

第2回 変わるソフトウェア開発現場

高根英哉／コラムラッド

高根さんは自らソフトハウスを設立し、Windows上のソフトウェアコンポーネント開発に挑戦しておられる。実践経験を踏まえて、開発現場の問題点、新しいソフトウェア開発現場のあり方、その息吹を語っていただいた。

エディタ：青山幹雄

私がソフトハウスを設立してから、まもなく10年になろうとしている。設立当初に目指したものは、良質のソフトウェアを開発する受託型企業。それが会社を成長させる最も重要な要素だと考えていた。とにかくスケジュール通りに請負を消化することの繰り返し。C言語を使いこなし、一通りの設計が行え、不具合の少ない成果物があれば、仕事には事欠かない。当時はそんな時代だったと記憶している。振り返ってみれば、自分自身も営業の必要性すら感じずに開発に明け暮れ、経営というものを放棄していたように思う。

そんな経営状態のまま2、3年が過ぎていき、いつしか、これといった得意分野を持たない器用貧乏な会社になっていた。そう自覚したときには焦りを感じたものである。経営も赤字でこそないが、半年も仕事が滞れば倒産するかもしれないという、不安定な状態が続いた。おそらく、小規模ソフトハウスにおける典型的な内部事情だったのではないだろうか。受託して開発する毎日、そんな日々が無駄の繰り返しにすぎないということによりやく気づき、それらを解消する手段としてコンポーネントに着目したのが、90年代前半のことである。

私たちが着目したのは、Windowsで動作するVisualBasicという言葉だった。その上で動作するVBX[★]というコンポーネント規格を研究してみると、最も現実的なコンポーネント規格であることが判明した。希望と期待の中、とりあえずVBX製品を開発

してはみたものの、製品の流通や値付けの難しさを痛感することになる。最初の試みは成功とは言い難いものになってしまったが、決して後ろ向きな市場ではないことも確信できた。その後、図-1に示すようにWindowsが32ビット化されるにともない、OCX^{★2}、ActiveX^{★3}と開発を重ねた結果、徐々に市場の認知を得ることができ、今日に至っている。ここまで来るのに5年以上もかかった計算になる。

また、一方でユーザの間でコンポーネントを使用しないという気運が最近の日本で高まりつつある。それは、コンポーネントベンダがいつ販売・サポートをやめるかもしれないという不安によるものだ。確かに、ブラックボックスのコンポーネントが自社の財産ともいえるアプリケーションに組み込まれるのだから、途中でサポートが打ち切られてはユーザは困ってしまう。それに、OSの進化などにもリアルタイムに反応できないことも十分予想できる。そういったことを理由に、ユーザ自身で開発されたもののみを利用するというわけだ。なるほど、領ける話ではあるし、今現在では賢明な処置といえるかもしれない。しかし、コンポーネント開発のメリットは、それを補って余りあるものだというのを忘れないでいただきたい。ソフト開発の現場は刻々と変わりつつある。昔のように、3年で開発し10年利用できるソフトウェアなどはもう不可能なのだ。目覚ましい進歩の中でコンポーネントと上手に融合していかない限り、今後のソフト開発はすべて欧米中心になることは間違いないだろう。

そんなことにならないうちに日本のソフト産業が健全な方向に歩めることを期待して、私は1997年にActiveXベンダ会を設立した。今は、有志と共に市場の基礎作りに参加している。コンポーネントによるソフト開発こそが、ソフトウェア革命に必須の技術である。そう信じている私にとって、この市場が立

★ VisualBasicで利用されることを目的に開発された16ビットコンポーネント規格。市場の広がりから他のあらゆるコンテナから利用された。

★2 VBXをOLEのコンポーネント規格に拡大したもの。当然32ビットが基本となるが、VBX同様VisualBasicで多く利用される。

★3 OCXにインターネットで利用される仕組みを加味したもの。ActiveXからサーバで利用されるコンポーネントが定義される。

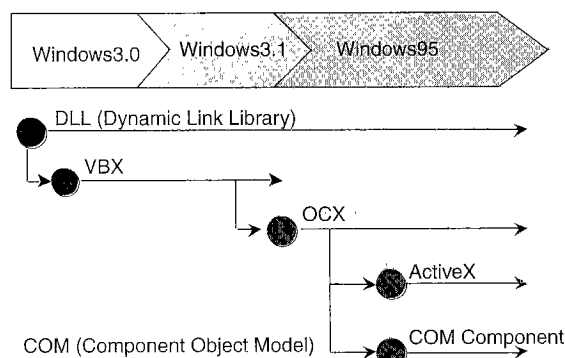


図-1 Windowsにおけるコンポーネントの進化

ち上がることが切なる願いなのだ。

教育のない現場

ソフトウェア業界の新人教育として、スタンダードな方法がOJT (On the Job Training) だ。その基本は、仕事の分かるベテランエンジニアとコンピュータを初めて触る数人の新人でチームを組み、1つの仕事を請け負うというピラミッド構造である。完成される仕事には必要以上の労働力と工数がかかるのは必至であり、当然ベテランには多大な負荷がかかる。ピラミッドの底辺に位置する新人に期待されるものは、ベテランの仕事の仕方や技術を観察し、体得することだ。

だが、理論的に考えてみよう。仕事ができる＝指導が上手、とは必ずしもいえないのが現実だ。教育より仕事の完成を第一義としてきたベテランが、新人を足手まといに思うことの方が、むしろ自然である。結論として、このような教育方法で年数だけを積み重ねたエンジニアは、仕事のつじつま合わせとドキュメント整理にのみ長けるようになる。ソフトウェアの本質を理解しないまま、ベテランの仲間入りを果たし、ピラミッドの上層に位置することになるのだ。現実のソフトウェア教育がこういったお粗末なものであるため、コンピュータやOSの原理を理解するといった基本的な能力が一向に育たない。「なぜ、そのようにソフトウェアを設計したか？」という問いかけに対し、「過去にそのように設計して開発した実績があるから」という、なんとも危うい答えが返ってくるのもそのせいである。

もちろん、OJTにも利点はある。OJTでこそ培われる勘や経験も重要であることに間違いはない。しかし、ベースの何もない新人を数週間の研修経験だけでいきなりOJTに組み込むほど、ソフトウェアは単純なものではないのだ。客先で肩身の狭い思いをし、わけもなく残業ばかりを強いられる新人たち。その苦悩を会社はもっと考える必要がある。

以前、このような場面に遭遇したことがある。大手メーカーA社で大規模プロジェクトが発生し、中堅ソフトハウスに対してUNIXとC言語のベテランエンジニア数十人の派遣を依頼したときのことだ。数日後、若く元気なエンジニアがたくさん派遣されてきた。しかし、彼らの経歴を確認すると、C言語を扱えるエンジニアはその中の数人のみ。残りはC言語はおろかUNIXに触れたこともない新人だったのである。80年代後半、日本経済は好景気で沸き返っている時期のことだった。このような劣悪な条件でも仕事に着手し、使えない新人を含む膨大な人件費は支払いながらも、プロジェクトを完成しなければならなかったA社に同情する。

これは別に珍しい例ではない。多くのソフトハウスがOJTの名のもとに、顧客に教育コストまで負担させていたのである。ソフトハウスの社員にとっては、自社の社員の顔すらほとんど知らず自分専用の机も持たないまま、あちらこちらの客先へと派遣されていくのが日常茶飯事であった。

10年後の今日でも、ソフトウェア業界の教育方針は基本的には変わっていない。相変わらず、OJTを名目に実質的な教育を放棄しているのだ。しかし、顧客の方は少し様子が変わってきている。人余りといわれる今日では、仕事のできないエンジニアを頭数に計算するような余裕はないのである。今まさに、ソフトウェアの教育方針を転換する時期に差しかかっている。

米国のカンファレンスに参加すると、日本とのギャップに驚かされる。米国ではセッションが終わると同時に多数の人間がさまざまな質問をあげるのだ。セッションの不手際は厳しく批評するし、不明な点はとことん究明する。そのカンファレンスにおいて、自分の技術力を高めようとする意識が非常に高い。

それに比べ、日本のカンファレンスは単なる業務命令による参加という意味合いが強い。単純に時間だけを消化し、セッションが終了するや退出し、一番重要なQ&Aタイムを無駄に過ごしているのだ。

米国のエンジニアは自分のためにソフトウェアを勉強し、技術力を高めるように努力していると思う。教育は自分自身で自分のために行っているのだ。カンファレンスにも自費で登録している個人名が目立つ。会社で自分自身を売り込むためには、それに見合った技術が必要になり、今後の技術ニーズがどこにあるかを研究する必要があることも彼らはよく知っている。終身雇用制度のない米国では、自分自身が唯一にして最大の商品なのだ。

最近ようやく日本でも、同様の意志を持ってカンファレンスに臨むエンジニアが徐々に増えてきた。マイクロソフトの調査でもカンファレンスに個人で登録する数は年々増えているという。業務命令でいわれたことのみを消化している旧世代のエンジニアは、今後は

間違いなく取り残されていくであろう。

組織構造の無駄

日本はいわゆる縦社会といわれる組織構造を持つ。自動車産業や家電メーカーなど、あらゆる産業の企業がピラミッド型の組織構造を基本としている。これが自然な企業の成長スタイルであることは間違いない。合理的な管理方法であり、役割分担もはっきりする。ただし、この中に大きな欠点が含まれていることを忘れてはいけない。

まず、身軽に仕事を裁量できないのがその1つ。すべての作業に許可が必要になり、創造性を著しく阻む。会社にとって有益な情報や発案であったとしても、実現までには多くの手続き（多くは管理上必要という、ハンコの回覧板である）を踏まなければならない。ましてや、それが組織の構造そのものに影響を及ぼすような発案である場合には、構造維持を優先するという理由で発案が却下されることも少なくない。組織を維持することが使命となってしまった企業は、命ともいえる人材の生み出す創造性を犠牲にするわけである。

また、横の連携がとれていないことも重大な欠点だ。組織の中にある1つ1つのピラミッドがすべてになると、その外側にある大きなピラミッドの存在が見えにくくなる。ましてや、同列に並ぶ隣のピラミッドなどは別会社にも匹敵する。結論として、同じ会社でありながら、周りの動きは見えにくくなり、同じような仕事や調査をお互いの協力なしに行うことになる。この無駄こそが企業の合理性を阻んでいるのだ。

このような欠点を抱えながらも、伝統の縦社会を維持しつつ日本の企業は成り立っているわけだ。さて、ソフト産業においてはどうかだろうか。

ソフトハウスも会社が成長するにつれ、同様の組織構造を持つようである。さすがはニッポン株式会社の一員、多くのピラミッドを築き上げ、管理能力と経験則でプロジェクトを推進するようになる。保険システムは保険システムのみ、流通システムは流通システムのみを研究し、個別に開発を繰り返す。互いに横のつながりはなく、技術協力は生まれない。ここでも縦社会構造を持つ同様の弊害が生じているようだ。

私は企業に技術セミナーの講師として訪問することが多い。企業から依頼を受けた技術的なタイトルで、2時間から3時間程度の解説を行う。最後にQ&Aを行い、ある程度の理解を得られれば成功だ。

しかし、ときどき意味不明な依頼を受けることがある。意味不明というのは依頼内容ではなく、その申し込み方法だ。つまり、同じ企業から同じ内容で同じ依頼を受けるのである。ときには時期までも重複するようなことがある。最初は、同じ講義依頼に対して担当

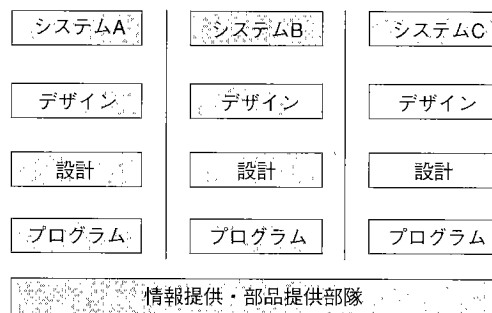


図-2 開発モジュールの共有

者が2人存在するのかと勘違いしたほどである。しかし、出所は異なるらしい。どうやら、依頼をした当人たちも自社内で依頼が重なっていることに気づいてないようだ。こうなると、私が交通整理をして依頼をまとめるのが一番効率的だ。もし、このまま2件の依頼がどこかでリンクすることなく、話が進んでいたらどうなるのだろうか？

同様の話は尽きない。苦心の末に解決したトラブル情報を企業内でまったく共有できなかったり、同じようなアプリケーションをさまざまな課で新規から開発してみたり……そうなるともう、何をかいわんやである。

ソフト産業は情報収集と迅速さが命だ。にもかかわらず、同一企業内ですら情報が共有できていないのが現実。これでは今後のソフト産業に大きな不安を残すことになる。縦社会の無駄を根本から見直す必要に迫られているのではなかろうか？

今、中堅SIベンダを中心に、これらに関する大きな変化が始まっている。図-2のように、どの組織にも属さず、情報を整理し社内に広めることを業務とする部が発足しつつあるのだ。目的は、ソフトウェアの開発に関する基本指針を定めたり、OSやコンピュータに関するあらゆる情報を収集し、社内に普及することである。また、同じようなモジュールを重ねて作成しないように、基本モジュールの提供まで管理している会社も存在する。

これらの活動が現実的に効果を上げるためには、1つの基本方針が必要とされる。それがコンポーネントである。たとえば、共通のモジュールを提供するにしても、どの組織がどのような切り口でそのモジュールを必要としているかは千差万別である。ソースコードを提供したところで、利用には開発に匹敵するくらいの労力が必要になる。だが、コンポーネントであれば、常に閉じたモジュールとして提供されるため、組み込みの手間が省けるのである。また、市販コンポーネントによる情報やモジュールの提供も可能になる。それらを繰り返し活用することにより、数多くのコンポーネントを蓄積することになり、しいては情報やモジュールの共有化に繋がるのである。

中堅SIベンダを中心に広がりつつある「コンポーネント専門部隊」。その存在が、ソフト産業の組織構造の無駄を払拭できるかもしれない。

見えない開発コスト

ソフトウェアの価格ほど不透明なものはない。価格差が10倍以上あっても性能的になんら変わらないという話はよく聞く。標準価格らしきものも存在しないから、「高い・安い」の根拠が希薄になる。なぜこのようなことが起きるのだろうか？

ソフトハウスが請負業務の広告を出しているという話はあまり聞かない。通常の産業であれば、「こんな仕事をこれくらいの金額でやります」という広告を元に営業を行うはずである。しかし、ソフトハウスの営業は少し異なる。大手ソフトハウスの請負営業の基本は「ブランド力」と「お知り合い」である。つまり、これまで頼んでいたところに継続的に仕事を依頼する傾向があるのだ。本来は、発注する際に最も重要になるべきである仕事のクオリティや費用などは、二の次どころか検討対象にすら上がらないことも多々ある。依頼主としては、面識度や会社のブランド力以外にソフトハウスを判断する材料を持っていないのだから仕方ない。ソフトハウスが「1,000万でやります」といえば、その金額をベースに交渉するものだと思います。もしこのとき、「弊社は同様のソフトウェアを100万円で開発します」という新鋭のソフトハウスが登場しても、胡散臭がられるのが落ちだ。ゆえに、費用に10倍の開きがある同性能のソフトウェアというものが存在しても不思議ではないのである。

しかし、仮にソフト産業が面識とブランド力でしか競争が行われないのだとすれば、この産業に長続きは望めないだろう。まず、新規に参入してくる会社にチャンスはなくなる。技術とアイデアというベンチャーの基本が無視されるからだ。新規に参入したソフトハウスは、大手企業の請け負った「お知り合い」開発を小規模な価格で下請けするのが関の山。これではあまりに基盤が貧弱ではないだろうか。

また、大手ソフトハウスは生産性や技術力の向上という、重要な努力を怠る結果になる。失敗さえしなければ次の仕事もある程度保障されているのであれば、新たな試みなど行う必要はなくなる。その結果、納期を最優先に、要求仕様を満たすことだけを考えるようになるからである。顧客の要求を把握する能力を軽んじる気はない。だが、手数があまりにも少なく、同じシステム同じ仕様同じプログラムを繰り返し、予算がつく限りなんの疑問も感じずに行われる開発には、危惧を感じる。

よく「お役所仕事」という雑言を吐く人がいるが、これは効率を考えずに仕事を行ったり、競争原理が働かないようなケースで使われる言葉である。現在のソフト産業が楽な仕事だとは決して思わないが、競争原理から外れていることは事実だ。このような原理が業界をリードすべき大手ソフトハウスに蔓延している限り、生産性の向上などは机上の空論と化してしまうだろう。

請け負う会社のブランド力がここまで影響を及ぼすのは、世界の中でも日本だけではないだろうか？ 数人で経営するようなソフトハウスが、技術とアイデアを武器に大きな契約を結ぶというのは、欧米ではよくある話である。逆に、会社が大きくなりすぎて技術力が低下することを恐れる傾向さえある。今後さまざまな産業の自由化が進むにつれ、これら欧米で技術を磨いたソフトハウスと同じ土俵で戦わなければいけない日がやってくるだろう。そのときでも、「開発に3年、予算にして2億円になります」となどと平気で答えられるのだろうか？

ソフトの開発技法は星の数ほど存在するし、今日でも技術は日々進化し続けている。納期を満たし、顧客の信用を保持できたとしても、それをさらに半分の期間で開発する方法があるかもしれない。これらを切磋琢磨することの重要性を問われる時代に差しかかっている。

OSを無視する開発手法

前述のように、ソフト産業は国際競争力をつける時代だ。設計開始から3年後5年後にソフトウェアを供給しても使われないことは明白である。効率よく確実にソフトウェアを開発する準備が必要なのだが、残念ながらその緊迫感がこの国にはあまりない。

さらに日本ではもう1つの不安材料が残っている。日本の開発スタイルは顧客要求の実現を第一義とする。これはもちろん正しい判断である。しかし、ソフトウェアの場合、そのためにあらゆる手段を高じる必要が生まれたとき、少し様子が変わってくる。

Windowsに限っての話かもしれないが、日本はC言語大国と呼ばれている。つまり、世界で一番C言語を利用するという意味である。ではなぜ、C言語が好まれるのだろうか？ Windows上でアプリケーションを開発するのに有名な言語として、Visual Basic (以下VB) が存在する。おそらく世界中のWindowsアプリケーションの半分以上がVBを利用して開発されているだろう。日本でも、最も愛用されている言語製品であることに間違いはない。

それでもC言語の普及率が他の国と比較して突出しているのはなぜだろう。それは、「VBにできてC言語

で実現できないことはありえない」，逆にいうと「C言語でできないことは何もない」，つまり「C言語万能説」というお題目を日本のエンジニアたちが信じ込んでいるからにはかならない。VBはOSに最も柔軟な言語であるがゆえに，Windowsの機能を直接利用する方法を不得手とする。ましてや，OSの管理にまで影響を与えるようなプログラムは記述しにくいように設計されている。しかし，C言語ならどんなことでも隙間をかいくぐっても実現できると彼らは信じているのだ。

Windowsを使用しないわけにはいかないという現実と，顧客の要求にすべて答えなくては行けないという企業倫理の狭間で揺れ動くソフトハウス。いかなるテクニックを利用しようとも実現する方法，というものを，彼らは常に模索しているのである。これが「あらゆる手段」といわれる部分である。

では，一見企業努力とも思えるこの方法に潜む危険性を考察してみよう。

まずOSのバージョン変化に注目してみると，図-3に示すように，年々早くなっていることに気づく。しかも，サービスパックやブラウザのバージョン改訂などを含めると，OSのバージョンは刻々と進化する時代に突入していることになる。そのような時代に，OSが推奨していない手法や非公開のAPIを利用して実現しているアプリケーションがあったとしたらどうなるだろう。

これらのアプリケーションは，決してOSの保障を受けることはない。それはつまり，何かのタイミングで突然動作しなくなる危険性が非常に高いことを意味する。当然この場合，アプリケーションの寿命は大変短くなる。が，顧客はこれを許さない。結果として，開発を担当したソフトハウスは開発に要した工数と同等，もしくはそれ以上を要してメンテナンス作業を継続することになる。こうなると開発には終わりがなくなることになる。いったんスタートしたプロジェクトは終わることなく更新を義務付けられ，エンジニアはどんどん疲弊していく。「忙しいが利益が出ない」という最悪の結果に繋がっていくのは自明の理である。

そもそも，一番初めのOS選択の時点でWindowsを採用した理由が，環境の安さと普及率だけだったとしたらどうなるだろう。目的のアプリケーションに適したOSかどうか判断せずに開発を行っているのだから，当然しわ寄せは開発エンジニアに向かう。

「ファンクションキーを利用したい」

「Enterキーで処理を進めたい」

「10msごとに割り込みをかけたい」

「10年間使用し続けたい」云々。これらが必須であるならば最初からWindowsを選択すべきではない。それに適したOSを調査し選択すべきなのである。

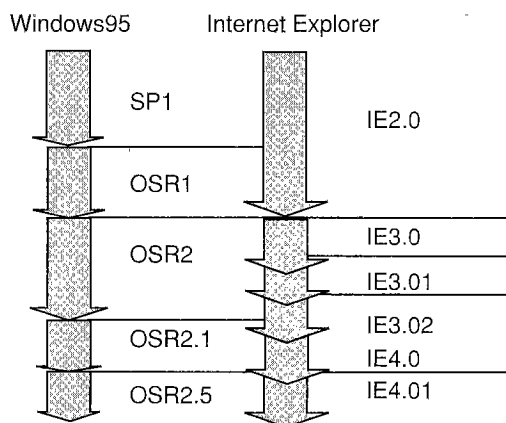


図-3 Windows95の内部バージョン

とはいうものの，Windows以外のOSが現在心許ないことも事実である。OS/2やMacintoshにしても，一時の勢いはなくなった。では，現場はどのような姿勢で開発に臨めばいいのだろうか？ この答えは，これも欧米のアプリケーション開発にありそうだ。

欧米のさまざまなアプリケーションは，概してシンプルである。日本のアプリケーションのように，Windowsの特性を無視してまで実現しているきめ細かい機能はないが，目的を達成するには十分である。また，これらのアプリケーションがとても早いタイミングで市場に投入されていることにも驚かされる。それは，彼らがWindowsの特性を最優先に設計を行っているからだ。ゆえにOSのバージョンアップにも対応しやすいし，VBで十分にアプリケーションを開発することができる。Windowsが進化すれば，アプリケーションもそれに合わせて変化することを当然としているし，ユーザもこれを認めている。メニューの配置が変わっただけで納品を拒否される日本とは大違いである。

変わるソフトウェア開発現場

日本のソフトハウスは，OS（Windows）のバージョン問題を真剣に取り組み始めた。1つの回答として，Windowsのバージョンを固定するという愚策に出た会社もあるが（これも長い目で見ればほぼ不可能であろう），多くのソフトハウスはOSへの順応路線を歩み始めている。ユーザも市場の自由化にともない，意識を少しずつ変えてくるだろう。もう少して開発現場の様相は一変するかもしれない。

（平成10年4月2日受付）