『マンスリー・レポート』1995/07［特集］

クライアント・サーバーシステム構築のノウハウ集

今年１月、当社で初の大規模クライアント・サーバーシステムが実現した。大型汎用機で稼働していたY市の港湾管理システムをダウンサイズしたものである。開発には、DOA（データ中心アプローチ）を中心としたIE（インフォメーションエンジニアリング）手法を採用し、またスパイラル開発方式によってユーザーニーズをできるだけ反映させた。そこではいろいろな「初めて」に挑戦し、数々の試行錯誤を経て、貴重なノウハウを獲得した。その一部を箇条書きノウハウ集の形で紹介したい。

IE手法の現実的な適用

（1）業務分析、データ分析を行わずして、ER（エンティティー関連）図を描くことはできない。古典的な分析手法によって業務フローを理解することが重要である。

（2）まずシステムの全体図を作成し、データフローを中心とした図を作成する。これがないとER図は描けない。

（3）始めから詳細なER図は作成しない。簡単なデータベースの箱に主要項目、関連項目を描いた概略ER図を作成し、徐々にレベルアップしていく。

（4）IE理論にはバッチ処理の考え方が希薄である。実際のシステムでは、月次締めバッチ処理や大量の印刷処理が存在することに注意する。

（5）マトリックスダイアグラム（業務処理とエンティティーを行列で表したもの）を書いたうえで、業務上おかしな点があれば組織を変更するという考え方は現実的でない。

（6）IE理論によるラフなデータモデルを提示するだけでは、ユーザーは納得しない。ユーザーと仕様を検討するためにも、必ず業務フロー図を作成する。

（7）初期の段階では処理性能を気にしない。あくまで論理設計に重点を置く。

（8）IE手法を使いながらも、「概要設計」「基本設計」という従来の言葉の枠内で作業を進める。

スパイラル開発方式の進め方

（1）スパイラル開発方式の適用は、「ユーザーインタフェースの改良」に絞る。

（2）レビュー前に、「データの更新・参照方法」「標準化の大枠」「採用するツールの“くせ”への共通認識」だけは決めておく。

（3）仕様を先送りできるのは、簡単なチェックや操作性、見やすさについてだけである。

（4）レビューの回数は１画面につき２～３回が妥当。多すぎると仕様がなかなか決まらず、スケジュール遅延の原因となる。

（5）プロトタイプはあくまでプロトタイプ。作り捨てと考え、仕様が決定した後にきちんとしたものを作り直す。

（6）レビューでは、だれがOKすれば作業完了か、キーパーソンを明確にしておく。

［著者・執筆時所属］
野村総合研究所
地域・生活システム部
三田雄仁（みたゆうじ）

［著者現職］
野村総合研究所
システムコンサルティング事業本部
産業ITマネジメントコンサルティング部
上級コンサルタント

ベンダーの効果的活用

（1）外国ベンダーの日本語マニュアルの記述をすべて信用してはならない。誤訳や間違った記述もある。できれば原文を読むことでより早くより正確に情報がつかめる。

（2）新しい機能は必ず自社のマシンでテストしてから使用する。

（3）基本的なことだが、ベンダーとのやりとりは必ず紙で行う。

（4）外国ベンダーの日本法人で解決できないトラブルの場合は、外国の本社に解決してもらう。必要ならば日本に来てもらう。

（5）ベンダーが提供する種々のツールのバージョンはなるべく統一させる。特にクライアント側とサーバー側でバージョンが異なると、機能が限定されることがある。

性能評価・改善の常識

（1）性能評価に影響を与えるパラメーターの設定を適正化する。たとえばオラクル社の場合、「init.ora」というパラメーターファイルが性能を左右する。

（2）アプリケーション（適用業務）ソフトのチューニングは、まずSQL（構造化照会言語）文を疑う（表1参照）。

（3）アプリケーションソフトのチューニングで効果が得られない場合、ハードウェアの増強で解決することも必要である。

（4）ラッシュテストや物理的に回線をまたぐテストは早期に必ず行い、処理方法の実現を確認し、現実的な応答時間を把握しておく。

（5）UNIX上ではゴミプロセスを絶対に放置しない。また、放置させない仕掛けを作る。ゴミプロセスはCPU（中央演算処理装置）を食い続け、性能の劣化を招く。

開発環境の整備

（1）開発環境は本番環境とできるだけ同じにする。特にサーバーやクライアントの機種、性能は、本番と同じでないと正確な性能テストができないだけでなく、コマンドが異なったりして、ソフトウェア開発に影響が出る。

（2）リリース直前の作業を円滑にするため、アプリケーションソフトのリリース方式をあらかじめ決め、実現可能か確認しておく。

（3）クライアントに配布するアプリケーシ

表1　SQL（構造化照会言語）文の十戒

- インデックスを殺すべからず。
- テーブルは大きい順に並べるべし。
- AND条件はゆるい順、OR条件はきつい順に並べるべし。
- NOTや！、=は避けるべし。
- 副問い合わせは避けるべし（すべからくJOINすべし）。
- テーブルにはエイリアスをはるべし。
- すべからくEXISTSを用いるべし。
- NULL検索すべからず。
- 小さいテーブルにインデックスはるべからず。
- パース（解析）回数を減らすべし。

（資料）Rogers & Ulka, *A Database Developer's Guide*、London & Tina, *Guidelines and Good Practice for Developing SQL*、「ORACLE 7 Server アプリケーション開発者ガイド」、および「ORACLE 7 Server 概要」

(1) 処理概要
　入出力（画面、帳票、データベース）がわかる図と処理概要の説明文を記述
(2) 主な機能
　登録、変更、削除、問い合わせの別に機能概要を記述
(3) テーブル更新・参照……表で記述
　○:更新　△:参照

テーブル名	登録	変更	削除	問い合わせ

(4) 入出力画面・出力帳票……表で記述

	名　称	備　考
入力		
出力		

(5) チェック仕様
　①単体チェック……表で記述

ブロック名	フィールド名	必須	更新禁止	存在	チェック内容

　②関連チェック……表で記述

項番	タイトル	チェック内容	トリガー

(6) 備　　考
　その他特記事項があれば、ここに記述
(7) 画面・帳票イメージ
　作成した画面のハードコピーまたはテスト出力した出力帳票

図1　詳細設計書の目次（書き方の標準化の例）

ョンソフトのリリース方式も重要である。

標準化の考え方

(1) 4GL（第四世代言語）開発ツールにも標準化は必要である。ただし、コーディングの自由度が低いツールは標準化を控えるなど、ツールの特性を考慮する。

(2) 標準化の目的は、「ユーザーインタフェースの統一」「開発のガイドライン」「ツールの特性への共通認識」「性能の向上」「保守性の向上」におく。

(3) オンライン画面であれば、機能を参照系、更新系など数種類に分ける。それぞれ雛形を用意して、自動生成ツールを用いて生成させ、生産性の向上を図る。

(4) 画面遷移などのユーザーインタフェースから内部のエラー処理部分までを記述した「標準化手引書」と、雛形の使用方法などを記述した「開発手順書」を作る。

(5) 画面や帳票のレイアウトは、単体開発が完了してからユーザーに提出する。基本設計時に提出すると、画面や帳票が変わるたびに修正しなければならない。

(6) 詳細設計書は簡略にする（図1参照）。

ユーザー端末としてのパソコンとGUI

(1) パソコンの頭であるOS（基本ソフト）の開放には覚悟が必要。クライアントに設定した各種のパラメーターをユーザーが変更する恐れがある。

(2) 業務の内容を考えて、使用頻度の高い、ユーザーのためになる、しかも使いやすいソフトウェアを厳選して提供することが大切で

ある。基本的には、ワープロ、表計算、データベースアクセスツールが必要だろう。

(3) GUI（グラフィカル・ユーザーインタフェース）に向くのは画面上で作業が発生し、人間の判断が必要な業務である。単なるデータ入力が目的の画面は、GUIにすることで逆に煩雑になる危険性もある。

(4) GUIの操作インタフェースは、すべての画面に共通した同じものにする。そのためには、なるべく操作インタフェースが同じ開発ツールを使用する。

EUCの重要性

(1) EUC（エンドユーザー・コンピューティング）を、End User Confusingにしてはいけない。ユーザーはEUCの本質を十分理解できずに混乱している。

(2) 自分で自由に検索・加工するためには、テーブル選択、項目選択、抽出条件指定といった煩雑なことも自分で行わねばならないことを、ユーザーに理解してもらう。

(3) 定型的な業務では「情報はボタン一つで参照したい」というユーザーの要求にこたえるためには、新たに個別の仕組みを作る必要があることに注意する。

(4) EUCに対する開発者とユーザーのずれを少しでも埋めるためにも、また定型的な業務をなるべくEUCで行ってもらうためにも、EUC研修の支援が不可欠である。

(5) ユーザーのレベルごとに、表計算やデータベースアクセスツールなどの基本的な操作研修を何度も実施すべきである。

(6) EUCとしばしば抱き合わせで用いられるマクロという言葉は、開発者側の売り文句になる。しかし、マクロも一つの高級言語なので、安易に導入すべきではない。このことをユーザーに理解してもらうのも大切である。

(7) 簡単にEUCが行える定型業務をマクロ対応で引き受けていては、EUCではなくなる。

(8) ユーザーがデータベースの結合検索を行うのはむずかしい。条件設定を誤りやすく、そのため膨大な検索時間を要したり、ゴミプロセスとしてシステムに多大な負荷を与える原因となってしまう。また、間違った検索結果でも正しいと判断してしまう。こういったことを解決するには、時には正規化を崩してデータベースを設計することも必要である。

重要なノウハウの共有

現在、あらゆる業務処理についてクライアント・サーバーシステムの構築が行われている。だが、どのプロジェクトでも過去の類似プロジェクトの調査が足りず、同じ過ちや失敗をくり返しているのが実情ではないか。

今回のプロジェクトでは、開発にたずさわった全メンバーが、システムのリリース後に社内の発表会でノウハウを披露した。今後は、こうした現場の最前線のメンバーによる発表会を多頻度に、かつ粉飾なく行うことが大切である。（三田雄仁）