成蹊大学理工学研究報告
J. Fac. Sci. Tech., Seikei Univ.
Vol.43 No.1 (2006) pp.9-15

# アスペクト指向プログラミングによる旅館予約システムの開発

飯田　善久*[1]，梅津　圭介*[2]

## Development of a Reservation System for Japanese Style Hotels using Aspect-Oriented Programming

Yoshihisa IIDA *[1], Keisuke UMEDU *[2]

**ABSTRACT：**A reservation system for Japanese style hotels using a new business model has already been reported by one of the authors. The reservation system consists of computer systems of travel agencies and hotels. Hotels keep their inventories of available rooms, offer services which respond to inquiry about room availability from travel agencies and accept reservation of specified rooms. Standard XML messages are used for communication between travel agencies and hotels. But hotels are using their own PMS (property management system) which have room inventory, and there are many kinds of PMSs. Development cost of a hotel system to cooperate with its PMS is very high. In order to decrease development cost of a hotel system, this report describes how to interface with PMS using Aspect-oriented approach.

**Keywords：**Aspect-oriented programming, Reservation system, AspectJ, PMS

(Received March 27, 2006)

## 1．はじめに

従来からの旅行会社に加え，数多くのリアル店舗を持たない，いわゆるネット旅行会社が数多く設立され，利用者はいつでも，どこからでも宿泊施設の予約が可能になり，ネット予約は大幅に増加している。しかし，現状の宿泊施設予約システムには次のような問題点がある。（図１参照）

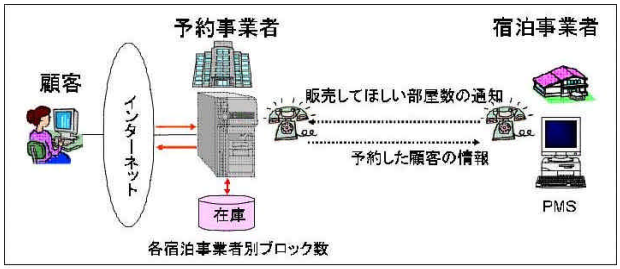


図１　現状の旅館予約システム

(1) 宿泊施設を運営する「宿泊事業者」は，「予約事業者」に対して，あらかじめ販売してほしい部屋数を事前に申告しておく。（これをブロックと呼ぶ）ネット旅行会社が増え，予約事業者の数が増大するにつれ，宿泊事業者は，各予約事業者に提供するブロック数の管理業務が煩雑になり，また全予約事業者へ提供したブロック数の総和が，実際の部屋数よりも多いこともあり，オーバーブッキングが生じることがある。

(2) 予約事業者は，提供されたブロックの範囲内では，顧客からの予約を自動的に処理することは可能であるが，それを超えた場合は，人手を介して宿泊事業者に空き部屋がないかどうかを問合せる必要がある。

(3) 予約事業者が顧客から受けた予約に関する情報（予約日，予約部屋数，氏名・年齢・性別等の顧客情報）は，FAX あるいは専用端末により送られてくるので，宿泊事業者は自社の館内管理システム（PMS：Property Management System）に再入力しなければならない。

*1：経営・情報工学科教授（Professor, Department of Information Sciences）, iida@is.seikei.ac.jp
*2：経営・情報工学科 2005 年度卒業研究生
(Undergraduate student, Department of Information Sciences)

これらの問題点を解決するために，旅行電子商取引促進機構[2]の検討をもとに，図２に示すような次世代型宿泊予約システムの試作システムを構築した。(文献[1]参照)

(1) 予約事業者からの要求にしたがって予約業務を行なう処理を，宿泊事業者自身が Web サービスの形で提供する。したがって，日々の予約可能部屋数を管理する在庫データベースは，宿泊事業者自身のシステムで管理する。

(2) 予約者に関する情報も，予約事業者からオンラインで受け取り，自社の PMS へ直接反映する。

(3) 予約事業者，宿泊事業者間でやり取りするメッセージは旅行電子商取引促進機構で策定中の標準 XML 形式[3]を使用する。

この試作システムを用いて，旅行関係者にデモンストレーションを行ない，その普及に努めているところであるが，各旅館に導入されている館内管理システム（PMS）は，多くのベンダーから提供されている。あるいは大規模旅館においては自作の PMS を使用している場合もある。したがって，このビジネスモデルを使用する場合には，図２に示した宿泊事業者システムを，各 PMS 毎に開発する必要があり，特に自作の PMS を使用している場合はその開発費用，保守費用も膨大になることが予想される。

これに対応するために，予約事業者とのインターフェースは標準 XML を用い，PMS とのインターフェースのみを独立して記述し，アスペクト指向プログラミングの考え方を導入してその両者を結合してシステムを構築することにより，システム開発が効率的に行なわれるのではないかと考え，その利点を検証することにした。

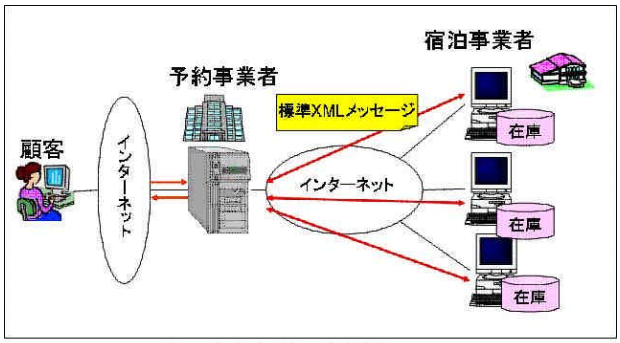


図２　試作システム

## ２．アスペクト指向プログラミング

ソフトウエア開発のパラダイムとして，オブジェクト指向の考え方が定着してきている。オブジェクト指向では，システム全体をオブジェクトに分割し，更にそれをサブオブジェクトに分割するといった手段が用いられ，システム全体はオブジェクトの木構造となる。しかし，そこからもれてしまった機能は，後から多くのオブジェクトにばらばらに実装されることになる。オブジェクト指向ではうまくできない例として，ロギング機能が挙げられる。[4]ログをとる処理そのものは，独立したオブジェクトとして作成することは容易であるが，そこで用意したメソッドを呼び出す処理は，ログ出力を必要とする各オブジェクトに分散して記述することになる。テスト段階が終了し，実運用に入った場合，開発者はテスト段階で必要としたログ出力メソッドの呼び出しを注意深く削除しなければならず，その際に新たなエラーを発生させることも少なからず生じている。ログ出力の呼び出しのように，多くのオブジェクトに関係する機能を「横断的関心事(crosscutting concern)」と呼び，オブジェクト指向の考え方ではうまく処理することが出来なかった。横断的関心事の処理方法の研究は，パルアルト研究所に在籍していた Gregor Kiczales らにより進められ，アスペクト指向プログラミングとしてまとめられた。[5]

アスペクト指向プログラミングでは，この横断的関心事をオブジェクトに分散させるのではなく，単一のモジュール「アスペクト」にまとめて記述し，更に横断的関心事を「いつ，どこで」利用するかを，それを利用するオブジェクト側で指定するのではなく，アスペクト側で指定する。(図 3 参照) 実行に際しては，その両者をツール（アスペクトコンパイラ）を用いて統合する方法が用いられている。

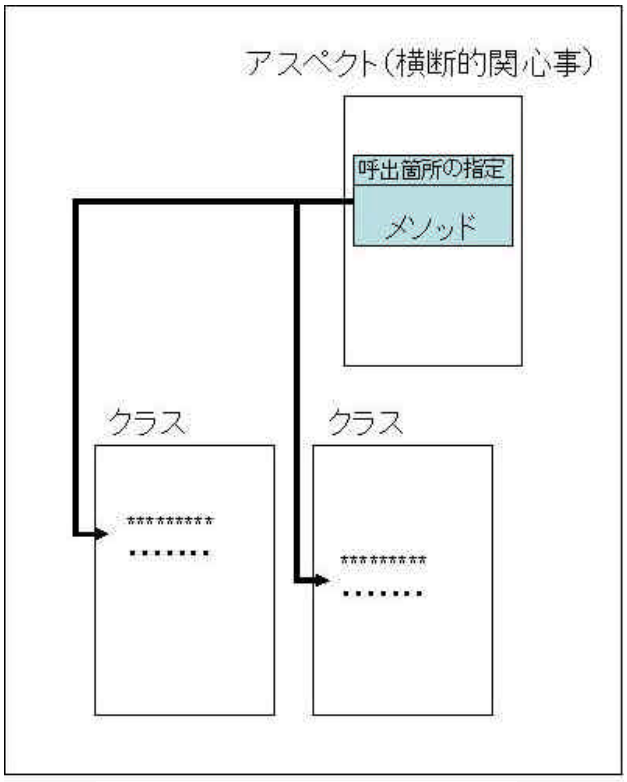


図３　アスペクト指向の考え方

Kiczales らが開発した，Java ベースのアスペクト指向の実装である AspectJ[6]を例にとって，実装方法を紹介する。　表 1 に示す，「Hello!」を出力するプログラム HelloWorld.java に，横断的関心事（「I am an Aspect」と出力）を織り込むことを考える。表１に示すように，HelloWorld.java は通常の Java プログラムであり，クラス HelloWorld のインスタンスを生成し，そのメソッド hello()を呼び出している。メソッド Hello()が，「Hello!」を出力している。

これに対して「横断的関心事」を織り込むアスペクトの一例である Aspect1.java を表２に示す。

①行目のキーワード aspect は，Java プログラムの class あるいは interface に相当するキーワードであり，このプログラムがアスペクトであることを示し，Aspect1 は作成者が名付けたアスペクト名である。

②行目でアスペクトの織り込み先を指定する。この例では，before()により，HelloWorld クラスの hello メソッド呼出しの前に織り込むことを表わしている。

③行目で，織り込むメソッド（アドバイスと呼ぶ）を定義している。

②に示した before()は，HelloWorld.hello()の実行の「前」に挿入することを示すが，この他に HelloWorld.hello()の実行「後」に実行することを示す after()，HelloWprld.hello()に「代わって」アドバイスを実行することを示す around()が用意されている。

表１　HelloWorld.java

```
public class HelloWorld {
    public static void main(String[] args) {
        HelloWorld helloWorld = new HelloWorld();
        helloWorld.hello();
    }
    void hello() {
        System.out.println("Hello!");
    }
}
```

表２　Aspect1.java

```
public aspect Aspect1 {                                    ①
  before():execution(void HelloWorld.hello()) {            ②
    System.out.println("I am an Aspect");                  ③
    }
}
```

表１，表２のプログラムを用意し，表３①に示すようにアスペクトコンパイラ ajc を呼び出し，両者を１つのアプリケーションとしてまとめ，それを表３②に示すように実行すれば，アスペクト Aspect1 の実行結果が先に，本体の HelloWorld クラスの Hello()メソッドの実行結果がその後に出力される。

このようにアスペクト指向プログラミングを利用すると，もとのコードはそのままにして，既存のプログラムに新しいコードを追加することが出来る。したがって，システム全体はそのままにして，一部の機能を改善したり，新しい機能を追加したりすることが容易になる。また，システム開発当初には想定していないシステムとの結合をしなければならない場合にも，結合部分のみをアスペクトで書き，既存のシステムに織り込めば，全体のシステムを再構築する場合に比べて容易に結合が出来ることが予想される。１章で述べたように宿泊事業者の予約システムの場合も，予約事業者とのインターフェース部分は，全宿泊事業者に共通であるので本体のプログラムとして記述し，各宿泊事業者によって異なる PMS とのインターフェース部分をアスペクト化することを考えた。（図４参照）

表３　コンパイルと実行

```
C:¥>ajc HelloWorld.java Aspect1.java          ①
C:¥>java HelloWorld                           ②
I am an Aspect ◀――――― Aspect1(表２②)の出力
Hello! ◀――――― HelloWorld.hello()の出力
C:¥>
```

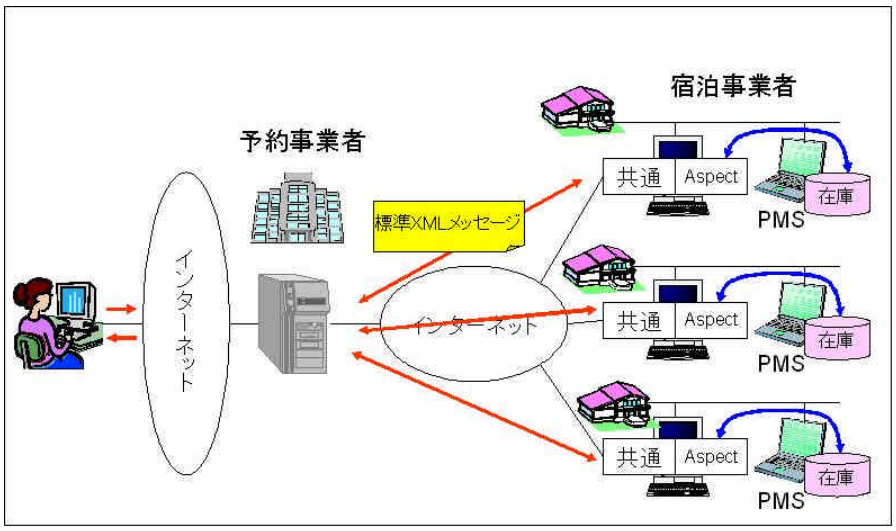


図４　アスペクト指向を取入れた旅館予約システム

## ３．アスペクト指向を取り入れた宿泊事業者予約システム

図２に示したシステム[1]の処理の流れを図 5 に示す。図 5 に示す処理の中で，宿泊事業者のデータベースにアクセスする必要がある(4)の在庫照会に対する回答，(5)の予約基本処理(予約要求)に対する回答，(6)宿泊者の詳細情報受入れ処理に対してアスペクト指向プログラミングの考え方を導入した。すなわち，旅行電子商取引促進機構が策定した，標準 XML 形式で予約事業者とメッセージ交換をする予約事業者とのインターフェース部分（図５太線部分の処理）と，各宿泊事業者固有の PMS とのインターフェース（図 5 の網掛け部分の記述）とを明確に分離し,後者は対象とする PMS により異なるので,アスペクトとして記述することにした。以下では，(5)の予約基本処理（予約要求）に対する回答処理について詳述する。

表４に，予約事業者から送られてきた予約要求に対する，宿泊事業者側の処理をするクラスの概要を示す。すなわち，

① 予約事業者から送られてくる標準 XML フォーマットのメッセージを解読し，予約に関するデータを取り出し，宣言された変数にセットする処理

② PMS に問い合わせ(③)をし，予約が取れる場合には空き部屋数を更新し(④)，宿泊事業者の予約番号を得て(⑤)，宿泊者情報を PMS のデータベースに書込む処理(⑥)を行なうメソッド AccessDB を呼び出す。

⑦ PMS の結果を受けて，回答用標準メッセージを作成する処理

から構成されている。このうちで，③の部屋残数を得る getRoomZan()メソッドは，ここでは常に 0 を返すように作成してある。したがって，④～⑥の処理を行なうことはないが，表４に示してあるように，何もしないことが記述されている。

これに対して，各 PMS に対応するアスペクトの枠組みを表５に示す。基本的には PMS のデータベースにアクセスする部分を，各 PMS に対応して記述することになる。

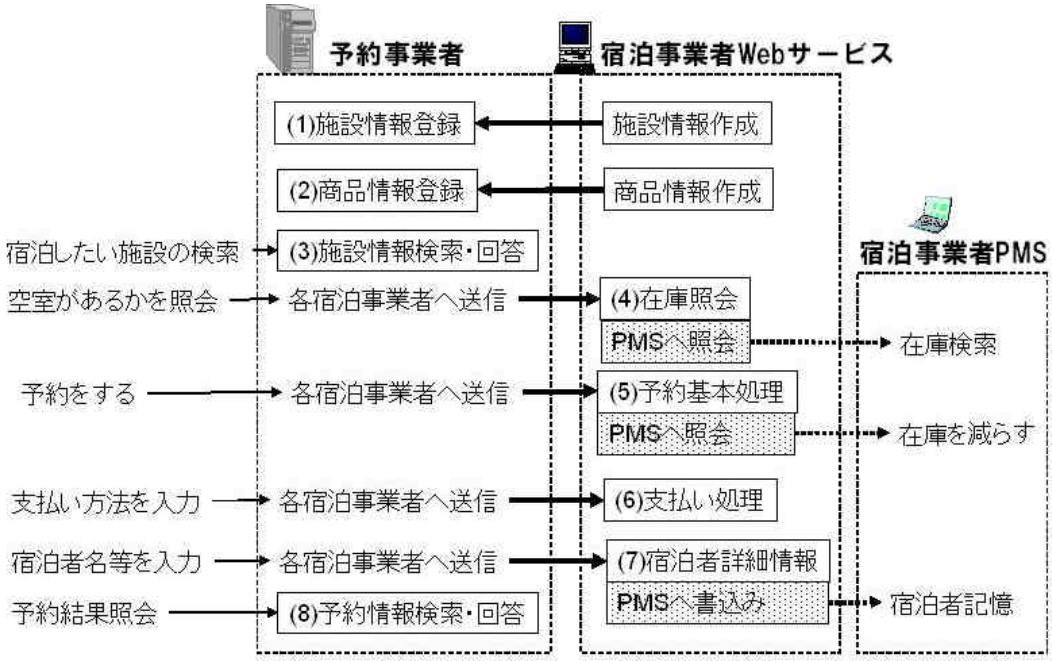


図５　アスペクト指向を取入れた予約システム

表４　予約要求に回答するプログラム

```
public class BookingResponse {
    static ・・☆・・      // 各種データをセットする変数の宣言
    static String confnum = null;          // 宿泊事業者の予約番号
    static String bookingcode = null;      // 空室があったら HK，満室だったら UC
    static int roomtype;                   // 部屋タイプ
    public Element [] answer(Element [] elems) throws Exception{
        XML を解読し，ホテル ID, 予約事業者 ID，宿泊日等を☆にセット   // ①
        AccessDB();                                                    // ②
        返信用 XML 作成                                                // ⑦
```

```
        return 返信用 XML;
    }
    static void AccessDB(){
        int RoomZan = 0;                                    // 空き部屋数を一時的に記憶する変数
        BookingResponse booking = new BookingResponse();
        roomZan = getRoomZan(checkinDate,roomtype);         // ③ 宿泊日，部屋タイプを与えて空き部屋数を得る
        if(RoomZan>0){
            updateRoomZan(checkinDate,roomtype,roomZan);            // ④ 空き部屋数を１減らす
            getConfNum(booking,checkinDate,roomtype,roomZan);       // ⑤ 宿泊事業者の予約番号を得る
            InsertGuestData(booking,checkinDate);                   // ⑥ 宿泊者情報書込み
            bookingcode = "HK";
        } else {
            bookingcode = "UC";
        }
    }
    private static int getRoomZan(String date,int room){
        return 0;
    }      // date:宿泊日　room:部屋タイプ
    private static void getConfNum(BookingResponse booking,String date,int room,int zan){
    }      // booking:本体クラスのオブジェト　date:宿泊日　room:部屋タイプ　zan:空室数
    private static void InsertGuestData(BookingResponse booking,String date){
    }      // booking:本体クラスのオブジェト　date:宿泊日
    private static void updateRoomZan(String date,int room,int zan){
    }      // date:宿泊日　room:部屋タイプ　zan:空室数
}
```

宿泊事業者の予約番号を得る getConfNum メソッドに対しては，アスペクトから本体の宿泊事業者予約番号，予約日時を記憶する変数にアクセスできるように，クラス BookingResponse のオブジェクトを引数として渡している。また，顧客情報を必要とする InsertGuestData メソッドに対しても，顧客情報は送信されてくる XML メッセージに詳細に記述され，本体のプログラムで解読されているので，パラメータとしてクラス BookingResponse のオブジェクトを与えている。

表５　予約要求に対する PMS とのインターフェースを記述するアスペクト

```
public aspect IGuestData {

//宿泊日，部屋タイプを与えて部屋の残数を得る getRoomZan を書き換えるアドバイス
    int around(String date,int room) :
        execution(private static int BookingResponse.getRoomZan(String,int)) && args(date,room) {
         int nRoomZan=0;
         日別の空き部屋数を記憶しているテーブルから，引数で指定された宿泊日の行を読み取る;
         nRoomZan =  上記により得られた ResultSet の，引数で指定した部屋タイプ・フィールドのデータ;
         return nRoomZan;
    }
//空き部屋数を１減らす updateRoomZan を書き換えるアドバイス
    void around(String date,int room,int zan) :
        execution(private static void AccoBookingResponse.updateRoomZan(String,int,int))
            && args(date,room,zan){
        日別の空き部屋数を記憶しているテーブルから，引数で指定された宿泊日の行，room で指定された列の値を
        zan-1 に更新する;
    }
```

```
//宿泊事業者の予約番号を得る getConfNum を書き換えるアドバイス
    void around(BookingResponse booking,String date,int room,int zan) :
        execution(private static void BookingResponse.getConfNum(BookingResponse,String,int,int))
            && args(booking,date,room,zan){
        宿泊施設が与える予約番号を記憶しているテーブルにアクセスし，その番号を booking.confnum にセットする;
        そのテーブルの予約番号を１増加する;
        booking.bookingdate（予約日）を更新する;
        booking.bookingtime（予約時刻）を現在時刻に更新する;
    }
// 顧客情報を書き込む InsertGuestData を書き換えるアドバイス
    void around(BookingResponse booking, String date) :
        execution(private static void BookingResponse.InsertGuestData(BookingResponse,String))
            && args(booking,date){
        商品情報からこの商品が指定する食事コードを得る
        顧客情報を記憶しておくテーブルに顧客情報(名前，連絡先，男女別，大人子供別，到着手段，チェックイン日時等)，選択した食事コードを書込む;
    }
}
```

## ４.アスペクト指向を取り入れた宿泊事業者予約システムの実装

文献[1]で報告した試行システムで用いた株式会社アルゴ２１のフロントくん Win[8]，およびアスペクト指向の利点を検証するために自作した簡易 PMS を対象にして，２つの宿泊施設側のシステムを開発した。(予約事業者側のシステムは，文献[1]のものをそのまま使用した。)

文献[1]で報告したシステムでは，宿泊施設側の予約処理は，予約事業者からの XML メッセージを処理する部分と，PMS とのインターフェース部分との切り分けがきちんとできていなかったので，宿泊事業者側のフロントくん Win に対するものも再構築したが，それが完成したあとの自作の PMS に対するアスペクトの作成は数日程度の作業で完成することが出来，アスペクト指向の考え方による開発期間の短縮が実現できたものと考えられ，今後他の PMS に対するシステムも容易に作成できる見通しが得られた。

なお開発は，Eclipse に AJDT（AspectJ Development Tools）[6][7]をプラグインしたものを用いて行なった。

## ５．考　察

アスペクト指向プログラミングのコードの正しさについては，本体のプログラムと各アドバイスのそれぞれの織り込み点での影響を考慮しなければならず，分かりにくいといわれている。また，開発に当たっても，本体のプログラムを充分に理解したうえで，アスペクトを作成しなければならない。したがって，この方法を実用化するにあたって，予約事業者との間のメッセージの解読方法およびその結果を記憶してある変数名等を知らなくても，PMS との間のインターフェースをつかさどるアドバイスを開発できるようにするために，各アスペクトの機能を明確に定義したマニュアルを用意し，更にアスペクトのアドバイスを各 PMS に対応して直接コーディングするのではなく，図６に示すように，必要なデータをユーザーの言葉で表わし，それをどのテーブルのどのフィールドに書込むかを記述するだけで，アドバイスを自動生成するユーザ・インターフェースの開発が必要であろう。

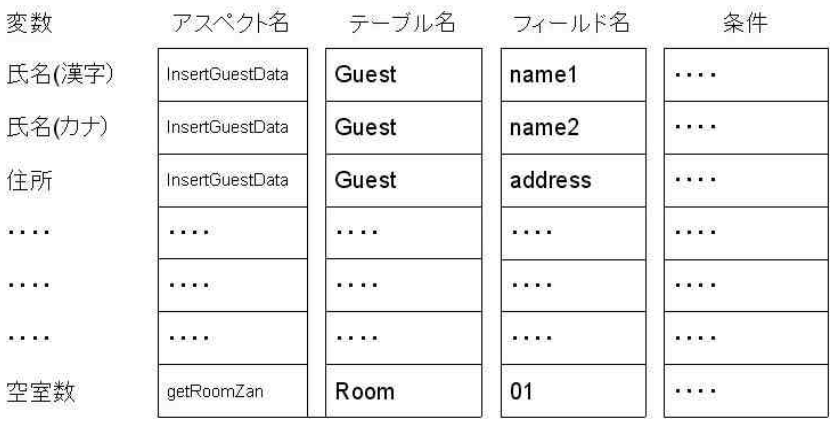


図６　アスペクト開発のためのユーザ・インターフェース

## 参考文献


[1] 飯田 善久，中島 玲子，宮本 聡子：次世代型宿泊予約システムの開発，成蹊大学理工学研究報告，Vol.42，No.1，pp13-18，2005 年 6 月

[2] 旅行電子商取引促進機構，http://www.travel-edi.com/

[3] ジェイアール総研情報システム：旅館施設情報提供および予約業務における業務プロセス及び情報モデル調査，2004 年 3 月

[4] 千葉 滋：アスペクト指向ソフトウエア開発とそのツール，情報処理，Vol.45，No.1，pp28-33，2004 年 1 月

[5] Gregor Kiczales: Aspect-Oriented Programming, http://www.cs.ubc.ca/~gregor/papers/kiczales-ECOOP1997-AOP.pdf

[6] 立掘 道昭：AspectJ によるアスペクト指向プログラミング入門，ソフトバンクパブリッシング，2004 年 4 月

[7] AJDT (AspectJ Development Tools), http://www.eclipse.org/ajdt/

[8] (株)アルゴ２１，http://www.argo21.co.jp/hotel/frontkun_win.html