システム開発の基盤構築における管理ポイント

Project Management Points of System Development

磯　部　　寛

要　約　オープンシステムのプロジェクト管理においては，新技術に対応できるプロジェクト体制，コミュニケーションの問題，新技術要件選定およびその後の利用技術の適用問題の比重が大きいと言える．本稿では，「アプリケーション開発を円滑に進めるための開発環境と工夫」をテーマにしてユーコープ事業連合での事例をあげながら，「オープン系＆マルチベンダ開発での工程間の独立した開発テスト環境の提供」，「アプリケーション開発チームと基盤構築チームのギャップを埋めるための開発管理，運用管理，構成管理などの各種標準化」，「プラットフォーム選定や性能評価の問題」，「SIを介在させない複数ベンダ参加組織のプロジェクト運営」について述べる．

Abstract　As for the project management of the open systems development, major important points exist in the project organization for the adoption and use of new technology, the communication problems, the adoption of new software and hardware products, and further application of adopted new technology.

This paper takes "the development environment and plan for smoothly promoting the applications development" as the underlying theme and discusses details of following four subjects, introducing a case study of some cooperative society:

1) Provision of the development test environment independent of separate development phases and/or processes in the open systems development performed by multiple vendors.
2) Standardization of the development management, operation management and configuration management in order to fill a gap between the application development and the infrastructure development teams.
3) Problems of platform adoption and its performance evaluation.
4) Project control of an organization composed of consisting multiple vendors without any system integrator.

1. はじめに

オープン系UNIXサーバのスケーラビリティの向上で，業務アプリケーションの実行スピードは，大型汎用機と比較して遜色ないどころか上回る時代になっている．また，オープン系システム開発でも1000人月を越える規模のシステム開発も珍しくなくなっている．

プロジェクト・マネジメントの観点では，レガシーシステム（汎用機）のシステム開発でもオープン系のシステム開発でも，管理項目はほぼ同一であるが，レガシーシステム開発と違う点は，基盤構築・運用保守のためのハードウェア/ソフトウェア選定にかなり注力が必要となることである．

その理由は，汎用機と比較してハードウェア/ソフトウェアのライフサイクルが短く，次から次へと新製品の評価及び導入をこなしていかなければならないからである．

またオープン系開発では自社製品を含まないマルチベンダでのシステム開発も視野に入れる必要がある．

プロジェクトの失敗の原因として，

1) 要求定義のあいまいさ・・・・機能実現のための範囲のもれ
2) プロジェクト体制の問題・・・リソースの確保，当事者能力の問題
3) コミュニケーションの問題・・決定までの時間，決定者/責任者の不在
4) 見積りの甘さの問題・・・・・・見えない範囲の扱い/見積り範囲の合意
5) 新技術要件への対応の悪さ・・調査・スキル保持要員の手配，新技術の利用技術の不手際
6) 最終結果としての製品の欠陥，品質の悪さ

などがあげられる．

これらの要因のうち，オープンシステムの場合は 2）の新技術に対応できるプロジェクト体制，3）コミュニケーションの問題，特にオープン系は開発者が若く大規模開発に慣れていない問題，5）の新技術要件選定とその後の利用技術の適用問題の比重が大きいといえる．

本稿では，「アプリケーション開発を円滑に進めるための開発環境と工夫」をテーマにしてユーコープ事業連合での事例をあげながら，プロジェクト進行上の管理ポイントとして

- オープン系かつマルチベンダ開発を円滑に進めるための工程間での独立した開発テスト環境の提供
- アプリケーション開発チームと基盤構築チームのギャップを埋めるための開発管理/運用管理/構成管理などの各種標準化
- プラットフォーム選定や性能評価などの問題
- SI 不在での複数ベンダ参加組織のプロジェクト運営

などについて，オープンシステムでの開発管理の一部なりとも述べていきたい．

2. 構築システム紹介

2.1 ユーザ紹介

生協　ユーコープ事業連合（加盟単一生協＝6 組合，関連会社＝7 社）

組合員数＝145 万人(主要 3 生協：コープかながわ, コープしずおか, 市民生協山梨)

全供給高＝1,617 億円　共同購入供給高　主要 3 生協合計＝738 億円（1999 年 3 月）

2.2 構築システム業務

対象業務は，表 1 のように組合員管理，未収金管理，共同購入，共通マスタの 4 システムである．超大型のホスト機で行っていた共同購入というミッションクリティカルな基幹業務をオープンシステムにダウンサイジングしたものである．

【共同購入業務の概要】

業務の主要部分は，大量トランザクションでのバッチ処理である．日々 60 万件のデータ（組合員別商品別）が OCR 注文書にて配送センタに回収され，集配信サーバ経由で本部サーバに届く．毎日午前 11 時ごろに受注の締めを行う．これは翌週及び翌々週の供給分のものである．受注締めの後，午後 2 時ごろを目指してホスト経由で

表 1　開発システム業務

対象業務	本部系/拠点系	開発分担			システムの形態		
		ソフトハウスJ社	メーカ系ベンダH社	日本ユニシス	リアル系	バッチ系	システム間連携
共同購入	本部サーバ系	○			△	○	△
	物流加工センタ系		○		○	○	△
	配送センタ系		○		○	○	△
未収金	本部サーバ系	○			△	○	△
組合員	本部サーバ系	○			○	△	△
共通マスタ	本部サーバ系			○	○	○	○
基盤＆運用システム	本部系＆地区クライアント系			○	○	○	○

システム形態の○は主，△は従

商品供給ベンダに対して，翌週納品の組合員に対する供給品の発注データを送り，物流センタ，加工センタに対し翌日分の物流指示データを送る．2〜3 時間で，60 万件（年末ピーク時 100 万件）のバッチ処理を行っている．

【未収金業務の概要】

業務の主要部分は，やはり大量トランザクションのバッチ処理である．

共同購入で供給した商品の個々の組合員に対する未収金と，旅行/保険/車検/カタログ販売などでの各組合員の利用に対しての未収金を一緒にして，請求データを作成し，銀行口座や郵便局から引き落とすためのデータを作成する．業務タイミングは，週単位に行う．

平均的に，60 万件程度のバッチ処理で，処理時間は 4 時間以内に終了することが性能要件である．

【組合員業務の概要】

基本は，組合員や班の登録や改廃のオンライン・メンテナンス業務だが，半期に一度，利用高に応じた出資金の割戻計算のための大量バッチ処理が発生する．

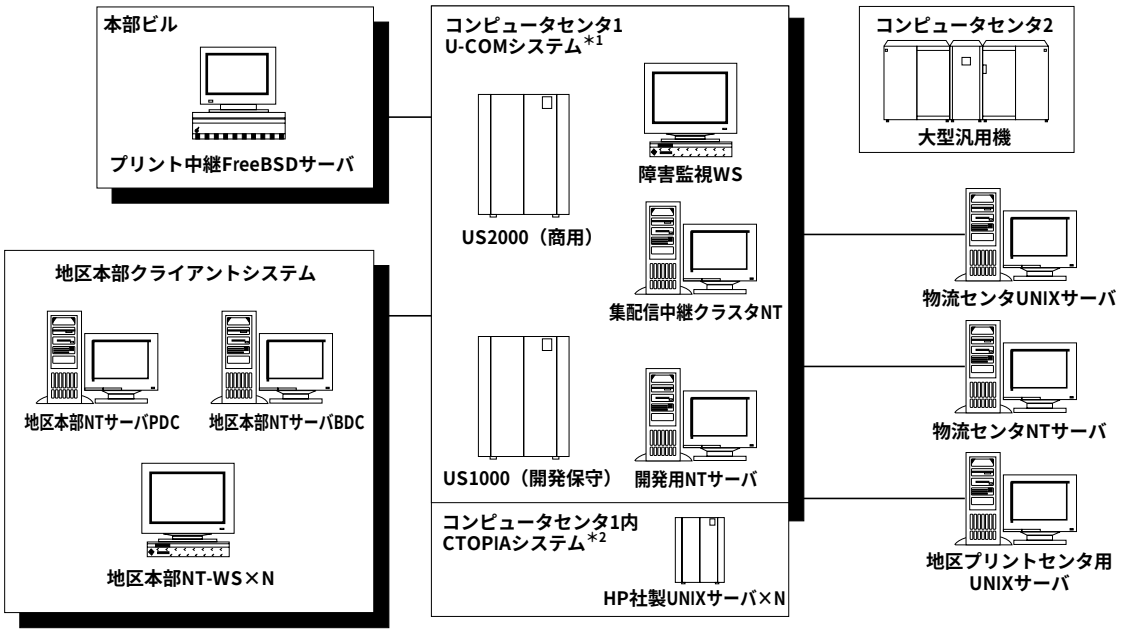
2.3　構築システム構成

本稼働時点で関連するノードのうち当社が担当した部分は図 1 の影の付いた部分のノードである．

1）　組合員・未収金・共同購入用 US 2000 サーバ
2）　集配信中継用及びプログラム配布用 NT サーバ（Microsoft クラスタ構成）
3）　障害監視 NT ワークステーション
4）　プリント中継用 FreeBSD サーバ
5）　事業連合本部・地区本部 NT サーバ＋NT クライアント
6）　開発保守用 US 1000 サーバ

2.4　クライアントサーバシステム（CSS）モデル

構築した CSS モデルは，安定性や運用性に対して，プログラム配布，バックアップサーバ，プリンタ共有などを考慮した．



図中の U-COM*1，CTOPIA*2 は文末参照

図 1　全体ノード

1）　クライアントサーバ方式：ネットワーク OS は WindowsNT 4.0，プロトコルは TCP/IP である．Unix サーバ，NT サーバと NT ワークステーションの三つを使用したクライアントサーバシステムである．

2）　プログラム共有：各ワークステーションにプログラムは置かず NT サーバで一元管理した．

3）　プリンタ共有：プリンタキューを NT サーバで管理し複数のクライアントから使用可とした．

4）　NT サーバは障害監視対象：24 時間電源オンが前提．障害に備えプライマリドメインとバックアップドメインを設置した．

5）　既存 CTOPIA システムとの共存：同一 LAN 上に既存システムの PC と共存可能とした．

3.　開発推進体制と進捗管理

3.1　PCT 会議

企画段階では，まだ決まらないものが多い．しかし，それを解決していくのもプロジェクトマネジャーの仕事である．まず，汎用機システムをダウンサイジングするという目標設定はされていたが以下の問題（リスク）があった．

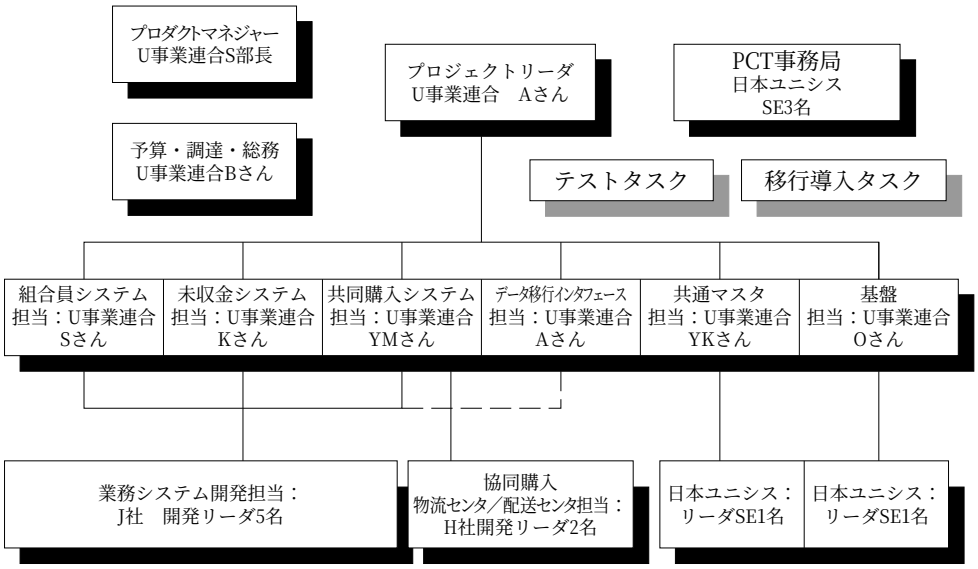
・推進組織とリーダが不明確
・仕事の進め方と複数の参加ベンダの役割分担が不明確
・ハードウェアが決まらない
・ソフトウェアが決まらない
・業務範囲が不明確

・必要な要員スキルが不明確
・稼働中の既存 HP（ヒューレッド・パッカード）社製 UNIX システムとの共存の可否

問題解決の推進体制として，企画段階が終了した 97 年 10 月ごろから，参加ベンダとユーザの主要メンバによる PCT 会議（プロジェクトコントロールチーム会議）が発足した．目的はリーダーシップが必要なアイテム，合議制で調整できるアイテムなどを切り分けることである．

全体で 1000 人月を越える規模のプロジェクトである．SI ベンダが存在しないためユーザ自身が図 2 のようにリーダになった．当社が担当したのは，プロジェクト管理支援，基盤構築，共通マスタ開発の三つである．他の参加ベンダは，業務 AP（アプリケーション・プログラム）全般を開発する J 社と共同購入配送センタシステムとセットセンタ（生鮮品加工センタ）システムを担当する H 社である．各々ベンダは対等の立場で参加している．

この 4 社が集まってプロジェクトを開始したが，まず最初の調整事項はシステム開発管理のための物差しの共通化と各社個別に管理するものの分離である．つまり各会社の文化が違うため，言葉の定義や仕事の進め方の意思合わせから始めている．



図中の U 事業連合は，ユーコープ事業連合の略

図 2　PCT 会議体

当社は SI ベンダの立場ではなかったが，プロジェクト推進会議（PCT）の事務局として，各参加ベンダの主要メンバ及びユーザ側担当者とともに品質管理や進捗管理を行った．また基盤担当ベンダとしてプラットフォーム構築，共通マスタ担当としてマスタメンテナンスシステムの構築を担当した．

PCT の主要な責務は，問題点の抽出と警告及びチーム間調整機能である．プロジェクトリーダと会議前には会議の進め方の調整をしたり，利害の異なる参加ベンダ間での調整を行うため会議以外のところでの情報収集にも努めた．

3.2 開発進捗管理

進捗管理は，毎週1回定例のPCT会議（プロジェクトコントロール会議）で各チームが実績と予定を報告して行った．また月に1回全体スケジュールに線を引いて各ベンダからの進捗をまとめて報告した．

進捗管理上の工程定義は重要である．なぜなら同じような言葉で表現しても，発言している会社によって，工程の意味や成果物などが微妙に食い違う．マルチベンダ開発体制のため，最初に定義しておかないと混乱してしまう．開発工程定義は，当社標準版と違い，J社及びH社との調整で表2のように決まった．これらのうち，要求分析とシステム分析工程以降については工程前，工程後に全体レビュを行って品質確保をしている．

各工程の意味付けは，表2のとおりで，システムテストと運用テストを各々2段階に分割している．これはテスト工程を細分化して段階的に仕事を進めることにより，成果物ベースでの進捗の遅れや進みを表面化させるためである．

表 2　開発工程と開発期間：1次システム 1997年4月〜1999年3月末

ユニシス標準工程	今回の工程管理	略称	成果物	作成責任
システム企画	システム企画 97年4〜9月	SP	システム開発計画書	ユーコープ事業連合
要件定義	要求分析 97年9月〜12月	SA	要求機能分析書(以下を含む) 機能+ビジネスルール+概念データ+制約条件を表現したもの	J社,H社
論理設計	基本設計 98年1月〜3月	UI	設計書((以下を含む)I0関連図,コード設計書,ファイル設計書,論理DB構造,画面,帳票,パンチ設計,各種一覧,共通仕様(各種標準化),概要プロセスフロー	J社,H社,ユニシス
物理設計	詳細設計 98年3月〜6月	SS	設計書(詳細DB+詳細プロセスフロー)	J社,H社,ユニシス
プログラミング	詳細設計 98年4月〜7月	PS	単体仕様書	J社,H社,ユニシス
	プログラム製造 98年4月〜8月	PG	プログラムソース+オブジェクト+シェル	J社,H社,ユニシス
		PT	(単体テスト結果報告書)	J社,H社,ユニシス
統合テスト	結合テスト 98年6月〜8月	IT	結合テスト計画/結果・報告書	J社,H社,ユニシス
	システムテスト1(システム内)98年9月	ST1	ジョブネット&ジョブ登録資料, 結合テスト計画/結果(準備データ,開発機)	J社,H社,ユニシス
	システムテスト2(全体連動)98年10月	ST2	システムテスト計画書（準備データ,開発機）	ユーコープ事業連合
運用テスト	運用テスト1（移行ブリッジ含む）98年11月〜12月	OT1	移行計画書，運用テスト1計画書（商用機）	ユーコープ事業連合
	運用テスト2（エンドユーザ検証）99年1月	OT2	運用テスト2計画書（商用機）	ユーコープ事業連合
保守，評価	保守	ME	アウトソーシング引継資料	ユーコープ事業連合

4. 開発環境の提供

多数のメンバが参加する開発を円滑に進めるためには，その多数のメンバがなるべく分離独立した環境のほうが，お互いの影響を少なくできる．そのためには工程別の

開発環境を初めから想定しておく必要がある．分離する環境として想定するものは大きく分けてサーバ環境，ネットワーク環境，クライアント環境の三つである．これらはコストがかかるものなので，開発に入る前から考慮しておくべきと考える．

本来，テスト環境と本稼働の商用環境は同一のものを用意する必要があることを強調しておきたい．なぜならサードパーティのハードウェア/ソフトウェアを含むシステムの性能検証や信頼性検証が本稼働前に必須と考えるからである．

中小規模のプロジェクトではコスト問題のため最終稼働用の商用サーバを用いて開発を行い，本稼働時点で同一サーバ内で本稼働環境に切替えることがある．このプロジェクトでは最初から開発環境と商用環境のサーバを設置し並行的に環境を整え，本稼働後も二つの環境を維持するようにした．

4.1 サーバ環境

開発テスト用のプラットフォームとして，UNIX サーバ 2 台（内 1 台は本番へ移行）と NT サーバ 3 台を準備し，その中に工程別に使用する資源が全く異なる環境を構築した．業務 AP 開発チームは NT もしくは UNIX のログインユーザを変えることによって使用する環境を使い分けるように工夫した．

工程別の環境を構築するにあたっては表 3 のように以下の要件を考慮する必要がある．

・使用するメンバとその人数
・使用データボリューム
・使用期間
・使用場所（ネットワーク考慮）
・本番業務との接続可否とセキュリティ

表 3　開発工程別の使用条件

環境	環境	使用データサイズ比	プログラムソース有無	スタブ提供	使用メンバと人数	使用期間	使用場所 その他条件
開発単体テスト	開発保守環境	数 100 件程度 各サブシステム毎に必要	○	○	リーダ 設計者・プログラマ 60 名程度	本番稼働後も継続的に使用	ベンダー事務所 ユーザの指定開発場所 本番環境の接続禁止
結合テスト システムテスト 1, システムテスト 2		0.1	○	○	リーダ 設計者 15 名程度	本番稼働後も継続的に使用	ユーザの指定開発場所 他システムの開発環境との接続が必要. 本番環境との接続禁止
運用テスト 1	商用環境	1.0 本番想定量を 1.0 とする.	×	×	リーダ 設計者 ユーザ 20 名程度	本番稼働まで	ユーザの指定開発場所（情報システムオフィス） 他システムの本番環境との接続ができること
運用テスト 2		0.05	×	×	ユーザ エンドユーザ 10 名程度	本番稼働後も継続的に使用	エンドユーザの操作教育場所 画面操作教育ができるだけの環境でよい

これらの環境では，業務 AP の変更書き換えや再コンパイルすることなくテストできるように考慮しておくことが必須である．これを実現するためにはどの情報を環境変数や初期化ファイル（INI ファイル）に保持するかという設計を十分行う必要がある．

4.2 擬似部品（スタブ*3）の提供

開発テスト工程毎にどこのプラットフォームのどの環境と接続するかが決定される．業務 AP 開発チームとしてはこれらの相手先を意識せずにテストを実施する必要性がある．また，誤ってテストデータを本番環境へ送ってしまうような危険性を排除する必要性もある．

このため，基盤構築チーム側では開発工程ごとに同一インタフェース仕様であるが実際の振る舞いについて異なる部品を工程毎に提供することで業務開発 AP チームの作業負荷を軽減した．

また，プロジェクト全体としてもプラットフォーム間のインタフェースについては基盤構築チームで集中管理することで顧客及び他ベンダとの調整窓口も一元化された．

さらに，接続のための認証情報についても部品に隠蔽することで機密性も高め，データ容量資源の制約がある場合は部品内でチェックすることで相手プラットフォームの資源を必要以上に使用することも防止した．

4.3 ネットワーク環境

ネットワーク環境やクライアント PC 環境も，テスト工程別に厳格に区分けして環境設定した．その理由は，テストの各工程環境と本稼働環境が混在すると何らかの事故が発生した場合，業務を止めてしまう可能性があるからである．管理ポイントは，以下のとおりである．

・クライアント PC の台数と置き場所
・ローカルネットワークセグメントとハブ，ルータ
・接続先ノードの設置場所
・接続先ネットワークセグメント，ハブ，ルータ
・回線スピード，回線使用状況
・セキュリティ

4.4 基盤構築チーム組織

要員体制としてプロジェクトマネジャーは基盤関連では以下の 2 チームを同時に立上げ，運用保守設計をシステム開発と同時にスタートさせるのが理想的と考える．

その理由として，とかく目の前の開発作業に注力し，稼働後の運用システム設計は開発工程の中ごろから着手というように遅れがちだからである．運用設計が終了していないとアプリケーション設計も終了せず中途半端にテストや本稼働を迎えることになる．

1） 基盤チーム：

プラットホームに依存する各種ハードウェア/ソフトウェア設定作業，基本部品作りと提供，及びデータベース設定とチューニング，ディスクや保存媒体のリソース管理などをするチーム

2） 運用適用チーム：

運用に関する全てのミドルウェアの実証実験と適用，関連する部品作り，業務AP チームへの適用支援（ジョブ組み込み，バックアップシステム適用など）を行うチーム

5. 業務 AP 開発チームと基盤構築チームとのギャップの埋め方

5.1 標　準　化

オープンシステムのシステム開発で，業務 AP 開発チームが設計やプログラム開発の前に明らかにすべき各種標準化の項目として今回のプロジェクトの例をあげる．

標準化作業は多岐に渡り，かなりの検討期間が必要になるので，作成のためには参加ベンダの共同作業にならざるを得ない．標準化の共同作業では各参加ベンダの利害問題も発生し各種調整が必要になった．それは，各ベンダが先行着手したケースでは標準化に合わせる手戻りを避けたいという思惑が発生したためである．

このため，今回のプロジェクトでは 3 ヶ月から 5 ヶ月の期間をかけている．なかでも方式論と呼ぶものは詳細設計の指針とされるものなのでそれなりの適材適所の技術力が要求される．

また運用管理基準や構成管理基準などは，システム設計標準（基準）の方向性や稼働後運用の設計に影響を与えるため早めに着手すべきである．

以下，各分野別に重点管理項目を明示して参考とする．

1） 共　　通（表 4）

表 4　共通標準（各種命名規則）

	作成した標準書	管理ポイント
1	命名規約	業務用語関連の命名付与基準 ファイルの命名付与基準 オラクル DB ファイル関連の命名付与基準 画面関連の命名付与基準 帳票関連の命名付与基準 メッセージ関連の命名付与基準 マクロの命名付与基準 ホスト名の命名付与基準/ドメイン名の命名付与基準
2	用語辞書	業務で使用する全ての用語と対応する略称，アルファベット名
3	ユーザ ID の体系の考え方	認証に関連して，以下の利用形態に合わせて整理する． 業務システム利用ユーザ NT ドメインユーザ 業務 DB 利用ユーザ Unix ログオンユーザ FTP ユーザ その他（電子メールユーザ）

2） 方式論

表 5 に示す各種方式論は，各業務 AP 開発グループがプログラム開発着手以前に，基盤構築チームの設計が終了していないと，各業務 AP 開発グループの開発作業が停滞することになるため特に急ぐべきものである．

表 5　設計方式論

	作成した標準書	管理ポイント
1	C/S 処理方式基準書	a）クライアント側で直接 SQL 文を発行するタイプ b）サーバ側のストアドプロシジャを利用するタイプ c）サーバ側でのバッチ処理起動と処理結果をクライアント側で受け取るタイプ などの処理方式を整理する.
2	バッチ処理方式基準書	a）バッチ制御を運用管理ソフトを用いて行うか，手作りシステム（シェル等）で行うか b）処理の排他制御の仕方 c）異常終了時の取扱 d）再起動の考え方 e）プロセス間インタフェース f）ファイルアサイン法 g）帳票作成プロセスと帳票システムとのインタフェース h）プロセスの優先順位の整理　i）走行ログの扱い
3	メニューと認証処理方式	メニュー方式のランチャープログラムの定義 a）階層構造によるメニュー b）認証及びメニュー起動 c）画面の移動 d）アプリケーションの実行 e）画面制御情報による実行の制限（権限） f）同一クライアント上の複数起動制限
4	オープンバッチ処理方式	リアルタイムに結果を要求せず，一定時間後に結果をクライアントに返すための全体のしかけを整理し定義する. a）クライアント側プログラム定義 　専用メニュー＋クライアント業務 AP b）サーバ側プログラム定義 　常駐デーモン形式プログラム＋サーバ業務 AP c）ファイル 　制御用パラメタファイルと業務データ受渡ファイルの定義 d）データベース e）ストアドプロシジャ f）関連部品説明
5	印刷処理方式基準書	a）印刷サーバの有無定義 b）XEROX プリンタ, KPR, ネットワークプリンタの使い分け定義 c）書式指定ファイルの定義 d）印刷データの定義 e）再出力の定義 f）複数台プリンタの利用方法定義
6	ノード間インタフェース基準書	a）現状分析と業務要件 b）対象ノード定義 c）想定処理性能値 d）処理規定 e）ファイル形式（固定/可変，ヘダー付き無し） f）運用規定 g）処理方式 （起動トリガ，ファイル形式，コード体系，転送モード，後続処理起動の扱い） h）提供部品

3） 設計基準

プログラム設計のための設計基準としては，以下のようなものを予め用意し各業務 AP 開発リーダに周知徹底させておく．ここでは特に説明しない．

画面設計基準　帳票設計基準　ファイル設計基準書　データベース設計基準書
世代管理方式基準書　ユーザログ基準書
プログラム設計基準書（クライアント）　プログラム設計基準書（サーバ）

4） プログラム基準

プログラマのためのコーディング規約を作成したが，ここでは特に説明しない．
コーディング規約（C，COBOL，KSHELL，SQL，VB，PL/SQL）

5） 構成管理基準

構成管理は，設計・開発と比較すると地味な事務作業という印象があるが，管理されたプロジェクトにするには避けて通れない部分である．システム開発全般に及ぶ構成管理のうち，成果物に関連するものとして，プログラムの配布管理，仕様変更管理，プログラムバージョン管理などを開発の早めの段階に決めておき，

表 6　構成管理基準

	作成した標準書	管理ポイント
1	構成管理	a）構成管理の定義 b）構成管理対象物の定義 c）構成管理担当者の定義 d）開発サーバ・構成管理サーバでのディレクトリ体系 e）バージョン管理の定義 f）保管対象物の定義 g）保管申請ルール・保管手順・申請書書式
2	配布管理（論理設計）	a）配布の全体イメージ b）クライアント側配布の手順 c）Unix サーバ側配布の手順 d）申請ルール e）配布対象外の資材
3	仕様変更管理	a）仕様変更管理手順
4	サーバ側バージョン管理（RCS 利用規約）	a）管理者の定義 b）主なコマンド c）RCS 管理対象物の定義 d）RCS 記述ルール e）管理単位と格納領域 f）管理手順
5	クライアント側バージョン管理 説明書(VisualSourceSafe)	a）VSS 概要 b）運用方法 c）VSS のバージョンと EXE のバージョン
6	クライアント側 PG 配布方式（NetM/DM）	a）全体図 b）NetM/DM が取り扱うノードの範囲 c）NetM/DM のエンティティ概念 d）NetM/DM エンティティ命名規約 e）NetM/DM 操作手順 f）NetM/DM のデータベース保守 g）配布エラー時の対応ルール

業務 AP 開発チームに周知しておくことは重要である．何故ならテスト工程や導入工程でのスケジュールの切迫している段階での混乱を少しでも減らし，システム全体の品質向上に役立てるためである（表 6）．

6)　運用管理基準

まず運用管理全般の整理学として，本稼働を想定するとユーザの運用チームは様々なカテゴリでの準備調整が必要になる．ここでは調整の全カテゴリではなく開発工程の初期段階に限定して，業務 AP 開発チームの開発業務に直接影響のあるジョブ運用と障害管理について基盤チームが提示した例を示す（表 7）．

この運用管理標準の中のジョブ運用とバックアップ運用等は結合試験やシステム試験工程に影響してくるため，開発工程での初期段階での策定が必要である．プロジェクトマネジャーが気配りする項目のひとつと考える．

5.2　基盤としてのスタビリティの向上と性能評価

5.2.1　サーバ配置の方針

サーバ配置を決めるために以下の項目を検討した．

1)　サーバの役割（機能）

2)　業務アプリケーションとサーバ分割方法の関連

として，

①　AP サーバと DB サーバの分割方法

②　業務サブシステム単位でのサーバ分割方法

③　バッチ処理サーバとリアル処理サーバでの分割方法

この検討結果，システム間インタフェースなどの運用保守を最重点にして一つのサーバに業務を集中させることに決定した．ただし，運用後でも，業務単位にサーバ分割できるように設計全体を考慮した．ひとつのサーバとするため，メモリーや CPU 数も大きめに設定している．

5.2.2　ネットワーク方針

1)　徹底したネットワークセグメントの分離

テスト工程ごとの分離，既存システムとの分離，テストセグメントと本稼働セグメントの分離など，多少の運用上の使い難さよりセキュリティの確保を最優先した．これは互いに関連することで業務を止めてしまうようなことを避けるためのユーコープ事業連合の方針に従ったものである．

2)　IP アドレスの管理の徹底

ユーコープ事業連合の方針で可変 IP アドレスを使用せず，管理された固定 IP アドレスの管理方法とした．理由は可変（DHCP）方式を採用すると個人用の PC を持ち込んで接続することが可能となるため，これを防止するためである．全接続端末の管理台帳を作成し IP アドレスを管理している．

5.2.3　実現可能性の確認

開発着手の前段階で OS やハードウェア選定のための調査作業を行い，実現可能性を確認した．特にマルチベンダ方式で自社製品と他社製品が混在しているため事前調査での動作確認が非常に重要であった．システム開発のスピードの速さがオープンシステム開発の一分野で叫ばれているが，ミッションクリティカルな基幹業務システム

表 7　運用管理基準

	作成した標準書	管理ポイント
1	ジョブスケジュール設定基準書	a）日次・週次・月次・年次・随時処理の定義 b）運用日・休業日の定義 c）基準日・基準時刻の定義 d）実行ホストの定義 e）異常終了・警告終了のしきい値の定義
2	ジョブネット設計標準	a）ジョブネット設計概要 b）設計手順 c）スポット処理設計 d）システム間インタフェース設計 e）最遅監視設計 f）自動運転設計 g）ID 付与基準
3	ジョブ登録操作説明書	a）ジョブ管理ソフトウェア概要 b）前提・制約条件 c）ジョブ登録方法 d）ジョブ実行方法 e）再実行方法 f）登録ジョブ要素のバックアップ g）ファイル連携
4	運用・障害設計基準	a）ユーザログ b）再始動ポイント c）トランザクションの考え方 d）バックアップ e）FTP 異常終了対応 f）業務規制 g）ファイル/SQL ステータス h）データ交換 i）ソフトウェア配布/メンテナンス時間帯 j）オンライン/バッチのコンカレント処理 k）ロードモジュール・データセグメントサイズ l）再処理と日付
5	バックアップ運用処理方式	a）バックアップの定義 b）バックアップ対象範囲 c）バックアップ前提条件 d）バックアップ方法 e）バックアップジョブ制御 f）リストア方法 g）バックアップ媒体管理 h）バックアップ処理時間見積り i）バックアップ対象登録方法
6	障害監視方式	a）障害検知/対応の範囲 b）監視対象の定義 c）障害監視のレベル d）障害監視要員の定義 e）障害通報先の定義 f）障害監視ソフトウェアに必要な要件 g）障害別対応条件の整理

の手作り開発には，腰を据えてじっくりと機能確認しておくことになる．逆説的に言えば実証実験ができたからこそ，リスク回避ができたと言える．

性能評価のポイントとしては，以下の 3 点がある．

- ハードウェアディペンド：ディスクチャネル/RAID の性能
- ソフトウェアディペンド：主にデータベースソフトウェア oracle の性能
- 業務 AP ディペンド　：アプリケーションロジックの工夫

基盤構築担当として，指定の処理時間内にジョブを終了させるという性能要件を満足させるため，

- 選定ハードウェアの処理効率測定（ハードウェア）
- 業務チームの ER モデルレビュー（業務 AP）
- 業務チームのプロセスモデルレビュー（業務 AP）
- 性能基礎数値を業務 AP チームに提示し処理プロセスの想定処理時間の見積依頼とレビュー
- ジョブの多重度の見直しとテーブル構造の見直し（業務 AP）

などを実施した．

中でも注力したのは基盤構築チームの責務として選定候補ハードウェアの処理時間計測と評価及び選定候補ミドルウェアの動作確認である．

1)　ディスクアクセスパス数と多重処理の相関関係試験

業務要件は，大量トランザクションのバッチ処理である．業務 AP 設計では処理効率を高めるために，数百万件のトランザクションを N 分割して，N 多重で並行処理させたり，10 万件程度以下のマスタデータは，業務 AP のメモリーに常駐させるなどの工夫を行うことが予想されていた．

これらの業務案件を満足させるには，まず I/O 処理の多重処理効率を高める目的で多数のアクセスパスを持つディスク装置で，I/O 負荷を分散させる必要があった．つまり，大きな容量でひとつのアクセスパスしかないディスク装置よりも 1 ドライブあたりのディスク容量は少なくても，アクセスパスの数の多いもののほうが，多重処理に強いと考えて，実証実験を行った．

【ディスクの比較検証試験】

比較試験したディスクは，当社が代理販売している H 社のモデル D と S 社のモデル R である．

①　テスト時点のそれぞれのディスク構成の特徴（表 8）

②　試験パターン

表 8　ディスク構成

ディスク装置名	H 社のモデル D	S 社のモデル R
テスト時容量	123 GB	318 GB
RAID	Raid 5	Raid 5
CU の数	6	2
1 CU 当たりのトレイ数	1	3
インタフェース	Ultra/Wide-SCSI（40 MB/sec）	Ultra/DFW-SCSI（40 MB/sec）

COBOL と C でテストアプリケーションを作成し，5 CPU と 10 CPU でそれぞれ多重度 1，5，10 で READ/WRITE を実行する．

（条件）
- READ/WRITE は DB アクセスはなく，UFS ファイルのみである．
- 5 多重，10 多重での入力ファイルと出力ファイルのディレクトリは多重毎に分離する．
- ファイルは，レコード長＝310 バイト，レコード件数は 10 万件，50 万件，100 万件の 3 種類である．

③ 試験結果

1 多重（シングルラン）の Write と 10 多重の Read では，モデル R のほうが速いが，5 多重/10 多重の他の処理では，モデル D のほうが速い．業務パターンで一番多いと想定した 5 多重では，比率で約 60% 弱の相対処理速度となった．業務要件が 5 多重から 10 多重の COBOL シーケンシャル処理なので，モデル D のほうが業務に合っていると判断した．

2） 選定候補ミドルウェアの動作確認

従来からユーコープ事業連合では統合運用管理ソフトとして JP 1 を採用していたため，オペレーションを統一する目的で，今回の開発でも引き続き同じ JP 1 の採用となった．ただし，同じ JP 1 でも，親和性があるかどうかは別の問題であり，当初，旧来の HP 社製サーバと混在させ，HP 社製サーバ側の JP 1 マネジャーから SUN（サン・マイクロシステムズ）社製サーバをコントロールできないか，動作確認を行った．まずバージョンが違う，内部コード体系が S-JIS と EUC と違うなどの問題がある．同一製品といっても，バージョンや動くプラットフォームが違えば，細かい部分で動きが違う可能性もある．

基盤担当ベンダのプロジェクトマネジャーとしては，技術担当者に確認をしておく必要があり，テストを実施・確認した．

① 確認した環境イメージ

② 確認した内容

	使用目的	ミドルウェア名称	動作確認内容	結　果
1	システム 自動運転	JP 1/AJS, JP 1/AOM, JP 1/SES	バージョン/コード体系の違うマネジャーからのジョブ自動運転指示と SUN サーバ内の EUC コード確認	ジョブ自動運転 OK EUC ログメッセージ表示 OK
2	異常検知	HP OpenView IT/オペレーション	エージェント内の日本語異常メッセージがマネジャー上で S-JIS に変換されるか	S-JIS 表示 OK
3	バックアップ	JP 1/OmniBackII	JP 1/OmniBackII が SUN のテープ装置に対して有効か	MediaAgent（エージェント側でテープ read/write を行うソフト）が動作せず．

AJS：Automatic Job Scheduler　AOM：Automatic Operation Monitor　SES：System Event Service

③ 結果

全体の整理として，オペレータなどの運用体制も勘案し，運用管理体系は既存の HP システムとは分離して，SUN サーバシステムは独立して考えることとした．

JP1 シリーズで対応させるのは，システムの自動運転/プログラム配布/障害監視で，バックアップソフトはジュークボックス対応の当社 S-BACKUP を採用した．

5.2.4 システム障害対応の要件定義

円滑な本稼働を迎えるためには，運用管理としての障害対応方法が明確になっているべきである．障害対応要件をまとめるためには，次に示すように監視対象の定義，障害監視端末の定義，障害環視のレベル定義，対応組織手順などの定義など，ユーザと合意をとっておく必要がある．人間が判断する部分が多いこと及び関係する人間が多いことなどから開発工程の早めの段階から整理しておくべきと考える．

- 障害監視対象定義
  - ・監視対象のハードウェア→生き死にのみか/縮退を含むか
  - ・監視対象から外すハードウェア
  - ・監視対象のソフトウェアログ→特にログデータのフィルタリング内容
- 障害監視（検知）端末の定義
- 障害監視のレベル定義
- 障害対応レベルを満足させる仕組み，体制，ルール作りの定義

1） 障害監視のレベルの定義例

各ノード毎に要求される信頼性レベル，監視レベル，対応レベルを表 9 のように定義し整理した．監視レベルや対応レベルの定義次第で必要ソフトウェアやしかけが変化するので，この部分の整理は重要である．

表 9　ノード別障害監視レベルと対応レベルの整理例

NO	監視対象ノード	数量	信頼レベル	監視レベル	対応レベル	摘　要
1	SUN サーバ	1	5	A	5	
2	ディスク　モデル D	14	4	B	4	LAN カードが 14 枚
3	テープ装置	0	4	B	4	
4	センタ 1 内　同一セグメント内の通信機器	?	5	C	4	
5	集配信中継＋配布元サーバ	1	5	A	4	非常時は代替手段あり
6	K 第 1 ビル・プリント用中継 PC	1	4	C	4	
7	K 第 1 ビル商用 NT サーバ A	1	3	B	3	代替機あり
8	K 第 1 ビル商用 NT サーバ B	1	3	B	3	代替機あり
9	K 第 2 ビル 4 F 情報システム部 NT サーバ 1	1	3	B	3	
10	〃　NT サーバ 2	1	3	C	3	
11	K 第 2 ビル 5 F コープファイナンス NT サーバ 1	1	3	B	3	業務は照会のみ
12	〃　NT サーバ 2	1	3	C	3	

13	K 第 2 ビル 5 F 共済運用部 NT サーバ 1	1	3	B	3	業務は照会のみ
14	〃　　NT サーバ 2	1	3	C	3	
15	S 本部 NT サーバ 1	1	3	B	3	組合員の更新業務あり
16	〃 NT サーバ 2	1	3	C	3	
17	Y 本部 NT サーバ 1	1	3	B	3	組合員の更新業務あり
18	〃　NT サーバ 2	1	3	C	3	
19	関連会社カタログ・NT サーバ 1	1	3	B	3	業務は照会のみ
20	〃　　NT サーバ 2	1	3	C	3	

（ルータやハブは別の監視システムで監視しているため，監視対象外とした）

信頼レベル（信頼性要求レベル）　ハードウェアの MTBF（ソフトウェアや業務 AP は別）

5：1 年に 1 回程度の障害/回復は翌朝まで．最高レベル

4：1 年に 1 回程度の障害/回復は翌日の午前中まで

3：半年に 1 回程度の障害/回復は翌日の午後いっぱいまで

2：2〜3 ヶ月に 1 回程度の障害/回復は翌日いっぱいかかってよい

1：数週間に 1 回程度の障害/回復は翌日および翌々日までかかってよい

監視レベル　A：稼働/停止＋ハード障害＋ソフト障害の 3 レベル監視

B：稼働/停止＋ハード障害の 2 レベル監視

C：稼働/停止のみの 1 レベル監視

対応レベル　夜間の無人運転中に障害発見した場合の対応状況を想定する．

5：即時対応必要　（最高レベル）

4：即時対応必要だが，内容によっては翌日対応でも可能なレベル

例）1 個のディスクドライブが故障し，縮退運転中

3：内容を調査し，即部品を手配し翌朝技術員が到着するレベル

2：内容を調査し，部品の手配をし交換を日程調整できるレベル

1：内容を調査し，警告を出してユーザが待てるレベル（性能監視レベル）

2）　障害対応レベルを満足させるための仕組みとルール作りの例

①　障害監視要員の想定

監視要員，運用要員，業務担当者，DB 保守要員，リモート監視要員の存在と役割を定義した．

②　障害通報先の想定

障害が発生した後，1 次切り分けをして次に実際に対応すべき人に通報することになる．

障害内容のカテゴリ別に通報先及び通報手段を整理した．カテゴリとは

ハードウェア別/UNIX か NT

ソフトウェア別/OS かミドルソフトウェアか業務 AP か部品か

通報手段とは

P＝ポケベル，T＝電話，F＝FAX，M＝メール，W＝掲示板，P＝紙，O[*4]

＝お知らせメッセージ

③　障害監視ソフトウェアに求められる想定要件

第1段階の障害検知の表示および警告のことを意味する．

- ノード監視画面上，赤く光る．(JP 1/NetM/CM 2 の基本機能)
- ポップアップ画面で異常を通知する．(障害監視 PC に手作りプログラムで表示)
- 誰かに障害を知らせるためにポケベルを鳴らす．(AT コマンドで発呼するプログラムおよびモデムが必要となる．

④　障害発生時の連絡ルート

図3のように，監視要員→業務システム担当者→ユニシスの技術員またはシステム担当の経路，及び障害監視端末からポケベルでリモート監視センタの2経路がある．

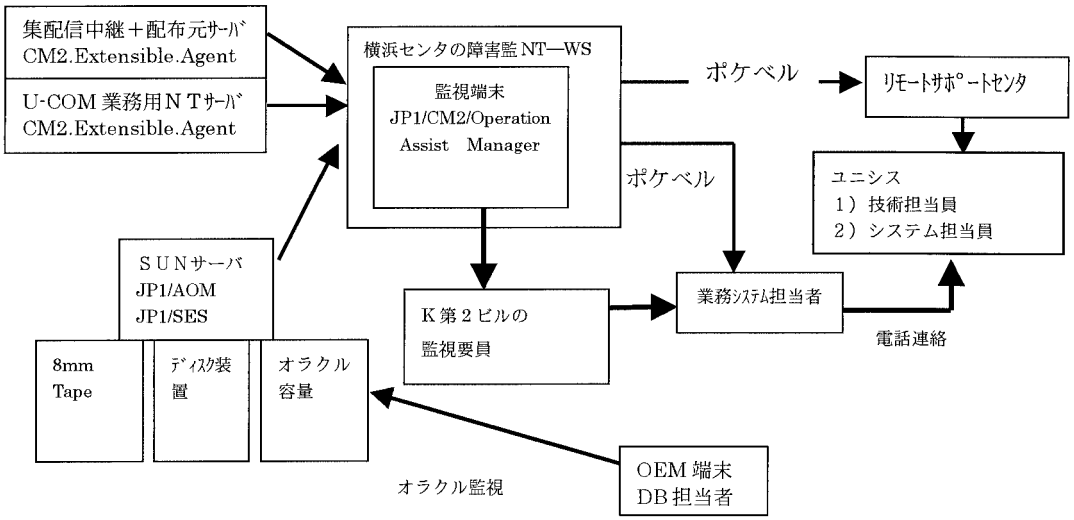


図 3　連絡ルート概念図

6. 構　成　管　理

構成管理の対象物は，開発に必要なものとして場所や設備類，開発用のハードウェア/ソフトウェアの全ての資材，開発用のマニュアル類などである．開発したものとはプログラムそのもの及び設計書類である．開発後の変更管理とは仕様変更やバグ対応などで変更が発生したプログラム及びドキュメント類のバージョン別管理や，本稼働に移行した後の各種資材管理が中心になる．

構成管理は，システム開発の全工程に渡って発生する管理業務だが，開発段階での構成管理と開発後の保守段階での構成管理を分けて述べたい．

6.1　開発時点の構成管理

6.1.1 構成管理担当組織

構成管理を始めるために，構成管理のための人と物が必要になる．人の問題として

開発要員とは別に構成管理を行う役割を持つ専任要員 2 名をプロジェクト期間中確保した．この構成管理要員は，チーム名称を「システム統合チーム」と呼び構成管理，バージョン管理，プログラム配布などの事務作業と運用作業を中心に行った．

事務作業と言いながらも事務管理のための各種ソフトウェアや OA ソフトを使いこなすスキルが必要な職種である．サブシステム毎の開発リーダ，設計者，プログラマ開発要員以外の要員として，こうした構成管理全般を担当する要員のコストもプロジェクトマネジャーはリソース確保時点で留意すべきだろう．

人以外のものとして構成管理のための専用サーバも検討したがコスト上の問題で開発用 NT サーバで代替した．

6.1.2 構成管理対象物の定義

管理するには，まず何を管理するのか決め，記録し，変更があれば記録文書の変更も実施する．

開発時点のカテゴリとして，設備・ハードウェア，ソフトウェア，マニュアル類，工程別納品ドキュメント，開発プログラムと分類した．但し管理の中心とすべきものはプロジェクトの製品の品質に寄与するプログラムそのものの構成管理だろう．

プロジェクトの開始段階早々に，構成管理の対象物全ての定義と管理ルールを徹底させておかないと開発者全体に混乱が生じる．プロジェクトそのものの品質を上げるためにも早めの作業が必要である．

1） 設備・ハードウェア

設備の内容では，まず開発場所のビルフロア，机，椅子，電話，FAX，コピー機，紙などの消耗品類，ハブなどのネットワーク設備，各種サーバ，PC，プリンタ類がある．詳細内容は構成管理用（開発用）NT サーバ内の表計算ソフトで管理した．

2） 管理対象ソフトウェア

ソフトウェアの管理は，バージョン番号，ライセンス番号，個数，どのサーバまたはクライアント PC に入っているか管理することになる．

ソフトウェアのライセンス管理は，ライセンス番号の紙だけを集めて一箇所にまとめておく．今回はソフト専用のスチール棚を用意し鍵をかけて，ソフトウェアの箱，ライセンス，各種媒体（CDROM や FD）を管理した．バージョンや個数など詳細内容は構成管理用（開発用）NT サーバ内にて表計算ソフトで管理した．

3） マニュアル類

マニュアルは，開発のための必須のもので多数の開発者が必要とする．マニュアルは冊数管理するため共有棚で鍵をかけて管理したほうが良いだろう．このようにしないと，必要なときに持ち出した後，返却してもらわないとすぐ行方不明になる．マニュアルタイトルや冊数など詳細内容は構成管理用（開発用）NT サーバ内にて表計算ソフトで管理した．

4） 工程別成果物ドキュメント

システム開発計画書で，工程別に何を出力するかユーザとベンダ間で合意しておく．このドキュメントを格納する場所をあらかじめプロジェクト発足時点で確

表 10　構成管理対象のドキュメント類と格納ディレクトリの一部の例

工程名	ドキュメント名	NT サーバ格納先
システム企画(SP)	システム開発計画書	E : \ Doc \ 030Keikaku
要求分析（UI）	要求機能分析書	E : \ Doc \ 040Yokyu
設計（SS＋PS）	論理＋物理 DB 構造	E : \ Doc \ DB
	ファイル設計書	E : \ Doc \ [システム名] \ 010FileIO
	コード設計書	E : \ Doc \ [システム名] \ 020Code
共通	パンチ仕様書	E : \ Doc \ [システム名] \ 030Punch
	共通仕様書	E : \ Doc \ [システム名] \ 040GeneralSpec
	帳票	E : \ Doc \ [システム名] \ 050UI \ Prt
	画面	E : \ Doc \ [システム名] \ 050UI \ Dis
	プロセスフロー，プロセス関連図	E : \ Doc \ [システム名] \ 060Process
	シェル関連，ジョブフロー	E : \ Doc \ [システム名] \ 080JOB
	JP 1 登録申請用紙	E : \ Doc \ [システム名] \ 090JP1
製造（PG＋PT）	プログラム管理表, プログラム単体仕様書	E : \ Doc \ [システム名] \ 070PgSpec

保しておく必要がある（表 10）.

今回は格納しておく場所は NT サーバ内のホルダとし，開発用 NT サーバにディスクを増設してドキュメントサーバとした.

サーバ内のディレクトリルールでは，開発途中では開発者が誰でも触れるホルダをドキュメント別に決め，公開した．アプリケーション開発を円滑にするためには，開発に入る前に各種環境が整っているべきである.

注意点として，毎日の出入りが激しいドキュメント類のウィルスチェックを毎日定期的に行うべきである．このプロジェクトでも外部から持ち込むドキュメントもあったせいかウイルスに汚染されたケースが 2 回ほど発生した．被害を大きくしないため毎日深夜に F-PROT を用いてサーバ内の全ドキュメントを検査した.

5） 開発プログラム

成果物ドキュメント同様，開発プログラムも格納する場所を開発着手する以前の段階で決めておく必要がある．誰もが管理しやすいように，サブシステム別/種類別または種類別/サブシステム別のツリー構造のディレクトリ設計，及び工程別のディレクトリ設計をするべきだろう．この部分は基盤構築チームと構成管理チームの共同作業になる.

開発時点では，まだベースラインが確定していないため構成管理対象とは言い難いがプロジェクトマネージメント上では体系立てられているかがチェックポイントと考える.

開発プログラムの工程別ディレクトリ管理方法を，このプロジェクトでのルールとしては表 11 のようにした.

表 11　構成管理対象のプログラム類と格納ディレクトリの一部

対象マシン	リソース区分名	格納先	工程	管理ツール
UNIX サーバ	COBOL ソース・Makefile, C ソース・Makefile	/export/coop/dev/［システム名］/src/work	PG, PT, IT, ST1-2	
		/export/coop/dev/［システム名］/src/public	OT1, OT2	RCS
	COBOL コピー句	/export/coop/dev/［システム名］/cpy/work	PG, PT, IT, ST1-2	
		/export/coop/dev/［システム名］/cpy/public	OT1, OT2	RCS
	C ヘッダ	/export/coop/dev/［システム名］/include/work	PG, PT, IT, ST1-2	
		/export/coop/dev/［システム名］/include/public	OT1, OT2	RCS
	シェル	/export/coop/dev/［システム名］/ksh/work	PG, PT, IT	
		/export/coop/test/［システム名］/ksh	ST1, ST2	
		/export/coop/dev/［システム名］ /ksh/public	OT1, OT2	RCS
	実行モジュール	/export/coop/dev/［システム名］/bin/work	PG, PT, IT	
		/export/coop/test/［システム名］/bin	ST1, ST2	
		/export/coop/dev/［システム名］/bin/public	OT1, OT2	RCS
NTサー	［illegible］ファイル	D : \Work\Src\［システム名］	PG, PT, IT	
	ダイナミックライブラリ			
		D : \Public\Prog\Dll	ST1～OT2	
	実行ファイル	D : \Work\Prog\［システム名］	PG, PT, IT	
		D : \Public\Prog\［システム名］	ST1～OT2	
	初期化ファイル	D : \Work\Ini	PG, PT, IT	
		D : \Public\Ini	ST1～OT2	VSS

6.1.3　バージョン管理

プログラムバージョン管理は，構成管理の一部として必須のため，開発チーム要員，特に若手メンバに対して指導教育を常に心がけなければならない．特にバージョンは仕様変更や納品行為といったイベントでは必ずチェックすべきと考える．

6.1.3.1　Unix 側プログラムのバージョン管理

Unix のバージョン管理ソフトウェアは SCCS と RCS に二分される．どちらも結果としては同程度の機能を提供しており，今回は経験者の多い RCS を採用した．以下 RCS 利用時の標準化として決めたルールを説明する．

1）　ライブラリ管理者

ライブラリ管理者は開発ベンダ側の各サブシステム開発リーダとする．

各サブシステム開発リーダは RCS でシェル及び実行モジュールの履歴管理を行う．

2）　RCS 管理で使用する主なコマンド

ci	リビジョンのチェックイン（登録）
co	リビジョンのチェックアウト（取り出し）
ident	ファイルのバージョン識別
rlog	リビジョン履歴

rcs　　ファイル属性変更

3) RCS 管理対象物

COBOL ソースと COPY 句，c ソース及びヘダーファイル，シェル，SQL 及び SP（Stored Procedure）

4) 管理対象物内に定義すべきキーワード

ident コマンドで必要とするキーワードとして ＄Id＄ をソース内に記述する．

例) 〈COBOL ソース〉 の指定方法

```
WORKING―STORAGE　　SECTION.
＊プログラム（Ver―Rev）管理
01 VERHEAD PIC X（70）VALUE
　　"＄Id＄ ".
```

5) RCS 管理単位

開発する各業務のサブシステムを１単位として管理する．

各業務サブシステム毎にソース用ディレクトリを指定し，ソース及び Make ファイルを格納する．その１階層下に RCS ディレクトリを作成し RCS のリビジョンファイルが入るようにする．

例)

	ディレクトリ例	バージョン管理対象物
1	${HOME}/include/public/xx/RCS	C ﾍｯﾀﾞﾌｧｲﾙ
2	${HOME}/src/public/xx/RCS	C ｿｰｽ，C ｿｰｽ用 Makefile,COBOL ｿｰｽ，COBOL ｿｰｽ用 Makefile
3	${HOME}/sql/public/xx/RCS	SQL・SP
4	${HOME}/ksh/public/xx/RCS	ｼﾞｮﾌﾞ(Korn ｼｪﾙ)

xx はサブシステム ID のアルファベット２桁

${HOME}=/export/coop/dev/各業務ｼｽﾃﾑ

6) 管理手順

以下のルールを決めたが，納品行為としてのバージョン管理と関連していたこと，及びプロジェクトメンバがバージョン管理に慣れていたため，ルールは尊守されている．具体的手順などのサンプルを早期提示することでルールが徹底される．

① 開発・単体試験・結合試験段階

個人用ディレクトリもしくは＄{HOME}/資材区分/work で作成，修正作業を実施する．

② システム試験準備（商用機配布準備）

システム試験時に，＄{HOME}/資材区分/public へコピーし，チェックインとチェックアウトを実施する．Make（コンパイル）が必要なものは Make を実施する．

③ システム試験段階

＄{HOME}/資材区分/public よりシステム試験環境（/export/coop/test）へコピーする．

④ 不具合修正時

$｛HOME｝/資材区分/public から$｛HOME｝/資材区分/work へコピーして修正及び再度単体試験を実施する．修正確認後 b）に戻る．

⑤ 商用機（本稼働機）への稼働

$｛HOME｝/資材区分/public の資材を配布するように配布申請依頼をする．配布担当者が配布依頼書の内容に従い配布を実施する．

6.1.3.2 クライアントプログラムのバージョン管理

クライアントプログラムのバージョン管理については，Microsoft 社の Visual SoureSafe という製品を評価し，採用した．Microsoft Visual SourceSafe for Windows（以下 VSS）は，ソフトウェアアプリケーションのチーム開発を支援するプロジェクト指向のバージョン管理システムである．ファイルに対する変更内容が記録されるため，ファイルの履歴表示，以前のバージョンの取得，複数の開発者との並行開発が可能になる．

1） VSS での管理対象物

VB，VC＋＋のフォーム，モジュール，のバージョンを管理した．

ベースラインとしてのバージョン 1 が登録されると，その後は差分のみが VSS のデータベースで管理される．ドキュメントも管理できるが，Word や VISIO の文書は差分管理されず，前のバージョンと次のバージョンとファイルそのものがフルに格納されディスク容量が非常に要求されるため，ドキュメントは VSS の管理対象から外すべきだろう．

2） バージョン番号

クライアントプログラムのバージョン管理は大きく分けて三つの分野があるため，管理基準を明確にしないと混乱する．その三つとは

- VB で実行ファイルを作成するときに与えるバージョン番号及びバージョン属性情報
- VB のソースを管理する VSS 内のバージョン番号，
- 出来上がった VB の実行プログラムを配布するソフト NetM/DM 内の配布バージョン番号

である．それぞれのバージョン番号の管理体系及び桁数を以下に示す．

① VSS のバージョン番号体系

1 からの連番で，2,3,4 と増加していく．VSS のバージョン管理は，VSS の世界の中でだけ使用する．連番のため VB のバージョンとは紐つけできない．このため，VSS に格納するときのコメント情報として，VB のバージョン番号や年月日を入力することをルール化した．

② VB のバージョン番号体系

バージョン番号はメジャー番号．マイナー番号．リビジョン番号の三つに分けられる．

例）1.01.45

このプロジェクトの番号付けルールでは，結合テストやシステムテストに提出するときは，リビジョン番号を付けないこととした．またマイナー番号は小数点以下 2 桁から開始する．

③ NetM/DM のバージョン番号

半角の数字 6 桁で小数点は無い．例）123456

NetM/DM でプログラム配布するときのバージョン登録としては，バージョン番号 6 桁に意味を持たせて紐つけした．頭 2 桁がバージョン，次の 2 桁がマイナーバージョン，下 2 桁は 00 固定である．

例）V 1.00（VB 側）⟶010000（NetM/DM）

V 1.01（VB 側）⟶010100（NetM/DM）

3） VSS 内の管理ホルダ

VSS の良いところはソースの一元管理ができる点である．誰かがチェックアウトしてソース修正している間は，他のメンバはアクセスできない．これは，VSS が自分で独自のデータベースを持っており，その中で各ソースの利用状況が管理されているためである．

6.2 保守時点の構成管理

開発が一段落し導入稼働段階以降は，保守時点の構成管理(変更管理)に移行する．保守時点ではプロジェクトメンバも縮小し設備関連も縮小，ハードウェアやソフトウェアも固定化されるため，構成管理の主体は，成果物としてのプログラム及びドキュメントの変更管理が中心になる．これを怠ると最新ソースの所在が不明，関連する仕様書類の所在が不明という，よく巷のプロジェクトでありがちな事態になってしまう．

以下，稼働後の構成管理を中心に実例を述べていきたい．

6.2.1 構成管理（変更管理）の目的

1） 混乱の防止

開発スケジュールが厳しいとプログラムを先に開発変更して，仕様書は後回しにすることが多々ある．その仕様書変更がきちんとされないと次の変更時点で，時間は経過している，前回保守したメンバは既にいないということになると変更作業そのものが混乱してしまう．プログラムと仕様書の同期をとって構成管理(変更管理）することで混乱を防止できる．

2） 品質の向上

構成管理（変更管理）が不十分な場合，複数の担当者が二重に変更してしまったり，変更し忘れたりの事故が起きる可能性がある．構成管理（変更管理）がされることで，全てのプログラムや仕様書がルールとおり整備され，システムの総合的品質が向上する．また，開発コストの超過防止やスケジュール遅れを防止するための手段にもなる．

6.2.2 本稼働後の構成管理対象物

管理対象物をソースプログラムと仕様書関連のドキュメントに限定した．実行モジュールは別途プログラム配布システムで管理するためである．ソースプログラムは Unix プログラムとクライアントプログラムに二分される．ドキュメントは，プログラム保守時点のプログラム内容と同期がとれている仕様書を保管するべきである．

1) プログラム関連の整理

	管理方法		管理対象	ｻｰﾊﾞ別構成管理	
	UNIX	ｸﾗｲｱﾝﾄ		UNIX	ｸﾗｲｱﾝﾄ
cobol ｿｰｽ(cob)	RCS で管理	−	○	○	−
Cobol ﾒｲｸﾌｧｲﾙ	RCS で管理	−	○	○	−
cobol ｺﾋﾟｰ句	RCS で管理	−	○	○	−
C ｿｰｽ(c/pc)	RCS で管理	−	○	○	−
C ﾒｲｸﾌｧｲﾙ(mak/mk)	RCS で管理	−	○	○	−
C ﾍｯﾀﾞ(h)	RCS で管理	−	○	○	−
ｼｪﾙ(sh)	RCS で管理	−	○	○	−
Sql(ｽﾄｱﾄﾞを含む)	RCS で管理	−	○	○	−
ﾛｰﾄﾞﾓｼﾞｭｰﾙ(bin)	配布ｼｽﾃﾑで世代管理	−	×	−	−
Lib(so を含む)	配布ｼｽﾃﾑで世代管理	−	※	○	−
制御ﾌｧｲﾙ(prm)	配布ｼｽﾃﾑで世代管理	−	○	○	○
その他ﾌｧｲﾙ(申請ﾍﾞｰｽ)(事前許可が必要)	−	−	○	○	○
Vb のｿｰｽ	−	VSS で管理	○	−	○
Exe	−	配布ｼｽﾃﾑで世代管理	×	−	−
Dll	−	配布ｼｽﾃﾑで世代管理	※	−	−
Ini	−	VSS で管理	○	−	○
Etc(ﾊﾟﾗﾒｰﾀﾌｧｲﾙ等)	−	VSS で管理	○	−	○

○構成管理対象　　×構成管理対象外　　※ファイル種別”etc”で申請する。

2) 仕様書関連ドキュメント

	現状管理方法		管理対象	ｻｰﾊﾞ別構成管理	
	UNIX	ｸﾗｲｱﾝﾄ		UNIX	NT ｻｰﾊﾞ
XXXXX.doc	−	ﾃﾞｨﾚｸﾄﾘ別世代管理	○	−	○
XXXXX.xls	−	ﾃﾞｨﾚｸﾄﾘ別世代管理	○	−	○
XXXXX.vsd	−	ﾃﾞｨﾚｸﾄﾘ別世代管理	○	−	○
XXXXX.txt	−	ﾃﾞｨﾚｸﾄﾘ別世代管理	○	−	○

○構成管理対象　　×構成管理対象外

6.2.3 構成管理ツール

構成管理のツールは，手作りした．保存申請を EXCEL ワークシートで行うため，申請内容の検査を行うツールを EXCEL マクロと unix シェルで作成した．検査内容は，全角/半角チェック，申請ファイルの存在確認チェック等である．クライアント側の申請ファイルのバージョンチェック，及び構成管理サーバへのファイルコピーは人間の手作業で行っている．Unix 側のバージョンチェック，構成管理サーバのディレクトリへのコピーは作成したシェルで行っている．

参考例として，図 4 の unix 側構成管理イメージを示す．

6.2.4 その他・手続きに関連する項目

保守時点の構成管理として変更管理を徹底していくには，さらに業務フローや申請用紙の内容を詳細に定義していく必要がある．

1) ベンダー作業定義

担当者レベルの作業定義及びベンダ側責任者の作業定義とチェック内容，承認ルールを明文化する．

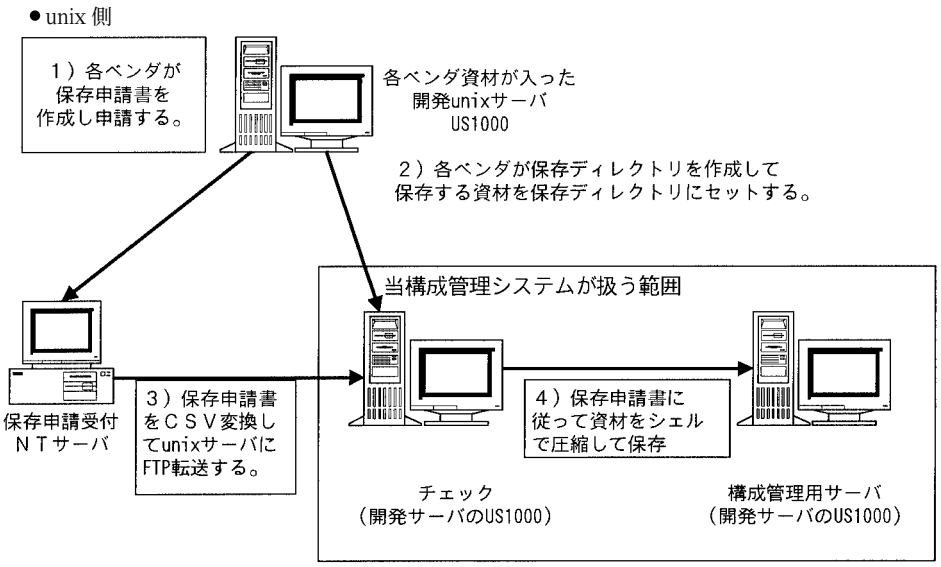


図 4　unix 側構成管理イメージ図

2)　ユーザ側システム開発担当としての受入側作業定義

保存申請内容の確認方法，承認ルール，保存担当者への引渡し方法などを明文化する．

3)　ユーザ側保存担当者

申請受付方法，内容審査方法，差戻し規約，保存サーバへの格納方法，保存サーバから旧バージョンの格納物の提供規約と提供方法などを明文化する．

7. お わ り に

プロジェクト全体を通して，基盤構築チームが業務 AP 開発チームに対して，円滑に開発が進む環境を提供するには，業務 AP チームが開発着手する以前に全ての方式論や標準化問題，運用設計や各種技術問題が解決されているべきと述べてきた．

しかし，その分野は多岐に渡るため，プロジェクトマネージメント上は優先順位を付けて作業していくことになる．運用管理/構成管理のソフトは，各ソフトウェアベンダから常にリリースされ続けているため，こうした業界動向を注視し，常に新技術の検討を行うべきと考える．特に自社製品をほとんど使わず，他社製品を組み合わせたマルチベンダ製品でのサービスを提供するには，その時点での実績が高く評判の良いツールや実績のあるソフトウェアを使うべきだろう．

このプロジェクト開始時点で，ミッションクリティカルな業務を汎用機から UNIX サーバにダウンサイジングして移行するのは時期早尚ではないかという声がユーコープ事業連合の内部から上がっていた．従来どおり汎用機を使い，業務 AP だけを再構築した方が安全のはずだという意見も寄せられていた．

そこで，筆者等はプロジェクトの正式キックオフまでの期間に，プラットフォームや各種ソフトウェアの実証実験を行い，UNIX サーバでの実現可能性について十分やっていけると確信を持つに至った．自分自身で確信し，ユーザを説得できたのはやは

りリスクを避けるための実証実験を通じて得た事実であった．

稼働後，約1年が経過しようとしているが，運用も安定し大規模なミッションクリティカルな業務でもUNIXサーバで稼働できる実績を残せたと考えている．全チーム一丸となったスケジュールにのっとり納期どおり稼働したうえ，ユーコープ事業連合に対し組合員から個別配送対応がよくなったいう声が届き，本当の顧客から評価されたため情報システム部は内部的にも面目を施し顧客満足度も高い．

最後に，このプロジェクトを推進したユーコープ事業連合の情報システム部の方々，業務AP開発を担当したJ社の開発リーダの方々，配送センタと物流センタシステムの開発担当だったH社のSEリーダの方々，及び当社の本プロジェクトの方々に感謝の意を表する．

＊1　U-COM　ユーコープ事業連合が独自に97年～99年に開発した基幹業務システムで大型汎用機からダウンサイジングしたもの，共同購入，組合員，未集金の業務でSUNのUSシリーズで構築したもの．

＊2　CTOPIA　ユーコープ事業連合とコープBが共同で94年～96年に開発したシステムで，会計/業績管理（部門別損益管理会計)/商品企画（商品供給仕入実績蓄積&検索)/共通マスタの業務をHPのTシリーズで構築したもの．

＊3　スタブ　テスト時点で，テスト対象モジュールが呼び出す下位のモジュールがまだ完成していない場合は，仮にその代わりになるスタブと呼ばれる簡単なモジュールを作成して対応する．

＊4　お知らせメッセージ　VBの業務画面で最初にログインしたときに表示されるテキスト画面のこと．

参考文献

［1］　日本ユニシス,「システム開発作業体系解説書」, 日本ユニシス, 1996年9月.

［2］　日本ユニシス,「Next 4 TEAMmethod TEAMprogram研修参加者ガイド」, 日本ユニシス, 1998年8月.

［3］　小野村英敏, プロジェクトマネージャ教科書, オーム社, 1996年.

［4］　戸田忠良, 上級SEになるための50のポイント, 共立出版株式会社, 1991年3月.

執筆者紹介　磯　部　　寛（Hiroshi Isobe）

1951年生．1974年芝浦工業大学電気通信工学科卒業．1974年日本ユニシス(株)入社．現在，I&Cソリューションサービス部ソリューション開発1室に所属する．主に中小型汎用機での流通業ユーザ（卸売業，物流業）のシステム開発に従事．最近数年間は小売業のオープン系での大規模システム開発に従事している．特種情報処理技術者．