スプレッド開発による
リポジトリシステムの構築について

グローバルフォーカス（株）

■ 執筆者Ｐｒｏｆｉｌｅ ■

清　水　　朗

1984年　（株）オリエントファイナンス（現㈱オリエントコーポレーション）入社
同年　システム管理担当
1995年　システム環境開発担当
1999年　グローバルフォーカス（株）出向
1999年　次期システム開発プロジェクト開発技法担当
2001年　ニューカードビジネス推進グループ
アシスタントマネージャ

■ 論文要旨 ■

データ中心アプローチ（DOA）の要となるリポジトリシステムの構築にあたっては，通常は，その準備段階である，データ整備・体系化などに多大な時間と労力がかかり，途中で挫折してしまうケースが多い．

そこで，ビジネスモデルとしてのエンティティ-リレーションシップダイアグラム（ERD）を核とし，実際の業務システム開発に適用しながら情報を体系化・整理して拡大していく，という“スプレッド開発”によって，リポジトリシステムの構築を行った．

この手法により，現在では，設計情報とソフトウェア開発の情報を中心とした項目を管理し，またシステム開発の実態に即した多様な機能を持つ，独自のリポジトリシステムが運用されている．

今後は，既存管理システムの吸収も含め，ソフトウェア開発だけでなくプロジェクト管理に関わる情報も管理対象として拡大し，情報資源管理（IRM）への発展を目指す．

■ 論文目次 ■

■ 図表一覧 ■

１．はじめに

１．１　当社概要

当社は，クレジット業界大手である母体企業の情報システム部門が 1999 年に分離し，富士通社など，数社のソフトウェア会社の出資で設立した，クレジット業務に特化した情報処理システム会社である．

これまでは主に母体となるクレジット企業の基幹系システムの保守・運用を担ってきたが，今後は，クレジット業務開発のノウハウを活かし，「コンサルティング～システム構築～運用までのトータルなシステム」を提供できるビジネススキームへも注力していく予定である．

１．２　リポジトリシステムとは

リポジトリシステムとは，一般的には「システム開発に関わるソフトウェア機能実現のための情報をもつシステム」と定義できる．

しかしながら，ここではリポジトリシステムの管理スパンを拡大し，「システム開発に必要なすべての情報・ノウハウを司るシステム」と位置付けている．つまり，①業務設計書／プログラム仕様書／プログラムモジュール／データベース実装情報などのソフトウェア機能実現のための情報とその構造を決めるための技術，に加え②要員のスキル／単価／工数／スケジュールなどのプロジェクトマネージメントの情報，およびそれを推進するための技術・ノウハウ，に関してもリポジトリが管理対象と考える．

１．３　本稿のポイント

当社ではこれまで，“データ辞書”の作成，“コピー句の統一”，“波及分析ツールの導入”などによって，ある程度のデータ管理を図ってきた．しかし，ソフトウェア情報を「調査する」レベルのものであり，データを「統制する」レベルではなかった．

そのためメンテナンスを続けていくと，データ，プロセスが重複・複雑化し，またドキュメントと実態との乖離が発生してしまうという問題が顕著になってきている．

当リポジトリシステムは，上記問題解決の一環として次期システムへ適用する，ということを目的に構築することとなった．

本稿では，(1)確立されたデータモデルを中心として，リポジトリシステムを段階的かつ有機的に結合・拡大しながら構築していくという手法（プロトタイプともスパイラルとも違うので「スプレッド開発」という），(2)プロジェクト開発の実態に即して構築した現実的なデータ構造，(3)業務システム構築と並行してリポジトリを構築することで，現場ニーズに対する創意・工夫から生まれた独自のリポジトリ機能，の３つをポイントとして述べる．

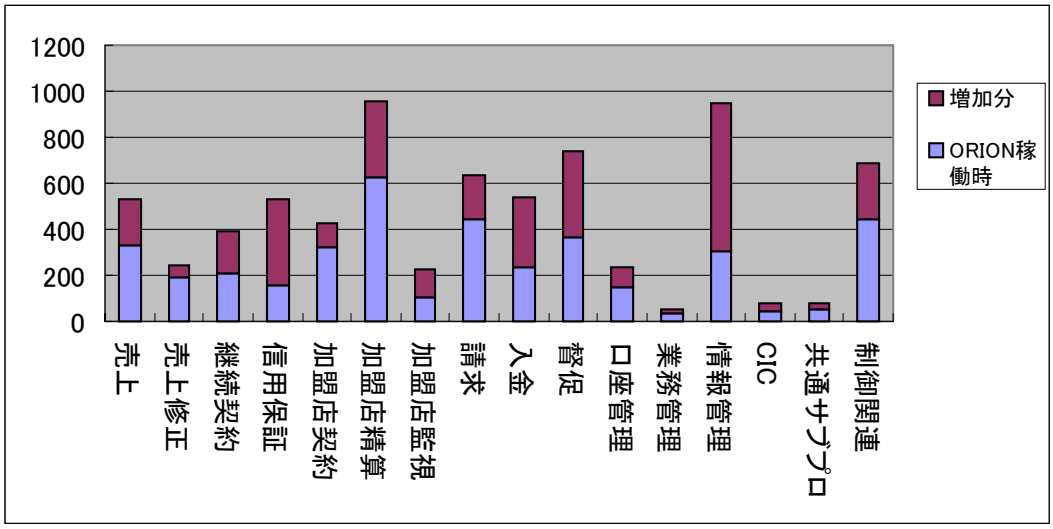
２．背景

２．１　現行システムの問題点

当社基幹システムは，リリースから既に十数年が経過しているが，機能追加以上にシステム規模が大きく，かつ複雑になってきている．**表１**は縦軸にプログラムステップ数（単位：KS）横軸にサブシステムを示したものである．これを見るとシステムリリース時（表１における“ORION 稼動時”の部分）に比べ，約 1.8 倍の規模となっていることがわかる．また同一項目（重複項目）を持つファイルも，年々これを上回る率で増加していることがわかった．これらは明らかにデータ，プログラムの重複によるものであり，いわゆるスパゲッティ状態が顕著となってきていることが判る．この結果，既存システム変更の影響調査や，新規案件の対応に時間がかかるなど，開発生産性の低下が問題となってきていた．

そこで，この状況の打開のため，次期基幹システムの構築を検討するプロジェクトが発足した．

表１　現行システムにおけるプログラム増加度



２．２　問題の解決手段

究極の次期システムは，それが最後の次期システムとすることである．といってもシステムの刷新をしないわけではなく，堅牢なデータ構造を確立し，アプリケーションだけを変更していくことができれば再構築は必要なくなるということである．

現状の問題点の打開策としてのデータ中心アプローチ（以下 DOA という）は，現状の問題解決だけでなく，将来的にも肥大化，複雑化を防止できる手法として採用することとなった．リポジトリシステムは，DOA が着目するデータ，情報を一元管理するものであり，この手法の実行に必須のしくみである．

２．３　リポジトリシステム開発の目的

一面だけ捉えると，リポジトリシステムは DOA 開発を実践するための単なるツールである．しかしその真の目的は，システム本来の姿である「重複がなく，多重管理もされない」状態を保つことであると考える．つまりシステム開発における無駄な作業・処理の排除もリポジトリシステムの目的であるといえる．

３．開発経緯

３．１　開発指針

次期システム開発における結論を受け，本格的にリポジトリに関する調査に着手したが，他社事例を調査していくと，一時の情報資源管理（以下 IRM という）ブームがあったに関わらず，有効に機能しているリポジトリの例はほとんどなかった．また市販パッケージについても，必要なレベルに達しているものがなかったため，社内で構築することとした．

構築にあたり，他社事例，特に失敗例を参考にして，「スプレッド開発」を実施することとし，以下の方針をたてた．

(1) 管理対象の段階的な拡大

いくつかの実際の業務開発へ適用しながら，徐々に管理する対象を拡大する．また単に拡大するのではなく，場合によっては全く別のシステムで管理対象を広げ，後で本体に吸収することも考慮にいれる．

(2) 実態に合ったシステムの構築

システム開発の実態と乖離しないよう，現場の開発担当者の意見をとりいれながらの開発とする．なお現在まで，「開発工数管理システム」，「情報系システム」，「大規模カードシステム」という三つの業務システム開発に合わせて構築を進めてきた．

(3) 拡張性の重視

徐々に管理対象を広げていく，つまり一種のスパイラル開発であるので，拡張性を重視した仕組みを考える．実際にはリポジトリ自身を管理するためのリポジトリ（リポジトリ自身はメタデータなのでメタメタデータという）を構築し管理することにした．

３．２　DOA と ERD

現状の問題点（スパゲッティ化，生産性低下）を解決するための方法論である DOA(Data Oriented Approach 以下 DOA)，その方法論の核となる手法である ERD(Entity Relationship Diagram 以下 ERD)，そして実現するためのリポジトリ，という三者は非常に密接な関係がある．以下リポジトリにとっての重要な要素である DOA と ERD についてその定義と関係を簡単に述べる．

(1) DOA(データ中心アプローチ)

DOA は一般的には「プロセスよりも安定性のあるデータに着目した開発手法」と定義づけがされている．本当の意味で「データに着目」するためには，「確立されたデータ構造」と「データを一元管理するしくみ」(リポジトリ)が必要になる．

(2) ERD

上記 DOA で必要なもう一方の「確立されたデータ構造」を表現する方法が ERD である．ただし，これまでの ERD では十分なルールが規定されていないこともあり，十人いれば十人の ERD が作成されてしまう場合が多かった．今回はこの解決のため，ルール・作法が明確であり，ERD そのものがビジネスモデルとしても利用を考慮し「T 字形 ERD 手法」を採用した．

３．３　スプレッド開発

通常は一つのプロジェクトが終了すると，個人的なスキルは蓄積されるが，組織として

生かされることは少ない．当社においても，年間数回の大規模なシステム開発が行われているが，必ずしもその資産・ノウハウが継承され，再利用されているといえない．この大きな原因のひとつに，組織的にコアとなる開発手法が確立されていないことが挙げられる．

確かに，プログラムの作成方法，画面のスタイルなどは，時代とともに変化してしかるべきだが，企業が扱う情報そのものは，ほとんど変わるものではなく，これを管理する手法は統一されるべきである．この管理方法が確立により情報の継承・再利用が可能となる．

スプレッド開発は，このデータ管理手法を確立した上で，徐々に管理対象データを拡張していく手法であり，機能拡張によって一部のプログラムが廃止，追加されることはあってもデータの変更はない（通常は項目またはエンティティの追加となる）．

今回のリポジトリ開発では，前半で手法確立に十分な準備期間かけ，システム開発に近い管理項目から，徐々にプロジェクト管理項目へと範囲を広げていった（**図１**）また常に実際の業務システム構築と並行して構築してきた（**表２**）．これは開発指針のとおり，開発実態と乖離しない考慮とともに，開発担当者の意見を吸い上げるためでもある．

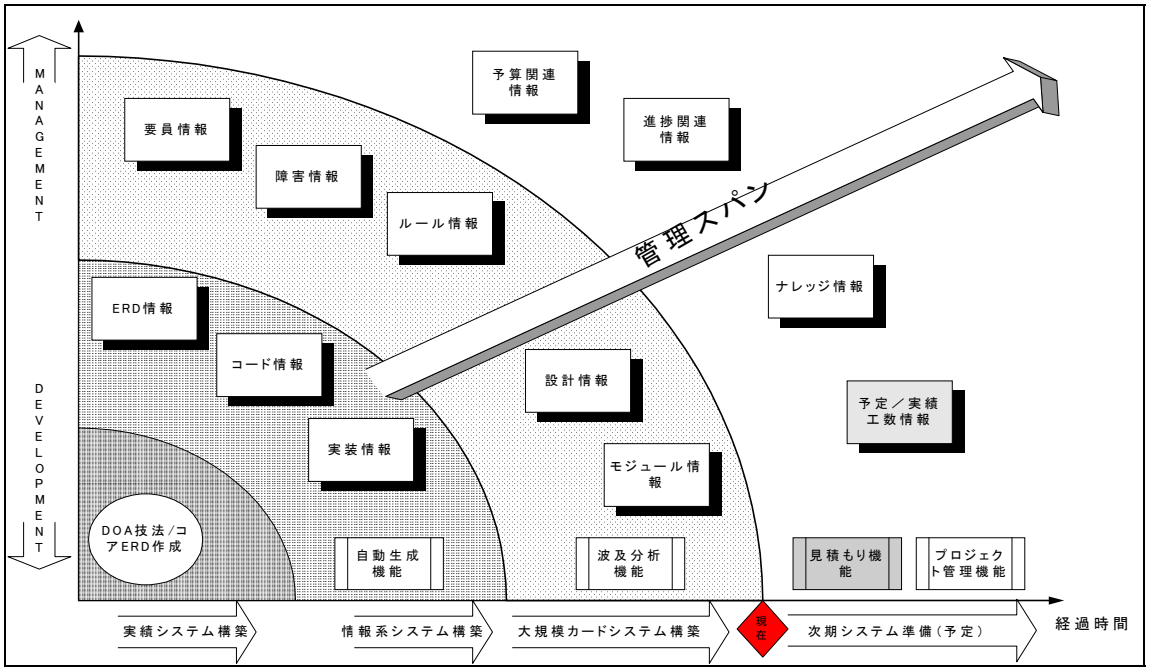


図１　スプレッド開発の推移

表２　各ステージにおけるリポジトリ開発概要

ステージ	0 ステージ	第一ステージ	第ニステージ	第三ステージ
期間	1997-1999	1999-2001	2001-2002	(未定)
業務システム	実績システム	情報系システム	大規模カードシステム	次期基盤システム
実施概要	教育・スキルアップ	コア ERD 完成	設計，実装の一致	全システム開発情報の統合
対象スパン	―	ERD，DB	設計，モジュール	プロジェクト情報
主な機能	―	ERD 管理，実装	モジュール管理	プロジェクト管理

４．リポジトリシステムの特徴

４．１　システム環境

システム環境は，拡張性，移植性を重視し，サーバ，クライアント，データベースに極力依存しないシステム環境とした．当初は教育，トレーニング用に DS/90 上に ORACLE７で構築し，現在は，NT サーバ(DB は ORACLE 8)へ移植している．また社内の他部署への公開用として WEB サーバ(NT)を一台利用している．

現在メインで適用しているカードシステム構築の場合，業務サーバ(UNIX5 台)，リポジトリサーバ(NT2 台)，クライアント（現在約 WIN 系約 300 台）が使用しており，すべて同一 LAN 上にある．なお一部のサテライト開発においては，社外サイトへ DB 環境ごと持ち込み，リポジトリ環境を提供している．

４．２　基本コンセプト

リポジトリシステムはシステム開発に関するすべての情報を一元管理するものであり，**図２**のとおり，分析工程～運用情報における全開発工程での利用を想定している．

この実現のため，①システム内でのデータ，ルールの重複の完全な排除，②ERD とルールと実装情報の完全一致，を基本コンセプトとしている．

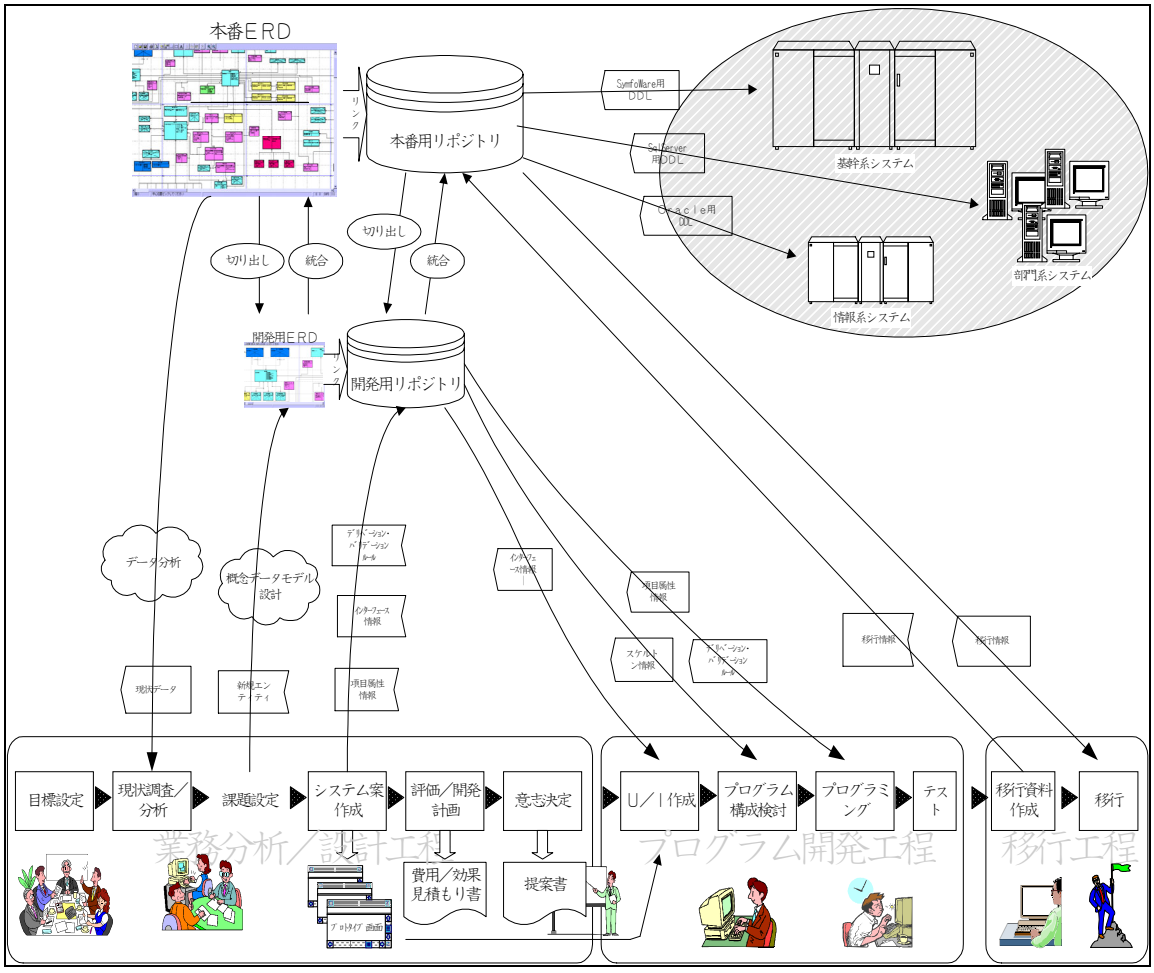


図２　全システム工程におけるリポジトリ利用イメージ

４．３　機能概要

４．３．１　データ構造

ERD，リポジトリ，実装情報の関連を表す最も基本的なデータ構造部分を**図３**に示す．現在この情報を核に，システム開発に関わるモジュール情報～プロジェクト管理情報まで約 800 の項目（230 エンティティ）を管理している．なお実際には業務 ERD と同じく，統一された ERD 表現方法（T 字形 ERD）で描かれている．

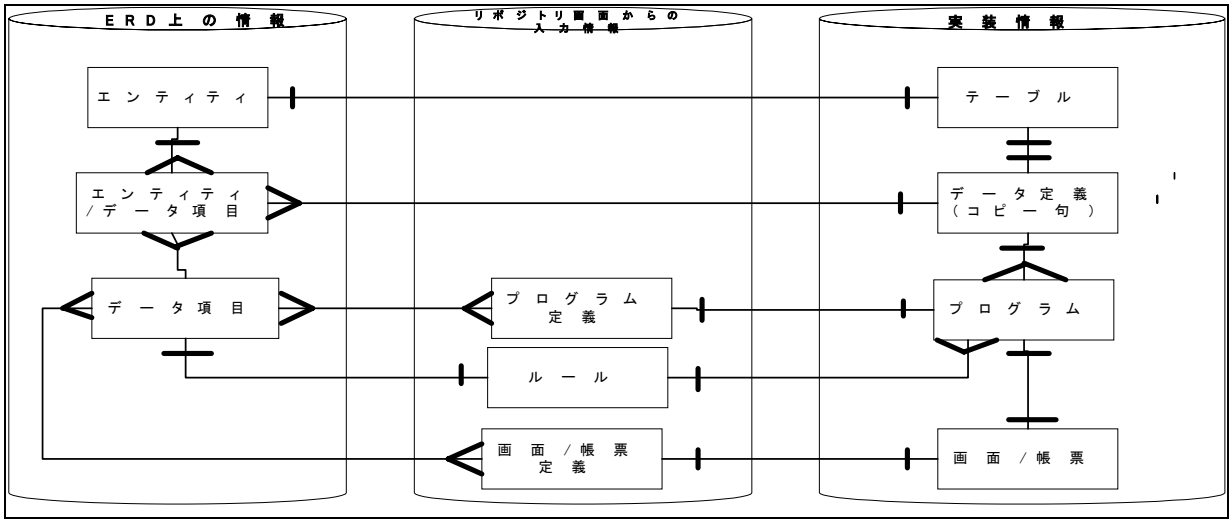


図３　リポジトリの基本データ構造

４．３．２　システム構成

リポジトリシステムは①ローダ，②コミュニケータ，③クリエータという３つのコンポーネントから構成されている（**図４**）．すべて Visual Basic（以下 VB という）で作成されている．リポジトリデータベース本体は ORACLE であるが，ドキュメントとモジュール管理のデータベースには履歴を管理するため Visual SourceSafe（以下 VSS という）を使用している．

① ローダはデータ管理者（Data Administrator，以下 DA という）のクライアントにのみインストールされる．専用エディタで作成した ERD の情報を，リポジトリデータベースにロードする．

② コミュニケータはリポジトリの本体機能であり，全開発者のクライアント(現約 300)にインストールされている．主にルール，プログラム仕様などの情報の入力を行うとともに，モジュール管理 DB，ドキュメント管理 DB と連携したリポジトリ情報を提供する．

③ クリエータはデータベース管理者（DataBase Administrator，以下 DBA という）のクライアントにのみインストールされる．リポジトリに登録された情報から実装情報を生成し，業務データベースへ実装連携する．

またこれ以外に，照会機能のみであるが WEB サーバを一台たてて，社内の他部署（実際には主に親会社のユーザ部署）への情報公開の手段としている．

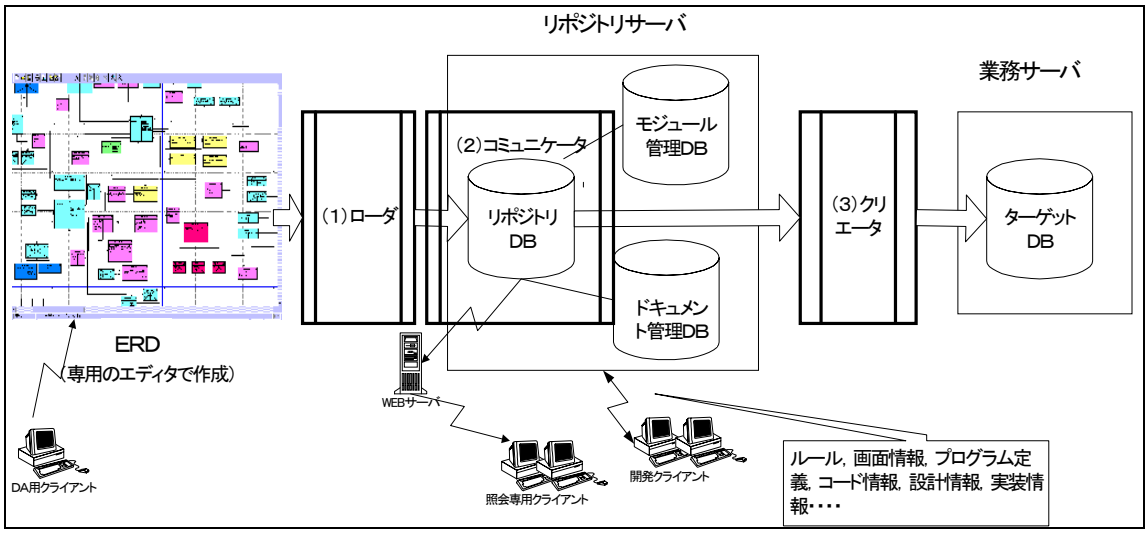


図４　システム構成

４．３．２　主な機能

以下に現リポジトリでの主な機能とその特徴を示す.

(1) ローディング機能（ローダ）

ERD で表現されているエンティティ，項目，リレーションシップ，カーディナリティの情報を，既存のリポジトリ情報との整合性をチェックしながら反映する.

入力となる ERD は表記上のルールが規定されている．またこの情報が実装情報に直接影響することから，独自の規約・基準のチェック機能も行う．なお既存リポジトリデータとの差分のみを反映する仕組みとなっている.

(2) ERD 情報管理機能（コミュニケータ）

リレーションシップ，サブセット（サブタイプ）など，ERD から取り込む図形情報を表示するとともに，ルール，定義などの入出力を行う．本来は図形情報であるため，画面上もできるだけビジュアルに表現し，本来のダイアログを再現できるように気を配った.

(3) コード照会機能（コミュニケータ）

各システムで使用しているすべてのコード情報を管理している．単にコード値の意味を表示するのではなく，ERD との関連，プログラムとの関連をとっているため，コードの変更による波及分析が可能である．なおメタメタデータでは全社のシステム（基幹系，情報系など，現在４システムある）のコードを管理できるようにした.

(4) インターフェース定義機能（コミュニケータ）

この画面では，システムを鳥瞰できるよう，アイコン種類やツリー形式のインターフェースなどを工夫している（**図５**）．画面レイアウトとリンクされているため，ユーザへのレビュにも使用できる．なお画面はドキュメント DB とのリンクで修正履歴もとれるようになっている．

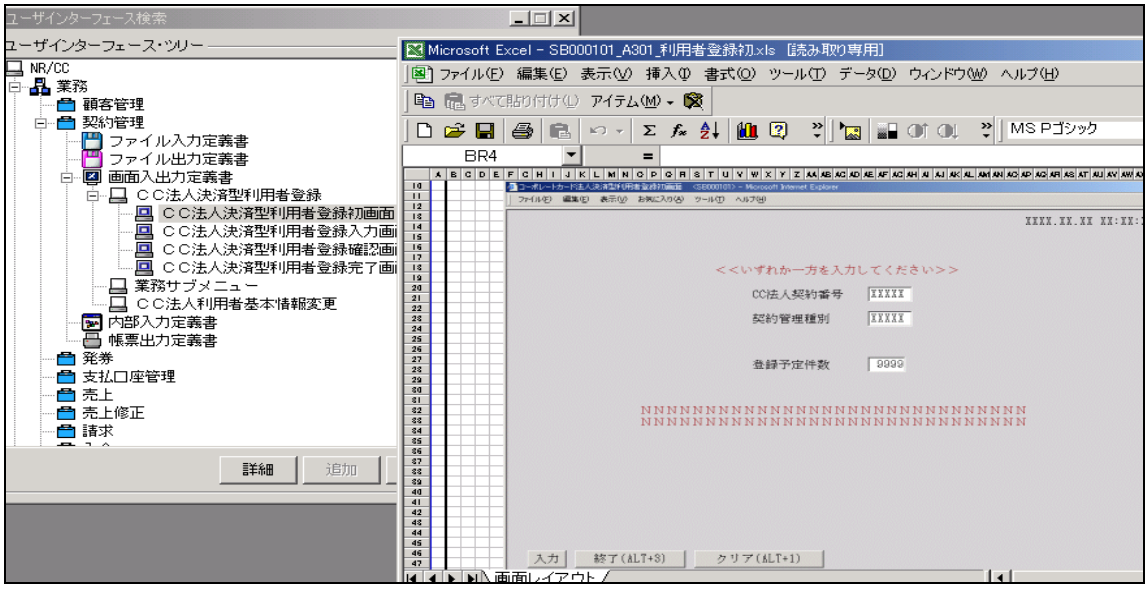


図５　インターフェース定義画面

(5) プログラム仕様，プログラム使用手引き照会機能（コミュニケータ）

元々はプログラム仕様書情報を格納する画面であったが，開発担当者の意見をとりいれ，プログラム使用手引書としても使えるインターフェースとした（**図６**）

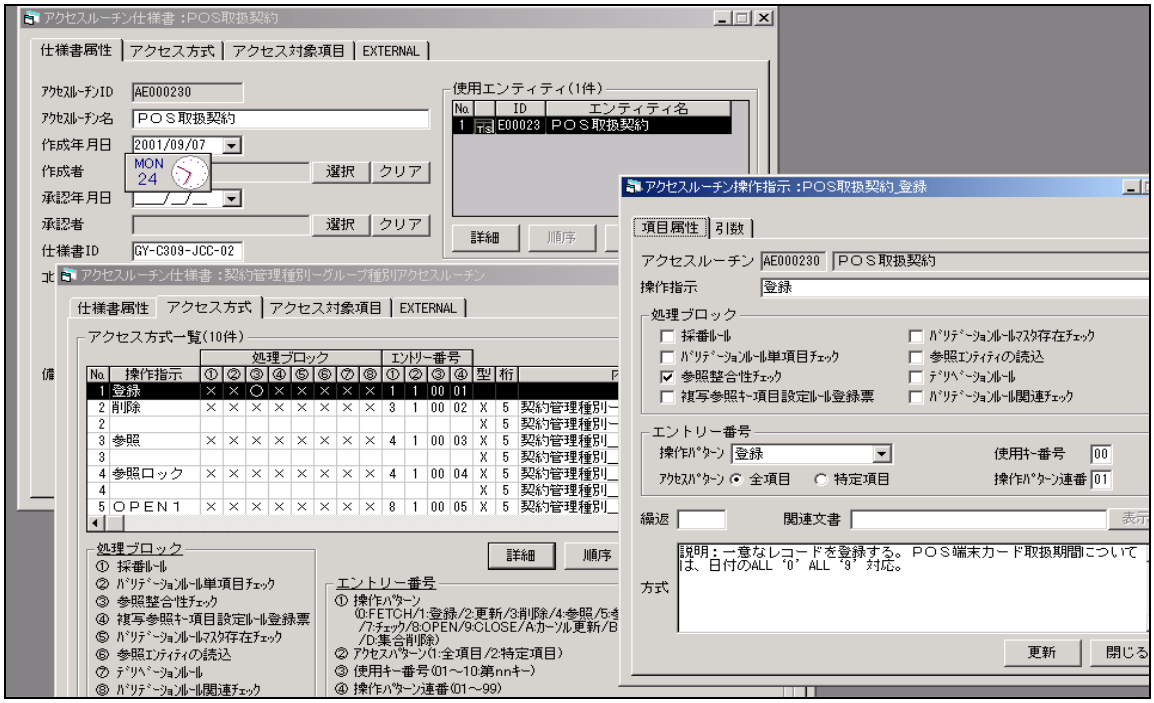


図６　プログラム仕様，使用手引き画面

(6) コピー句照会，生成機能

エンティティの情報を基にコピー句を自動生成する機能．「この機能を使わないと開発できない」という制限で，設計情報と実装情報の完全一致を図った．

(7) 障害管理機能

プロジェクト管理に必須となる障害を管理する機能．モジュール情報と連携する．今後は進捗管理，品質管理も含める予定である．

(8) 要員管理機能

要員の所属会社，使用端末のアドレス，インベントリなどを管理する．ライセンス数の管理などはプロジェクト管理上必須と考えて追加した．

(9) データベース定義・実装機能

テーブル単位に件数を入力すると予想容量を計算するなどで，手修正が一切不要な DDL を生成する．またテーブルスペースの割り当て，実行などの実装機能を持つ．DBA のみ使用が可能とし，設計との乖離を防止している．

４．４　工夫した点と効果

実際のシステム開発のニーズにあわせてリポジトリを構築していく過程で出されたアイディアや，論理と実態の整合性を維持する工夫，及びその効果を述べる．

(1) 設計情報と実装情報を一致させる工夫

リリース時には一致している設計情報と実態が，メンテナンスを重ねるにつれて乖離してしまうということは，当社現行システムでも，他の多くのシステム開発においてもよくある問題である．こうなるとプログラム仕様書はもちろんデータベースの設計書までも更新されずに陳腐化し，最後には所在さえわからなくなる．

そこで当リポジトリでは業務 ERD と実装テーブルを完全に一致させるため，①“論理構造と物理構造を一つの ERD で表現し唯一のもの”としている．また変更時は，②“ERD からリポジトリ経由で自動生成される DDL でしか変更できない仕組み”とし，完全に一致させている．これにより開発者は実テーブルを確認する必要なくリポジトリ情報のみで開発することが可能であり，また保守工程以降においても設計情報が維持される効果は大きい．

(2) コード管理に関する工夫

コードは，そのシステムにおいて非常に重要なファクターである．特に今回の手法ではコードの存在が ERD 設計の前提・基本となっている．

しかしこのコードを教条的に扱うと，とても現実的な ERD やデータ設計ができない．もちろん一枚の図面に全て表現することも物理的に困難である．

そこで，コードと呼ばれていても単に分類を意味するものについては，①“一括して同一テーブルで管理し ERD 上ではエンティティを持たない”こととした．ただし ERD の捉え方を重視し，②“リポジトリ上ではそれがエンティティとして存在するかのように仮想的に表示”させている．またこの仮想エンティティへのアクセスは，リポジトリ上のテーブルとレプリカをとる③“特別なコードテーブルとアクセスルーチンを作成”することで，プログラムレベルでのコードアクセスを容易にしている．

(3) メタメタデータの管理

前述のとおり，業務開発に合わせながらリポジトリの管理範囲を徐々に拡大し，機能を追加，変更していく手法をとっているため，リポジトリ自体の変更は柔軟かつ迅速に行えなくてはならない．また各業務システム単位のリポジトリの管理，全社共通情報の管理があるため，リポジトリのリポジトリであるメタメタデータを管理することとした（**図７**）．このメタメタデータにより，開発・本番のリポジトリも含め柔軟で，効率的な機能追加ができている．

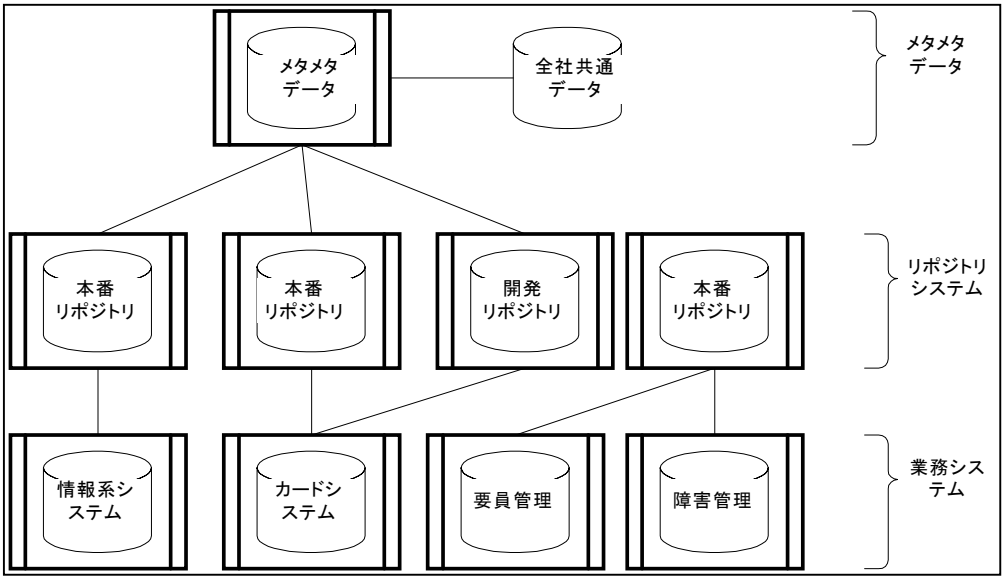


図７　メタメタデータ概念図

(4) 履歴の扱い

ERD の弱点の一つに“履歴”がある．ERD で履歴を表現するのは煩雑であり，誤解を生じやすい．

そこで図面上の ERD では履歴の概念を排した純粋な ERD を描かせ，履歴の情報についてはリポジトリで管理することとした．具体的には，リポジトリで履歴を持つとしたエンティティについては，テーブルを生成する際に履歴項目（一律！履歴区分”，“登録日付”，“更新日付”）を付加した上で，二つのテーブル（最新テーブルと履歴テーブル）を生成するしくみとした．この結果，履歴の表現が容易になり，共通認識のもとで設計することが可能となった．

(5) リポジトリ構造の公開

リポジトリのデータ構造も，業務が作成する ERD と共通の作成基準に則って作成し，メタメタデータとともに公開した．当初は業務開発者への，DOA 開発手順の説明を補完するものであったが，①業務開発グループが自発的にリポジトリデータを利用したプログラム自動生成ツールを作成，②制御グループから制御情報のリポジトリ管理の提案がある，などリポジトリ自体への関心の深まりが開発者のニーズを引き出し，結果的にリポジトリシステムのブラッシュアップに寄与することとなった．

５．評価

最新のリポジトリシステムを，“大規模カードシステム構築”への適用状況を例に評価する．

(1) 情報の一元化

システム内でのデータは，一時ファイル・性能ファイルを除き，完全に重複を排している．今回はこれを日本語名で実現したのは，ERD をデータモデルだけでなくビジネスモデルとして位置付け，アイデンティファイアを主語としたコンテキスト（文脈）を重視したためである．これにより，例えば ERD 上のアトリビュート名はコピー句の項目名と実装されるテーブルのカラム名と一致することとなり，設計情報～実装情報まで，項目名でも一環したシステムを具現している．定量的効果としては，波及分析の容易性による調査工数の削減がいえる．

(2) 情報の共有化

開発要員は，リポジトリを通してシステム開発に必要な情報を，また管理者は同様にプロジェクト管理に必要な情報を，整合性がとられかつ最新の状態で入手することができる．三百人以上のプロジェクトにとって， ERD・設計情報はもちろん，障害の情報まで共有できるメリットは大きい．また今回は一部サテライト開発が行われたが，先方へリポジトリをインストールすることで容易に情報共有することができた．

(3) メタメタデータの管理

メタメタデータを管理したことで，リポジトリの変更・機能追加を，業務システムの構築ペースに合わせ実施できた．また全社的な視野での管理は，他のシステムの情報（主にコード）へも波及し，全社的なコード統合の契機となっている．

(4) 自動化機能

テーブル，インデックスなどの DDL(Data Definition Language)の自動生成，容量自動計算，コピー句などの定義体自動生成によって，手作業ベースで行う工数を大幅に削減できた．これに加え人的な誤りによる手戻りもなくなったことも評価される．また，一括自動生成は一括変更の取り込みを容易にし，実際に何度も共通項目に変更があり有効利用された．

６．今後の展開

リポジトリシステムはこれまでいくつかの業務プロジェクトを通して構築され，基本コンセプト部分と，データ構造の部分では，ほぼ安定してきたと考えている．当面は，初期段階に構築した工数管理のシステムの取り込みや，予算管理への拡張を考えている．

また今後の展開が予定される次期システムへの適用では，プロジェクト管理に関する情報を主に拡大する予定である．現在プロジェクト管理手法は，個々のプロジェクトでは実践されているが，いわゆるナレッジエンジニアのレベルで管理されていないため，個人レベルでの情報流通としかなっていない．これらを整理・統合した上で，テスト管理，品質管理への拡大も考慮し本格的な IRM へと展開していきたい．

７．おわりに

コンピュータの父フォン・ノイマンは「世界は，“０”と“１”の信号ですべてを写しとることができる」と言ったそうだが，「世界は“エンティティ”と“リレーションシップ”ですべて写しとることができる」と言えるかもしれない．これは冗談じみた話であるが，もしエンティティとリレーションシップで，システム開発に関わる情報だけでなく，企業が持つ資産，及び活動の情報をすべて表現できたら，それを当リポジトリシステムで一元管理することはそんなに不可能な話ではないだろう．元々はソフトウェア機能の開発に目的が限定されていたリポジトリではあるが，システム開発における成否の要因が全体的なプロジェクト管理に関わる要因へとシフトしてきたことに伴って，その範囲を広げてきている．またそれを可能とした理由に，ERD がビジネスモデルとして確立，認識されてきたことも大きい．

リポジトリシステムのような，システム開発支援系のシステムは，業務システムを構築するケースとは違い，直接企業の収益に寄与するシステムではない（少なくともそう見られる）ため，周りの理解を得ながら構築・推進していくのはなかなか困難である．今回の「スプレッド開発」は苦肉の策であった．ただしこれをうまくやるためには，今回本稿ではその内容にあまり触れなかったが，リポジトリシステムと業務システムが共通認識に立てる，確立された ERD 作成手法が必須となる．

さて今回は，実際にクリティカルな業務プロジェクト開発と並行してリポジトリシステムを構築できたことは，リポジトリ開発にとって大きなメリットであった．一方，業務開発側には当初戸惑いがあったかと思われる．しかし，工程が進むにつれリポジトリの意義やメタデータの理解が深まり，自発的にツールを開発して頂くなどの協力が得られた．最後にこの場を借り，お礼を申し上げたい．

参考文献

［１］佐藤　正美：論理データベース論考，ソフト・リサーチ・センター，2000年3月25日