ソフトウェア工学 136－14
（2002．3．7）

Web情報システムのソフトウェアアーキテクチャ*

熊崎敦司† 野呂昌満†† 張 漢明††

Webブラウザを使用者インタフェースとするソフトウェア (WIS) の開発がさかんに行われており，その使用者インタフェースのもとでアプリケーションを統合する技術が重要となっている．WIS開発者は無秩序に存在するWeb技術に混乱しており，Web技術の理解，選択等に多くの労力を費している．本研究では，ソフトウェアアーキテクチャ上で適切なWeb技術を使用するための枠組を示すことを目的とし，WISのオブジェクト指向ソフトウェアアーキテクチャを構築する．本ソフトウェアアーキテクチャにしたがえば，必要な箇所に適切なWeb技術を用いたWIS開発が可能である．

The Software Architecture for Web-based Information Systems

ATSUSHI KUMAZAKI ,† MASAMI NORO †† and HAN-MYUNG CHANG††

Software engineering community becomes to put much more emphasis on the development of web-based information systems(WIS). That is, the key issue for developing WIS becomes how to select and coordinate existing web-based technologies such as CGI scripts, applets, PHPs, servlets, etc. There is unfortunately a bit chaotic situation on the coordination and it makes the development difficult. This paper concerns about a domain specific software architecture we constructed for WIS. The architecture is a guide map for the selection and the coordination. It eases the development of WIS as a result.

1. はじめに

Webブラウザを使用者インタフェースとするソフトウェアを開発することがソフトウェア開発の現場で標準となってきた．そのような背景の下，Web上でアプリケーションを統合する技術が重要となっている．本研究ではWebブラウザを使用者インタフェースとするソフトウェアをWIS(Web Information System)とよび，そのソフトウェアアーキテクチャについて議論する．

現在，WIS開発者は無秩序に存在するWeb技術に混乱しており，それぞれの技術の理解，選択等に多くの労力を費している．ソフトウェア開発におけるおもな課題は応用の論理 (Application logic) の適切な設計と実現である．しかし，WIS開発においては，それ以外の部分，すなわち，実現技術の理解と選択に多くの労力の投資が強いられている．

この問題は，

- Web技術に依存しない一般的なWISの構成を示し，
- どの箇所にどのWeb技術が有効であるかを示すことで解決できると考えられる．

本研究では，ソフトウェアアーキテクチャ上で適切なWeb技術を使用するための枠組を示すことを目的とし，WISのオブジェクト指向ソフトウェアアーキテクチャを構築する．本ソフトウェアアーキテクチャにしたがえば，適切な箇所に適切なWeb技術を用いたWIS開発が可能である．

2. ソフトウェアアーキテクチャ

ソフトウェアアーキテクチャ[3]は，システムの構造を示すだけではなく，実現の方法を含む実現のプロセスを内包すると我々は考える．

ソフトウェアアーキテクチャの構成要素の粒度や抽象度に関する定義は曖昧であり，一般的には同一粒度，同一抽象度の構成要素を持つと理解されている．しかし，我々は，ソフトウェアアーキテクチャは異なる抽象度の構成要素を集めたものであると考える (図1参照)．

† 南山大学経営学部情報管理学科
Department of Information Systems & Quantitative Sciences, Nanzan University
†† 南山大学数理情報学部情報通信学科
Department of Information & Telecommunication Engineering, Nanzan University

* この研究は，一部，2001年度南山大学パッヘ研究奨励金 I-A の補助を受けて行った．

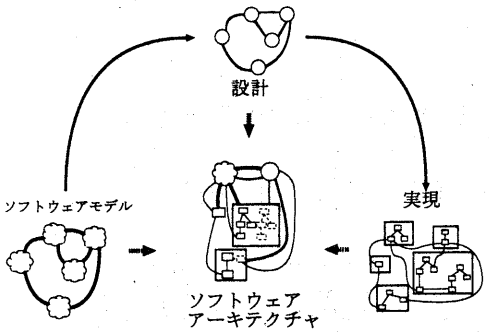

図 1 ソフトウェアアーキテクチャ
Fig. 1 Software Architecture

この前提を受け入れれば, ソフトウェアアーキテクチャは以下に示すように開発プロセスを宣言的に示すことになる.

- 仕様段階の構成要素について, 設計, 実現を行う.
- 設計段階の構成要素について, 実現を行う.
- 実現段階の構成要素を適切にあつらえる.

これらの半順序関係は構成要素間の関係 (メッセージの送受信, データフロー等) から決定される (図 2 参照). 構成要素は, その構成要素が依存する構成要素, または, その構成要素がメッセージを送信する構成要素を開発した後, 開発されるべきである. 相手先の構成要素が明らかな場合 (メッセージプロトコル等, 外部に公開する情報が決まっている場合) は開発順序は制約されない. すなわち, 構成要素間の関係は, その構成要素実現の半順序をも示すと考えられる.

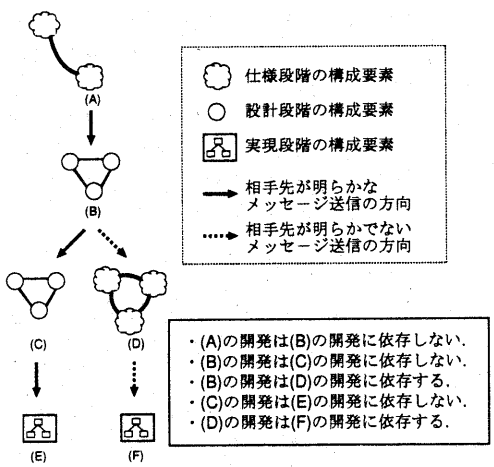

図 2 構成要素実現の半順序
Fig. 2 Partial Oder of Implementation

上述の通り, ソフトウェアアーキテクチャは構成要素を集めたものであるが, 同時に, その構成要素の抽象度と粒度を定義し, その開発プロセスをも含んでいる. このことから, 我々は 4+1 の側面[5] をさらに標準化し, 以下の 3 側面からソフトウェアアーキテクチャは記述可能であると考える.

- 抽象側面,
  構成要素と構成要素間の関係を静的, 動的に示す.
- 具象側面,
  どの構成要素が半製品となりうるか, また, どの構成要素に開発技術を適用できるか等, 実現に関する情報を示す.
- プロセス側面
  具象側面にしたがって, 構成要素を実現するさいの開発プロセスを示す.

3. WIS のソフトウェアアーキテクチャ

3.1 抽象側面

領域に依存した (Domain-Specific) ソフトウェアアーキテクチャを構築する場合には大きく分けて,

- 新たに領域に依存したソフトウェアアーキテクチャを最初から構築する手法, と
- すでに知られている (Well-Defined) ソフトウェアアーキテクチャを統合する手法

の 2 方法が考えられる. アプリケーションの特徴を満たすソフトウェアアーキテクチャがすでに構築されているならば, 後者を選択するのが必然である.

WIS の特徴は以下の 2 点である.

- データベースを利用するオブジェクト指向アプリケーションである.
- Web ブラウザを使用者インタフェースとする対話型アプリケーションである.

データベースを利用するオブジェクト指向アプリケーションは 3 層アーキテクチャ[1] によって構造が整理可能だといわれている. 応用の論理を実現するアプリケーション層, 情報データを配置するデータ層, 両者のインピーダンスミスマッチを吸収するファンクション層に分割することで整理可能である.

一方で, Smalltalk-80[4] のライブラリで提供されている MVC(モデル-ビュー-コントローラ) アプリケーションフレームワークとそのソフトウェアアーキテクチャは使用者インタフェースを持つ対話型のオブジェクト指向アプリケーションの開発に有効であるとされている. 使用者インタフェースを実現するコントローラ, ビューとモデルを分離することで, 使用者インタ

フェースに起こる変更を局所化することができる.

WIS は両者の特徴を持ち, これらを Web 上でうまく統合することで WIS に特化したソフトウェアアーキテクチャが得られる. 3 層アーキテクチャでは使用者インタフェースを含む応用の論理の実現までは述べていない. アプリケーション層に MVC アーキテクチャを適用し, 使用者インタフェースを実現する構成要素を詳細化することで両者の統合が可能である.

MVC アーキテクチャにおけるモデルが 3 層アーキテクチャと MVC アーキテクチャを統合する鍵となる. 3 層アーキテクチャでは, ファンクション層を介してデータベースにアクセスした結果をアプリケーション層で使用者に提示する. MVC アーキテクチャの場合, 使用者への提示は, モデルの変更によって行われる. 両者の接点は結果提示におけるモデルの変更である.

MVC アーキテクチャと 3 層アーキテクチャの統合として考えうる方法を以下に示す.

- アプリケーション, モデル分離型
  応用の論理を分離し, データベースアクセスは応用の論理を実現するオブジェクト (以後, アプリケーションオブジェクトと記す) が行う. アプリケーションオブジェクトは, 処理の結果としてモデルに変更を施す.
- アプリケーション, モデル一体型
  モデルが応用の論理を含んでおり, データベースアクセスはモデルが行う. データベースアクセスの結果として, モデルは自分自身の状態を変更する.

我々は前者を選択する. 応用の論理が単純な場合には後者が有効であるが, 現在の WIS における応用の論理は単純なものではなく, 今後も大規模化が予想される. その場合, モデルの変更と応用の論理の変更が直結する後者は好ましくない.

モデルはデータベースアクセスの結果として変更されると前述したが, WIS の場合, 使用者に提供する情報はすべてが動的に生成されるわけではない. すでにわかっている静的な情報に一部, 動的な情報を付加して提示する機会も少なくない. この場合, モデルはあらかじめ決まった形をしており, その一部がアプリケーションオブジェクトによって生成される.

以上の議論から以下に示すクラス構成 (図 3 参照) と動的な挙動 (図 4 参照) が得られる

3.2 具象側面

具象側面では実現に関わる情報を記述する. 一般には

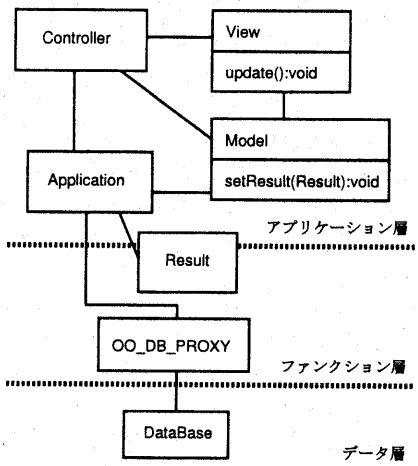

図 3 クラス構成
Fig. 3 Class Composition

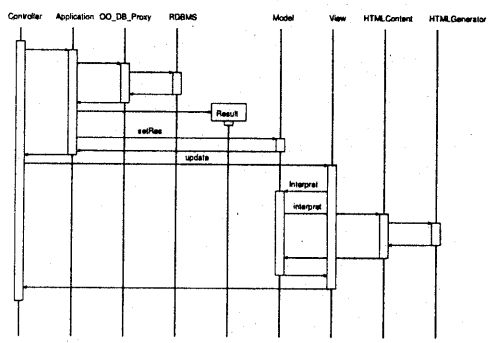

図 4 動的な挙動
Fig. 4 Sequence of Messages

- 抽象側面が示す構造に則り, システムを実現するさいに用いる開発技術,
- 開発技術を用いて実現する半製品

を記述することになる. しかし, WIS 実現時には上記の問題に加えて, Web 技術に関する情報が必要である. これらを併せて具象側面で記述する.

3.2.1 WIS のアプリケーションフレームワーク

オブジェクト指向ソフトウェアアーキテクチャを前提にソフトウェア開発を支援する方法としては以下の 3 つが考えられる.

- 構成要素間の関係を記述し, アプリケーションフレームワークを作成する.
- ソフトウェアアーキテクチャ内の構成要素の自動生成系を作成する.
- ソフトウェアアーキテクチャ内の構成要素を実現するためのクラスライブラリを作成する.

我々は以下の理由により, WIS のアプリケーションフレームワークを作成する. クラス間の関係をフローズンスポットとして提供することで WIS 開発者の労力軽減が期待できる.

WIS の仕様を厳密に記述することは難しく, プログラム生成系の適用は事実上不可能である. 仮にプログラム生成系を実現しても, プログラムと同程度の仕様が必要となることが予想でき, 労力の軽減にはならないと考える.

クラスライブラリは Web 技術に依存する側面が強い. Web 技術毎に必要とされるクラスライブラリを作成することになり, 本研究の趣旨である Web 技術の整理には向かない.

結果, 図 5 に示すような構造が得られた. 太線はフローズンスポットとなるクラスを表す.

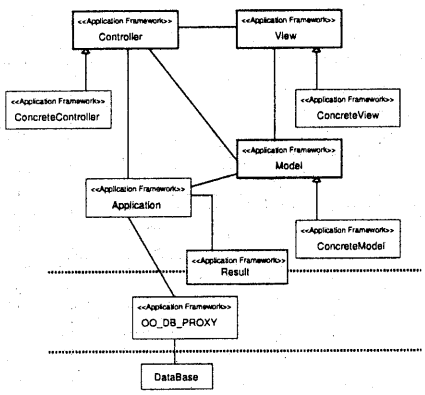

図 5 開発技術の割付け

Application of Development Technology

3.2.2 Web 技術の応用

必要な箇所に適切な Web 技術を用いるためには, どの構成要素の実現にどの Web 技術が有効であるかを考慮する必要がある. その解は,

- どの構成要素がクライアント側, またはサーバ側のどちらで実現されるべきか
- どの Web 技術がクライアント, またはサーバのどちら側の構成要素の実現に有効であるか

を考えることで得られる.

使用者からの入力はコントローラが受け付け, 使用者への情報の提示はビューが行う. WIS の場合, これらの処理をクライアント側で行っている. すなわち, コントローラとビューはクライアント側で実現されるべき構成要素である.

Web サーバは使用者インタフェースからの通知をきっかけに使用者の要求を処理するサブシステムと考えることができる. すなわち, アプリケーションオブジェクト, モデルはサーバ側で実現されるべき構成要素である.

WIS 構築に使用される代表的な Web 技術を以下に挙げる.

- CGI(Common Gateway Interface) (サーバ)
  CGI とは Web サーバがプログラムを起動するさいの起動方法 (環境変数の名前や値) を定めた仕様の名称である. この仕様に基づいて作成されたプログラムが CGI プログラムである. Web サーバはリクエストに応じて CGI プログラムを起動, 結果をクライアントに返す.
- サーバサイドスクリプト (サーバ)
  Web データベースで必要な処理を独自のスクリプト/タグを使い HTML 文書内に記述する. スクリプト/タグはサーバ側で実行され, その結果が Web ブラウザに送られる. 代表的なものに ASP(Active Server Page), PHP などがある.
- アプレット (クライアント)
  Web サーバからダウンロードされ, Web ブラウザ上で実行される Java プログラムである.
- サーブレット (サーバ)
  Web サーバ側で実行される Java プログラムである. Web サーバから呼び出され, 各種処理を行い, 結果を Web ブラウザに返す. リクエスト毎にプロセスを起動しないので CGI よりも効率良く動作する.
- JavaScript (クライアント)
  Web サーバからダウンロードされ, Web ブラウザ上で実行されるスクリプト言語. HTML 文書中に記述される.

これらをクライアント側の技術, サーバ側の技術に分類し, 整理すると図 6 に示す解が得られる.

3.3 プロセス側面

具象側面が示す構成要素 (図 5 参照) から以下の 6 工程が導き出される.

- データベースの実現
- データベースアクセスのためのオブジェクト指向インタフェースの実現
- アプリケーションオブジェクトの実現
- コントローラのサブクラスの実現

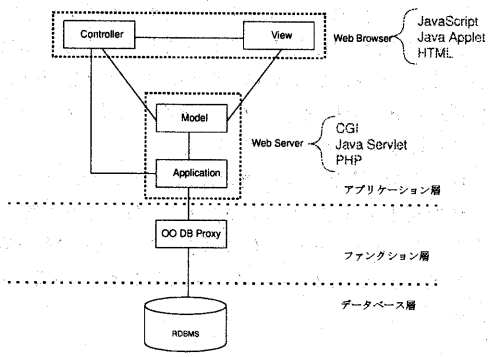

図 6 Web 技術の応用
Fig. 6 Application of Web Technology

- モデルのサブクラスの実現
- ビューのサブクラスの実現

これらの半順序関係は構成要素間の関係にメッセージのシーケンス (図 4 参照) を当てはめることで得られる (図 7 参照). 矢印はメッセージの送受信の方向を表す. 実線はアプリケーションフレームワークとして確定したメッセージ通信 (相手先, メッセージの名前が明らかであること) を表す. 破線はメッセージの送信先オブジェクトが明確でない (メッセージプロトコルが明らかでない) ことを示す. この場合は, メッセージの受信オブジェクトを実現後, メッセージの送信オブジェクトを実現する必要がある. 未確定クラス (細線) へのメッセージ送信が破線になることがわかる.

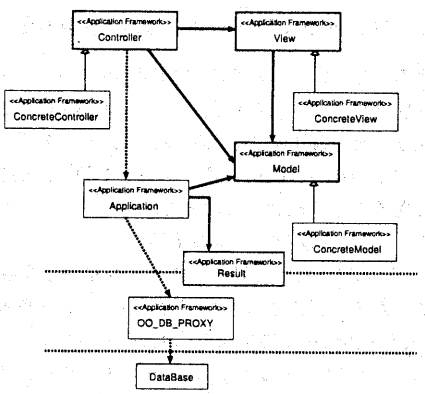

図 7 プロセス側面記述の過程
Fig. 7 Description of Process View

以上の議論から導かれる 3 つのサブプロセスを並行に結合することで図 8 に示す開発プロセスが得られる. 記述には RUP(Rational Unified Process) でワークフローを表すのに用いられるアクティビティ図を用いる.

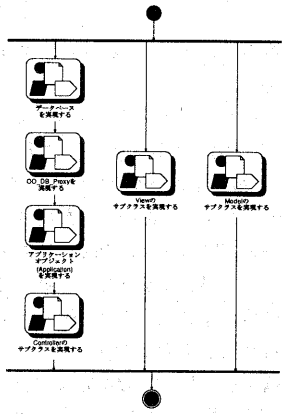

図 8 プロセス側面
Fig. 8 Process View

4. ソフトウェアアーキテクチャの応用

ソフトウェアアーキテクチャはソフトウェア開発支援ツール統合の鍵である. ソフトウェアアーキテクチャが,

- 開発支援技術, と
- それを用いた場合の開発プロセス

を示すことから,

- ソフトウェアアーキテクチャが示す開発支援技術にしたがい, 開発支援ツールを実現し,
- それを開発プロセスにしたがって統合すれば,

ソフトウェアアーキテクチャを中心とするソフトウェア開発環境を実現することができると考えられる.

本研究ではソフトウェア開発支援環境実現の初期段階として, WIS のアプリケーションフレームワークを実現する. アプリケーションフレームワークの実現にはプログラミング言語 Java を用いる. 今後の WIS 開発では Java を用いた開発が主流となることが予想される.

4.1 アプリケーションフレームワークの設計

アプリケーションフレームワーク実現の第一目的はソフトウェアアーキテクチャの妥当性を検証することなので, アプリケーションフレームワーク作成においては, 素直な設計, 実現を一義とする. とくに, 抽象側面が示す構成を素直に反映する.

WIS はクライアント/サーバ型のソフトウェアであり, 設計時には, その間の通信プロトコルを無視することはできない. Web ブラウザ, Web サーバ間の通信プロトコルとして以下の 2 通りの実現方法が考えら

れる.

- HTTP(Hyper Text Transfer Protocol) による通信
- それ以外のプロトコル (独自のプロトコル等) による通信

一般に Web ブラウザ, Web サーバ間の通信は HTTP によって実現されている. これは RFC で定められており, 事実上の標準となっている. しかし, アプレットとサーブレットを併用した場合, Web ブラウザ, Web サーバ間の通信は, Web ブラウザ上で実行されるアプレット, Web サーバ上で実行されるサーブレット間の通信に置き換えられ, 他のプロトコルを用いて通信を行うことも可能である.

上記の議論から, HTTP を用いて通信を実現する形態が Web 技術に特化しない, 汎用的な形態だと考えられる. アプレット, サーブレット間で通信を行う形態は特殊なものと考える.

Web ブラウザと Web サーバ間の通信, すなわち, コントローラ, ビューとモデル間の通信は HTTP で行うことになる. これを実現する方法としては以下の 2 通りが考えられる.

- コントローラとビューの代理 (プロキシ) を配置し, それらが HTTP を理解する.
- モデルが HTTP を理解する.

我々は以下の観点から前者の方法を選択する.

- モデル, ビュー, コントローラの明確な役割分担を行う.
- 遠隔オブジェクト通信を一番自然に実現する方法は ORB 方式である.

サーバ上にビュー, コントローラの代理を配置することで抽象側面が示す構造を実現する.

4.2 アプリケーションフレームワークの実現

アプリケーション層は MVC アーキテクチャを利用しており, これは Gamma[2] らが述べているように Observer パターンを利用することで実現できる. Observer パターンを利用した MVC アーキテクチャの実現として以下の 2 通りが考えられる.

- プッシュ型 MVC アーキテクチャ
- プル型 MVC アーキテクチャ

どちらを用いるかは MVC アーキテクチャのモデルの構造に依存する. モデルが単純な場合には, それに応じてアクセスメソッドも単純になる場合が多い. その場合にはプル型の MVC アーキテクチャが有効である. しかし, モデルが複雑になれば, それに応じてアクセスメソッドも繁雑になり煩わしい. その場合にはプッシュ型の MVC アーキテクチャが有効である.

WIS の場合, モデルは HTML 文書のモデルであり, HTML 文書が構造を持つことは明らかである. したがって, プッシュ型の MVC アーキテクチャを利用し, モデル自身に解析用のメソッドを持たせる. ビューはモデルに解析のきっかけを与える.

構造を持ったデータの解析には Interpreter パターン[2] を利用する方法が一般的であり, 最も自然である. さらに Visitor パターンを併用することでその振る舞いの変更を局所化し, 柔軟性を与えることができる (図 9 参照).

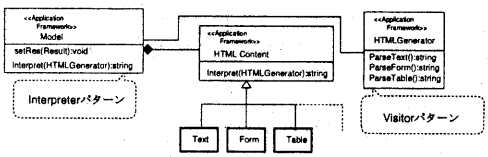

図 9 モデルのクラス構成
Fig. 9 Class Composition of Model

ホットスポット

開発者の労力を軽減するために一般的な HTML 用のモデル要素と解析オブジェクト (Visitor) はあらかじめ用意することにする (図 9 参照).

ホットスポットは以下の通りである.

- `Controller` クラス
- `Model` クラス
- `HTMLGenerator` クラス
- `View` クラス

5. ソフトウェアアーキテクチャの実践

構築したソフトウェアアーキテクチャにしたがって, 受注管理システムを試作する. これは Web 上で商品の発注とその確認をするためのシステムである. 試作の目的は, 適切な Web 技術を使用した WIS 開発が可能であることを確認することである.

5.1 受注管理システムの概要

今回, 試作した受注管理システムは登録と検索を行うことができ, 図 10 に示す 4 画面 (メニュー画面, 登録画面, 検索画面, 結果画面) から構成される.

5.2 アプリケーションフレームワークを使用した受注管理システムの実現

プロセス側面にしたがい, 以下の作業を行うことで受注管理システムを試作した.

(1) データベースの実現

データベースには一般的な関係データベースとして, PostgreSQL を利用する. ファイルシステムを用いてデータ

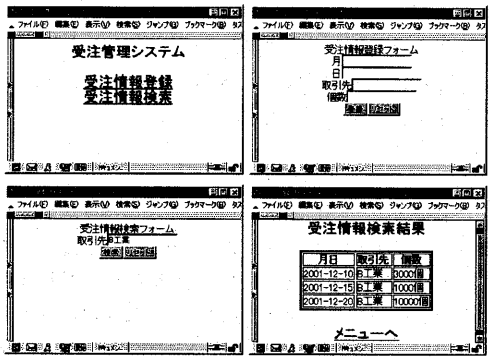

図 10 (左上) メニュー画面 (右上) 登録画面
(左下) 検索画面 (右下) 結果画面
Fig. 10 Menu, Resigtration, Search, Result

ベース機能を実現することも考えられるが, データベースを実現することが本研究の本質でない.

(2) オブジェクト指向インタフェースの実現
オブジェクト指向インタフェースの実現には JDBC(Java DataBase Connectivity) を利用する.

(3) アプリケーションオブジェクトの実現
受注管理システムが行うサービスは登録と検索である. それぞれに応じたメソッドを実現する.

(4) コントローラのサブクラスの実現
コントローラは入力処理を行うので, すでに Java で提供されている HttpServlet クラスを利用する. HttpServlet クラスのサブクラスとすることで HTTP リクエストの処理などにかかる労力を軽減することができる.

(5) モデルのサブクラスの実現
受注管理システムはつの画面から構成されるので, それぞれに応じてモデルのサブクラスを実現する

(6) ビューのサブクラスの実現
モデルと同じく, 画面に応じてビューのサブクラスを実現する

以上の結果, 受注管理システムを試作することができ, Web ブラウザからサービスを受けることができることを確認した.

6. 議　　論

実際に本ソフトウェアアーキテクチャを用いることで,

- Web 技術を整理することができ,
- 適切な Web 技術を利用することが可能であるかについて考察を行う.

試作した受注管理システムの特徴は以下の通りである.

- データベースを使用して, 受注データを管理する.
- 使用者インタフェースの外観, 操作性は問題としない. (HTML で実現可能な使用者インタフェースで必要十分である.)
- 使用者認証は行わない.

この特徴をふまえ, WIS を以下の 4 つに分類し, 考察を行う.

- データベースアクセスが必要なもの.
- HTML では実現不可能な使用者インタフェースが必要なもの.
- 使用者認証が必要なもの.
- その他.

データベースアクセスが必要なもの

これは検索エンジン[7)11)], 掲示版[9)] 等, 近年, よく見られる WIS の典型例である. 構築したソフトウェアアーキテクチャに最も自然に当てはまる. これらは, 本論文で紹介したように Web サーバ上の技術を用いることで実現が可能である. Web サーバ上で処理を実行し, その結果を Web ブラウザに返答する.

Web サーバ上の技術を選択するさいには, 各 Web 技術の特徴に加えて, サーバの環境を考慮する必要がある. CGI は多くの環境で利用することができるが, PHP, Java サーブレット等はまだ利用できる環境が限られている.

CGI の利点は多数のプログラミング言語で記述可能なことである. 開発者は慣れたプログラミング言語を使用することができる. ただし, リクエスト毎にプロセスを発生するので, PHP, Java サーブレットに比べて効率が悪い.

PHP は Web データベースアクセスに特化したプログラミング言語で, 簡単なデータベースアクセスシステムの構築に有効である.

Java サーブレットの利点は豊富に API が用意されていることである. さらに, Java という記述力の高いプログラミング言語を利用することができ, 複雑な処理の実現にも有効である. ただし, これを利用できる環境はまだまだ限られている.

今回は Java サーブレットを利用したので, JavaServlet, JDBC など Web やデータベースに特化した Java の API を使用することで開発の労力を軽減することができた. 他の Web 技術を利用する場合は, これらの機能

は独自に実現する必要がある.

HTML では実現不可能な使用者インタフェースが必要なもの

オンラインゲーム[10] 等に見られる見た目を重視した使用者インタフェースや, マウス操作を念頭においた操作性を考慮した使用者インタフェースなどを実現するには HTML の機能だけでは不十分である. この場合はクライアント側の技術を用いて, 使用者インタフェース, すなわち, コントローラ, ビューを実現することになる.

使用者認証が必要なもの

Yahoo!オークション[8] に代表されるオンライントレードや, インターネット決算事業銀行[6] 等において, 使用者認証は必要不可欠である. これは処理内容を使用者に知られてはいけない処理の代表例である. Web ブラウザ上で実行する処理は, 内容を使用者に知られる危険性を孕んでいる. したがって, 使用者認証など内容を知られたくない処理は Web サーバ上で実現するべきである.

そ の 他

その他の場合として, サーバの負荷を分散させたい場合がある. その場合には, Web ブラウザ (クライアント) 上で処理の実行が可能な Java アプレット, JavaScript 等を用いることになる. これらは Web ブラウザ, Web サーバ間のデータを減らすことから, ネットワーク資源の有効利用にもつながる.

ま と め

以上の議論の結果, クライアント側の技術については以下の結論が導き出される.

- 使用者インタフェースの実現に使用する.
  クライアント側の技術を用いて, ビュー, コントローラを実現すればよい.
- 負荷分散の実現に使用する.

同様にサーバ側の技術については以下の結論が導き出される.

- Java サーブレット, PHP など, とくに Web に特化した技術を積極的に利用する.
  - Web に特化した API による労力の軽減
  - 新たなプロセスを発生しないことによる効率の改善

  が期待できる.
- 上記の技術が利用できない場合には CGI を利用する.
  この場合, Web, データベースなどに特化した部分を独自に実現する必要がある.
- クライアントに知られたくない処理の実行はサーバ上で行う.

上述のように Web 技術を整理できたことにより, WIS 開発の初期段階にあたる Web 技術選択にかかる労力を軽減できたと考える.

7. おわりに

本研究では, WIS のソフトウェアアーキテクチャを構築した. 本ソフトウェアアーキテクチャにより, 開発者の混乱の一要因となっている Web 技術を整理することができたと考える. その結果, 適切な箇所に適切な Web 技術を用いた開発が可能である. 本ソフトウェアアーキテクチャにしたがい, 受注管理システムを試作することで, 本研究の方向性に問題がないことを確認した.

今後の課題は以下の通りである.

- 他の Web 技術を使用したアプリケーションフレームワーク, WIS の実現
  今後は, Java サーブレット以外の Web 技術を利用し, アプリケーションフレームワークを実現する. また, それを用いて WIS を実現することで, 本ソフトウェアアーキテクチャの妥当性の検証を行う.
- WIS 開発環境の実現
  プロセス側面にしたがい, 開発支援ツールを統合することで WIS の開発支援環境を実現することができると考える.

参 考 文 献

1) Jhon J.Donovan, *Business Re-Engineering with Information Technology*, Prentice Hall PTR, 1994.
2) E. Gamma, R. Helm, R. Johnson, and J. Vlissides, *Design Patterns*, Addison-Wesley, 1994.
3) D. Garlan and D. E. Perry Eds., *Special Issues on Software Architecture : IEEE Transaction on Software Enginnering, Vol. 21, No. 4*, 1995.
4) A. Goldberg, *Smalltalk-80:the interactive programming environment.* Addison-Wesley, 1984
5) P. B. Kruchten, "The 4+1 View Model of Architecture," *IEEE Software*, Vol. 12, No. 6, pp.42-50, Nov. 1995.
6) イーバンク銀行, http://www.ebank.co.jp.
7) YAHOO! JAPAN, http://www.yahoo.co.jp.
8) YAHOO! Auctions, http://auctions.yahoo.co.jp.
9) YAHOO! MESSAGE BOARDS, http://messages.yahoo.co.jp.
10) YAHOO! GAMES, http://games.yahoo.co.jp.
11) goo, http://www.goo.ne.jp.