大規模開発を支えた開発プロセス

Development Process for Super Large Scale Development

小笠原　一洋

要　約　ANACore[*1] 開発プロジェクトは非常に大規模で，開発期間が長期に渡ったため，プロジェクトの途中で新規参入する開発者も多かった．またシステムに接続する外部インタフェースも多岐に渡り，それぞれに異なるアーキテクチャが用意されていた．このような状況を踏まえ，開発作業を円滑に進めていくために開発環境の整備とアーキテクチャ毎の開発プロセスを定義した．また各開発プロセスで作成される成果物についても，各種規約の順守性やバグ混入の可能性をチェックする仕組みを作ることで，高品質が保たれるようにした．

Abstract　The ANACore Development Project was large scale not only in the number of developers involved at one time but also in number of new developers joining the project due to the long development period. The system was to be used from several external interfaces which required different software architecture to support such interface. We defined different development process for respective architecture to enable wide variety of architecture to be integrated into one system. We have also created within each process check methods for each deliverables to secure high quality.

1. はじめに

ANACore[*1] 開発プロジェクトは，2 万人月の大規模，ピーク時 800 人の大人数，また 5 年間という長期間にわたるものであった．システムに接続する外部インタフェースも CUI[*2]，GUI，外部システム，バッチと多岐にわたり，それぞれに異なるアーキテクチャが必要であった．

本稿では，これらの特徴に対して本プロジェクトがどのように取り組んだかを紹介する．2 章で開発環境，3 章で開発チーム構成，4 章で開発プロセス，5 章で品質面について説明する．

2. 開発環境

ANACore 開発では全日本空輸株式会社（ANA）の社屋内に社内 LAN から独立した開発専用のネットワーク（以下，独自 LAN）を敷設し，その中に開発環境を構築した．ピーク時は 800 人規模の開発作業となるため，独自 LAN には AP サーバ，DB サーバや構成管理サーバなど直接開発に必要なサーバから，メールサーバやファイルサーバなどの開発作業をサポートするために必要なサーバまで，開発に必要となるサーバを全て用意した（表 1）．

設計書の作成やプログラム開発は各開発者の PC で行ったが，実装ツールや構成管理ツールについては，表 2 に示すように，基本設定を済ませた形のインストールイメージや導入手順書を用意することで，プロジェクト途中から開発者が参入する場合も容易に開発環境を構築し，スムーズに作業に入ることができるようにした．

表 1　開発環境サーバ一覧

サーバ名称	OS・SW	主な用途
ドメイン管理サーバ	Windows Server ・Active Directory	開発ユーザを管理する.
メールサーバ	Windows Server ・Exchange	開発者間での情報共有や，テスト環境のリリース状況の連絡などを行う.
ファイルサーバ	Windows Server /NAS	開発ツールや開発を進めていく上で必要な台帳などの非成果物，またテスト証跡等を保管する.
構成管理サーバ	Windows Server ・ClearCase	納品対象のソースコード，ドキュメントを保管/管理する.
ビルドサーバ	Windows Server ・Apache Ant	構成管理サーバ上のソースコードからテストに必要なライブラリを生成する.
APサーバ	RedHat Linux ・Weblogic Server	テスト対象のライブラリをデプロイし，機能テストを実施する.
DBサーバ	HP-UX ・Oracle	業務やシステムで利用するデータを管理する.
バックアップサーバ	Windows Server ・NetBackup	構成管理サーバ，ファイルサーバのデータバックアップを行う.

表 2　主な開発ツール一覧

ソフトウェア名称	主な用途
Eclipse	Java開発の統合開発環境. 単体テストを行うためのJUnitや，品質を担保するためのCheckStyle，FindBugsなどのプラグインが導入された形で用意した.
WebLogic Server	開発者のPCでテストを実施する際の実行環境となるアプリケーションサーバ. 初期設定まで行われた状態でインストールされるインストールイメージ，また導入手順書や動作確認用のライブラリも用意した.
ClearCase	構成管理クライアント. 一般的にはあまり使われていないSWのため，導入手順書から利用手順書まで用意した.

3.　開発チーム構成

大規模，大人数での開発作業を効率よく進めるため，開発者を業務チーム，ゲートウェイチーム，標準化/DB チーム，移行チーム，インフラチーム，統制チームの 6 チームに分け，開発作業を分担した（表 3）．業務チームについては業務特性から，更に予約，発券，搭乗，座席管理，共通の 5 チームに分けた．なお，移行チームとインフラチームの作業は，本稿での説明の範囲外とする．

表 3　チーム構成

チーム名称	主な役割	開発担当
業務チーム	要件定義，業務ロジックの設計，実装を担当. 更に予約，発券，搭乗，座席管理，共通のチームに分かれる.	サービス開発 GUI開発 バッチ開発
ゲートウェイチーム	端末や外部システムからの入力を受け付け，対応する業務ロジックの呼び出し，また呼び出し結果を返却する機能の設計，実装を担当.	CUI AP開発 他シス[*3]AP開発
標準化/DBチーム	システムの共通基盤となるフレームワーク，DBスキーマの設計/実装，また開発プロセスの策定を担当.	DB開発
統制チーム	プロジェクトの進捗管理や品質管理，またライブラリアンとして、テスト環境へのリリース作業も担当.	※ライブラリアン作業
移行チーム	現行システムを新システムに切替えるための方式検討から，本番切替え時のデータ移行作業までを担当.	※本稿での説明範囲外
インフラチーム	開発環境，本番環境のHW，SWの導入から保守までを担当.	※本稿での説明範囲外

4. 開発プロセス

ANACore 開発は長期に渡るため，開発フェーズをグループ 1 ～ 3，及び現行改修取込み 1 ～ 3 の六つのフェーズに分割し，段階的に行った．それぞれの開発フェーズ内の作業工程は一般的なプロジェクト同様に，要件定義，論理設計，物理設計，実装の工程であった．ただし要件定義が終わった以後の工程については，アーキテクチャに合わせてサービス開発，CUI AP 開発，GUI 開発，他シス AP 開発，バッチ開発，DB 開発の六つの開発方式に分け，それぞれに詳細なプロセスを定義した開発ガイドラインを作成し，それらに従って作業を進めた．実装が終わった後に機能テスト工程を設け，開発したそれぞれの部品を統合した機能単位でのテストを実施した（図 1）．

図 1　開発プロセス

4. 1　要件定義

ANACore 開発は，一言で表せば従来のメインフレームで動作するアプリケーションを機能はそのままに，AirCore[*4] パッケージを利用してオープン化する作業である．そのため要件定義工程ではシステム化する要件を一から定義していくやり方ではなく，メインフレーム機能と AirCore パッケージで提供される機能とを対応づけ，ANACore として実現する機能を選択していく形で進めた（図 2）．新たに追加する機能，メインフレームと変わる機能，あるいは廃止する機能などは適宜 ANA と調整し，システム化範囲を確定した．

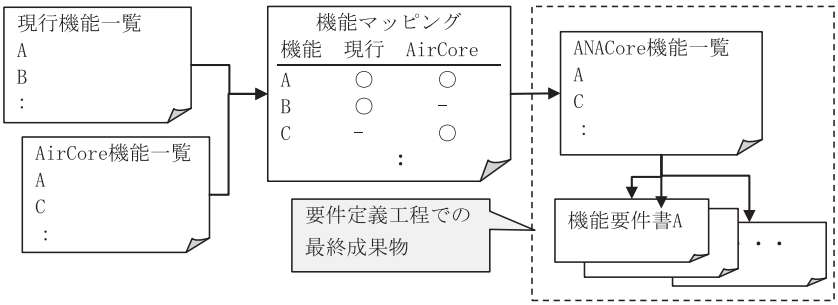

図 2　要件定義作業

表 4　要件定義工程の主な成果物

成果物名称	説明
機能一覧	ANACoreで実現する業務機能を一覧化した管理台帳． プロジェクト全体で一篇作成した．
機能要件書	各業務機能の入出力，及び業務処理の詳細をまとめた仕様書．機能一覧で定義した機能単位に作成した． バッチ機能については入出力ファイルの項目とレイアウト定義を入出力定義書として，また外部システム接続機能については送受信する電文の項目とレイアウト定義をフォーマット定義書として，別冊で作成した．

最終的に ANACore として提供する機能を機能一覧として定義し，それぞれの機能ごとに機能要件書と呼ぶ成果物を作成して詳細な業務要件を纏め上げた（表 4）．なお機能要件書は業務の大きな分類（予約，発券，搭乗，座席管理，共通）に応じて，それぞれの業務チームで作成した．

4. 2　論理設計/物理設計/実装

論理設計/物理設計/実装工程では機能要件書を入力に，ANACore アーキテクチャに合わせて業務要件を実現するための設計/実装を行った．

4 章冒頭で述べた通り，論理設計以降の工程についてはサービス開発，CUI AP 開発，GUI 開発，他シス AP 開発，バッチ開発，DB 開発の六つに分けた（図 3）．本節でそれぞれについて解説する．

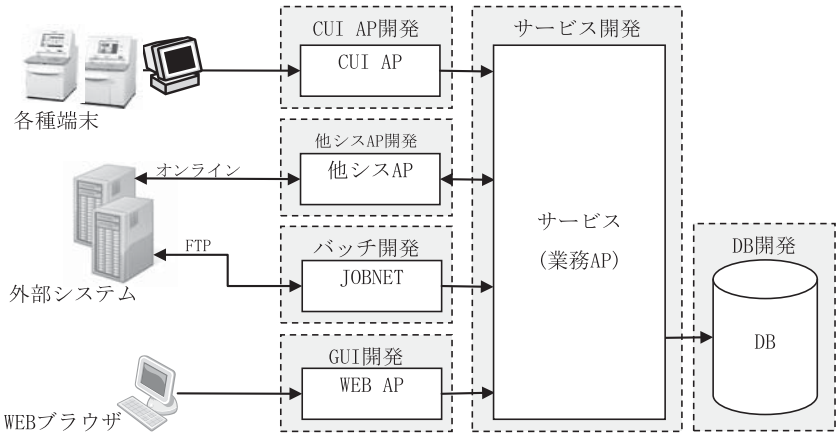

図 3　ANACore アーキテクチャと開発方式の関係

4. 2. 1　CUI AP 開発

CUI AP 開発では，各種端末からファンクションとして送られてくる入力電文を解析する処理，解析結果を基に対応する一連のサービス（業務処理）を呼び出す処理，そして処理結果を応答電文として構築し端末に返却する処理を担う，CUI AP（CUI アプリケーション）を作成するための設計/実装を行った．開発はゲートウェイチームが担当し，図 4 に示すように，入出力定義書の作成，インタラクション仕様書の作成，実装，という流れで開発した．

本開発の成果物を表 5 に挙げる．入出力定義書の作成では，ファンクションとして送られて

くる入力電文，及び業務処理結果として返却する出力電文について，構成項目とフォーマットを定義した．インタラクション仕様書の作成ではファンクションに対応して呼び出すサービスと，受け渡す項目を定義した．呼び出すサービスは 4. 2. 5 節で述べるサービス開発で作成するため，適宜業務チームと連携しながらインタフェースを調整した．

実装は ANACore の CUI フレームワークが提供する機能を利用し，Java のプログラムとして作成した．単体テストは CUI フレームワーク内に専用のテストツールを用意し，サービス開発とは独立して実施した．

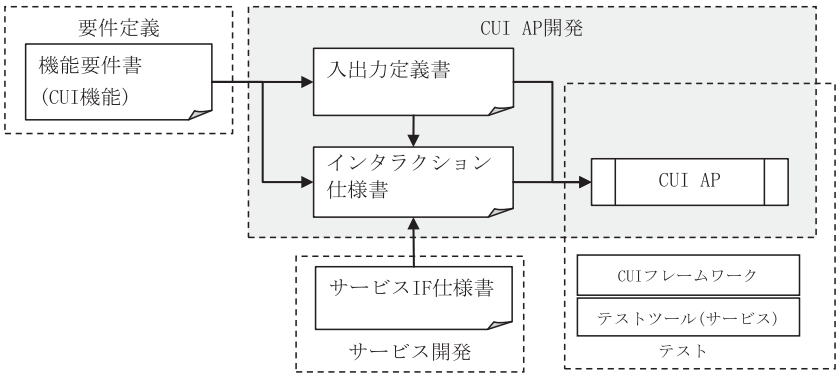

図 4　CUI AP 開発作業

表 5　CUI AP 開発の主な成果物

成果物名称	説明
入出力定義書	要求/応答電文の入出力項目，及び入出力フォーマットなどを定義した設計書． 端末からの入力ファンクション単位に作成した．
インタラクション仕様書	呼び出すサービスや，サービスと受け渡す入出力項目などを定義した設計書． 端末からの入力ファンクション単位に作成した．
CUI AP	インタラクション仕様書にしたがって実装したJavaプログラム．

4. 2. 2　他シス AP 開発

他シス AP 開発では外部システムが ANACore を利用する機能の開発と，ANACore が外部システムを利用する機能の開発に分類した．前者では問い合わせ電文の解析処理，解析結果に応じて対応するサービスを呼び出す処理，またサービスの処理結果を応答電文として構築し，返却する処理を，後者ではサービスからの問い合わせ内容を電文として構築，送信する処理，また応答電文を解析し，呼び出し元のサービスに返却する処理を担う，他シス AP（外部システム連携アプリケーション）を作成するための設計/実装を行った．開発はゲートウェイチームが担当し，図 5 に示すように，電文項目定義書の作成，コマンドクラス仕様書の作成，連携クラス仕様書の作成，実装，という流れで開発した．

本開発の主な成果物を表 6 に挙げる．電文項目定義書では外部システムと送受信する電文について，構成項目とフォーマットを定義した．コマンドクラス仕様書の作成では外部システムからの問い合わせ電文に対応して呼び出すサービスと，受け渡す項目を定義した．連携クラス

仕様書の作成ではサービスから外部システムに渡す項目や，外部システムの呼び出し方などを定義した．CUI AP 開発と同様に，呼び出す/呼び出されるサービスと関係するため，適宜業務チームと連携しながらインタフェースを調整した．実装は外部システムとの連携を実現する他システムフレームワークを利用する Java プログラムとして作成し，専用の単体テスト環境でテストした．

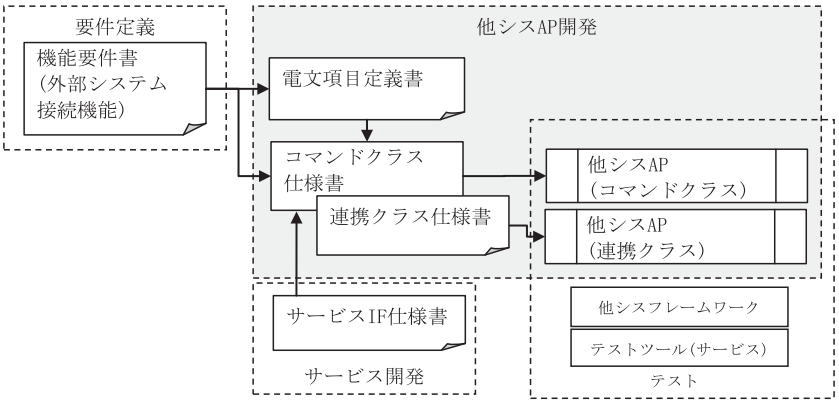

図 5　他シス AP 開発作業

表 6　他シス AP 開発の主な成果物

成果物名称	説明
電文項目定義書	要求/応答電文の入出力項目，及び入出力フォーマットなどを定義した設計書. 外部システムと送受信する電文単位に作成した.
コマンドクラス仕様書	呼び出すサービスや，外部システムとサービスで受け渡す項目などを定義した設計書. 外部システムから電文を受信する機能単位に作成した.
連携クラス仕様書	サービスと外部システムで受け渡す項目や，外部システムの呼び出し方などを定義した設計書. 外部システムに電文を送信する機能単位に作成した.
他シスAP	コマンドクラス仕様書にしたがってコマンドクラスと呼ぶJavaプログラムを，また連携クラス仕様書にしたがって連携クラスと呼ぶJavaプログラムをそれぞれ実装した.

4. 2. 3　バッチ開発

バッチ開発では，バッチ業務処理を呼び出すジョブネット（JP1/AJS2），及びジョブネット（ジョブ）から呼び出すシェルの設計/実装部分と，バッチ業務処理本体の開発に分類した．バッチ業務処理本体はサービス開発（4. 2. 5 項）としたため，ここではバッチの実行を制御するジョブネットとシェルについて記載する．これらはともに業務チームが開発を担当し，図 6 に示すように，ジョブネット定義書の作成，シェル一覧の作成，実装，という流れで開発した．

本開発の主な成果物を表 7 に挙げる．ジョブネット定義書の作成ではジョブネットの起動条件の定義，また業務処理を呼び出す順序をジョブフローとして定義した．シェル一覧の作成ではジョブネット定義書で定義したジョブの単位に処理内容を定義した．なおバッチフレームワークが提供する機能を利用する場合は，別途用意した生成ツールを利用し，シェル一覧から定義内容に従った処理を実現するシェルを自動生成できるようにすることで，シェル実装経験

の少ない開発者の負荷を軽減した．なお，その場合は単体テストを不要とした．

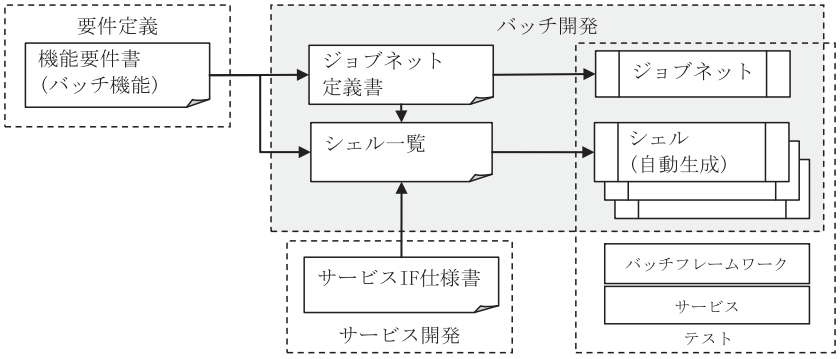

図 6　バッチ開発作業

表 7　バッチ開発の主な成果物

成果物名称	説明
ジョブネット定義書	ジョブネットの起動条件，及びジョブフローを定義した設計書． バッチ機能単位に作成した．
シェル一覧	ジョブネット定義書で定義した各ジョブから呼び出すシェルの処理内容を定義した設計書． 業務チーム単位で1篇ずつ作成した．
ジョブネット	ジョブネット定義書にしたがって実装したジョブネット．
シェル	シェル一覧からツールを利用して自動生成したシェル． 特殊処理が必要な一部のシェルについては手動作成した．

4. 2. 4　GUI 開発

GUI 開発では，GUI 業務処理を呼び出す Web AP（Web アプリケーション．JSP/Servlet）の設計/実装を行った．バッチ業務同様，GUI 業務処理の開発自体は次項のサービス開発に含めた．これらはともに業務チームが担当し，図 7 に示すように，画面遷移定義書の作成，画面定義書の作成，インタラクション仕様書の作成，実装，という流れで開発した．

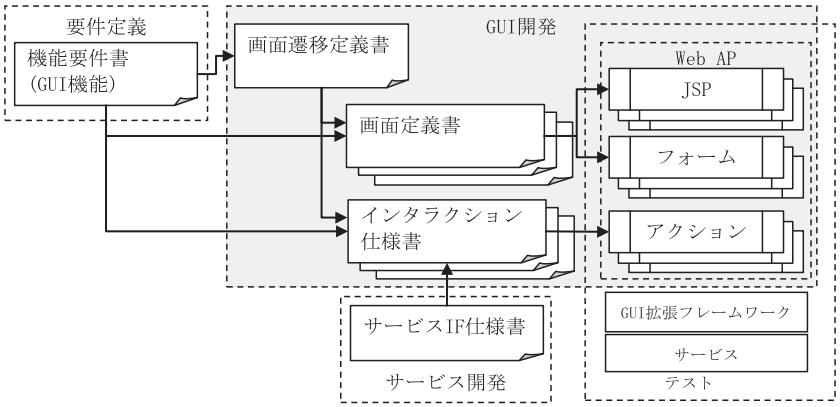

図 7　GUI 開発作業

表 8　GUI 開発の主な成果物

成果物名称	説明
画面遷移定義書	画面フローや画面遷移時の処理（インタラクション）を定義した設計書．一連のGUI業務処理単位に作成した．
画面定義書	WEB画面に表示する項目やフォーマットを定義した設計書． 画面単位に作成した．
インタラクション仕様書	画面間の遷移に応じて呼び出すサービスクラスや，受け渡す項目を定義した設計書． 画面遷移の単位に作成した．
Web AP	画面定義書，インタラクション仕様書を基に実装したJSP，Servlet．Strutsベースでの開発であったため，実際にはJSP，アクションフォーム，アクションクラスとして実装した．

本開発の主な成果物を表 8 に挙げる．画面遷移定義書の作成では，GUI 業務に合わせて表示する画面フローを定義した．画面定義書の作成では，画面遷移定義書で定義した画面ごとに，実際に WEB ブラウザ画面に表示する項目やフォーマットを定義した．インタラクション仕様書の作成では，画面遷移に応じて呼び出すサービスと，画面と受け渡す項目を定義した．実装は JSP と Java のサーブレットを作成し，サービスと連携させテストした．テストツールは画面操作をバッチとして自動化し，結果の検証まで行う Selenium[*5] を利用した．

4. 2. 5　サービス開発

サービス開発では，業務処理の中核となるビジネスルール実現の役割を担う，サービスを作成するための設計/実装を行った．業務チームが開発を担当し，図 8 で示すように，サービス IF 仕様書の作成，クラス仕様書の作成，実装，という流れで開発した．

本開発の主な成果物を表 9 に挙げる．サービス IF 仕様書の作成では，接続インタフェース毎に入出力，及び外部仕様を定義した．クラス仕様書の作成では，業務処理の詳細仕様，及び DB アクセス処理を定義した．実装は Java のプログラムとして行い，単体テストは JUnit 及び，サービスの呼び出しと戻り値を比較する専用の SI テストツールを用いた．

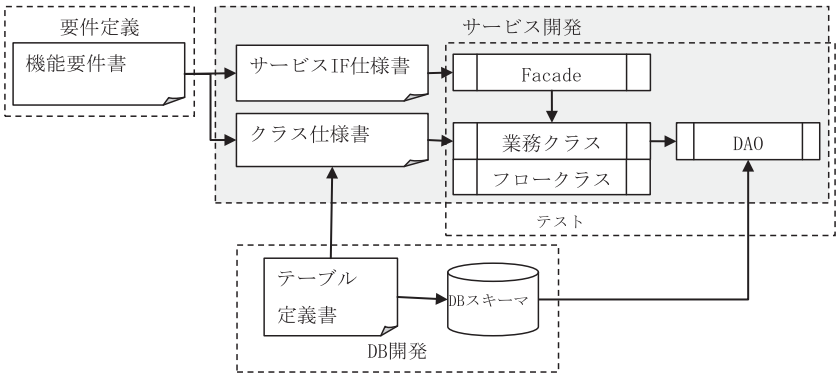

図 8　サービス開発作業

表 9　サービス開発の主な成果物

成果物名称	説明
サービスIF仕様書	システム外部との接続インタフェース毎に入出力項目や外部仕様を定義した設計書. 機能単位，および他チームからの依頼単位(後述)に作成した.
クラス仕様書	業務処理の詳細やDBアクセス処理について定義した設計書. 基本的にはエンティティ(意味のある業務データの集まり)の単位に作成した.
Facade/MF	サービスIF仕様書を基に実装したJavaプログラム. 外部との接続インタフェースとなるFacadeと，他チームからの依頼によって作成する，内部との接続インタフェースとなるMF(Module Facade)に分けて作成した.
業務/フロークラス	クラス仕様書を基に実装したJavaプログラム. 一つのエンティティに関連する業務を実現する業務クラス，複数の業務クラスやMFを呼び集めて一連の業務処理を実現するフロークラスに分けて作成した.
DAO	DBアクセスを行うJavaプログラム. iBatisを利用し，DBアクセスを行うDAO本体や，DBデータを受け渡すDTO(Data Transfer Object)をDBスキーマ情報から自動生成することで，基本的な処理についてはコーディングが不要となり，実装負荷が大幅に軽減した.

なお ANACore のベースとなっている AirCore では，旅客業務を 14 種類の業務に分類し，それぞれにモジュールを分けて作成することで，AirCore を採用する顧客が必要なモジュールだけを選択的に利用できるアーキテクチャとなっていた．ANACore でもそのアーキテクチャを踏襲し，機能要件を 14 のいずれかのモジュールに当てはめて開発した．ただし機能によっては他モジュール管轄の一部機能，あるいはデータを利用したいケースが出てくる．そのような場合は図 9 に示すように，他チームにサービス IF の作成を依頼してそれを呼び出す形とし，他モジュールへの直接のアクセスを禁止した．それによってモジュール間で連携する箇所がサービス IF に集約されて各モジュールの独立性が高まるとともに，チームごとの開発範囲が担当モジュールに限定されることで開発効率も高まった．

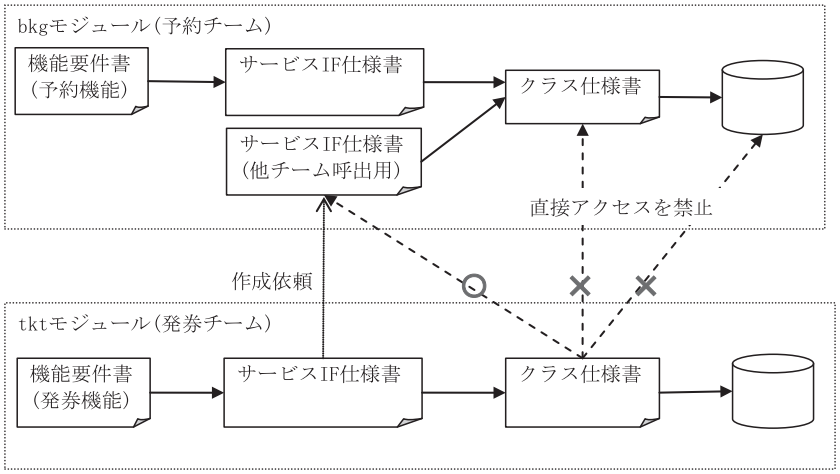

図 9　モジュール単位の開発の独立性確保

4.2.6　DB 開発

DB 開発では，ANACore で取り扱うデータをデータベースで管理するための設計/実装を行った．標準化/DB チームが開発を担当し，図 10 で示すように，論理 ER 図の作成，物理 ER 図の作成，各種データベースオブジェクトの定義書作成，実装，という流れで開発した．本開発の主な成果物を表 10 に挙げる．論理 ER 図の作成では，業務処理で取り扱うデータ項目を定義したテーブルと，その関係を定義した．物理 ER 図の作成では，論理 ER 図にシステム要件やパフォーマンスも加味し，実際に作成するテーブル項目や関連を定義した．各種データベースオブジェクトの定義書の作成ではテーブルやインデックスなどのデータベースオブジェクトを作成するために必要な情報を定義した．ER 図や各種データベースオブジェクト定義書の作成では，ツールとして ERWin を利用した．

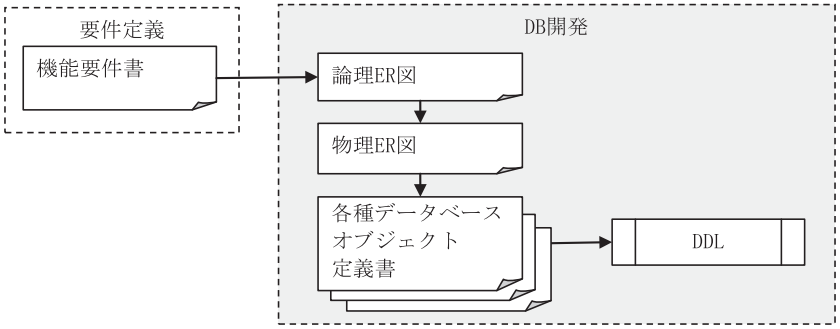

図 10　DB 開発作業

表 10　DB 開発の主な成果物

成果物名称	説明
論理ER図	ANACoreで取り扱う業務データとその関係を定義した設計書. 業務チーム単位，及び業務に依存しない共通機能で1篇ずつ作成した.
物理ER図	論理ER図にシステム要件やパフォーマンス要件等を追加した，実際のテーブル定義した設計書. 業務チーム単位，及び業務に依存しない共通機能で1篇ずつ作成した.
各種データベースオブジェクト定義書	テーブルやビュー，インデックスなどのデータベースオブジェクトの作成に必要な情報を定義した設計書. データベースオブジェクトの単位に作成した.
DDL	実際のデータベースオブジェクトを作成するためのDDL文. ERWinを利用し，各種データベースオブジェクト定義書から自動生成した.

なお DB 開発は標準化/DB チームで作業するため，業務チームが担当するサービス開発の過程でデータ項目やスキーマ構造に不足や扱いづらい個所が出てくるケースがある．その場合も適宜業務チームから DB 変更依頼を受け付け，標準化/DB チームで一元的に対応することで，統一感のあるスキーマを保つようにした．

4.3　テスト時の考慮

ANACore 開発では，テストは開発者自身の環境で行う単体/サービス結合テスト，その後に専用の機能テスト環境を用いる機能テストの順に実施した．

単体/サービス結合テストは，サービス開発，CUI AP 開発，GUI 開発，他シス AP 開発，バッチ開発の各開発工程の中で実施し，プログラム単体，あるいはいくつかの関連するプログラムを結合させてテストした．テスト時に利用する範囲はあくまでも自チーム内で作成された範囲とし，チームの独立性を確保した．

機能テストは，ANACore 機能単位のテストであり，該当機能に関するプログラムがすべて連携した状態でのテストを，開発チームとは別のテスト用のチームを準備し実施した．機能テストでは多数のテストを多数のチームで並行して実行可能とするため，機能テスト環境も多数準備した．また，接続経路に合わせて CUI 機能用に CUI テストツール，外部システム接続用に ES テストツールといったテスト専用ツールを作成し，実端末や外部システムと接続せずにテストできる環境を用意した．

加えて，最新のソースコードから自動でライブラリを作成し，それを毎朝ライブラリアンが各テスト環境にリリースする仕組みや，テストデータを初期化する仕組みを作り，テスト担当者がテストの実施に専念できるようにした（図 11）．

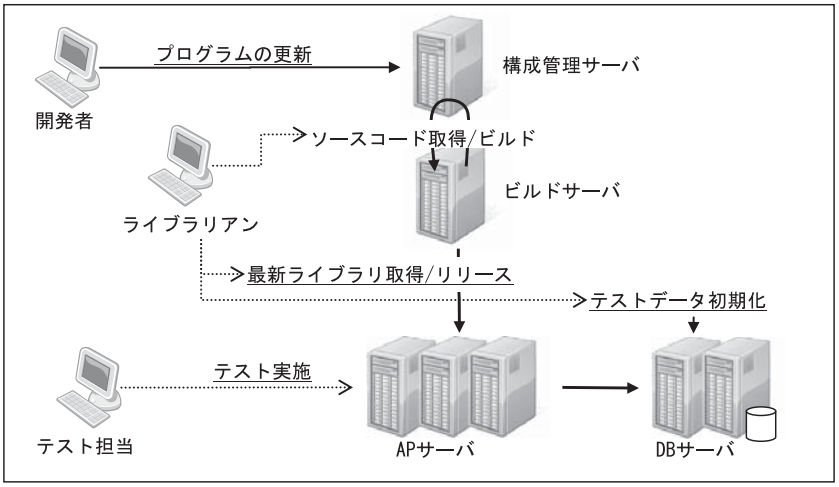

図 11　機能テスト実施環境

5.　品質面の考慮

実装段階での品質を担保するため，各開発者は実装の都度，コーディング規約の遵守性，バグ潜在の可能性を CheckStyle[*6]，FindBugs[*7] を利用してチェックした．この開発プロセスの一部として実施される品質チェック以外に，構成管理にチェックインされたソースコードに対して同様の品質チェックを自動で行い，結果をメール周知する仕組みを作り，チェック漏れの防止に努めた．

またプロジェクトの終盤ではテストで検出された障害対応によるコード修正が多くなるため，新規開発とは異なるいくつかの観点（表 11）でソースコードやジョブネットをチェックするツールを作成し，日次での品質チェックを実施した．これは専任チームとして立ち上げ，チェック結果に対し対応の要否を確認し，対応要項目について修正状況を管理することで，開発チームの負荷を増やすことなく品質確保の体制を作った．

表 11　主な品質検査の観点

品質検査観点	説明
DBリソース解放の順守	DBアクセスの実装において，例外の発生も考慮して正しくリソースの解放処理ができているかを確認する．
派生トランザクションの妥当性	一つのトランザクションから別トランザクションを起動する場合，例外発生時に起動元と起動先の処理で不整合が発生しないことを確認する．
排他制御の適切な使用	不必要なテーブルロックを行っていないか確認する．
コンポーネントの呼び出し関係の順守	ANACoreアーキテクチャで規定するコンポーネントについて，コンポーネント間の呼び出し関係がコーディング規約を順守しているか確認する．
大量データ取得によるメモリリソース圧迫の懸念	DB検索時に大量のデータを検索し，メモリに展開している処理がないか確認する．
適切な例外処理	例外発生時に不必要に例外を上位の呼び出し元に返さないようにしているか，上位に例外を返す際に下位で発生した例外情報を落としてしまったりしていないか確認する．
複雑なSQLの確認	多数のテーブル結合やROWNUMを利用するようなSQLにおいて，意図した処理を行うSQLになっているか，またパフォーマンスの懸念のあるSQLがないか等を確認する．
ジョブネット，シェル整合性	シェル一覧から自動生成するシェルとそのシェルを呼び出すジョブネット(ジョブ)の整合性が取れているか確認する．
適切なジョブネット設定	スケジュール設定や保留設定などのジョブネット設定が適切に行われているか確認する．
適切な判定ジョブの利用	判定ジョブの利用箇所が，先行ジョブや従属ジョブの設定が意図した処理フローと整合しているか確認する．

6. お わ り に

ANACore は，2013 年 2 月の本番稼働後，安定した運用を続けており，2013 年 10 月現在は保守運用フェーズに入っている．ただし新規案件の開発要求も続々と出てきており，今もなお相当規模の開発作業が続いている．本稿で説明した内容は開発当初に用意した開発環境や開発プロセスであるため，保守フェーズや拡張開発においては一部重厚長大と思える部分もある．今後は保守運用フェーズ以降の開発について，無駄なく，そつなく，軽量化された開発手法を定義できればと考えている．

＊ 1　ANACore は，米国 Unisys 社のエアラインパッケージ AirCore をベースに開発した，全日本空輸株式会社における新国内線旅客システムの開発コード名である．
＊ 2　CUI は，character user interface の略で，ここでは WEB ブラウザとやり取りする GUI 開発と区別し，空港設置の端末や自動機などからの電文による入力と応答を指す．
＊ 3　他シスは，外部システム連携を表わす ANACore 開発プロジェクト固有の略語である．
＊ 4　AirCore は，米国 Unisys 社が開発した航空業界向けの各種システムのパッケージである．
＊ 5　Selenium は，web ブラウザを使って web アプリケーションを自動的にテストするオープンソースのツールである．
＊ 6　CheckStyle は，コーディング規約への準拠を確認するオープンソースツールである．
＊ 7　FindBugs は，バグのパターンをチェックするオープンソースツールである．

執筆者紹介　小笠原　一洋（Kazuhiro Ogasawara）

1997年日本ユニシス(株)入社．いくつかのJava系開発プロジェクト，製品主管業務を経て2007年よりANACore開発プロジェクトに参加．標準化チームに所属し，開発プロセスの整備やバッチ機能に関するフレームワークの開発に従事．