WEB システムにおける SCM システム開発事例

富士ソフトＡＢＣ 株式会社

■ 執筆者Ｐｒｏｆｉｌｅ ■

1997 年　富士ソフトＡＢＣ株式会社　入社
2003 年　現在
システム事業本部 大阪事業所　所属

市　原　宏　一

1998 年　富士ソフトＡＢＣ株式会社　入社
2003 年　現在
システム事業本部 大阪事業所　所属

天　埜　哲　也

■ 論文要旨 ■

SCM（サプライチェーンマネジメント）とは，供給者から消費者までを結ぶ，調達・製造・配送・販売の一連の業務（サプライチェーン）を統合し，業務効率を改善する経営戦略である．

本論文は，サプライチェーンに関する情報を統合するために電子化を行い，それらの情報を登録／照会する WEB システムを導入したシステムの開発事例である．

情報の電子化に際しては，フォーマット化された数値・文字情報だけではなく，画像情報・Excel ファイルなどの付加情報も関連づけ，情報の多様性を持たせた．

また，WEB システムの構築に関しては，J2EE アプリケーションサーバを導入し，アプリケーション開発では各機能をオブジェクト単位に分割しスムーズな開発を行った．

■ 論文目次 ■

■ 図表一覧 ■

１．はじめに

１．１　当社概要

当社は 1970 年の創立以来，独立系ソフトハウスとして，幅広い分野においてソフトウェアの受託開発を行ってきた．

今年，流通システム事業本部を設立し，流通分野のシステム開発に積極的に取り組んでいる．

１．２　SCM（Supply Chain Management）

SCM（Supply Chain Managemant）とは，メーカ，原材料サプライヤなどの供給者から，卸店，小売店を結ぶ，調達・製造・配送・販売の一連の業務を統合し，業務効率を改善する取り組みを指す．

本論文において，弊社で開発を行った某大手小売業者の SCM システム開発事例について解説する．

１．３　SCM システムの目的

開発事例である SCM システムの特徴についてを説明する．

スーパーマーケットなどの食品小売店においては，毎日売れそうな商品を売れそうな数量を仕入れてその日のうちに売り切ることが重要である．つまり仕入れ数と販売数を一致させ，売れ残りによるロスを防ぐ必要がある．

しかし，売り切ればよいというものではなく．閉店までまだ時間があるというのに，陳列棚からそうそうに商品が消えていては問題である．つまり販売機会の損失を防ぐ必要がある．また顧客の「商品の選択ができる」，「目当ての商品がある」といった顧客満足も満たさなければならない．そうしなければ，次にこの顧客はそのスーパーマーケットに来てくれなくなるかもしれないからだ．この仕入れ数と販売数のバランスは，スーパーマーケットだけでなくあらゆる小売業に共通する命題である．

この命題は，売り場の担当者の経験と勘だけで実現することは難しい．以前にも増して，消費者のニーズは多様化しているため，同じ商品を数多く並べるだけではなく，充実した品揃えの商品を適量保持する必要に迫られている．

また，近年，商品のライフサイクルも短くなり，商品ごとの需要も激しく変動するため，需要予測はますます難しいものとなっている．そこで小売店や卸店，メーカ，原材料サプライヤなどの企業が協力して，理想的な仕入れ数と販売数を実現させるために情報システムを構築する必要が出てくるのである．

ではどのようにすれば，この理想を実現してくれるシステムを構築することができるのだろうか．まず小売店の理想的な仕入れ数と販売数を考えると，品切れしないぎりぎりの商品の在庫数を持つことである．正確な販売予測を立てることと，それに従った必要最小限の在庫を保持できるよう仕入れを管理する必要がある．

仕入れの面から考えると，自社の在庫を最適にするためにはシステムにより販売実績を把握し，細かな販売予測をたて，それにしたがって卸店に少しずつ発注をかけて，すばやく納品してもらうことが必要である．だが，これを実現するには卸店もまた，商品の最適な在庫を持つことが必要となる．小売店の要求に応えるため，大量の在庫を抱えることに

よる，経営の悪化を防ぐため，卸店も，商品別の販売実績や，発注予測を立てて最適な在庫を保つことが必要となる．卸店に商品を納入するメーカもまた同様である．つまり小売店が最適な在庫を保つためには，卸店，メーカ，原材料サプライヤを「繋ぐ」（Supply Chain）ための情報連携が必要なのである．

その上で，生産計画や物流計画をたてていく必要がある．

本 SCM システムでは，小売店が主導で開発を行う形となった．

また，小売店で販売する惣菜，弁当などの管理がターゲットとなっており，このシステムにおけるサプライチェーンは以下のようになる．

小売店←（惣菜，弁当）←製造ベンダ←（原材料）←卸店←（原材料）←原材料サプライヤ

また本システムにおける小売店のターゲットは以下のとおりであった．

・仕入先も含めた全体のロス・在庫圧縮

WEB 画面により仕入先は，店舗の販売動向情報を得ることができ，在庫の適正化を図ることができる．

販売実績に基いて商品別の需要予測・原材料発注を行い，安全在庫の圧縮を図る．これまで販売実績情報などの共有化は行われておらず，各製造ベンダ，卸店などは独自の方法による在庫管理や需要予測をおこなっていた．

・各種マスタの電子化とマスタの仕入先との共有

各種マスタの重複管理をなくし，マスタ管理事務作業などの間接作業の効率化を図る．

これまで顧客では，紙ベースの商品マスタ管理を行っており，関連企業へは FAX などで送付し，それぞれで管理する仕組みとなっていた．

・ローコストでのオンタイムの商品供給発注業務の統一を行う．

従来は，各卸店，原材料サプライヤごとに発注システムが異なっており，製造ベンダは各卸店ごとにそれぞれの仕組み(FAX を含む)での発注を行っていた．

２．システム要件

各企業との情報共有が目的であることと，導入が容易であることから WEB システムでのシステム構築となった．

客先本部に WEB システムを構築し，各関連企業はそこにアクセスすることによってマスタ情報や，取引先の在庫情報を参照することができる仕組みとした．また，各企業間の情報連携にはファイル転送用ソフトウェアを導入し在庫情報，廃棄，返品情報を各企業から小売店本部に集約して一括管理し，またマスタ情報や実績情報を小売店本部から各企業に配信する仕組みを構築した．

２．１　システムの課題

本システムは，惣菜・弁当を扱うシステムであるため，マスタ管理，とりわけ商品マスタが非常に複雑となることは避けられなかった．惣菜，弁当を製造するための製造工程，(原材料)レシピ情報，各工程での歩留まり，添加物，アレルゲン情報，商品貼付ラベル表示情報も保持する必要がある．また，出来上がり商品の画像情報も保持する必要があった．また店舗を全国展開しているため，地域によって同じ商品でも味付けが異なるなど商品仕様の差異を許容する必要があった．

また，商品のマイナーチェンジを行った場合でも，別商品とならないように世代も意識した作りとする必要があった．

商品マスタの中では，商品のレシピ情報の電子化が難しいものであった．WEB ブラウザ上への登録では，盛付け例などは表現が難しく，また商品によっても情報量に違いがあった．本 SCM システムではエクセル上に，レシピ情報を入力し，そのファイルをアップロードすることで登録を行っている．WEB ブラウザ上でエクセルを表示させることで対応した．また画像情報も同様にアップロードにより登録としている．

ユーザーはまた，このエクセルシートをダウンロードして参照可能にしている．

３．SCM システムの開発について

本 SCM システム開発では，利用者側の GUI 処理部，他システムとの通信処理部，スケジューリング化された業務バッチ処理部などがあるが，本論文では，利用者側の GUI 処理部として WEB システムを適用したシステム開発事例について解説する．

開発作業をスムーズに進め，設計工程から共通処理の作成作業効率を高めるために，画面遷移に特殊化した共通フレームワークを作成する．また，各業務機能から使用する共通処理関数として，DB 操作関数，ログ出力関数，文字チェック関数，HTML 制御文字変換関数を作成した事について詳細に解説する．

その他の設計内容については，特に重要と思われるポイントを箇条書きにて記述する．

３．１　SCM システム基本構成

３．１．１　システム基本構成

SCM システムでは，現在，主流となっている３層クライアントサーバシステム構成を採用する．３層クライアントサーバシステム構成を**図１**に示す．

従来のシステム開発では，２層クライアントサーバシステムを採用していた．

しかし，この方式ではクライアントからの要求やデータベースからデータの取得などの処理をすべてクライアント・サーバ間のネットワーク上で行うこととなり，ネットワークトラフィックが高負荷になる問題が発生していた．

また，セキュリティなども問題となっていた．

現在，３層クライアントサーバシステムは，これらの問題を改善するシステム構成の主流となっている．

３層とは，クライアントの GUI 関連処理のみをプレゼンテーション層，業務処理を行うファンクション層，データベースシステム関連処理を行うデータ層から構成される．

サーバ側の処理をファンクション層とデータ層に分割することで，各機能の処理を分散することが可能となる．また，ファンクション層とデータ層間の接続に冗長性を持たせることで柔軟な機能変更も可能となっている．

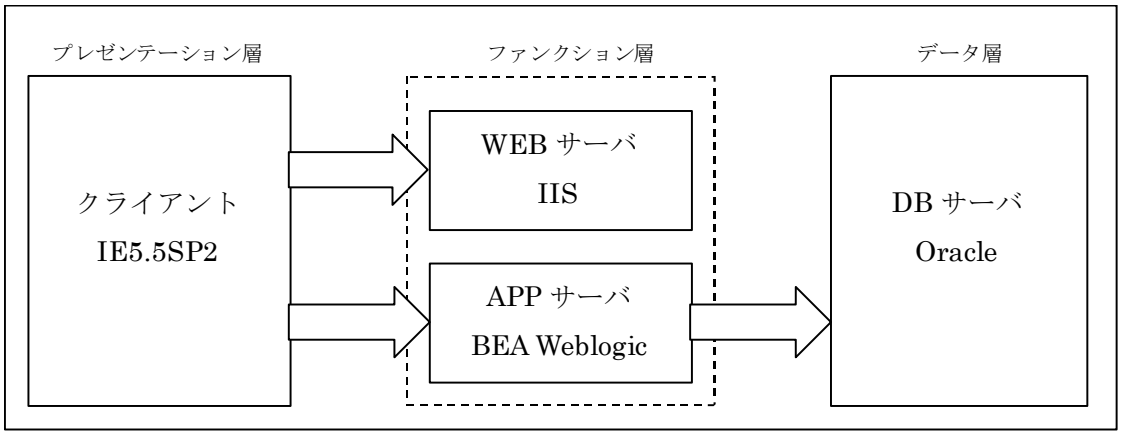

図１　３層クライアントサーバシステム構成

３．１．２　ネットワーク構成

３層クライアントサーバシステム構成を考慮したネットワーク構成を**図２**に示す.

ネットワークの特徴としては，セキュリティ面を考慮し，外部公開サーバを DMZ（DeMilitarized Zone）セグメントに配置する．DMZ を取り入れたネットワーク構成を取ることで，外部使用者／社内利用者／公開サーバのいずれからもファイアウォールを通過することになり，詳細なアクセスの制御を行うことが可能となる.

外部公開 WEB/APP サーバについては，ロードバランサを配置し，負荷分散を行う.

DB サーバについては，２台によるクラスタリングを行う．クラスタリングを行うことで，負荷分散を行い，且つ，障害時の対応を容易に行うことが可能となる.

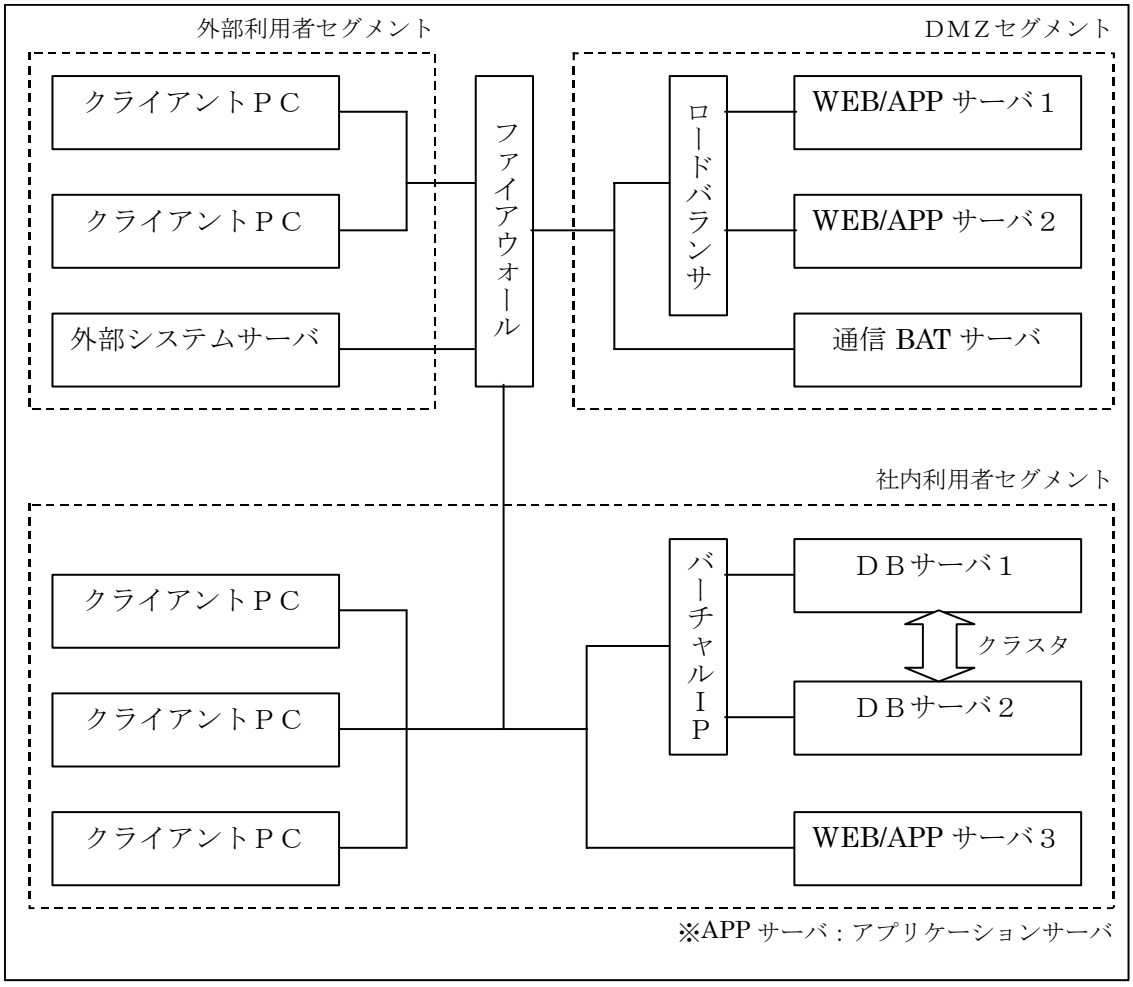

図２　ネットワークシステム構成

３．２　SCM システムの設計方針

SCM 業務の電子化に当たり，現状の業務フローの作成，ユーザへのヒアリングを行った結果（要件定義書）を基に機能設計・詳細設計を行う.

設計の流れを**図３**に示す.

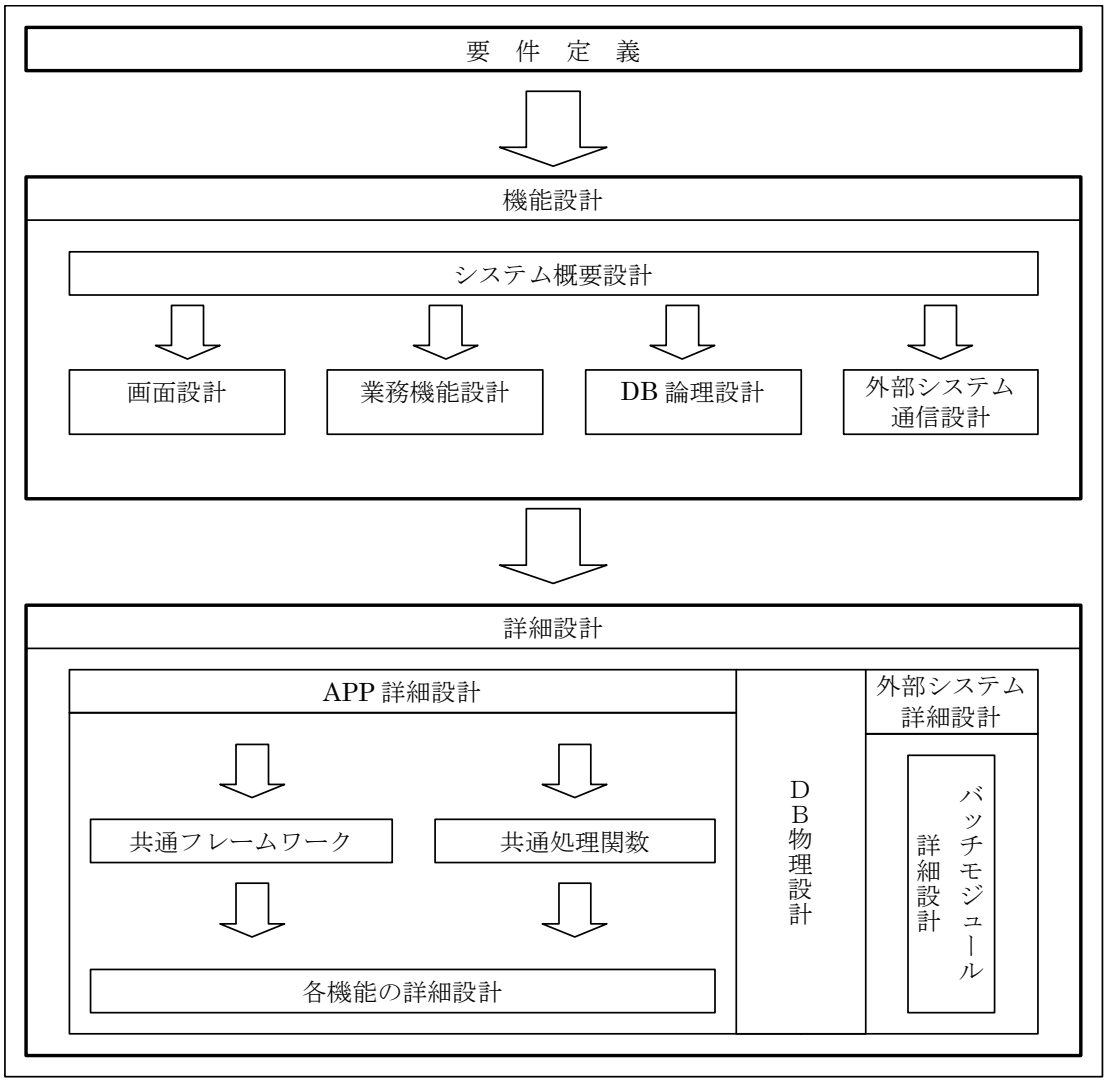

図３　設計フロー

３．３　機能設計

機能設計では，システム全体に対するシステム概要設計を行う．

システム概要設計完了後に，設計内容を３層クライアントサーバシステム構成に合わせ，画面設計，業務機能設計，データベース設計に分割し，更に詳細な設計を行う．

（１）システム概要設計

要件定義でまとめられた機能を業務機能の体系化を行い，開発する業務機能の最小単位を明確にする．

また，システム全体の概要を把握するために，次のドキュメントを作成する．

・システム全体を表記したコンテキストダイアグラム
・データフローダイアグラム（レベル０）
・他システムをまとめた外部エンティティ一覧表
・システム利用者をまとめたプレイヤ一覧表

（２）業務機能の体系化

要件定義でまとめた業務機能を，サブシステムという分類体系にまとめる．

サブシステムとしては，業務の基本データを登録／照会する機能，基本データから作成される業務データを登録／照会する機能，過去の業務データを照会する機能などで分類する．また，システム内の各機能を業務単位に分割する．

（３）コンテキストダイアグラム

本開発対象システムを中心として，外部とのアクセス要素を図形化して表記する．

外部とのアクセス要素を明確にする．

（４）データフローダイアグラム（レベル０）

SCM システムと他システム間のデータの流れを把握するために作成する．

（５）外部エンティティ一覧表

他システムをエンティティとして一覧にまとめる．

（６）プレイヤ一覧表

業務機能を扱うユーザを業務権限ごとで分類する．作成した利用者分類をプレイヤグループとし，グループが一意となる ID を割り当てプレイヤリストを作成する．

業務機能を利用するユーザをグループ単位で管理する．

３．３．１　画面設計

画面設計では，画面共通仕様書を作成する．

・画面の共通レイアウト

　画面の表示色，文字フォント，文字フォントサイズ，タイトル表示フォーマット，ボタンの表示位置，検索条件の表示，検索結果の表示イメージ，エラーメッセージなどの項目をまとめる．

・画面遷移の共通仕様．

・SCM システムで扱う文字コードの決定．

３．３．２　業務機能設計

各業務機能について，次の項目について設計を行う．

・業務機能の目的

・業務機能を利用するプレイヤ

・プレイヤごとによる業務機能の制限内容

・各業務機能画面の画面遷移

・業務機能画面のレイアウト

・業務機能画面の入力項目

（特に入力文字サイズ，入力文字種別を明確にする）

・業務機能の実行処理の概要説明

・業務機能で表示するメッセージ

３．３．３　DB 論理設計

DB 論理設計を行う．

・各業務機能単位でデータの流れを把握するために，フローダイアグラムを作成．

・各エンティティのデータ項目定義書を作成．

・論理 ERD を作成．

３．４　詳細設計

J2EE プラットフォームにて WEB システムを実現するに当たり，一般的な J2EE アーキテクチャである Model-View-Controller(MVC)アーキテクチャを適用した.

SCM システムでは，MVC の Controller 部についての処理テンプレート（共通フレームワークとする.）を作成する.（MVC アーキテクチャを**図４**に示す.）

各業務機能は，共通フレームワークを使用することで，各業務に特化した処理の実装に専念することができ，作業効率を高めることができる.

また，SCM システムで使用する共通処理関数も提供することで，処理の統一化と作業効率の向上を行う.

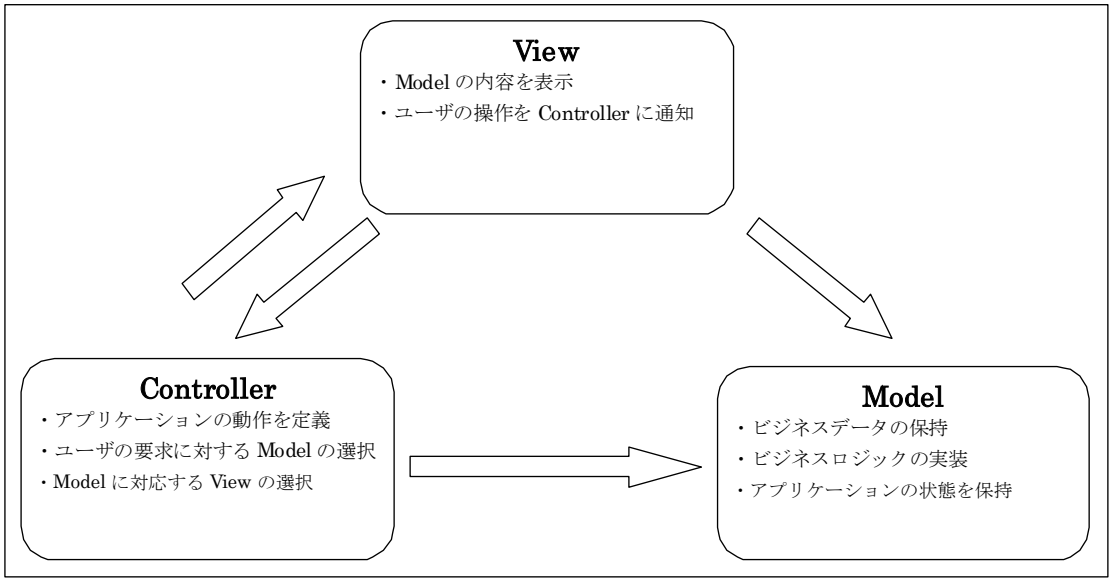

図４　Model-View-Controller(MVC)アーキテクチャ

３．４．１　共通フレームワークの設計

共通フレームワークは，SCM システムの共通機能として，下記の機能を実装する.

・画面の遷移制御
・セキュリティ管理
・アクセス管理
・エラーメッセージ表示機能

各業務機能の開発者は，ビジネスロジックと Model の内容表示処理のみを実装するだけで良くなり，作業効率を高めることが可能となる.

実際に SCM システム開発では，最も難易度の高かった商品レシピ管理機能についても，ビジネスロジックの実装に専念することができたため，作業効率を上げることができた.

共通フレームワークの機能シーケンス（**図５**）を基に処理内容を解説する.
共通フレームワークは，View からの操作内容の通知を受け取る.
その際に SCM システム（アプリケーション）の Controller として，下記の動作を行う.

（１）画面からの操作要求に対して，セキュリティチェック，アクセスチェックを行う．（セキュリティ管理機能，アクセス管理機能の実行）
（２）View に対応する Model の選択を行う.
（３）選択した Model のビジネスロジックを実行する.
（４）ビジネスロジックの実行結果に対応する View を選択する.
（２－４：画面遷移制御機能）
（５）ビジネスロジックの実行でエラーが発生した場合，エラーメッセージを表示する．（エラーメッセージ表示機能）

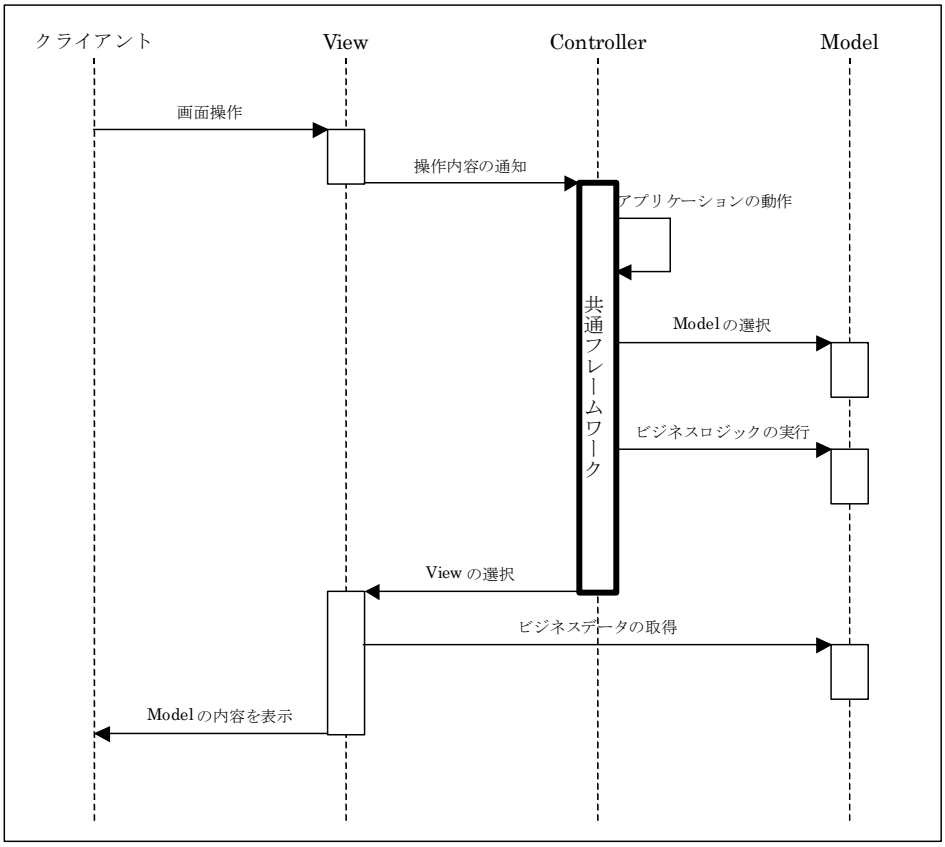

図５　共通フレームワークの機能シーケンス

３．４．２　デザインパターンの適用について

共通フレームワークには，View の操作内容の通知より対応する Model を選択する処理がある．この機能を実現するためにデザインパターンで定義されている Command パターンを適用する．

３．４．３　共通処理関数の設計

共通処理関数として次の５つの関数を共用化した．

（１）システムの入力可能文字チェック関数

・SCM システムでの入力禁止文字チェック関数
・数文字列チェック関数
・数値チェック関数
・全角文字入力チェック関数
・全角カタカナチェック関数

（２）HTML 制御文字変換関数

HTML 制御文字変換関数とは，文字列を WEB クライアント上で表示できる文字に変換する関数である．具体的には，＜，＞，＠，“といった文字列については，そのままでは正しく WEB クライアント上に表示することができない．

本関数を使用することで，HTML 制御文字変換一覧表（**表１**）に該当する文字列は変換処理が行われる．

SCM システムでは，プログラミング開始前に HTML 制御文字変換数についての説明を行い，プログラマに変換処理の必要性と変換用の共通関数の存在を意識させることで，事前に問題回避を行った．

表１　HTML 制御文字変換一覧表

変換前の文字	変換後の文字
＜	<
＞	>
“	"
&	&
半角空白	
¥	¥

（３）DB 操作文字変換関数

DB 操作文字列変換関数とは，文字中に’（シングルクォート）が含まれている場合，’’（シングルクォート）を重ねるように変換処理を行う関数である．

（４）DB 接続共通機能

SCM システムでは DB 操作情報としてログ出力を行う要望があったため，SQL 文をログ出力する必要があった．DB 接続共通機能にログ出力機能を実装することで，DB を操作した内容をログとして出力可能とした．

（5）ログ出力機能

ログ出力機能は，インフォメーションログ出力，エラーログ出力，デバックログ出力を行うことができる．

３．４．４　各機能の詳細設計

各機能の詳細設計では，共通フレームワーク，共通処理関数を使用することでビジネスロジックと Model の内容表示処理の詳細設計に専念することができる．

３．５　単体テストツールの活用

本開発においては，各業務機能のプログラミング開始前に共通処理関数をライブラリとして提供している．そのため，各業務機能の開発者からの要望などで共通処理関数に修正が入ることがあり，その場合は早急に対応する必要がある．

しかし，機能の追加・変更ではソース修正に必要な時間以外に既存処理の動作テストが必要になり，修正後のモジュールをリリースするには時間がかかっていた．

これを早急に対応するために共通処理関数ライブラリでは単体テストを自動実行することができるツールを利用している．そのため，修正後のモジュールをリリースするのに必要な時間は，モジュールの修正と自動実行した単体テストの結果検証にかかる時間のみとなり，早期に修正後のモジュールを提供することが可能となっている．

４．今後の課題

SCM システム開発では，作業効率を高めるために，画面遷移に特化した共通フレームワークを作成し，また，共通処理関数をライブラリとして各業務機能開発者に提供した．

更なる作業効率を高めるためには，共通処理のライブラリ化や処理テンプレートを導入する必要がある．フレームワークについては様々なタイプのものが存在するが，データの表示処理に特化したフレームワークも存在する．本 SCM システム開発では View に関するフレームワークを導入していないため，これを採用することで更なる作業効率を高めることができる．

また，Java の開発効率を高める別の方法としては，高機能な統合環境の導入があげられる．コンパイル・デバック機能やソースエディタ機能での入力文字の補完機能，コンパイルエラー行へのジャンプ処理などがある統合環境を導入することで，更なる作業効率アップが期待できる．

今後の課題としては，開発効率向上のため，ツール・ライブラリ導入を積極的に行い，常に最新の技術動向に目を向けて，いかに情報を収集することが重要であるかを認識し，実践するつもりである．