題がある。 |
| 言語 | Perl | 成熟期 | スクリプト言語 |
| 言語 | PL/I (IBMメインフレーム用LE) | 成熟期 | 北欧ではスキルが充実している。そのほかの地域ではスキル上の問題が発生している。 |
| 言語 | PL/I (LE以外) | 老年期 | スキルおよび互換性に関する問題がある。 |
| ベンダー独自の言語環境 | PowerBuilder (Sybase) | 成熟期 | スキル上の問題が発生している。古いバージョンは老年期に移行しつつある。 |
| ベンダー独自の言語環境 | Progress OpenEdge | 成熟期 | スキル上の問題が発生している。古いバージョンは老年期に移行しつつある。 |
| 言語 | Python | 成年期 | 動的言語。まだ成年期に達したばかりである。 |
| 言語 | RPG (ILE以外) | 老年期 | サポートおよびスキル上の問題がある。 |
| 言語 | ILE用RPG | 成熟期 | スキル上の問題が発生している。 |
| 言語 | Ruby | 成年期 | 動的言語で、主にRailsで使用されている。まだ成年期に達したばかりである。 |
| 言語 | Smalltalk | 成熟期 | コミュニティが大幅に縮小している。言語自体は、InstantiationsやCincomなどのベンダーによってサポートされているバージョンについてはまだ成熟期である。サポートされていないバージョンは老年期に分類される。 |
| 言語 | TCL | 老年期 | スクリプト言語。スキルおよびサポート上の問題がある。 |
| ベンダー独自の言語環境 | Visual Basic .NET (Microsoft) | 成年期 | |
| ベンダー独自の言語環境 | Visual Basic 6 (Microsoft) | 老朽化 | 互換性およびスキル上の問題がある。開発プラットフォームはサポートが終了している。 |

出典：ガートナー (2007年12月)

備考：表2を見ると、サポートは一般的に現行リリースと一段階前のリリースに集中していることが分かる。それ以前のリリースは陳腐化していると判断してよく、存続力を検討する場合には特に注意が必要である。これらの評価は、一般的な市場を対象に行われている。老年期にある製品についても、ベンダーが十分なサポートを提供しているケースがあり、その場合は継続的な利用にも適する。ただしサポート・コミュニティは、成熟期や成年期にある製品に比べて規模や活動の活発度が大幅に縮小している。

大企業の場合、陳腐化したソフトウェアによる影響は、ある程度緩和される。非常に特殊なスキル・セットに対して追加料金が発生したとしても、大企業の予算規模の中では目立たない。さらに、製品の入れ替えやマイグレーションにかかる膨大なコストが回避されることで、容易に相殺することができる。大企業の中には、社外のトレーナーが市場から撤退するとともに、広範なトレーニング・プログラムを自社で展開できるケースもあり得る。スタッフも充実しているため、複雑かつ多様なテクノロジを扱うことができる。新旧のテクノロジを組み合わせてコンポジット・アプリケーションを構築することも可能であろう。中堅・中小企業のIT部門では、保守に伴うコストや内部運用コストの増大を回避しなければならず、また旧来のスキルを保持しながら新たなビジネス・ニーズに対応することは困難であるため、早めの移行が必要になる。移行に当たっては、上位20%と下位20%のユーザーのための人員確保、保守、移行作業にかかわるコストは、そのほかのユーザーよりも10%高くなるであろう。

アプリケーション・ポートフォリオ管理のベスト・プラクティスは、言語およびツールのポートフォリオの範囲が限定されている場合にも適用できる。現在所有するパッケージ製品やツールは詳細に分類して、ライフサイクルの範囲で有効に活用しなければならない。製品のライフサイクルは、ビジネス・ニーズの変化、パッケージ・ベンダーの代替計画、あるいはその言語またはツールの経年劣化や推定寿命の状態によって区切られる。資産を置き換えるまでの時間とコストを見積もり、その結果を置き換えスケジュールに位置付けることで、置き換えに要するコストを効果的に予測できるようになる。財務計画担当者またはCFOと協力して、財務的なリスクに適切に対処するための方法を計画すべきである。

## 推奨リサーチ

- 「Plan Legacy Application Retirements Carefully」
- 「コストとリスクの削減を通じてビジネス利益を改善するAPM」(APP-07-65、2007年8月10日付)

(監訳：川辺 謙介)