エンドユーザ主導の Web アプリケーション開発方式の試作
－ビジネスロジックの定義と実装方式－

許杰 †　　中所武司 ‡
明治大学大学院理工学研究科　ソフトウェア工学研究室
E-mail：　†133525225@qq.com　‡chusho@cs.meiji.ac.jp

あらまし　近年のインターネットの普及により、様々な業務が Web アプリケーション化されている。それに伴い、Web アプリケーションを効率的に開発するための基盤として、MVC モデルとストアドプロシージャを使うことが基本的な技法となっている。本研究では、外注する予算の確保が難しい中小規模の Web アプリケーションを対象として、サービスの頻繁な変化にも対応できるようなエンドユーザ主導型開発のためのビジネスロジックの実装方式を提案する。エンドユーザである業務専門家が、自らの有する業務知識に基づき、Web アプリケーションを簡単に構築できるようにすることをめざしている。本稿では、エンドユーザによる Web アプリケーション開発技法の研究の一環として, ビジネスロジックの実装方式に関する研究アプローチとその試作について報告する。

キーワード エンドユーザ、Web アプリケーション、ビジネスロジック、MVC モデル

End-User-Initiative Development for Web Application
－The Implementation of Business Logic－

Jie　Xu †　　Takeshi　Chusho ‡
Software Engineering Laboratory,Graduate School of Science and Technology,Meiji University
E-mail：†133525225@qq.com　‡chusho@cs.meiji.ac.jp

Abstract In recent years,　many web applications are used in various business fields.　These web applications are often developed by the MVC model and stored procedures with reusable components. Furthermore, it is increasingly required that business professionals build their web application by themselves. In this paper, the solution about the implementation of business logic for end-user-initiative development is proposed to support small and medium businesses. Then, a visual tool for the end-user is built.

Keyword End-User, Web Application, business, MVC model

1.　はじめに

今、インターネット上で Web アプリケーションが普及し、クラウドコンピューティグが注目されるなど、ソフトウェアのサービス化が促進されている。エンドユーザになじみのある Web ブラウザをユーザインタフェースとする Web アプリケーションに関して、我々の過去の研究では、コンポーネントベースの技術としてのアプリケーションフレームワークとビジュアルモデリング技術を用いて、ユーザインタフェースおよび比較的簡単なデータベースを構築する方法を実現してきた。一方、ビジネスロジックに関しては、頻繁な変更を伴う保守に対応することが重要になるという観点から、試行錯誤的に種々の実装方式を研究してきた。

2.　エンドユーザ主導開発

エンドユーザ主導開発は、ホストコンピュータを用いた定型業務処理から知的創造業務支援の利用形態へ変化する中で、そのニーズや重要度が高まっている。業務の専門家が自らソフトを開発し、保守を行うことにより、コストの削減や開発リスクの減少などのメリットがもたらされる。これまでの研究から、ビジネスロジックをアプリケーションのユーザインタフェースやデータベース管理

から独立させることはとても大切になっていることがわかる。基本的なアプローチを図 1 に示す。UI、ビジネスロジック、DBを基本的な構成要素とするアーキテクチャを前提にアプリケーションを作成する。

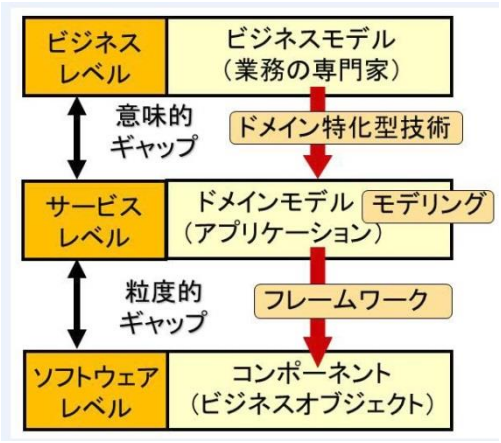

図 1. エンドユーザ主導開発のアプローチ

本研究を通じてのエンドユーザ主導開発の主要な課題として、多様なビジネスロジックの定義方法とその実装方式があげられる。これまで、スクリプト言語の開発、ルール表現の導入などを試みてきたが、プログラミングの概念は必要である。最近では、特定分野対応のタイルテンプレートを組み合わせて業務処理を記述する方法で問題の解決を図ったが、十分なテンプレートの集合を用意するという課題はある。本報告では、データベースのストアドプロシージャ技術と関連付けたビジネスロジックの実装方式について述べる。

3. 2 種類の基本的アーキテクチャ

3.1 例題の概要

今回、ビジネスロジック分析のため、不用品再利用システムを例題としてとりあげ、C＃で試作した。昨今、情報技術(IT)を応用して持続可能な社会のための環境保護に貢献する(Green-by-IT)が期待されており、地方自治体が運営する地域住民のための不用品再利用サービスやローカルに運営される不用品再利用のための中古物品の販売店に関して、もしその担当者自身が Web サイトとして立ち上げることができれば、大きな効果が期待できる。この例題の構築方式としては、従来方式と MVC モデル方式を検討した。図 2 は不用品再利用システムの画面遷移図であり、以下の機能を含む。

(1)メールアドレスにおける本人チェック
(2)ユーザ登録及び情報変更
(3)不用品の登録及び更新
(4)不用品の検索及び申請
(5)受け取り希望者の決定処理

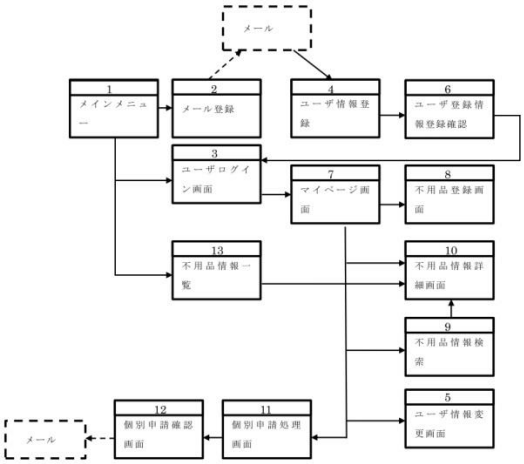

図 2. 不用品再利用システムの画面遷移図

3.2 基本設計

従来方式は、ASP.NET の開発環境における 3 層アーキテクチャを採用しており、最も典型的な構築方式である。図 3 は従来方式を利用したシステムの構成図である。ASP.NET では、HTML に出力する GUI 要素とページの制約処理は別々のファイルに分割することが推奨されている。

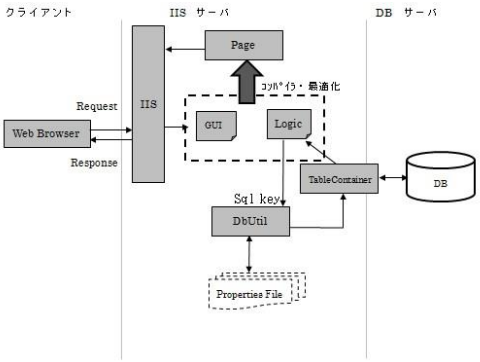

図 3．システム構成図(従来方式)

3.3 MVC モデルによるシステムの構築

近年、Web アプリケーションの開発技術が発展

してきて、従来方式の ASP.NET Web Forms に代わり、ASP.NET MVC モデルをベースにした Web 開発が盛んになってきた。MVC モデルベースの不用品再利用システムのシステム構成図を図４に示す。

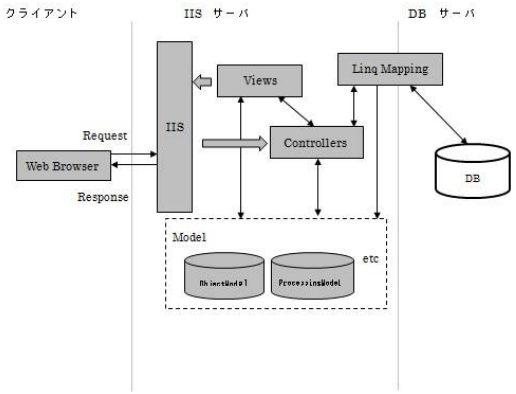

図４．システム構成図(Asp.Net MVC)

3.4 ビジネスロジックの考察

不用品再利用システムにおいては、不用品の申請に対する受付処理があり、流れ図は図 5 のようになる。この例からも、判断が多くなるとビジネスロジックの複雑さも高まっていくことがわかる。

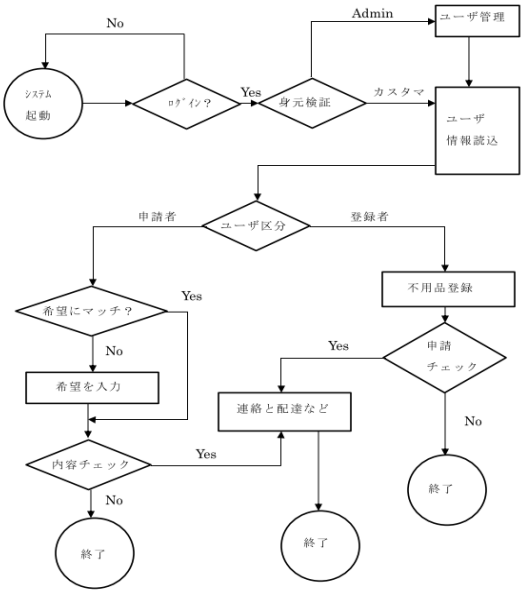

図 5. 申請及び確定の仕組

3.5 ビジネスロジックの構造化

ビジネスロジックは「ビジネスモデル」が変われば、幾らでも作り出せる。だが、ビジネスロジックの構造が定められたら、どんな変化にもよらず、その構造に従う必要がある。

ビジネスロジックにはダイナミックなビジネスロジックとスタティックなビジネスロジックがある。ダイナミックなビジネスロジックとは他の処理結果に依存した処理を行うロジックである。スタティックなビジネスロジックはほかの要素や処理などに関係なく、常に存在しているロジックであり、条件チェックと判断処理の場合が多い。

この二つの概念でビジネスロジックは二つタイプに分けられて其々の実行も違う。ダイナミックなビジネスロジックは目標明確で特定的な処理で選択的に実行されることが多い。スタティックなビジネスロジックの結果は真と偽しかないのが特徴でチェック処理などを指している。本研究で取り出した例はスタティックなビジネスロジックである。

4. ストアドプロシージャによる記述

以上に述べてきた２種類のシステムの構成に関して、エンドユーザによるビジネスロジックの定義を支援するという観点でみると、ビジネスのロジック処理は、従来方式と MVC モデル方式のいずれの場合も、ビジネスロジックの処理はあらゆるところに含まれている。

本研究においては、ビジネスロジックを反映しやすいデータベースの処理に注目し、データベースプログラミングを１つの支援方式として取り上げる。その代表的な技術はストアドプロシージャである。ストアドプロシージャの利点は、頻繁に DB のアクセスを行う処理から解放することである。手続き型ステートメントとセット指向ステートメントの両方を組み合わせたものであり、極めて複雑なビジネスルールにも対応できる。複雑すぎて他の制約では実現できなかったものや、セット指向的な処理が混在しているものなども、ストアドプロシージャの形で実装できる。

4.1 典型的なビジネスロジックの説明

例題として取り上げた不用品再利用システムでのビジネスロジックの定義については、システムの利用者や取り扱う対象や処理の制約などをサービス提供者（エンドーユーザ）が検討していくものである。ここでは、人やシステムに対して特定の条件下でアクションを引き起こすような、「もし～ならば～する」と表現される典型的なビジネス

ロジックとして、次の３つを取り上げる。

(1)もし、登録者がすでに不用品登録数の上限(例えば 5 個)の不用品を登録していれば登録を断る。

(2)もし、一定時間経過後にある品物を希望するユーザが複数いれば､リストから当選者を決める処理を行う。

(3)もし、不用品登録後、一定期間(例えば 3 か月)経過したものがあれば削除処理を行う。

この三つの例について、まずはプログラムによる実装方式を考えて見よう。一番目の例はユーザの登録制限であり、プログラムの処理においては、まずそのユーザに関連したデータをデータベースから取り出して登録数をチェックし、５個以上の場合は登録不可の処理を行い、５個未満の場合は登録処理を行う。DB のアクセスはステップの数と比例している。その次の 2 番目と 3 番目の例はデータベースの一括処理を要求していると考えられるので、実際にプログラムの実装手順は其々以下の通りである。

希望者のリストから当選者を決める処理

STEP1:希望者のリストを作る

STEP2:ランダム関数を利用して当選者を決める

STEP3:当選者に送信する、当選者以外の人にも送信する

STEP4:当選者の申請情報を更新、当選者以外の人の申請情報を削除する

不用品登録後、一定期間(例えば 3 か月)経過したものの削除処理は次のようになる。

STEP1:登録日付をチェックし、削除リストを作る

STEP2:削除リストを確認し、削除処理を行う。

ステップ毎に DB アクセスを行う必要があるので、ビジネスロジックによるステップ数が増えれば増えるほど、DB アクセスの頻度も上がっていく。そこで、ビジネスロジックを反映するビジネスルールの処理をデータベースに持っていった場合について、次に検討する。

4.2 ストアドプロシージャ

ストアドプロシージャを利用してビジネスロジックを実現する。4.1 に示した提案の一番目は DB 側に持って行くと、その実装内容は以下のようになる。

実装の例:

```
ALTER PROCEDURE
  [dbo].[ap_CheckNumberGoodsbyUser]
  @count INT = 0、
  @user_No varchar(50)、
  @IsAlowed Bit OUTPUT
AS
BEGIN
SET NOCOUNT ON;
    select @count = COUNT(*)
    from dbo.User_Tbl u、dbo.Goods_Tbl g
    where g.User_No = u.User_No
    And u.User_No = @user_No
    if   @count >= 5
    begin   set @IsAlowed = 0
          select * from dbo.Goods_Tbl
          where user_no = @user_no
    end
    else   set @IsAlowed = 1
END
```

開発環境はSQLServerであるので、CLRデータベースオブジェクトを作成してプログラムの統合を行う。CLRは(Common Language Runtime:共通言語ランタイム)の略である。ビジネスロジックの処理をDBに持っていくことによってプログラムの処理は軽くなり、その後のメンテナンスもより容易になる。SEにとっては簡単でありがたいことだと思う。エンドユーザ主導の立場では、いくつかの解決しなければならない課題が存在する。

4.3 ストアドプロシージャの自動生成

エンドユーザ（業務の専門家）は通常の業務にパソコンを利用しているが，プログラミング言語の知識はない。そこで，ストアドプロシージャの自動生成及びエンドユーザによるカスタマイズ化を実現することが必要になる。その方式として，今回のストアドプロシージャの記述例の考察に基づき，関数の概念や論理構造(判断，ループ)などを合わせてコンパイラツールを作成する方式などが考えられる。

具体的には、コード自動生成の概念を用いて、不用品再利用システムにおける四つの基本的なストアドプロシージャにより各テーブルの選択、新規、更新、削除操作を動的に実現させることができる。

5. エンドユーザへの支援

5.1 基本アーキテクチャ

基本的なアーキテクチャはデータベースに基づ

くとともに、ルールエンジンの概念を入れてエンドユーザ向けの自動生成 GUI ツールを開発することを検討する。

簡単に言えばＷｅｂアプリのビジネスロジックに関する処理を DB に持っていき、一つのビジネスロジックは一つのストアトプロシージャに対応させて処理するということである。自動生成ツールを利用してこれらのストアドプロシージャを生成することにより、エンドユーザ主導開発を実現することができると考えている。対象ユーザはプログラミングできないユーザなので、GUI の使いやすさとわかりやすさが求められる。

図 6 に上の部分はウェブアプリの構成である。下の部分は自動生成ツールのイメージ図である。稼働中のシステムはビジネスロジックを処理する度にデータベースに行ってストアドプロシージャを呼び出す。これらのストアドプロシージャはデータベースに保存されている。エンドユーザは自動生成ツールを利用してこれらのストアドプロシージャを生成する。原理は基本ストアドプロシージャを呼び出して引数などを設定してから、作成していくという流れである。

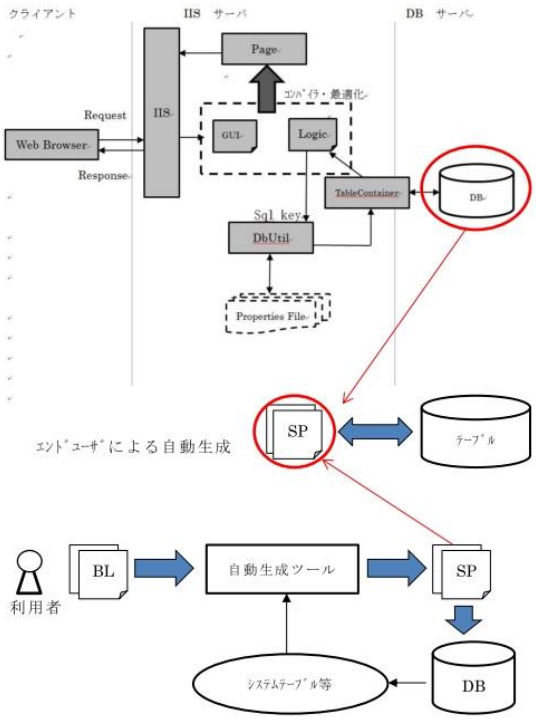

図 6.エンドユーザ支援への支援方式

5.2 基本ストアドプロシージャ

基本ストアドプロシージャはルールエンジンの中で一番基礎となるストアドプロシージャである。新しいビジネスロジックが必要となった場合にルールエンジンに新しいストアトプロシージャを追加する。ビジネスロジックを構造化することによって特徴が抽出され、新しいビジネスロジックはどんな特徴を持つかということが分かる。そこで、その特徴を満たす抽象的な基本ストアトプロシージャを予め用意すれば、エンドユーザが自動生成ツールを利用して、対応する基本ストアトプロシージャを呼び出すことにより新しいビジネスロジックを実装できる。図 7 は、同じ特徴を持ったビジネスロジックを実装する場合は同じ基本ストアトプロシージャを呼び出すだけで済むことを示し、基本ストアトプロシージャの汎用性を表している。

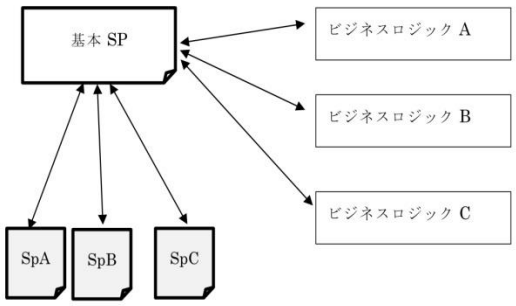

図 7. 基本ストアドプロシージャのアプローチ

5.3 基本ストアドプロシージャの構築

以下の例を用いて基本ストアドプロシージャの構築方法について述べる。

例：登録者がすでに上限(例えば 5 個)の登録をしていないかのチェック(4.2 節の例)

構造化するために「キーワード」を取り出して考察する。まず、登録者に対応する「一般ユーザ」は範囲が狭いから、ここは「誰」ではなくて「何」という言葉に取り換える。「登録した品物」は「一般ユーザ」に関係があるため、「関連するもの」という言葉を採用する。「個数が 5 個以上」にはチェックという意味がふくまれている。それから、上限だけではなくて下限などの場合もあるから、抽象的な「制限を満たした場合」という表現にする。登録処理に限らず、他の処理にも適していることを明確にするために、「処理させない」という言葉にした。まとめると以下のような構造化された文章になる。

「何が何に関連するものの数は制限を満たした場合は処理させない。」

ビジネスロジックはスタティックとダイナミックの二種類がある。例の仕様を定義してみる。この例はスタティックに分類されるチェック処理である。上にあるキーワードが抽出された文書を基盤として少しずつ、このビジネスロジックの実装仕様を完成していく。

5.4 GUI 設計

GUI には、エンドユーザを対象とするので、使いやすさが求められる。操作開始からストアドプロシージャ自動生成までの流れは図８のようになる。

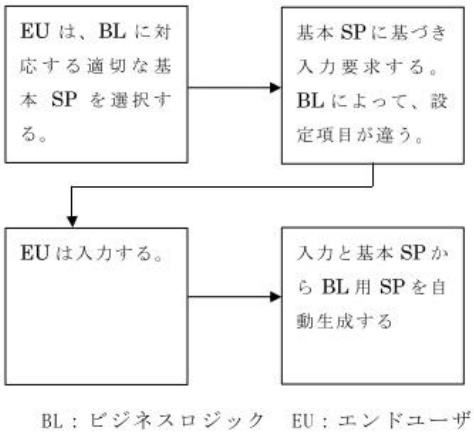

図 8. 自動生成する操作の流れ

5.3の例はチェック処理の中の数量チェックであるが、さらに時間チェックの基本ストアトプロシージャも必要である。そのため、入力画面には選択肢を設けるべきだと考える。このように分析してそれなりの画面も設計する必要がある。要するにビジネスロジックは画面との関係も一対一になっていることである。

6. おわりに

本稿では、エンドユーザ主導開発におけるビジネスロジックの実装方式の研究の一環として、２つの実装方式を採用し、不用品再利用システムを例題とするアプリケーションを試作した。ストアドプロシージャプログラミングを用いてビジネスロジックの定義方法に関する技術課題を解決することにより、エンドユーザ主導開発の実現可能性を高めた。

そして、特化したストアトプロシージャの実装と自動生成の可能性を検討し、エンドユーザにとって一番馴染み深いビジュアルツールとして、業務の専門家を対象とする自動生成ツールを提案をした。

参考文献

[1] 李静、中所武司、エードユーザ向け Web アプリケーションフレームワークの提案と試作 FIT2011 第 10 回情報科学技術フォーラム、B-013 (Sep. 2011)

[2] 佐藤修、エンドユーザコンピューティング、日科技連出版社 (July 1996)

[3] 中所武司、エンドユーザ主導開発のためのビジネスロジックの定義方式の提案、電子情報通信学会 技術研究報告 Vol.112(July 2012)

[4] 中所武司、Green-by-IT の観点からの地域活性化に関する要求分析、情報処理学会ソフトウェア工学研究会要求工学ワーキンググループ ワークショップ (Jan. 2012).

[5] 中所武司、特定分野向けフレームワークにおけるビジネスロジックのカスタマイズ機能に関する考察、情報処理学会ソフトウェア工学研究会要求工学ワーキンググループ ワークショップ (June 2011).

[6] Dino Esposito、プログラミング Microsoft ASP.NET MVC ASP.NET MVC 3対応版 (マイクロソフト公式解説書) 日経BP社 (2012/5/8)

[7] デヤン・サンデリック、SQL Server 2005 ストアドプロシージャプログラミング (SQL Server Books) 翔泳社 (2007/3/21)

[8] .NET Reflection:Dynamically Bind.Your Data Layer to Stored Procedures and SQL Commands Using .NET Metadata and Reflection

[9] Kalen Delaney、 Inside Microsoft SQL Server 2000.

[10] 目時秀典, 鈴木和久著‖キソ カラノ、基礎からの ASP.NET / ASP.NET ソフトバンククリエイティブ , 2010.9

[11] 許杰, 中所武司： エンドユーザ主導開発におけるビジネスロジックの実装方式の試作, FIT2012 第 11 回情報科学技術フォーラム, B-034 (Sep. 2012)