情 報 シ ス テ ム 52－3
（１９９５．１．１７）

オブジェクト指向（ＧＵＩ）開発ツールを利用したアプリケーション開発事例

安達／正敏　＆　前村／克己

カテナ株式会社

ＰＣハードの著しい性能向上、次々と登場する新しいＯＳ、極めて短いサイクルでのハード・ソフトのバージョンアップ等の状況の中で、高品質低コストかつ短期間での開発が要求されるＰＣシステム開発に、従来のトップダウン方式による構造化手法及びトランザクション処理に適した手続き型言語を用いての開発手法を踏襲していたのでは多くの問題が発生する。

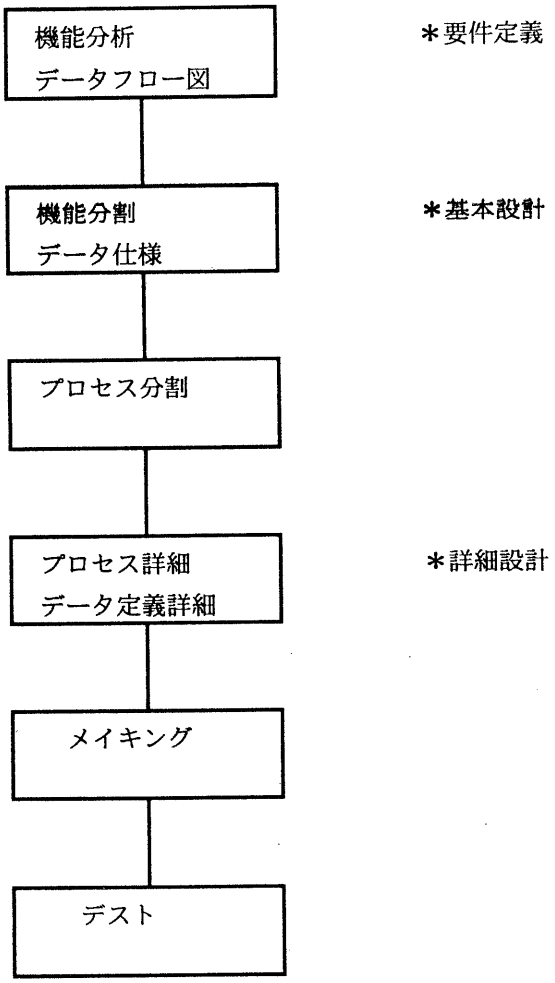
そこで、それらの問題を解消する手段として有効であるオブジェクト指向の開発ツールを使用しての効果の程を、ＷＩＺＤＯＭと言うツールを使用して実際に開発した事例として示す。

1．まえがき

ＧＵＩ開発ツールを用いてのＰＣシステムの開発と、その方法による評価について述べる。

従来の開発手順は、汎用大型システムに浸透している構造化手法（図－１）を用いて、開発を行っていた。

（図－１）



これらには様々な標準化された開発技法が存在するが、現在我々が大規模システムの設計にあたり使用しているのが、データの流れを中心にシステムの設計をしていくことに特徴のあるＤＯＡ技法（Ｄａｔａ　Ｏｒｉｅｎｔｅｄ　Ａｐｐｒｏａｃｈ）である。これは、

まずシステムの論理的な解析を行った後に物理的な解析を行い、それらをデータフローダイアグラム（ＤＦＤ図）で表し、またデータそのものはＥＲ法などで正規化していく、各フェーズにおける成果物等も決められており、システムの上流工程での技法として高く評価されている。ところが、これらの技法が一つづつステップを踏んでいき内部ロジックまであぶりだしていくのに対して、ＰＣでは内部のロジックは（たとえば表での計算、ＤＢとのインターフェイス）等はＰＣのツールが行い我々はその外側を作成する場合が多い。また、これらの技法が数十～数百人という大規模プロジェクトを対象にしているため、プロジェクト管理・スケジューリング等の位置づけが大きいのにたいして、ＰＣの場合ほとんどが１０人以下の少数で行うためプロジェクト分割、モジュール分割、といった従来の概念に当てはまらない事象が多くなってきている。そのため、実際にはこれらの優秀な手法を直接ＰＣに当てはめることはできないのである。

以上をまとめるならば下記のように整理することができる

- 開発の初期段階で実行順序を決定してしまうので、仕様理解が十分でない場合が多い
- 詳細設計の過程で仕様を確定するような目的が曖昧な場合が多く、多くの後戻り発生する
- キー押下など、イベント処理でプロセスを制御するケースが多いので、仕様の記述が難しい
- データ定義とプロセスが密接に関連しているので、変更の内容によっては、手戻りの影響が大きい

現在の我々には、オープン化の波の中でＰＣでの独自の開発技法を作り出して行くいことが焦眉の課題となっている。

２．担保管理システム

今回本レポートの対象である担保管理システムは、短い期間で高密度の内容が期待され、既存の開発手法では対応できない性質のものであった。以下、そのシステムの内容に関して概略を記す

２．１　背景

土地価格の崩壊以降金融業界の深刻な問題として不良債権の問題が表面化してきている。これらは土地の値段が高いときの土地評価額によって融資した物件に対して、土地の価格の下落による評価割れを起こしてても放置してきたことに原因があるものが少なくない。しかしながら為替やオンラインといった部門に比べると担保管理に関してはきわめてシステム化が遅れているのが現状である。そこで、担保の管理と評価を行えるシステムの構築が現在の課題の一つとなっている。実際、これらのシステム化へのアプローチは様々な会社から様々な形で出されつつあり、極めて今日的なテーマである。

そこでそういった要求に応えられるシステムの構築を行うべく、某社の発注により担保管理システムのデモバージョンの開発を行った。その際の重要な考慮事項として

- デモバージョンであることから、各ユーザの要求にあった変更が容易にできる構造であること。
- 開発期間は短期間であること。
- また、担保管理や土地建物の評価においては様々な手法やが有り、現状では固定化できないものもあるため、各種手法（特に土地評価など）の変更に耐えられるつくりであること。

などがあげられた。

２．２　業務設計

業務設計では不動産鑑定士のアドバイスをうけながら設計に取り組んでいった、これらの専門知識を必要とする部分に関してはどうしても専門家との打ち合わせが必要であり。本システムでも、この不動産鑑定士との打ち合わせが全体の骨格に対して大きな影響を与えた。

２．３　主要機能

１）　管理機能

担保内容を融資先、融資情報、土地評価、建物評価に分けて管理しそれらのリレーショナルをとることで有機的な管理をおこなう。また、これらのデータを元に、火災保険の契約が終了した物件の検索といった必要な情報を取り出せる検索機能がある。これによって必要な情報を格納するのみでなく取り出す面でも強化されている。

２）　評価機能

土地を評価する場合の評価手法は数え切れないほどの手法があるが、本システムでは公示価、標準化を元にした事例比較法を組み込んだ。これは、設定された公示価や標準化と現在の評価対象土地を比較して評価額を算出するものである。また、銀行独自の取引事例も公示価や標準化と同様の手法で組み入れることができるようになっている。そして、他の手法もシステム内に簡単に取り入れることが可能となっている

２．４　システム構成

ＰＣ同士でのＬＡＮシステムであり、銀行の融資課（入力）と審査部（判断）を結んでのＬＡＮを想定している。

管理機能で述べたように概念としてＤＢの種類は、担保ＤＢ、融資先ＤＢ、土地ＤＢ、建物ＤＢと４つで有りそれぞれがＩＤで結合している。しかし、後述するようにＷＩＺＤＯＭでは１画面１ＤＢであるという制約から物理的には画面数だけのＤＢより構成されて

いる。（担保ＤＢ：１、融資先ＤＢ：２　土地ＤＢ：３　建物ＤＢ：２の構成になっている。キャラクタ単位でのデザインであるため１画面での情報量には制限があり（８０＊２５）どうしてもＤＢ数が多くなっていった。

３．　開発経過

３．１　ツールＷＩＺＤＯＭの概要

今回の開発で我々が採用したのがＷＩＺＤＯＭと言うツールである。これを使用することで生産性と効率の向上に期待するものがあった。

ＷＩＺＤＯＭは英国ウイズダムテクノロジィー社によって開発されたオブジェクト指向アプリケーションジェネレータである。従来のシステムがファイル設計／画面設計といった各種工程を経てプログラミング、テスト、運用といった工程にいたるのに比べ、ＷＩＺＤＯＭはスクリーンフローという概念に基づいており、システムにとって必要なオブジェクトをスクリーンや帳票単位で作成し次々と作り出して結合していくことでシステムを形作っていく。大きな特徴として、画面や帳票デザイン時に作成時に直接画面を直接エディットする事で必要な画面ないし帳票を得ることができる。

データの管理部分はＢＴＲＩＥＶＥ（米ノベル社）を使用しているが、これによってＮＥＴＷＡＲＥとの相性も良くＮＥＴＷＡＲＥ系のＬＡＮ配下で動かすのにきわめて好都合である。

ただし、現在のバージョンではＤＯＳ対応版しかないため、最近主流となりつつあるＷＩＮＤＯＷＳでは使用できない。また、画面のデザイン等がキャラクタ単位でしかデザインをおこなえないのでデザインする際にかなりの制約がある（とりわけ一画面中の情報量はかなり少ない。）

以上利点／欠点はあるものの、当面の開発すべきシステムの開発期間が短いこと、デモンストレーションバージョンと言うことで画面回りの修正が多数発生することが予測されることから、画面回りの修正がきわめて容易なＷＩＺＤＯＭを選択した

３．２　ＤＢ設計と画面設計の同時性

既存の手法では、システム設計上におけるファイル設計はＤＯＡなどの標準的な開発技法でもまず第一に設計するべきものとして位置づけられており、そのための手法としてもＥＲ法や正規化などの手法が提示されている。

ところが、ＷＩＺＤＯＭでは、画面がそのままデータベースになることから、漏れのない、論理的矛盾のないＤＢ設計のみを先行して行うことは、その設計通りに実現できないという意味では無意味な作業となってしまう。我々も当初は今までの設計技法から抜けきれずにＤＢ設計を行っていったためかなり無駄な作業を費やしてしまったが、最終的には、ＤＢやデータの関連は今までのようなファイル設計書として作成するのではなく、本シス

テムにおけるデータ相関図を作成し、その関連に基づいて画面の設計を行っていった。すなわちシステムで必要なデータの相関関係をを実現するための画面の遷移（ツリー図）などを開発していった。データベースである。リレーショナルである。ＳＱＬであるなどという認識はぬぐい去り、ＤＢそのものはあまり意識せず、画面を通してのデータの受け渡しをいかにスムーズに行うかを中心に作成していった。このため、通常ならば２段階にわたっての設計が画面だけの設計で終了し。画面はＷＩＺＤＯＭの画面作成機能によってワープロ感覚で作成できるため、画面デザインは特に作成せず直接画面を作成した。これによって開発工数はその部分だけを取り出してみると通常の３～４倍の生産性を得た。また、画面の作成に早くとりかかれたことから、顧客とともに実際にそれらの画面を見てもらいながら開発を進めて行くことも可能であった。

ただし、画面作成において相互に関連するデータの整合性をとるために非表示のデータ領域をもうけてそれに値をいれて行かねばならず。構想としては４つのＤＢの美しいリレーションとなるべきところが、実際には画面１（ＤＢ１）画面２（ＤＢ２）・・・と多数のＤＢの乱立とそれらに関連を持たせるために画面に非表示領域を多数設定しなければならないなど、作成の過程においてかなりのストレスを感じたのも正直な感想である。

３．３　プロトタイプの功罪

今回の開発はデモンストレーション用のシステムであり、銀行などのエンドユーザの方へのプロトタイプとしての位置づけであったのだが、本システムそのものを作成していく過程で画面デザインが早くて修正も容易という特徴から、発注者と互いに確認しあって作成していった。その意味では本システムを作成するためのプロトタイプが存在しプロトタイプのプロトタイプといえるものが一時期存在した。

このプロトタイピングというものは、パソコンの様に小回りのきくシステムでは極めて重要なものであり。ユーザとの意見の調整やお互いの誤解をなくし作業時の無駄な空間を極力小さくしていくために有効な手段である。実際に我々も画面を作成した段階で発注者に確認してもらい意見の交換を行っていった。

プロトタイプは実際に有効な方法ではあるが、時としてプロトタイプがあるが故に問題が発生する場合がある。最も多いのが、プロトタイプの位置づけを曖昧にして開発を行ってしまったために、その位置づけが開発側と発注元との間で食い違った捉え方をした場合などである。プロトタイプでは実際の中身（データのやりとり、加工等々）は作成せず見た目の画面などでそれらのイメージをとらえてもらうのだが、本来の目的を逸脱してそれらを実際のシステムのように、数字や結果、相互の関連を問題にしたり、それらがそのまま実現すると思われたりすると、まだ設計をきちんと行っていないにも関わらずプロトタイプが一人歩きしてしまう場合が多々ある。これらの解決にはは、プロトタイプの位置づけを相互に確認し、目的をはっきりさせるといった基本的なことの徹底に帰着する

３．４　ツールに規定される開発

今回に限ったことではないのだが、ＰＣでの開発ではその特殊性からベースとなる開発ツールによって大きくその開発手段が変わってゆく、そもそも開発に使う「言語」にしてもツールのマクロによって大きくその機能が異なっている。今回の場合もその例に漏れずＷＩＺＤＯＭというツールによって規定された部分が多かった。それは単に画面がＤＢの単位であることから画面設計を優先しなくてはならないと言うことだけではなく。エラー処理、入力方法にしてもそのアプリケーションの機能の中で行わなくてはならず、設計してしまっても実現性がない等という経験も少なからずあった。汎用機ではＣＯＢＯＬやＦＯＲＴＲＡＮといった言語の骨格はほとんど変わっていないため、ある程度の経験者が設計をすれば実現性のない物を設計してしまう等と言うことはあり得ないのだが、ＰＣでは十分考えられる。ＰＣの場合は様々なツールアプリケーションの開発速度が日進月歩であり、それぞれに特徴があり、必ずしも同じではない。そのため、開発の過程におけるツールの選択はきわめて重要である。また一度選択してしまったら今度はそれに規定された開発課程を踏んで行かねばならない。ＰＣの場合は開発技法そのものがツールに大きく規定されてしまうのである。

４．おわりに

４．１　今回の開発の考察

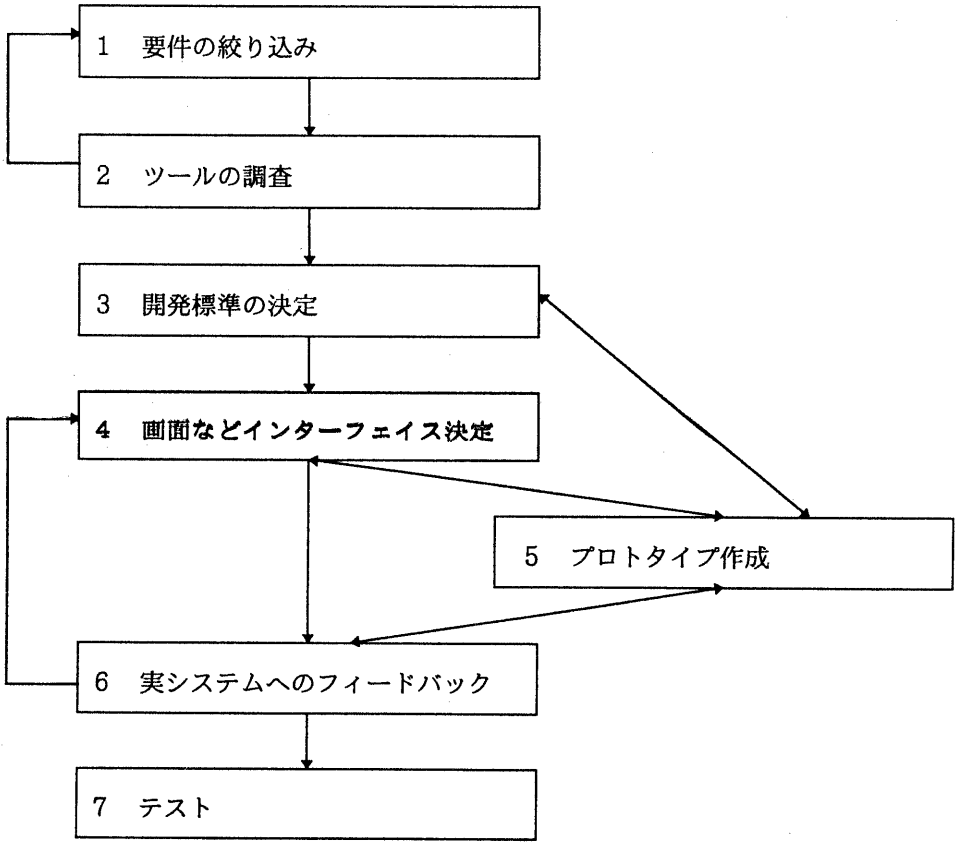
ＷＩＺＤＯＭ等を使った場合では、まず画面の設計が第一となる。どのようなデータをどういった順序で入力していくのかということを決定し、それからそれらにあわせて最適な帳票やデータ構成と行ったものを作成していくのがもっとも適した技法であるといえる。特に帳票はどうやっても作成できないものもあるので、そのデザインの決定はシステムに対する要望を実際のＷＩＺＤＯＭの機能とつきあわせながら決めて行かねばならない

今回の開発を省みると、全面的に肯定するわけにはいかない部分が多数ある。一つにはＷＩＺＤＯＭ自身の問題である。ＤＯＳのキャラクタベースでのツールであるため画面デザインに制限が多く、ＷＩＮＤＯＷＳの美しい画面を見慣れた目にはどうしても汚くみえてしまう。（一昔前のＴＳＳ端末の画面を想像していただくとよい）また、ＭＩＤのようなマルチウインドウが本来の意味では使用できないので、裏で画面を開いて参照しながら入力といった現在では当たり前のようなことができないなど、ＤＯＳを基盤としたツールであるが故の限界がある。

また、ＰＣの場合は色やデザインが自由なのでそれらに対する標準化の作業が重要になっている。画面のデザイン、ファンクションの定義、フォントサイズ、フォント等を細かく規定していく作業も重要である。

以上をまとめるならば、今回の開発手順は次のようであった（図－２）

今回の開発手順（図－２）



既存の図（図－１）と比較すると上流工程に回帰するケースが多い。ＰＣ開発の実際ではこのように、様々なフェーズが密接に絡まりあいながら螺旋状に開発が進んで行く場合が多いのである。このようなやり方が最適であるかどうかはさらに試行を繰り返しながらの検討の余地があるが。今回における開発はまさに上図のような流れによって開発した。

４．２　今後の方向性

冒頭でも述べたように、我々は開発現場の最前線で日々試行錯誤を繰り返しながらシステム開発に奮闘している開発部門であり、ＰＣにおける開発技法を専門に研究している部門の人間ではないので本日のレポートも散漫な部分が多くなってしまったことをお許しいただきたい。

開発の現場では、ＷＩＮＤＯＷＳ，ＷＩＮＤＯＷＳ－ＮＴ、ＯＳ２等の最先端のＯＳの進歩と、嵐のように出てくる新しいツール／アプリケーションの波の中で、一体何がもっ

ともよいツールであり、開発技法であるのかが常に現実の問題として我々に突きつけられている。そして、前述したようにツールやＯＳに開発過程が拘束されてしまわざる得ないのか？普遍的なＰＣの開発技法は存在するのか否か？で毎日悩んでいるというのが現実である。そのような状況下で、実際の開発現場での実践を通しながら新しいＰＣ開発の理論を探っていきたいというのが我々の思いであり、今回のレポートはそのような意味での第一歩であると考えている。