中規模流通業システムにおける
コンポーネント・ベース開発の事例

（株）東洋紡システムクリエート

■ 執筆者Ｐｒｏｆｉｌｅ ■

坂　口　丈　幸

1998 年　(株)東洋紡システムクリエート 入社
現在 ソリューション事業部所属

■ 論文要旨 ■

本部および支所全 12 拠点にて稼動し毎週約 100 万件を処理する中規模流通業システムの 100% Windows NT ベース・システムでの実装事例を中心に，そのソフトウェア開発において試みたコンポーネント化技法を，高再利用性・部品の組み替え容易性・適応業務への展開容易性という視点から考察した.

■ 論文目次 ■

■ 図表一覧 ■

１．はじめに

１．１　当社概要

当社は，生活文化を担う魅力ある企業をめざす TOYOBO の情報システム室が独立したシステム・インテグレータである．TOYOBO の１世紀にわたる事業基盤と潤沢な知的情報資産をバックボーンに，経営課題や業務課題を最適なシステム力でソリューションする未来志向のテクノ・シンクタンクをめざしている．ユーザーと情報システムをつなぐ架け橋となるべく，こころ豊かなパートナーシップを築くことを社是としている．

１．２　本システム開発の背景

(1) 顧客

Ａ生活協同組合

・本部，物流拠点の他 10 営業拠点

・会員(組合員)数 約 13 万人

(2) システム化対象範囲

無店舗流通業(共同購入)業務システム

・事業規模 約 140 億円／年

・約 100 万トランザクション／週

対象業務系

・営業準備系：主にマスタを保守

・注文～受注：顧客(組合員)からの注文を処理

・集品～出荷～供給：顧客への商品の仕分けと配達を処理

・発注～入荷～保管：業者への発注と在庫を処理

・債権系、債務系：売掛・買掛を処理

(3) 本システム化前の状態

・約 20 年前に導入されたホスト系システムで，保守・管理が困難になっている

・業態変化への即応性に欠ける

(4) 顧客側の構想

事業のモデル化

・事業をビジネス・プロセスの集合で定義

・差替え可能なプログラム部品で業務フローを構成

・100% Windows NT ベースで構成し，ダウンサイジングを徹底

(5) ツールの利用

顧客の構想モデルに可能な限り忠実に実装するため，開発ツール「COOL:Plex」を選定

１．３　コンポーネント化（部品化）の方針

一口に「コンポーネント化」あるいは「部品化」と言ってもさまざまな切り口が考えられるが，本システムの開発において最も重要視したのは以下のような点である．

(1) 再利用性

類似処理の重複開発を防ぐため，再利用可能な単位を抽出して可能な限りカプセル化

する．

(2) 組み替え容易性

インタフェースを明確にし，顧客要求の変化などに応じてコンポーネントの組み替え・差し替えが容易になるようにする．

(3) 展開容易性

同一コンポーネントをそのまま再利用するのではなく，差分を加味して類似の別コンポーネントを作成しやすくする．

２．本システムの概要

２．１　業務の基本フロー

図１は，顧客の構想モデルに基づく業務の基本フローの概要である．

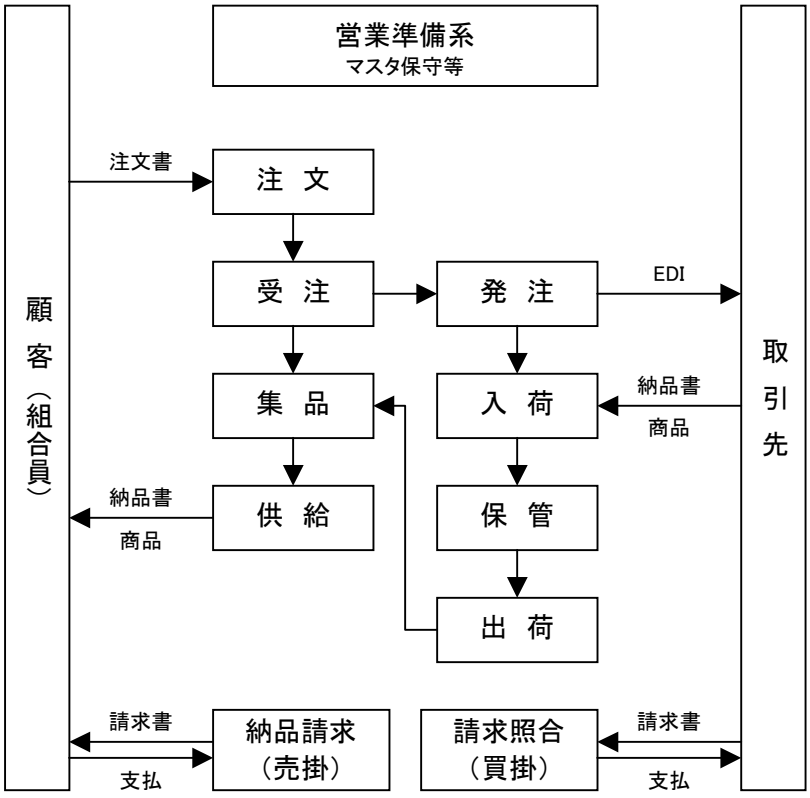

図１　業務の基本フロー

このフローは，モデル化した業務の基本フローであり，実装システムは各拠点別のサーバ上に必要な業務機能だけを配置した分散システムに構築する．

２．２　ソフトウェア規模

本システムのソフトウェア規模は，機能数でみて**表１**のようになる．なお，この数字にはバッチ処理数の他，画面数および帳票数が含まれている．

表１　システム規模（機能数）

系	本部拠点	物流拠点	営業拠点	全拠点計
準備系	150	0	0	150
営業活動系				
注文	20	0	0	20
受注	25	3	16	44
集品	14	80	66	160
出荷	0	1	2	3
売上	18	4	8	30
発注	0	48	8	56
入荷	0	39	6	45
保管	0	4	2	6
債権・債務系				
納品請求(売掛)	46	0	0	46
請求照合(買掛)	67	0	0	67
総合計	340	179	108	627

３．ソフトウェア開発フェーズ

本システム開発のフェーズ（工程）は一般的なシステム開発の場合と大きくは変わらないが，顧客の構想モデルを基本にしているのと，開発ツールとして COOL:Plex を使用しているという点で特徴的である．ここではそれらの特徴を中心に説明する．

３．１　現行業務分析フェーズ

このフェーズでは，現状業務で使用されている入出力帳票・画面を元に，それらを使用目的，使用者，使用場所，使用タイミング，使用方法でまとめることで，業務の実態を整理した．後続フェーズは，全てこのフェーズでの分析結果が基本になる．

３．２　モデリング・フェーズ

これは従来型開発にはないフェーズである．

前述のように，今回の開発では開発ツール COOL:Plex を用いた．このツールは開発プロジェクトの設計情報を「モデル」という単位で管理し，モデル内あるいはモデル間で設計情報を継承できる，オブジェクト指向のツールである．顧客の構想モデルでは，生協業態に依存しない普遍的な性質と業態依存の性質，さらに実装を意識した性質の３階層構造であり，それをツールの持つモデルで表現させるために，以下のようなモデル体系とした．

(1) 基本モデル

生協に共通的に必要な基本コンポーネントと基本データ構造を抽象的に定義する．

(2) 業態モデル

業態（無店舗流通業(共同購入)）特有の全業務のコンポーネントとデータベース定義，および各業務の雛形を定義する．

(3) 実装（本部・物流・支所）モデル

業態モデルを特化して，各拠点で必要な業務（営業準備系，営業活動系，債権・債務系）の細部を実装する．

その上で，分析フェーズで整理した各業務を抽象化し，より本質的な属性や機能を基本モデルに，業態共通の属性や機能を業態モデルに定義した．

分析フェーズで整理した各業務は，基本および業態モデルを継承して実装モデルで定義するものとした．

３．３　設計フェーズ

このフェーズでは，データベースと業務プログラムをスムーズに構築できるようにするため，コンポーネント化の視点から以下のような作業に力点を置いた．

(1) 業務共通部品の洗い出し

各業務で共通な機能を洗い出し，業務共通部品として括り出す．ここでは主に，再利用性の高いコンポーネントにするため，入出力インタフェースの共通化に特に注意を払った．

(2) 業務のパターン化

各業務の入出力パターンを洗い出し，処理の基本パターンを抽出する．今回抽出した処理の基本パターンは以下の三つである．

① 入力データを，業務処理予定データとして蓄積する

② 蓄積した予定データを確定データに変遷する

③ 確定データから，次の業務に引き渡すデータを出力する

ここでは主に，上記パターンの組み替えあるいは組み合わせによってあらゆる業務に対応できるようにすることを目的とした．例えば，上記の①の入力部と③の出力部を組み合わせることによって，蓄積データを持たないフィルタ業務も実現できるようになっている．

(3) 業務パターンに沿ったプログラム設計の標準化

各プログラムの設計は全て業務パターンの枠組みにのっとって行い，詳細レベルの定義はデータベース編集仕様などのみにとどめる．具体的な成果書類は，業務パターンの組み合わせと業務共通部品の関連を記述した図とデータの編集仕様書からなる．

各業務パターンには，処理フローの中で差分コーディングを挿入できる個所があらかじめ決められており，編集仕様に関してだけはプログラム単位で大きく異なる場合が多いのでそれぞれの差分実装の対象とした．それ以外の差分コーディング，例えば処理フローそのものの変更等は原則として禁止することとした．

３．４　実装フェーズ

各個別プログラムの実装作業は，大別して二つのグループに分けて実施した．

(1) 部品作成グループ

設計フェーズで洗い出された業務共通部品と業務パターンを継承可能な部品としてツール上に実装するグループである．これら部品は，全ての業務プログラムの基礎となるものである．そのため，このグループは設計フェーズにも参画し，全体のフローから各種部品の詳細仕様にいたるまで熟知したエクスパートが担当した．

(2) 量産（部品合成）グループ

標準化された設計書通り，部品グループが作成した部品を継承・組み合わせることによって，最小のコーディング量で業務プログラムを量産するグループである．開発ツールの操作の習熟が必要であるが，業務共通部品やパターン部品を使用することで，比較的スキルの低い作業者でもプログラムを量産することを狙っている．

４．問題点と将来への課題

前述のような経緯を経て進んできた本システムの開発であるが，現在最終フェーズである運用テストの最終段階を迎え，コンポーネント化の成果（長所）および問題点（短所）が明らかになっている．

４．１　コンポーネント化の成果（長所）

構造の明確化

従来のような処理プログラム中心の分析・設計に比べ，事業のモデル化からコンポーネント重視の分析・設計の流れを通じて，業務のフローとソフトウェア構造との関係を明確にしやすいという利点があった．

実装フェーズの工数短縮

特に以下のようなプログラムで，工数短縮が著しかった：

- 量産グループで実装したプログラム
- 業務部品の変更で済む場合の，プログラム仕様変更
- １拠点モデルで実装済みのプログラムの，他拠点モデルへの移植

ただし，以下のような場合の実装工数は従来と変わらなかった：

- 業務部品の実装および業務パターンの雛形定義
- 複雑な制御を伴うオンライン画面プログラム
- 類型化が難しい，複雑な処理

ただ，全プログラムの約 80%を占める類型バッチプログラムで工数短縮が見られたので，実装フェーズ全体の工数短縮にはある程度の成果があったと言える．

４．２　コンポーネント化の問題点（短所）

(1) パフォーマンス問題

今回のコンポーネント化開発の最大の問題点は，パフォーマンス（実行速度）問題であった．従来平面的に実装されたプログラムと違って，業務部品の呼出を多数含んでいるので，プログラム・ロジック的にも，データベース・アクセスの見地からも実行速度最適であったといえなかった．

もちろん実行速度要件の厳しい業務に関しては対応を施したが，これらはコンポーネント化と必ずしも相容れない個別プログラム依存の対応がほとんどであった．

将来の開発では，開発時最適であり，なおかつ実行時最適でもあるコンポーネント化開発を考える必要がある．

(2) コーディング・レス化

実装工数の短縮は，主に業務パターンと業務部品の組合せでプログラムの骨格を形作ることができる類型プログラムで著しかった．そのような場合でもデータ編集処理等いくらかのコーディングが必要であったので，今後は更に進歩させてコーディングを一切排したプログラム実装が可能かどうかの模索を試みる予定である．

５．おわりに

前述の通り，本システム開発は現在も進行中なので，テスト等も含めたトータルな意味での工数短縮が十分図れたかどうかの検証はまだできていないが，恐らくは従来型開発の場合とほとんど変わらないかもしれない．しかし将来，拠点の増設や業務の拡張などに遭遇した際に今回の開発の成果であるコンポーネントを再利用できることを考えると，生産性と品質は従来型の開発より有利であると期待される．