情報システム更新のサイクル

蜂谷　博

はじめに

情報システム更新のサイクルについて

情報システムは，従来は数年の周期で更新されてきた。情報システムの更新は，主にハードウェア資源のレ，ンタル期限・リース期間の満了に発生して，まずハードウェアリソースの更新が企画されそれを機にソフトウェアもレベルアップを兼ねて更新されるのが一般的であった。しかし情報システムのプラットフォームであるハードウェア環境は，メインフレームと呼ばれる汎用コンピュータを主力とした時期から，廉価なパソコンによる時期を経て，LANに接続された多数のパソコンとさらにそれがインターネットに接続されたコンピュータネットワークの時代に変化してきた。このハードウェア環境のダイナミックな変化は，開発のベースとなるOSなど基本ソフトも大きく変化させた。その結果としてソフトウェア資産の継承的な更新では対応できなくなっている。開発ツールとしての言語も，GUIやオブジェクトオリエンテッドな開発ツールの提供によって，ソフトウェア環境も機能も大幅に変化している。

ここには，それらのハードウェア環境の変化とそれに対応するソフトウェア環境の変化を配慮しながら，情報システムの更新のタイミングと更新の際に考えなければならない問題点が生じている。

1.　情報システム運用後の特性と傾向

情報システムが構築され運用を始めると，それ以前の手作業には戻れなくなる。またそれ以外にもいくつかの現象と傾向があらわれる。情報システムの開発に携わってきた筆者の経験の中から，その特性を整理してみた。人間による手作業による事務処理か

ら，情報システムを開発し，それに置き換えて適用した現場では，次のような特性がある。

(1) 情報システムの麻薬性

従来人間が手作業で行ってきた仕事を，コンピュータによる情報処理システムを開発し，それに移行すると，概して以前の手作業の処理に後戻りすることはできない。あるいは後戻りすることはきわめて困難である。情報システムを開発し使い始めると，それまでその仕事に携わっていた担当者も，時間が経過するにつれて人事異動で他部門に移ったり退職したりして，手作業で行っていた仕事の手順を熟知していた人はいなくなる。また同一人物がそのシステムの運用に携わっていたと場合でも，記憶の中では，仕事の手順やノウハウは少しずつ消えて忘れられてゆく。仕事の手順や関連するノウハウがドキュメントとしてきちんと整備されている場合は比較的安全であるが，担当者の経験と知識の中にしか残されていないような職場では，熟練者がいなくなるにつれ，仕事の手順もノウハウも消滅してゆく。

また日本では，仕事の手順や技術・ノウハウを，詳細なドキュメンテーションを作って残しておくという習慣は少ない。「仕事の手順をドキュメントに記述して残し，担当者が代わってもただちに業務が引き継げるようにする」という考え方は日本のオフィスでは少ない。「知らしむべからず，寄らしむべし」と言われるように，昔から日本の役所では，仕事の手順は担当者の記憶の中に守秘的に保存され，門外秘とされることが多かった。仕事の手順やノウハウを第三者に伝授することは，仕事そのものの遂行能力を伝授することであり，仕事を引き継ぐ時であった。場合によっては，仕事を引き継ぐ際でも仕事の仕方やノウハウは教えないという傾向もあった。それが日本の事務処理のやりかたであり，オフィスの一つの傾向であった。この伝統的な文化は，近代日本にも継承され，ビジネスの現場でも作業のマニアル化や仕事のドキュメンテーションの作成は，あまり進められてはいない。

いっぽう，オフィスにおける仕事の手順などは，仕事の成果にくらべ，仕事のレベルとしては軽視される傾向が強く，作業のマニアル化やドキュメンテーションのために時間と労力を投入することは軽視される傾向が強かった。

したがって，多くの職場では仕事の手順のドキュメンテーションやマニアルといったものは作られておらず，情報システムの開発の際も，システムエンジニアが担当者に直接的にヒアリングしながら，システムの設計をするというケースが多い。情報システムを構築するような現代的な仕事の現場でも，その直前の仕事のドキュメンテーションは，かならずしも十分に作られているとはいえない。

そのため，いったん情報システムを開発しそれを適用した後は，以前の手作業による処理には戻ることは困難となる。

また幸い以前の担当者がいて手作業に戻れたとしても，担当者は概してその仕事に喜びを感じるよりは，「機械でできる仕事を，なんでまた人間がしなければならないのか」という心境に陥りやすく，「面倒だ」とか「厄介だ」という心理的飽和状態に陥りやすい。

(2) 情報システムは生きものである

情報システムは，設計・開発・デバッグ・テストランを経て本運用に入る。しかし運用されれば，その中から新たな機能のレベルアップの要求や，システムの現状に対する不満が発生する。使ってみてはじめて「もっとこんな機能があったら…」という要求が発見される。したがって運用に入ると同時に，レベルアップの期待が潜在的に発生する。このレベルアップの要求に応えて，適当な時期に機能の追加・使い勝手の改良などを行わなければ，エンドユーザの支持を得られない。そため情報システムは，構築されたら完成ではなく，常に一定の改良と更新を求められる。

このときに問題となるのは，システムの更新と改良のコストとマンパワーである。コンピュータが普及し低廉化して，ハードウェア価格はかってのメインフレームで情報システムを構築した時代に比べ想像以上に手に入りやすくなった。しかし情報システムを構築するには，ソフトウェア技術が必要であり，業務の複雑さや，固有の業務に特化した使い易いシステムを構築するためには，優秀なソフトウェア技術者による優れた技術とノウハウが必要である。一方ソフトウェア技術者の育成は，社会的なニーズには追いついておらず，慢性的な人材不足の状況となっている。しかも近年の大学生の学力不足と理数系離れによって，優秀なソフトウェア技術者が育成される土壌はますます痩せ細っている。情報システムを構築したい企業にとっても，特殊な専門技術を持つソフトウェア技術者を終身雇用的に採用することは，企業内の人権費の圧迫となりかねない。そのためソフトウェア開発をアウトソーシングして外注にすることになるが，ここには長短両面がある。ソフトウェアの開発や更新を外注化することと，技術者を雇用して内部製作をする方法とでは，次のような特性があげられる。

① 開発段階では，ソフトウェア開発を外注化する方法は，後年度負担の大きい人件費を抱え込む危険は少ない。開発の段階では，技術者を雇用してソフトウェア開発を内製化するより低コストで開発できる。

② 開発段階では，外部のマンパワーによって，多数の技術者を集中的にそのプロジェクトに投入して，短期間で集約的にシステムの開発をすることができる。技術者を雇用して内部で製作しようとすると，多数の技術者を集中的に投入することは難しい。

③ 完成後，運用をしてみて，改良や機能の追加のニーズが発生したとき，開発を外注化したケースでは，ソフトウェアの追加や改良は，高い費用を求められ，また納期も依頼者の狙いどおりにはできないことが多い。改良や機能追加のニーズそ

のものの詳細を把握することに手間取ることもある。

④ ソフトウェア技術者を組織内に配置していれば，システムの改良や機能追加に対しては，外注よりは速いレスポンスで対応しやすい。もちろん抱えている業務との兼ね合いによるが。

⑤ 重要な社内(組織内)情報の漏洩の危険性

外注化することによって，組織内の重要な情報を部外の技術者に扱わせることになり，社外あるいは組織外にもれてはならない情報の管理が，一時的であるにせよ部外者の手にゆだねられる。このときの業務上の守秘義務の管理には，十分な配慮が必要である。

特に地方自治体や行政組織では，ソフトウェア開発担当者を雇用することができにくく，ソフトウェア開発を外注化することが多いが，このときに依頼先に組織内の重要情報をサンプルデータとしてでも提供しなければならず，そこから重要情報の漏洩が生ずる危険性は高い。

ソフトウェアの外注化と内部製作とは，以上のような長所と短所を併せ持つ。

(3) 情報システムは完成したと同時に陳腐化が始まる

情報システムは完成したと同時に新たなニーズを生み出し，システムは完成した瞬間から陳腐化が始まると言っても過言ではない。したがって情報システムは，常に機能改良の要求を内包している。

この考え方は，日光の東照宮の陽明門を建築したときのいわれに共通する。陽明門は「完成した時から崩壊がはじまる」という考え方から，門に使われている柱を意図的に逆さに組み込み，「これはまだ未完成である」と主張し，それゆえ「永遠に崩壊が始まらないように」と祈念しているのだそうだ。崩壊しないように祈念しているというのは，いかにも仏教的であり観念的である。いかに理屈をつけようと祈念しようとも，木材で構築された建築物であるから，現実には永久に腐朽しないことはありえない。鉄筋で構築したとしても，いつかは崩壊する。崩壊してほしくないための祈りとして，このような理屈を取り入れたのであろう。これは仏教思想からもたらされた考え方であろうが，弁証法的でありまた論理的でもある。情報システムにもこの考え方と共通することが言える。

情報システムが完成し，運用されるようになると，運用前には潜在していた新たなニーズが顕在化する。手作業で行なっていた時代には考えられなかった新たな機能の要求や，情報システム化したことによって新たにできそうな機能の要求が発生する。

また情報システムのプラットフォームとしての情報機器のハードウェア性能と機能は，技術革新が加速度的にすすめられており，ニーズのキャッチから廉価な実用機の普及ま

でのサイクルは短縮している。

ニーズのキャッチ ⇒ 理論的な実現性検討 ⇒ 実験的な機器の開発
⇒ 商用試作機の開発 ⇒ 量産商用機の開発 ⇒ 普及（価格競争）

ハードウェア機器の技術革新によって，それ以前は技術的に実現の困難なあるいはコスト的に入手の困難なためにあきらめていたニーズも，徐々に手に入りやすくなっている。新たなハードウェアが手に入ることになれば，それに伴ってそのハードウェア機能を活用するためにソフトウェアの改良や追加が必要となる。このようにしてシステムは，常に改良と機能の性能向上が求められる。システムの機能改良をしなくなったときから，そのシステムの陳腐化が始まる。

2. 情報システムを取り巻く環境の変化とシステムのライフサイクル（寿命）

情報システムの再構築の原因となりうる要因

情報システムは常に改良と機能追加が求められる。導入した直後の段階では，それは改良と追加といった対処がなされる。しかし導入してしばらく時間が経つと，単なる改良や追加では対応しきれなくなり，システムの基本から設計しなおす必要性が発生する。追加や改良といった手直し的な対処では，解決できない大幅な質的な更新を求める新たなニーズが発生する。システムの再構築が必要となる要因は，つぎのようなものが挙げられる。

(1) ハードウェア・リソースの技術革新

－新しい機能の機器の出現。

－新製品による性能向上。プライス／パフォーマンスの向上

－従来は価格的にも性能的にも採用できなかったハードウェアや新しい周辺機器が出現する。機能的に従来は採用できなかった機材を廉価で組み込むことによって，エンドユーザにとってより使いやすい端末機器や周辺機器が現れる。

(2) レンタルあるいはリースしているハードウェアの契約期限の満了

レンタル／リース契約の期限の満了。レンタル契約の場合は，期間が満了しても価格は下がらないが，同一金額でより高性能なハードウェアリソースが手に入ることになるため，旧システムを継続して使用するより，新しいハードウェアに更新するほうが概して有利である。リース契約の場合は，契約期間で減価償却がされるため，期間を延長する契約ではリース価格は大幅に割り引かれる。しかし税務対策上は情

報システムのリース費用は経費として認められるため，予算規模を同じとしても新しいハードウェアに更新することを選択するケースが多い。

その時に同じ費用で新機材にリプレースするか，旧機器でコスト（リース価格）を低下させるかは，企業や経営主体が選択する。

新機材に移行するときに，同時にソフトウェアのレベルアップを行うかどうかも大きな選択肢である。

(3) ソフトウェア技術のイノベーション

ソフトウェア技術においても，イノベーションによってより使いやすいシステムに改良できる展望がでてくる。この場合は，従来のソフトウェアの継続的な更新ができにくい。設計からしなおす事になる。

Visual Basic, Visual c++, Delphi, java など，ユーザインタフェイスを GUI に構築できるソフトウェアツールが開発され，安価に入手できる。それを使って，ユーザインターフェイスの良いシステムを容易に構築できる（⇒システムの改良）。

このようなソフトウェア技術のイノベーションによる，システム再構築のニーズは，再構築と言っても，従来のシステムの設計書がほとんど役に立たないことが多い。システム設計の段階から改めて作り直す必要がある。

(4) ソフトウェアの重大な障害の発見

重大な欠陥の発見

［例］2000年問題

2000年問題は，ハードウェアおよびソフトウェアの両方に原因が内在していた。それもユーザーサイトの責任と言うよりは，ハードウェアメーカーと，システムプログラムの開発をした基本ソフトウェアメーカーによって作られた比較的単純な原因であった。応用ソフトウェアの開発段階でも，その障害は取り込まれており，ソフトウェア的にはソースコードから手直しする必要があった。修正そのものは比較的小さな部分であったものの，過去に構築され累積されたソフトウェアとデータの両方を点検・修正する必要があったため，業務量としては大きなものであった。

重大な bug の発見…場合によっては，ソフトウェアの修正・改良より，スクラップアンドビルドで作り直すほうが早いケースもある。

(5) 現場のユーザ（操作者）やエンドユーザからの機能のレベルアップの要求

新たなニーズに対応して，新機能を吸収するには，改良・更新より，ソフトウェアの再構築をしたほうが早いことがよくある。

「ソフトウェアがハードである」

ソフトウェアのレベルアップすなわち機能レベルアップは，既存のコーディングの修正・更新作業を必要とする。プログラミング技術は，手続き型プログラム，構造化プログラム，オブジェクト・オリエンテッド・プログラム…と発達してきたが，しかしそれでも他人がコーディングしたプログラムの修正・更新・機能追加は，かなり厄介な作業である。ソフトウェアの更新・改良は，人的コストと工数を必要とし，ハードウェアを増強するよりコスト的にも工期的にも高くつく。さらにソフトウェアを一部修正したり機能追加をしたりすると，それに伴って思いもよらないところに不整合やバグが生じたりして，それらのバグ対策には後々悩まされる危険性も高い。

－コスト
－工数
－複雑さ
－bug
－関連して発生する Bug，　整合性

それに比べると，ハードウェア・パーツは，今日ではモジュール化されており，ある部品を交換したり，増強・増設したりすることは，ソフトウェアの更新よりは比較的容易にできる。システムの動きが遅くてユーザから不満が出ているようなケースでは，ハードウェア・リソースを増設することによって解決できるとすれば，その解決方法は，ソフトウェアの改良よりはるかに容易に安いコストで短期間で実現可能（解決可能）である。

ハードウェア・リソースの増強：主記憶（RAM）の増設，ハードディスクの増設，ディスクファイルの増設，ファイルサーバ内の集合ディスクユニットの増設，回線の増設，端末装置やパソコンの増設…

そのことを指して「ソフトウェアがハードである。ハードウェアがソフトである」と言われる。

3. まとめ

ソフトウェアの更新は，担当者が離職したり移動したりしていると別の人によってソースコードを解読したうえで修正しなければならない。開発時の設計書やシステムのスペックが残っている場合は，それらのドキュメントは解読の手がかりとなる。しかし開発時を過ぎて途中で修正・追加された機能は，設計書やスペックが残されていないことが多い。とくに小規模の改良・修正が行なわれるときには，元の設計と異なった機能が追加されたり，方針が異なった試用に変更されたりしている。これらのドキュメント

は作られてなかったり，作られていたとしても完全でないことが多い。ドキュメンテーションがしっかりしていないシステムの手直しは，厄介であると同時に危険でもある。

ソフトウェア開発のスタッフが多い大企業では開発時のドキュメンテーションの管理もきちんとなされているであろう。しかし一般に小企業や個人企業では，情報システムを開発し，さらに保守運用するための選任のスタッフを抱えていないことが多い。このようなケースでは，開発をアウトソーシングすることになるが，システムの改良・機能追加などは要注意である。開発時と同様にアウトソーシングすれば，一部手直しとはいえ，新規開発に近いコストがかかる。一方，内部にシステム開発をするスタッフを抱えると，定常的な人件費コストの負担を大きくする。このあたりはどちらの方策が有利なのかは一概には言えない点が厄介である。

情報システムが生き物であるという考え方からは，先進的な開発ソフトウェアを使いこなせる技術者を少数精鋭的に内部に抱えることができれば，システムの常時のレベルアップようきゅうにも対応でき，ドキュメンテーションの管理にも目が行き届くはずである。しかし，少数の専門家に業務を全面的に委託し依存すると，ドキュメンテーションの作成や管理は，後回しにされがちである。それは経営者が配慮しなければならないことであろう。経営者や組織のトップは，システムの開発を一部の技術者に任せたり依存してしまうのでなく，全体の管理者としてのシステムに対する理解と，開発時の設計書やその後のメインテナンスのドキュメントを管理してゆく必要がある。

参考文献

1）オフィスオートメーション　2000年
「情報システムのライフサイクルについて」
—— ウォーターフォール型とスパイラル型 ——
九州産業大学　鷲頭　正憲