LUCINA for Java/WebLogic Server の構造

Architecture of LUCINA for Java/WebLogic Server

杉 野 順 清

要 約 LUCINA にはいくつかのプロファイルがあり，その中の「Web AP サーバプロファイル」として開発されたのが，LUCINA for Java/WebLogic Server である．この，WebLogic Server とは J 2 EE ベースの Web AP を構築するための米国 BEA 社のミドルウェア製品である．

本稿では，Web AP サーバの歴史と将来像に触れ，LUCINA for Java/WebLogic Server の開発に先立ってなされた決定事項とその開発方法の概要について記述する．さらに，現在問題になっている点について解説する．

Abstract The LUCINA methodology has some profiles, And " LUCINA for Java/WebLogic Server " was developed as the " Web Application Profile " one of them. WebLogic Server is a J 2 EE-based standard middleware product of BEA Systems, Inc. used to build Web application.

This paper, first, discusses the history and future of the Web Application Server, and then major decisions made prior to the development of " LUCINA for Java/WebLogic Server " and the outline of the product, finally problems to be solved at present time.

1. はじめに

Web アプリケーションの世界は今まさに発展しつつある最もホットな分野の一つである．「IT 革命」という言葉は既に一般的な言葉になったが，その基盤を支える最も重要な技術のひとつが Web アプリケーションであることは間違いない．

しかし，Web アプリケーションを構築するための技法や利用技術といったものは，まだ整理されておらず，開発者はそれぞれ現在手さぐりで開発しているのが現状である．LUCINA[*1] for Java/WebLogic Server（以下では LUCINA for Java/WLS）はその状態を少しでも緩和するために開発されたものである．

本稿ではまず，「Web アプリケーション」の目指す世界を説明し（2 章），次にそれを達成するために我々が考えた内容について説明していく（3 章）．さらに，現行の LUCINA for Java/WLS の問題点について考察する（4 章）．

2. Web アプリケーションの目指す世界

Web アプリケーションとは，一言でいうと Web の仕組みを利用して構築された，クライアント―サーバ型のアプリケーションである．ここで Web の仕組みとは次のようなものを指している．

1） クライアントには Web ブラウザを用いる．

2） クライアント―サーバ間のプロトコルには HTTP(HyperText Transfer Protocol（RFC-1945，RFC-2068 など）を用いる．

3） HTTP 上で運ばれる電文（コンテンツ）には主に HTML（HyperText Markup

Language（RFC-1866，RFC-1867 など）を使用し，Web ブラウザ上にはそれが直接表示される．

4） サーバでは何らかの処理を行った結果により「動的に」HTML などを生成し，Web ブラウザに返す．

ここで，1)～3)までに関しては，既存のインターネットやイントラネット及び各クライアント PC 上にインストールされた Web ブラウザの仕組みがそのまま使えることが重要である．Visual Basic などを使用したような，いわゆる FAT クライアントと呼びれるものとは違い，別段特定の（業務）システム用のクライアントソフトウェアを PC などにインストールする必要が無いのである．その結果，いわゆる DLL 地獄[*2] などと呼ばれる問題から解放され，さらには HTML さえ解釈できれば，Web-TV や i-mode のように PC 以外のものをクライアントとして用いる道も開けてくる．

一方，その分サーバの開発および運用は複雑になってしまうのはいなめない．上記 4)では一言で「動的に」HTML を生成すると書いたが，実際にそれを行うのはかなり複雑である．「Web アプリケーションサーバ」とは，それを少しでも生産性よく開発し，効率よく実行できるためのミドルウェア（もしくは，フレームワーク）である．

以下では，その Web アプリケーションサーバの発展の歴史と将来像について述べる．

2.1 Web アプリケーションサーバの発展の歴史

Web アプリケーションサーバは，Web サーバから次のように発展してきた．

1） 静的 HTML の時代

Hypertext の名の通り，HTML ドキュメント同士がリンクという形で相互に連結しあっている．バラバラに存在していたドキュメントがインターネットでつながったが，あくまで静的なもののみで「サービス」と呼べるものはなかった．

2） CGI（Common Gateway Interface）の時代

〈FORM〉タグにより WWW ブラウザからサーバへリクエストを出す仕組みが HTML の仕様に追加された．サーバ側ではそのリクエストを別プロセスに渡すことにより，動的に HTML を作成することも可能になった．このサーバ側の仕組みのことを CGI といい，Web アプリケーションのはしりになった（図 1）．

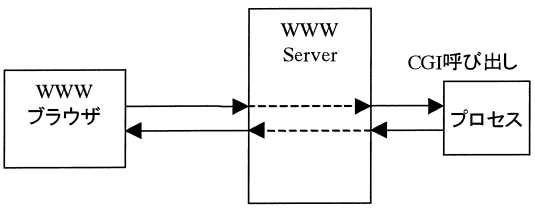


図 1 CGI の仕組み

3）　スクリプティングの時代

CGI は通常 Perl や C 言語などを使って作成されており，複雑な HTML を生成するプログラムを記述するのは困難が伴う．また，その CGI は通常プロセス起動によっており，効率面でも問題があった．そこで，生まれてきたのが HTML の中にプログラムを埋め込む方法（これをスクリプティングと呼ぶ）で，そのプログラムの起動もスレッドによるものになった. Microsoft IIS で採用された ASP（Active Server Pages）がその代表である（図 2）.

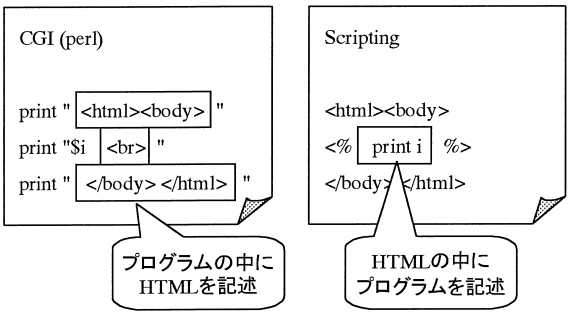


図 2　CGI とスクリプティング

4）　J 2 EE（Java 2 Platform Enterprise Edition）の時代

これらにより，複雑な Web アプリケーションも開発できるようになったが，検索エンジンなどから，より本格的な業務アプリケーションに応用するためには，データベースアクセス・トランザクション処理・ディレクトリサービスなどが必要になってくる．このようなエンタープライズ領域に応用できるために考えられた仕様が J 2 EE で，上記のものはそれぞれ JDBC・JTA/JTS・JNDI に対応する（図 3）.

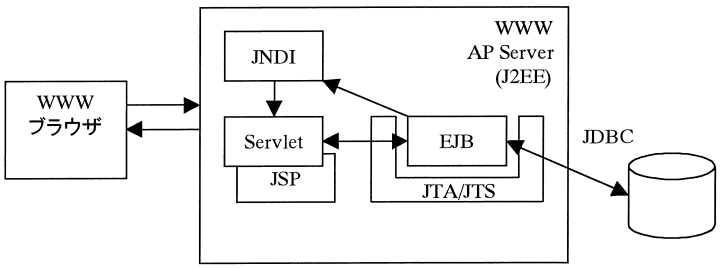


図 3　J 2 EE

2. 2　Web アプリケーションサーバの将来像

将来像を描くときの切り口は次のものがある.

①　サーバサイド

—コンポーネントの流通

—Web サービス

② クライアントサイド
—XML 対応ブラウザ
—クライアントサイドスクリプティング

本稿の趣旨とは外れるので割愛するが，クライアントサイドについては，WWW ブラウザの高機能化に伴い，XML 対応ブラウザ（XSL や CSS など）が発展していくと思われ，また，クライアントスクリプティング（JavaScript や Java Applet など）に関しても，特に携帯端末における J 2 ME（Java 2 Platform Micro Edition）のサポートなど，興味深い話題も多い．

以下では，サーバサイドにおける Web アプリケーションの将来像について述べたい．

2.2.1 コンポーネントの流通

一つの発展の方向として，EJB（Enterprise Java Beans）の仕様の充実があげられる．EJB 1.1 仕様から EJB 2.0 仕様[*3] に拡張された，重要な点は以下の通りである．

① メッセージドリブン EJB
JMS などから非同期で EJB を呼び出せる．

② CMP Entity Bean で EJB-QL（Query Language）をサポート
SQL の where 句に似た文法の EJB-QL を使用して Bean の検索条件を指定できる．

③ EJB home メソッドのサポート
通常の Java におけるクラスメソッドに対応する home メソッドが記述可能になった．

また，EJB の利用技術の蓄積を目指した「EJB™ コンポーネントに関するコンソーシアム」も日本国内のベンダ[*4] を中心に設立された．このコンソーシアムの各部会と目的は次のようなものである．

① ポータビリティ部会
EJB コンポーネントの各種 EJB コンテナ製品間でのポータビリティを技術的に追求・促進する．

② コンポーネント公開情報部会
EJB コンポーネントの仕様及び品質に関して，ユーザや市場にいかなる情報が公開されるべきかを技術的に追求する．

③ デザイン部会
再利用性の高い EJB コンポーネントを開発するための設計・開発標準を検討する．

つまり，EJB の仕様とその利用技術は充実しつつあるり，さらには，EJB コンポーネントがソフトウェア部品として流通するようになる可能性が高いとおもわれる．

2.2.2 Web サービス

コンポーネントの再利用の形態としては，部品として流通するだけでなく，ネット経由で利用することも考えられる．そのような視点で，最近，注目されているのが米国 Microsoft 社や IBM 社が中心になって仕様を策定している「Web サービス」である[*5]．

Web サービスとは,「既存の Web 環境をベースに，プロトコルとしては HTTP を，データ表現としては XML を利用したサーバ上のサービス」である.

Web サービスの主要技術は次の通りである.

① SOAP（Simple Object Access Protocol）
XML と HTTP をつかって RPC（Remote Procedure Call）をするためのもの

② UDDI（Uniform Description, Discovery and Integration）
Web サービスに関する情報を登録・検索できる電話帳のようなサービス

③ WSDL（Web Service Description Language）
Web サービスのオペレーションとメッセージを抽象化したものを記述する XML フォーマット

これらの関係を図示すると図 4 のようになる.

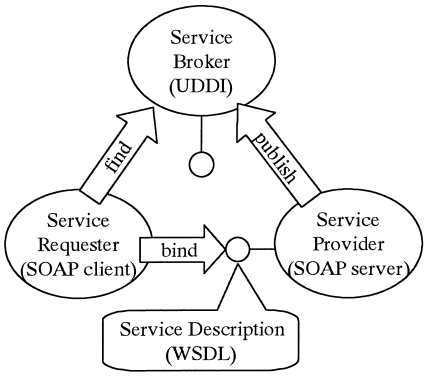


図 4　Web サービスのアーキテクチャ

筆者が新聞や雑誌やオンラインニュースなどを見聞きする限りにおいては，日本では EJB そのものをコンポーネント部品として流通させることに熱心で，米国では Web サービスを利用することに熱心であるように見える.

方向性としては細かいものもあわせるといくつもあり，どれが主流になるのかを見定めるのは難しい．しかし，大きく二つの流れがあることだけは認識しておく必要があるし，提供手段―コンポーネントそのものを流通させるか，Web サービスとして提供するのか―に違いがあるにせよ，両方が共存していくことには間違いない.

LUCINA はもともとコンポーネント指向の技法であり，提供手段にかかわらず適用できることを目指している． LUCINA の用語に対応させるとの表 1 のようになる.

表 1　LUCINA 用語と提供手段

LUCINA 用語	対応先	提供手段
コンポーネント	EJB コンポーネント	コンポーネントそのものをソフトウェア部品として流通させる
コンポーネントシステム	Web サービス	Web サービスとしてネットワーク経由で提供する

3. LUCINA for Java/WebLogic Server

LUCINA にはいくつかのプロファイルが含まれており，LUCINA for Java/WLS は「Web アプリケーションサーバモデル」である．その適用システムイメージとしては，インターネット，イントラネットでの本格的な業務システム構築向けのシステム・モデルであり，WebLogic Server が提供する J 2 EE 機能を使用して短期間にシステムを構築することが可能になることを目指している．

以下では，その LUCINA for Java/WLS について説明していく．

3.1 LUCINA for Java/WLS の必要性

J 2 EE には主な仕様だけで表 2 のようなものが含まれる．（図 3 も参照のこと）

表 2 J 2 EE の主な仕様

仕様名	概要
EJB (Enterprise Java Beans)	Java のコンポーネント化
Servlet	サーバサイド動的 HTML 生成
JSP (Java Server Pages)	HTML 内スクリプティング
JDBC (Java DataBase Connectivity)	データベースアクセス
JNDI (Java Naming and Directory Interface)	ネーミングサービス
JTA/JTS (Java Transaction API/Service)	トランザクションサービス
JMS (Java Message Service)	メッセージングサービス

それぞれ大量のドキュメントと API[*6] を抱えており，そのような膨大な API を一通りだけでも理解するだけでも大変である．

さらに，Web アプリケーションを開発する場合は，このような J 2 EE の知識のほかに，Java 言語の知識，Java 2 Platform Standard Edition Core API の知識，HTTP プロトコルの知識，HTML のタグの知識，JavaScript の知識，といった多様な知識も要求される．

つまり，旧来から存在した，Oracle など RDBMS の知識，Windows NT や Solaris などの OS の知識，さらにオブジェクト指向やデータモデリングの知識に上記のような知識が加わっている．

しかし，本当に開発者全員がこれらの知識を一通り学ぶ必要があるのであろうか．答えは「否」である．ただし，LUCINA で規定している「アーキテクチャチーム」の担当者は上記の知識を一通りは理解している必要がある．

すべての知識を持ちあわせていない技術者でも，LUCINA for Java/WLS に沿った形で開発すれば，求められる品質と期間で開発できることを目的にしている（図 5）．

さらには，Web アプリケーションという範囲に限定しても，目的のシステムを構築するために使用できるプロダクトの機能の組み合わせは多数存在し，それらは増加しつづけている．その中から，ベスト（あるいはベター）な組み合わせを見つけるのは，ますます困難になってきている．

LUCINA for Java/WLS ではそれを解決するために，実証済みの基本アーキテクチャを提供しており，それを元に各システムの要件にあわせてアーキテクチャを組み直

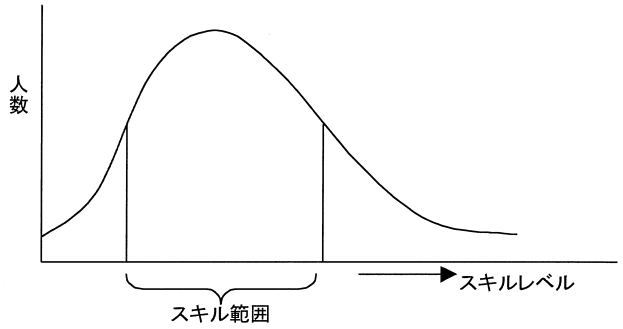


図 5　LUCINA が対象としている技術者像

すことになる．その結果，より短時間で確実なアーキテクチャを構成できる．

3.2　開発に先立って決めたこと（ターゲットマーケットと提供スケジュール）

本節では，このような LUCINA for Java/WLS のアーキテクチャの策定にあたって，どのような作業をしてきたのかを記述していく．

LUCINA for Java/WLS の開発を開始したのは 1999 年 10 月ぐらいであり，その最初に決めたことは「ターゲットマーケット」である．

当初，策定するに当たって決まっていたのは，LUCINA のプロファイルとして「アプリケーションサーバモデル」に対応する，ということだけである．ひとことでアプリケーションサーバモデルといっても，人によりその適用イメージはかなり異なってくることが想定された．よって，LUCINA for Java/WLS のターゲットマーケットがどこにあるのかを明確にすることで，その適用イメージの統一を図らなければならない．一方，LUCINA は開発のための技法であるので特定のマーケット（たとえば，銀行のネットバンキングや流通のモールサイトなど）に縛られるわけではない．

そこで，「ターゲットマーケット」＝「アプリケーションの要件定義（の一部）」（つまり，アプリケーションモデル）と捉らえて，次のような項目について定義を行った．

- アプリケーション領域
- 処理量
- 開発量
- 分散度
- RAS（Reliability, Availability, Serviceability）
- 連携性，他接

さらに，表 3 に示すような提供スケジュールも策定した．Web AP サーバは最新の技術であり，ドッグイヤーといわれるくらい技術の移り変わりは激しい．よって，ある程度未解決な部分が残ったとしても，なるべく早く市場に投入できることが重要である．つまり，いくら良いものであっても，早い時機に投入できなかったら意味が無い．

実際は，部分的に Ver 1.1 や 1.2 などで行おうとした内容を取り込みつつ，Ver 1.0 が 2000 年 5 月に提供され，その後は教育用の資料を作成した以外は，大きな改修なしに現在に至っている．

表 3　提供スケジュール

バージョン	タイトル	提供時期
Ver 1.0 α	Basic Policy	1999 年 12 月末
Ver 1.0	Limited Deployment	2000 年 4 月
Ver 1.x	General Release	2000 年 7 月
Ver 1.x 以降	Maintenance Phase	2000 年 7 月以降

繰り返しになるが，何か汎用的なものを開発するときに重要なのは，「ターゲットマーケット」と「スケジュール」と「コンセプト」である．さらには汎用的とは矛盾するが，いかに一番売れるところに絞り込めるかがもっとも重要である．企画や開発の担当者は，いろんなユーザに広く売り込めることを狙うがために，とかく様々な機能を盛り込みたがるが，これは大きな間違いである．

また，次節で示す「基本方針」のような一種の要件定義を行うにあたり，そのアプローチとして「マーケットイン」と「プロダクトアウト」があるが，今回はマーケットインアプローチをとりたいがために，ターゲットマーケットを最初に決めたという意味もある．

3.3　基本方針記述書

上記のような「ターゲットマーケット」＝「アプリケーションの要件定義（の一部）」の決定の次に行ったことは，「基本方針記述書」の策定である．

この基本方針記述書は，最終成果物ではなく，アーキテクチャ/プロダクトセット，ツールキット，ドキュメント（開発プロセス編/実践編/設計方針編など），といった成果物を開発するための要求仕様書，および設計書[*7] という位置づけである．

また，この基本方針記述書を作成するときには「ターゲットマーケット」を次のような点で参照することにした．

- このアプリケーションを開発するために必要十分な条件を満たしているか検証することで，内容の検証を行う．Nice To Have（十分条件以上のもの）といったものは含めない．
- 複数の選択肢が存在する場合，このアプリケーションを開発するときに最も適した選択肢を選ぶことに利用する．

この基本方針記述書は表 4 で示されるような章で成り立っている．

表 4　基本方針書目次

章番号	タイトル
1 章	メインスケジュールと目標
2 章	ターゲットマーケット
3 章	アーキテクチャ
4 章	コンポーネント
5 章	プレゼンテーション
6 章	開発
7 章	開発プロセス
8 章	その他

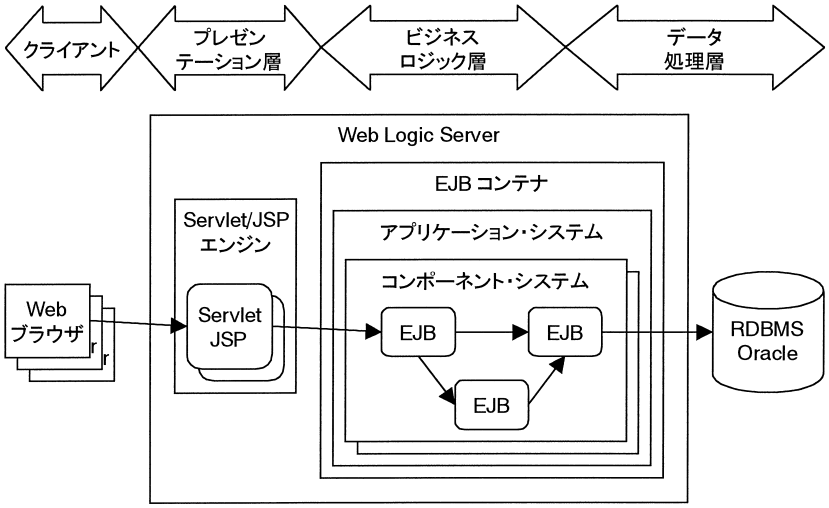


図 6　基本アーキテクチャ

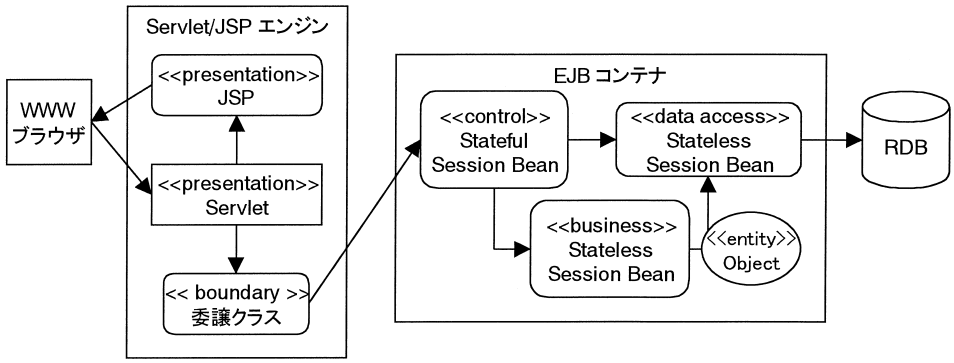


図 7　基本アーキテクチャ（詳細）

つまり，LUCINA for Java/WLS を開発していくに当たってのバイブルとしての位置づけにあり，設計開発が進む中で複数の選択肢の一つを決める必要があるときなどにこれを参照している．

3.4　アーキテクチャ

LUCINA for Java/WLS のアーキテクチャは図 6 に示すような構成になっている．

また，Servlet/JSP 及び EJB の部分のみを拡大したのが図 7 である．

各部の主な役割は表 5 のようになっている．

3.5　開発プロセス

LUCINA for Java/WLS の開発プロセスは図 8 のようになっている．

この中で，他の LUCINA プロファイルや他の開発方法論にはない特徴の一つとして「画面遷移記述」がある[*8]．

表 5　各部の役割

名前	役割
WWW ブラウザ	画面の表示。単純な入力チェック。Servlet 呼び出し。
Servlet/JSP エンジン	ユーザ認証・許可。Servlet/JSP の実行。
《presentation》Servlet	画面遷移の管理。《boundary》処理委譲クラスの呼び出し。JSP 呼び出し。
《boundary》委譲クラス	型変換。EJB の呼び出しとコントロールオブジェクトのハンドルの管理。
《presentation》JSP	画面(HTML)生成。
EJB コンテナ	トランザクションの管理。
《control》Stateful Session Bean	トランザクションの開始/終了。クライアント毎のステート管理。ビジネスオブジェクトやデータアクセスオブジェクトの呼び出し。
《business》Stateless Session Bean	ビジネスルールの実行。データアクセスオブジェクトの呼び出し。
《data access》Stateless Session Bean	永続データの取得・更新。
RDBMS	永続データの管理

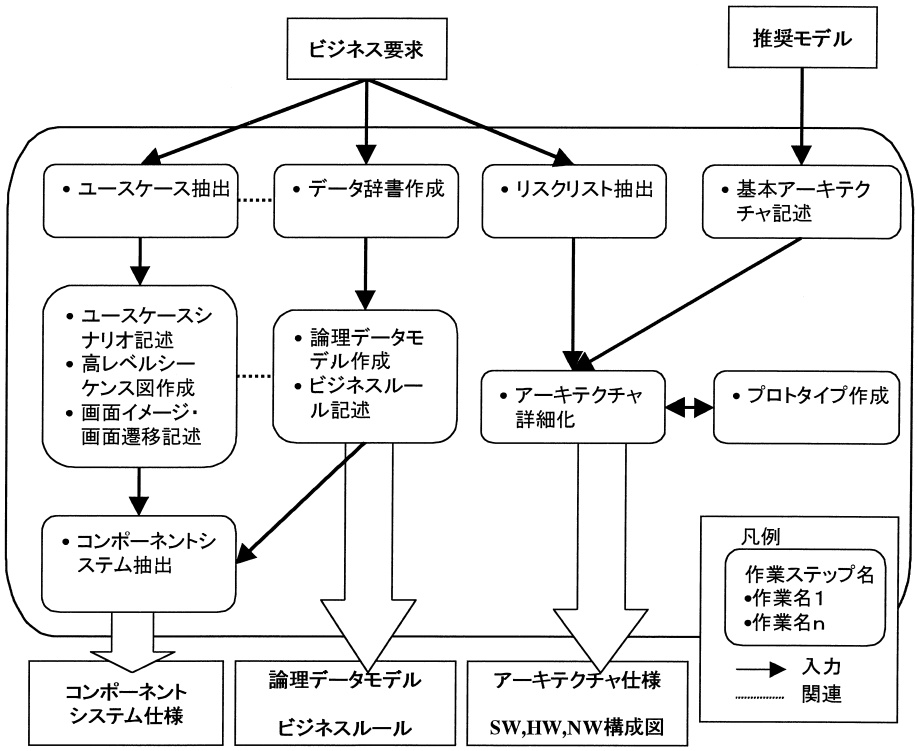


図 8　LUCINA for Java/WLS の開発プロセス

従来の手法では，図 9 に示すように，単に画面を複数書いてそれを矢印などで結んでいくことで画面遷移記述としている．

この図の場合，画面遷移分岐の原因は分からない．画面遷移が分岐する点はこの図の場合二個所あるが，①違うボタンが押されることで分岐する，②入力エラーがあったら分岐する，の二種類の原因がある．また，Web アプリケーションの場合，①呼び出すサーブレットが違う，②サーブレット内で表示するページを変える，という違

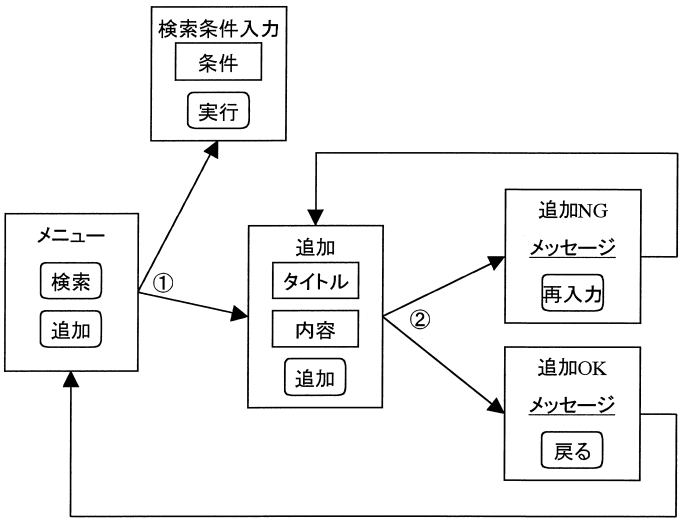


図 9　従来の画面遷移図

いをこの遷移図では的確に表現できない．

そこで，サーブレットを表現するものを導入した．ひとつのサーブレットにはそれを示す一つの小さい丸を記入する．さらにその小さい丸はサーブレットとは限らないため，インタラクションと呼ぶことにした（図 10）．

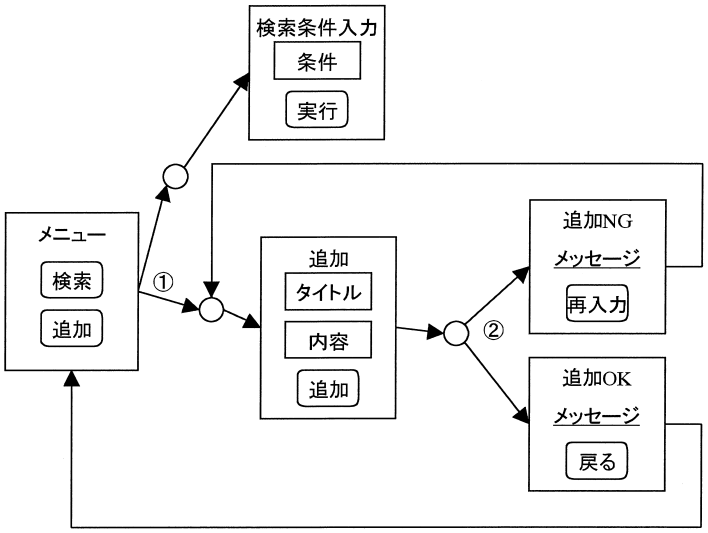


図 10　改良した画面遷移図

画面から直接分岐する場合は押されるボタンが違うこと，インタラクションから分解する場合は何らかの処理がなされた結果分岐していることを，表現できる．

4. 今後の課題

本章では，LUCINA for Java/WLS に残された課題のうち，重要な項目について説明する．

4.1 機能先選択方式と対象先選択方式

LUCINA では（for Java や WebLogic Server などに関わりなく），まず業務分析フェーズの最初で，ユースケース抽出とユースケースシナリオ記述を行う．要求定義書中にある機能は，通常一つのユースケースにマップされる．また，ユースケースシナリオは，対応するユースケースの作業内容を詳細化したものになっている．

LUCINA for Java/WLS では，どれか一つのユースケースを選択し，そのユースケースシナリオを書くときは，既にどの機能をこれから実行しようとしているかが決まっているとしている．そこでは次のことが暗黙的仮定になっている．

> 暗黙的仮定 T：ユースケース図とユースケースシナリオから画面とコントロールオブジェクトの設計ができる．（以下では単に仮定 T とする）

仮定 T の中心は，ユースケース図から画面遷移図を一意に導出できるという仮定である．この仮定 T は必ずしもいつも成り立つとは限らないため，実システムを開発する上で支障が生じている．

実システムにおいて，ユースケースの選択はメニューなどによりアクターに選択させた後ユースケースシナリオに沿った処理を行っていくことするならば，その処理を行うにはその処理の対象を選択することになる．対象より機能を先に選択することになるため，これを機能先選択方式と呼ぶことにする．（現在の LUCINA for Java/WLS は，機能先選択方式のみに対応している．）

一方，同じ処理をするにしても，その実行対象を先に選択してから，機能を後で選択する方式も考えられ，これを対象先選択方式と呼ぶことにする．

機能先選択方式の例として，銀行の CD 機があげられる．CD 機の画面上には「振り込み」や「入金」などメニューが表示されており，その中の一つを選ぶことから始まり，後はガイダンスに沿って振り込み先などを入力していく．

また，対象先選択方式の例としては，Web 仮想店舗システムでの買い物があげられる．ここでは，最初は適当に気の向くまま商品を見て回り，もし買いたくなるような商品を見つけたら，それを選択し仮想買い物かごに入れて，最後に決済をする．

この違いを，もう少し分かりやすく説明するために，さらに例題を用いて説明する．

■例題

- ノウハウ蓄積システム
- ノウハウは一つのテーブル（RDB table を想定）に集中して記憶.
- フィールドには ID，タイトル，内容（テキスト）だけがある．
- ノウハウは追加・参照・変更・削除が可能
- ノウハウの選択は，タイトルが表示された一覧画面で行う．

次に，ユースケース図は使わずに，画面遷移図とその処理プリミティブ（インタラクション）を用いて，この二つの方式で設計した結果を図 11 と図 12 に示す．この例

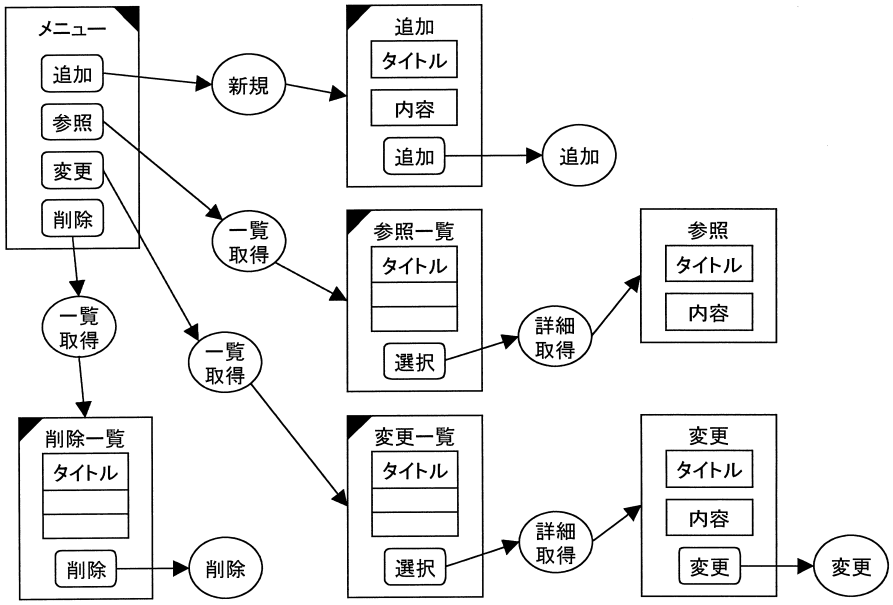
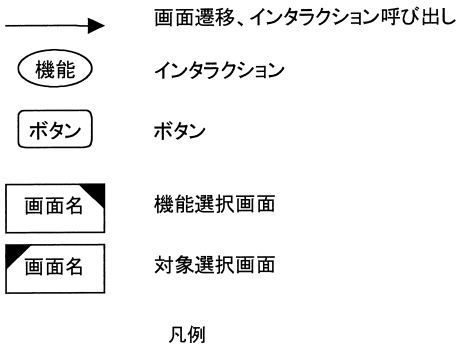


図 11　例題の機能先選択方式での画面遷移図



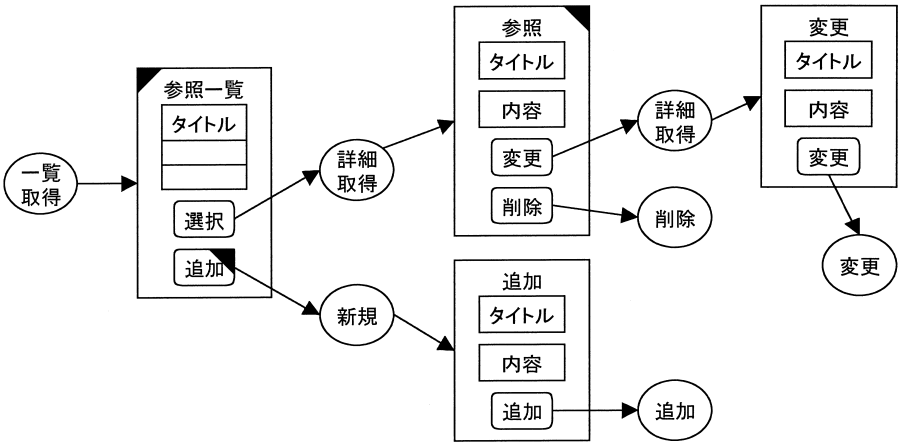


図 12　例題の対象先選択方式での画面遷移図

では，削除時確認画面や登録結果画面などは，単純化のために省略してある．

なお，LUCINA ではユースケース図とユースケースシナリオを先に記述し，画面遷移図はその後で記述することになっており，仮定 T が成り立つとすれば必然的に「機能先選択方式」になってしまう．そこで，その反証を例示するために，ここではあえて逆順にしている

どちらの方がより良いのかは，システムによって違ってくると思われる[*9]が，これにより，少なくとも二つの方式の違いで，画面遷移や画面に表示するボタンなどが変わってくることは理解してもらえるはずである．ただし，いずれの場合でも画面遷移図におけるインタラクションそのものは同一であり，そのインタラクションはユースケースに対応している．

ここで，図 12 の「追加ボタン」は「対象（空オブジェクト）を選択すると同時に，機能（追加）も選択している，ことに注意してもらいたい．機能先選択方式にそったかたちで，ユースケース図を導出し，記述してみると図 13 のようになる．

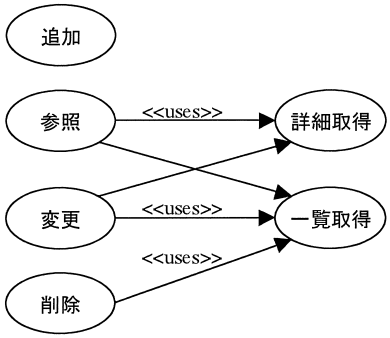


図 13　機能先選択方式でのユースケース図

しかし，対象先選択方式の場合（詳細な説明は割愛するが）適切なユースケース図を書くのは難しい．多少無理にでも記述するなら図 14 のようになる．

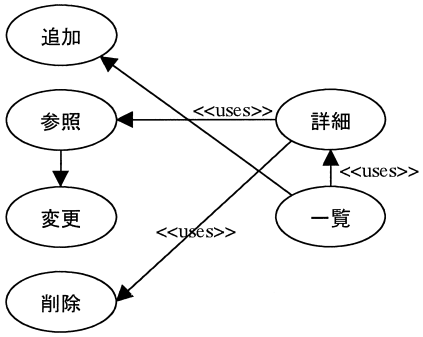


図 14　対象先選択方式でのユースケース図

明らかにこのユースケース図はおかしいが，これは，最初に述べた仮定 T が誤っているからに他ならない．本来ユースケースはあくまでアクターから見たときの最小機能単位を示すだけであり，画面遷移図とユースケース図とは切り離して考える必要がある．ユースケース図から，画面遷移図を一意には導出できないとするのが正しい．

つまり，ユースケースとそのシナリオのみを入力文書とするのではなく，画面遷移図をも導出できる「ユーザインタフェース記述書」のようなものも入力文書としなければならない．いまは，それの適切な記述方法は見つかっておらず，今後の課題となる．

4.2 開発/テストツール

1) 統合開発環境（IDE）

開発ツールに関しては，VisualCafe などの WebLogic Server 対応を IDE を使用することができる．しかし，次のような問題点がある．

① 初心者にとってハードルが高い

VisualCafe に限らず，この手の IDE は年々高機能化が進んでおり，それにつられてマニュアルもかなり分厚くなっている．LUCINA はもともと，「ある程度の知識でも，そこそこの開発生産性を得られる」ということを目的の一つにしているが，そのための Getting Started や適用マニュアルが作られていない．特定の IDE と使用する機能を絞ったうえで，よりよいマニュアルの作成が望まれる．しかし，それにしても IDE そのものも毎年のように新しいバージョンが発売されるため，かなりの工数を掛け続けないといけないという問題がある．

② IDE は特に必須ではない

もともと，VisualCafe に限らず IDE が支持されてきたのは，GUI の作成ウィザード機能によるところが大きい．GUI を多用するアプリケーションの開発生産性は IDE を使う結果，非常に高くなる．一方，Web アプリケーションの開発はサーバサイドが中心で，GUI 構築のための機能は不要であり，IDE を使ったからといって，大きな生産性向上は望めない．

Web アプリケーションの開発は，(1) コーディング (2) コンパイル&デプロイ (3) 実行&デバッグを繰り替えすことになる．熟練者にとっては (1) は慣れたエディタの方が良い，(2) は（csh や make などの）スクリプティングでカバーできる，わけで唯一 (3) だけは IDE を使ったほうが効率良さそうであるが，それにしても高機能な IDE は不要で JDK だけで十分なケースも多い．

2) テストツール

テストツールとして次のようなものが必要である．

① 機能テストツール（単体テスト・結合テスト用）

② パフォーマンス測定ツール

LUCINA for Java/WLS の開発開始当初では，上記のテストができるツールは豊富ではなかったが，現在はそれなりに使えそうなものがいくつか出てきている．それらツールの評価と WLS と組み合わせたときの使用方法に関するマニュアルの整備が望まれる．

3） Web 画面の作成

Web ブラウザを利用したユーザインタフェース作成のためのツールが現在あまり充実していない．

Web アプリケーションのユーザインタフェース作成機能としては，JSP 対応と Javascript 対応が望まれる．しかし，JSP そのものが現在発展中でありタイムリーに対応している IDE がない．Javascript は Internet Explorer と Netscape Communicator で動作が異なるし，同じブラウザでもバージョンが違うと同じようには動作しない．ブラウザ（IE の最新版など）を限定すれば，いくつかの IDE（MS Visual InterDev など）は実用的に使用可能だが，IDE での問題を同時にはクリアできない．

5. おわりに

Web アプリケーションの世界はまだまだ発展中であることは間違いなく，エンジニアが新たに習得しなければならない知識も非常に多い．いま WebLogic Server を利用した開発プロジェクトを眺めてみると，現実的に最も問題になっているのはその膨大な知識を習得しきれないことによる「エンジニアのスキル不足」である．LUCINA はもともとこのギャップを埋めるためのものではあるが，少なくとも Java や HTML/HTTP に対する基礎的な知識は持っていることを前提にしており，残念ながら「Visual Basic もしくは COBOL しかしらないエンジニア」に対する回答には至っていない．

その回答を得るためには，各エンジニアのスキルアップ，強力な開発ツールによるサポート，LUCINA 的技法の充実，と様々なアプローチが考えられるが，いずれもすぐに達成できるものではないと思われる．結局は地道にできることから一つずつ，しかしその方向性は間違えずに，こなしていくしかないと感じる．

最後に，LUCINA for Java/WLS の開発の主要なメンバである，細谷氏，山本氏，尾島氏，八木氏，萩谷氏と，様々な助言をいただいた，原氏，羽田氏，宮脇氏，福島氏に感謝する．

＊ 1 LUCINA とはシステム開発における生産性向上をめざした開発技法であり，その特徴は次にあげられる．
- 日本ユニシスで標準的に用いられるマネージメントプロセスに適合し，手順・技法・文書様式を実務的に提示する．
- アーキテクチャ（システム構造，アプリケーション構造，実行環境）を絞り込んでいる．
- 開発環境を提供している．
- コンポーネント指向開発であり，コンポーネントの実装方法は構造化手法も採用．

＊ 2 DLL 地獄：Microsoft Windows 上で動作するアプリケーションが用いる DLL（Dynamic Linking Library）は Windows のバージョン毎や開発環境のバージョン毎に，同じ名前で互換性のないものが存在する．DLL 地獄とは，PC にインストールされている DLL によって，アプリケーションの動作がより不安定になる事を指す．ただ，Microsoft は Windows の新しい版で解決しようとしている．詳しくは次の URL などを参照されたい．http : //www.galliver.co.jp/writing/vmx/sxs 2 k/index.html

＊ 3 EJB 2.0 仕様：2000 年 11 月現在で“ Proposed Final Draft ”になっている．（http : //java.sun.com/products/ejb/docs.html）

＊ 4 日本 IBM，富士通など 6 社が発起人メンバ．

＊5　SUN 社も 2001 年 2 月，オープンでスマートな Web サービスとして SUN ONE（Sun Open Net Environment）を発表した．
＊6　大量のドキュメントと API：例えば，EJB 2.0（Proposed Final Draft）の場合 Specification で 558 page，API で 21 クラス 62 メソッドある．JDBC の API はこれのさらに十倍以上もある．
＊7　Unified Process の Elaboration Phase の成果物に対応する．
＊8　他にも多数あるが，LUCINA for Java/WLS の様々なドキュメントにも書かれているため，ここでは割愛する．
＊9　少なくとも同一システム内では，ユーザインタフェースの観点からも，どちらの方針で行くかをあらかじめ決めておき，統一する必要がある．

執筆者紹介　杉　野　順　清（Junsei Sugino）

1990 年 3 月国立豊橋技術科学大学大学院修士課程情報工学専攻卒業．1990 年 4 月日本ユニシス(株)入社．1990 年 8 月システム技術本部知識システム部配属，知的支援プラットホーム Tippler 開発に従事．1995 年 4 月オープンソフトウェア事業部転属，SYSTEM ν［nju：］商品企画担当．1999 年 4 月生産技術部所属，LUCINA 開発に従事．現在に至る．