人間的な、あまりに人間的な
ーソフトウェア開発工程と設計ー

岐阜経済大学　経営学部 経営情報学科　井戸　伸彦

来歴:

０．０版　２００３年１２月５日

スライドの構成

はじめに

（１）ソフトウェア開発工程

（２）ユースケース図

（３）シナリオ、イベントフロー

（４）要求分析から設計へ

（５）DFD

（６）POPKINの使い方

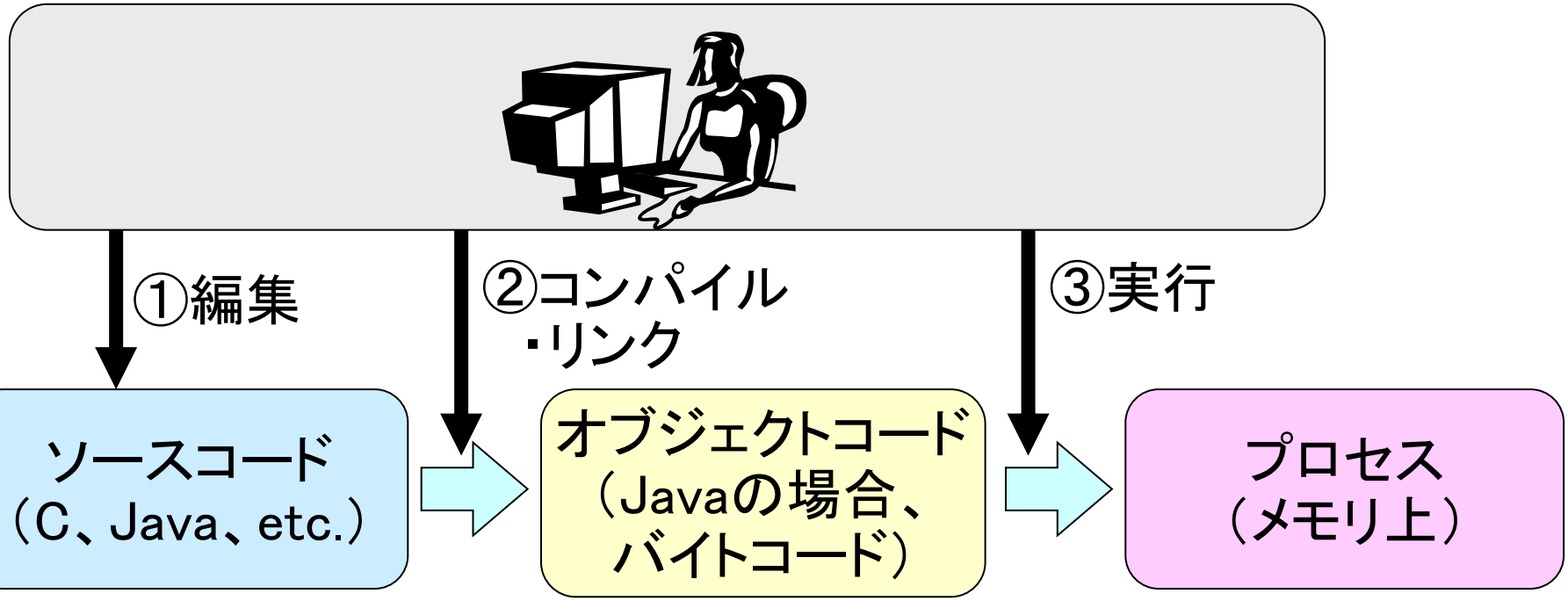
はじめに

- 受講者は、初歩のプログラミングの経験があるものとしています。
- 「ああ、オブジェクトにしあらましかばーオブジェクト指向プログラミングー」のスライドを学習済みであるものとしています。
- ツールとして、”Jude”、“POPKIN（デモ版）”を用います。
- 説明は直感的な分かりやすさを重視し、厳密には不正確な言い方もしています。

（1．1）“プログラムを作る”こと

■“プログラムを作る”という作業は、単純に考えることが出来ます。

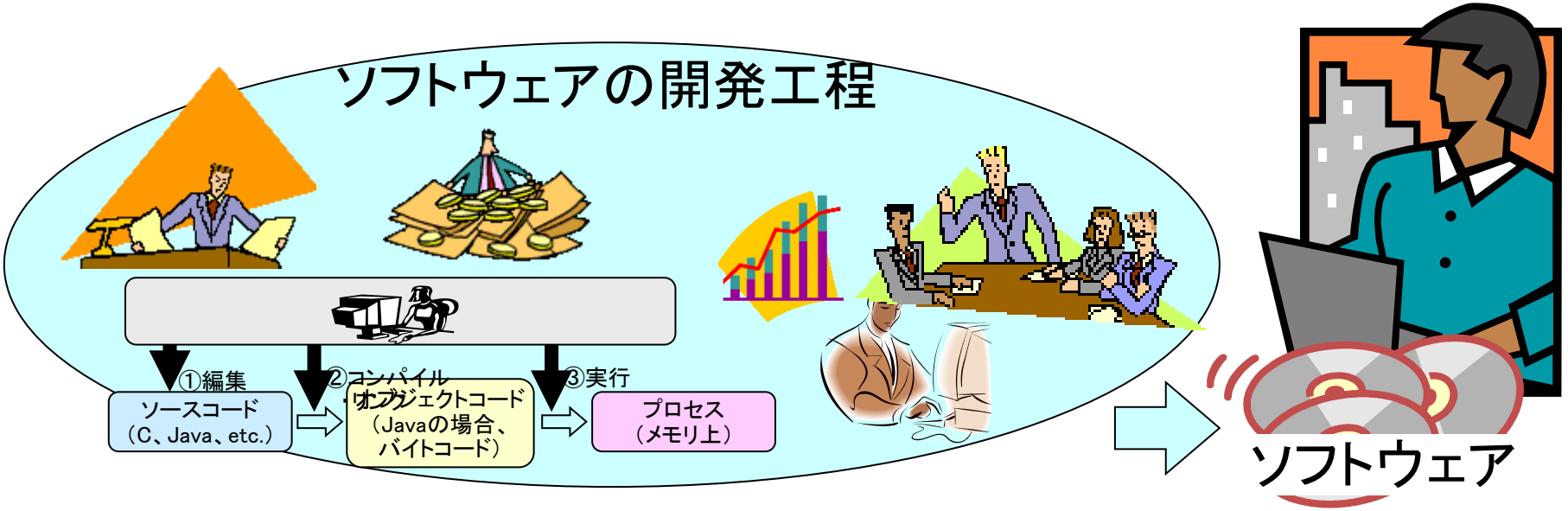
- ①ソースコードを編集し（メモ帳でJavaファイルを編集する）、
- ②コンパイル(リンク)を行い（ **% javac Window.java**）、
- ③実行させるだけです（ **% java Window**）。



（1．2）“ソフトウェアを開発する”こと

■日常私たちが利用する “ソフトウェア”を開発することは、もちろん、“プログラムを作る”ことも含んでいますが、それほど単純ではありません。

■この“ソフトウェアの開発工程”を、効率を上げることを目的に分析・整理していくことが必要になります。これを「ソフトウェア工学」と言います。



（1．3）ウォーターフォール(Water Fall)モデル

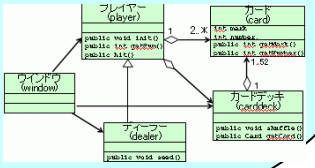
■ソフトウェア開発工程をどのように捉えるかについて、オーソドックスな考え方に、ウォーターフォール（＝滝）モデルといものがあります。

■ウォーターフォールモデルでは、滝が流れ落ちるように、各段階（工程）が一方方向に進んでいき、後戻りがないことを想定しています。

（1．3．1）上流工程

＜要求分析＞
ソフトウェアのユーザが、何を要求しているかを整理して、どのようなソフトウェアを開発するかを確定します。

このようなクラスを作ろう。クラスの関係は、こうしておこう。



＜仕様作成＞
ソフトウェアの機能・動作がどのようなものであるかを確定します。

＜プログラム設計＞
プログラムの構造を決めます。

ミニテストもね。

Web上で出席が取りたいの？

出席者は乱数を入力する。
講師は出席一覧表を閲覧する。

（1．3．2）下流工程

＜コーディング＞
プログラム（ソースコード）を作成します。

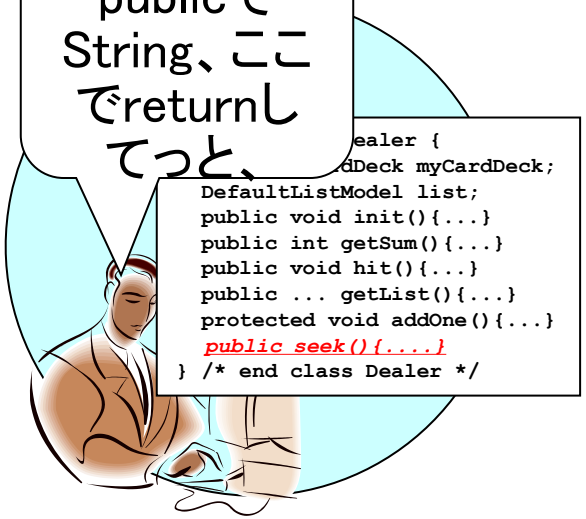
＜テスト＞
プログラムが意図したとおり正常に動くかを確認します。

運用開始

＜保守＞
運用中に見つかったバグを修正したり、新たに生じた要求に対応したりします。

これは不便でおかしいよ。直そう！

えーと、publicでString、ここでreturnしてっと、



```
ealer {
dDeck myCardDeck;
DefaultListModel list;
public void init(){...}
public int getSum(){...}
public void hit(){...}
public ... getList(){...}
protected void addOne(){...}
public seek(){...}
} /* end class Dealer */
```

あれっ？動かないぞ。。。そうか、ここで代入しなきゃ。

（1．3．3）成果物とレビュー

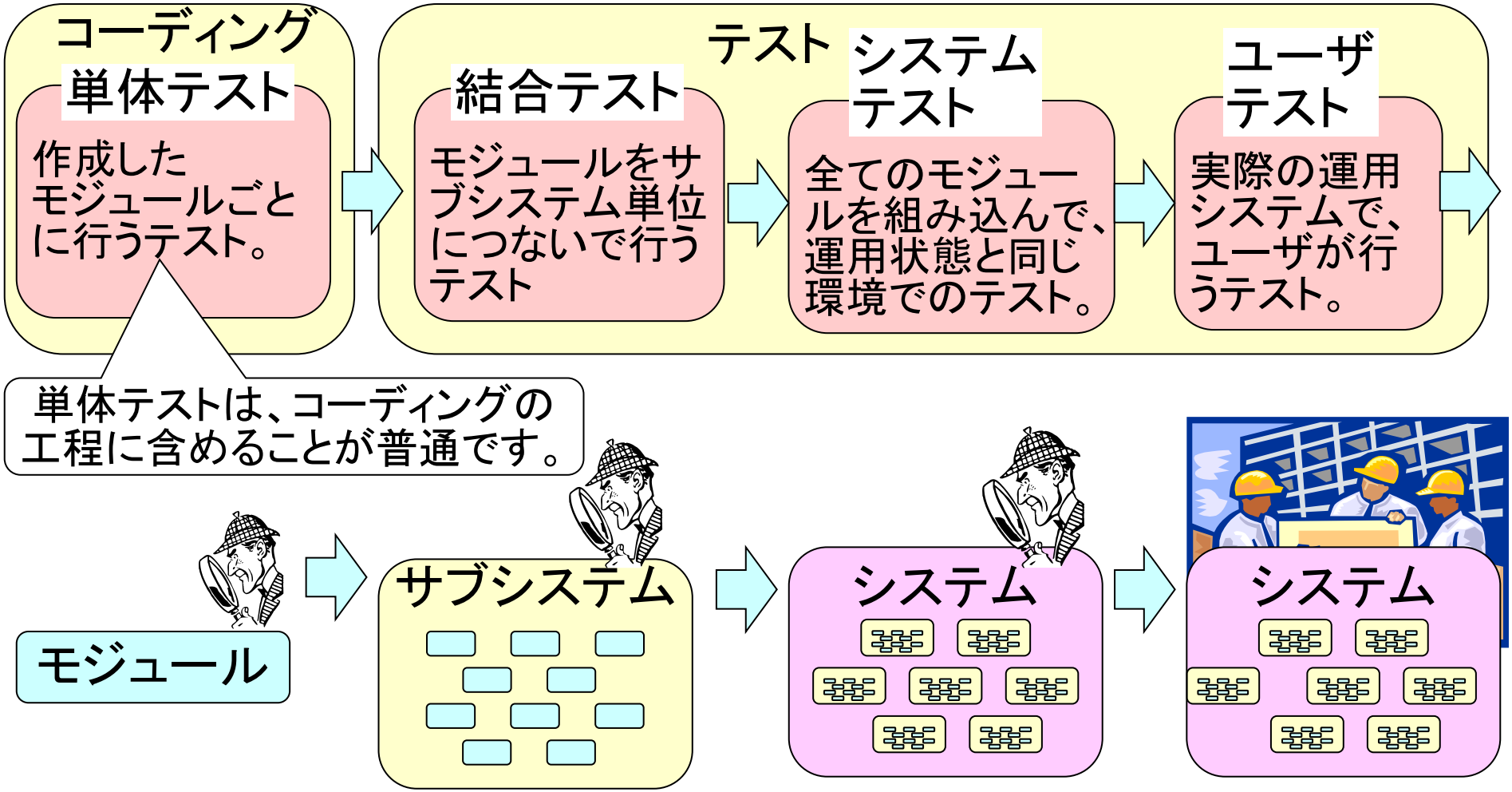
■ウォーターフォールモデルでは、各工程での成果物を文書化し、これをレビュー（点検）して、“後戻りしない”で良いことを確認していきます。

要求仕様書
要求分析
システム仕様書
仕様作成
基本設計書
プログラム設計
プログラム
コーディング
システム仕様書
テスト
保守

※　文書の呼び方はさまざまです。上記は一例です。

（1．3．4）テスト工程

■テストエ程は、さらに、分割して考えることが普通です。

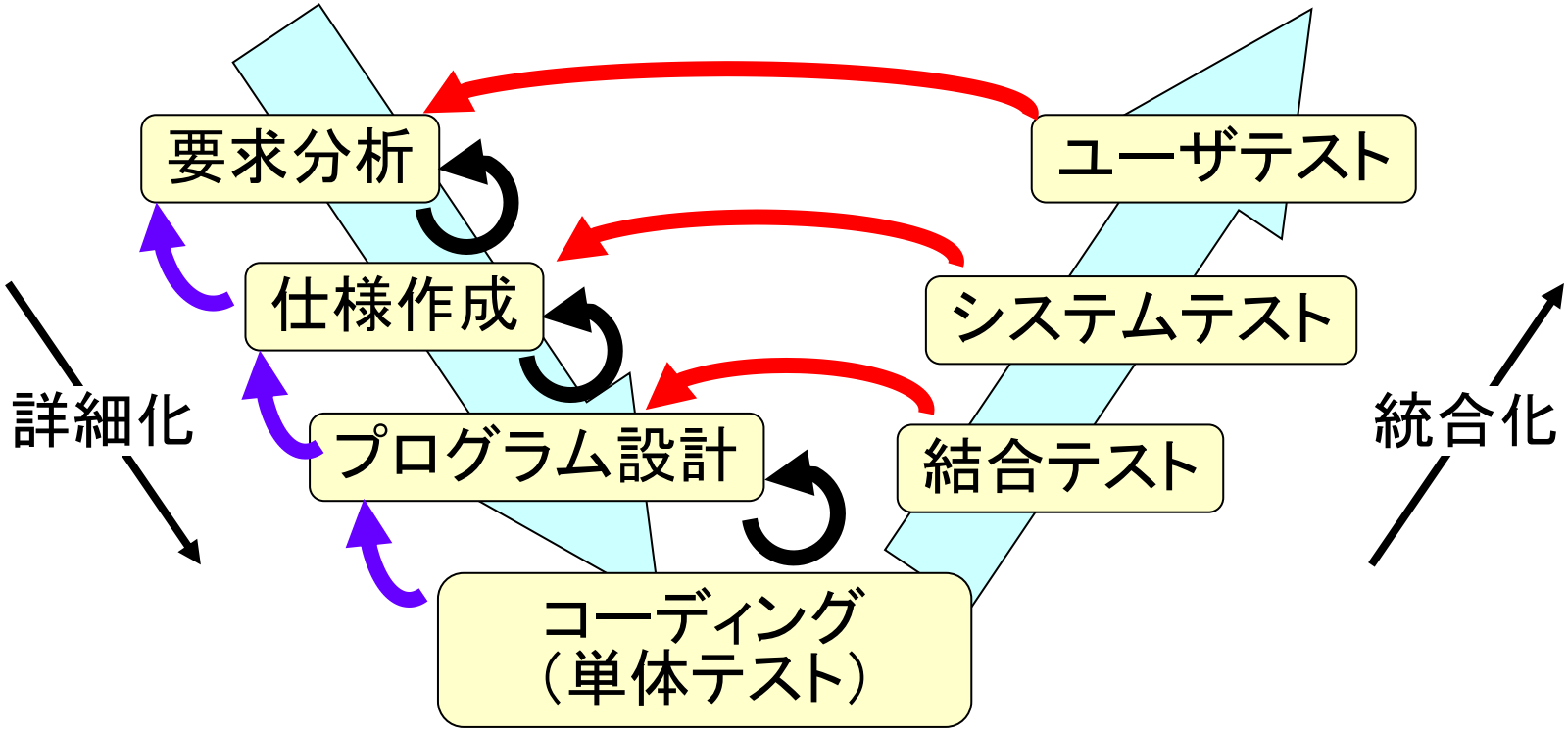


※モジュール：Javaのクラスのような、プログラムの最小単位のことです。

（1．3．5）V字モデル

■プログラムの作成過程では、①前の過程に沿った形で進められているか（ ）、②自己完結しているか（ ）をチェックしながら、徐々に詳細化が図られます。

■プログラムのテスト過程では、仕様が正しく実現されているか（ ）をチェックしながら統合化が図られます。

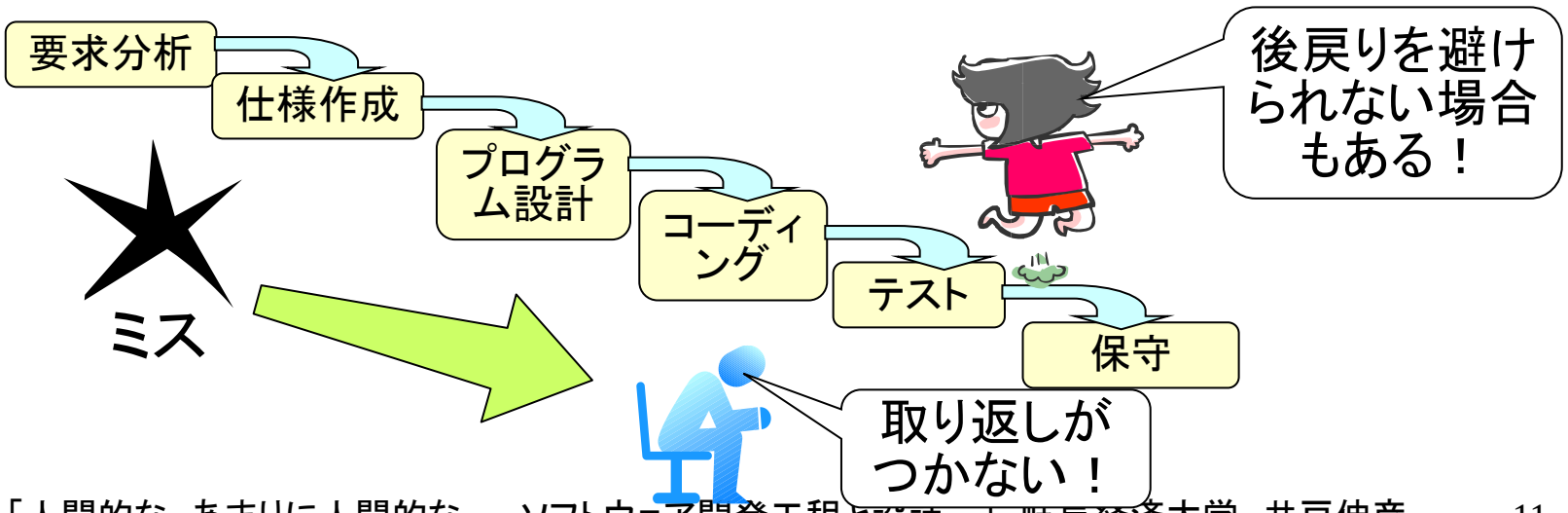


（1．4）ウォーターフォールモデルの限界

■ウォーターフォールモデルが機能しないという訳ではありませんが、次のような限界があることが知られています。

- 初期の段階での問題を見逃すと、後の段階で致命的なトラブルになってしまう。
- 実際には後戻りすることは避けられない場合がある。

■大雑把に言うと、「ウォーターフォールモデルの理想通りに整然とはことは進まない」という訳です。



（1．5）プロトタイピング、スパイラルモデル

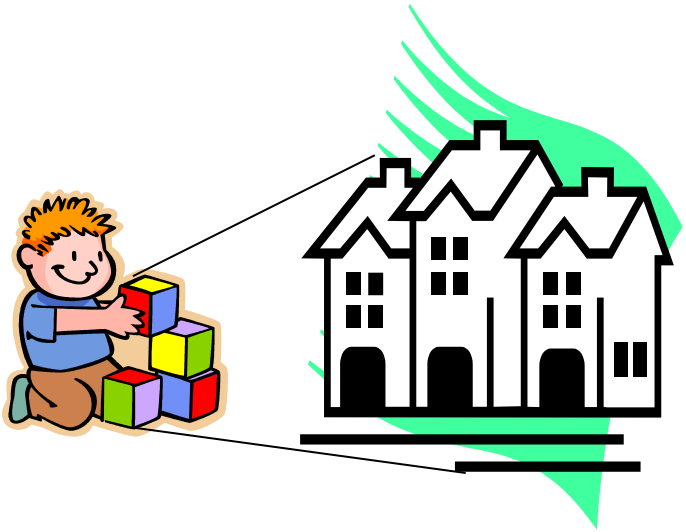
■前スライド(1.4)のような問題を乗り越えるためのモデルとして、いろいろな考えが生まれてきました。

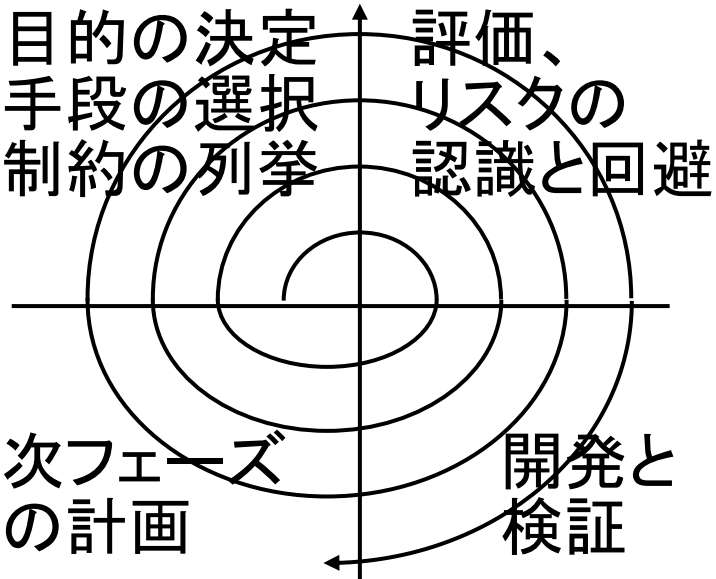
■プロトタイピング

- 初期段階に試作品（プロトタイプ）をまず作って、これをソフトウェアの発注者に示し、フィードバックを得ながら要求仕様を固めていく方法。

■スパイラルモデル

- 開発工程を、「計画」、「目的や手段の選択・列挙」、「手段・リスクの評価」、「開発と検証」の繰り返しと捉える方法。

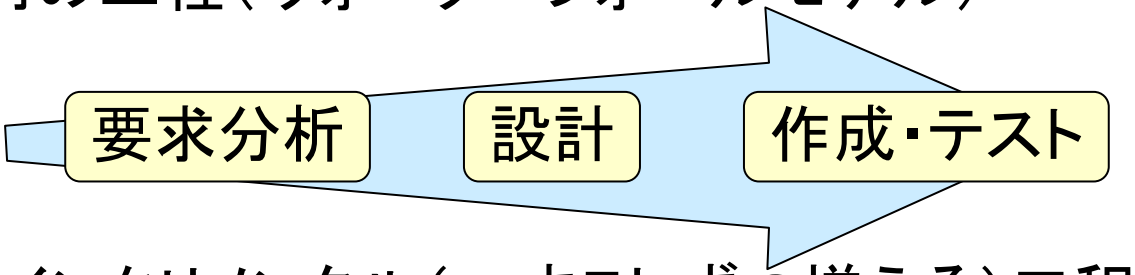




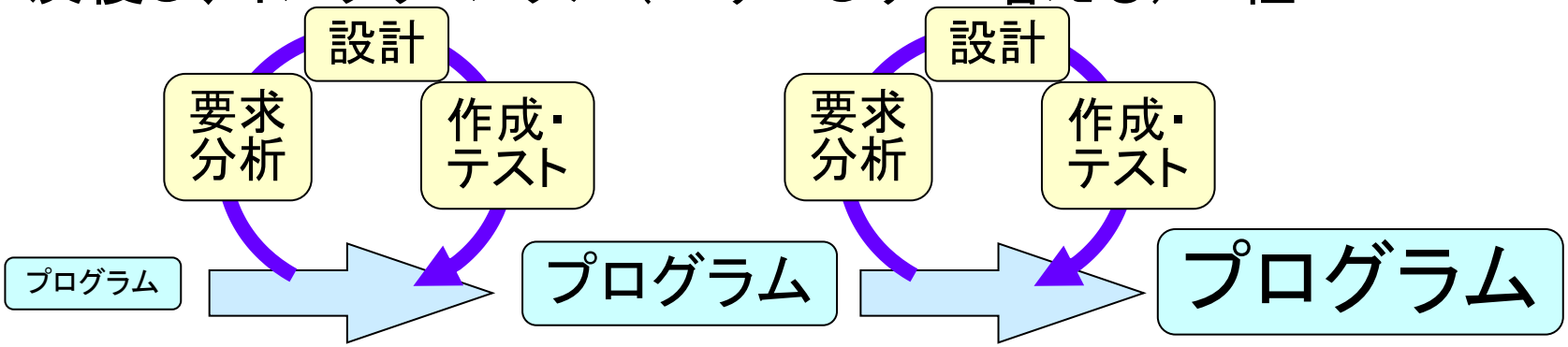
（1．6）反復的、インクリメンタル

■前スライド(1.5)の2つの手法は、ウォーターフォールモデルに対して、次のように対比することが出来ます。

- 一方方向の工程（ウォーターフォールモデル）



- 反復し、インクリメンタル（＝すこしずつ増える）工程



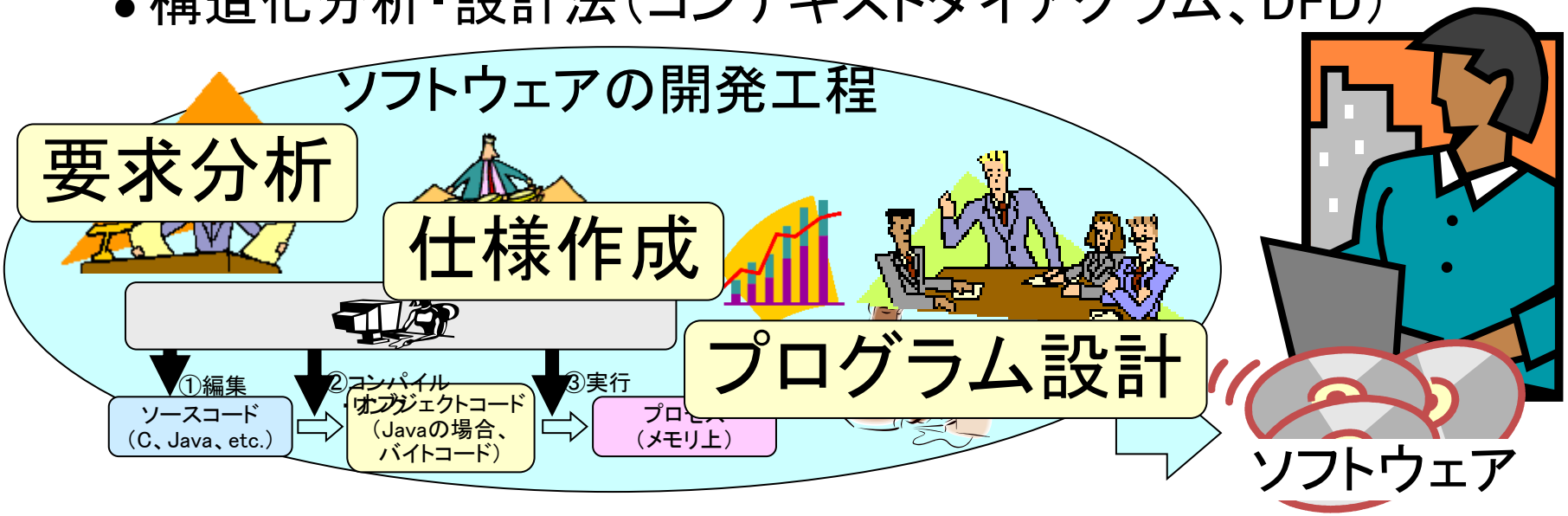
■反復的でインクリメンタルなモデルとして、UP（Unified Process）やXP(eXtreme Programming)などが現在注目されています。

（1．7）本スライドでの目標

■本スライドでは、ソフトウェア開発工程のうち、要求分析、仕様作成からプログラム設計への道筋に焦点を当てます。

■これらの工程の中で有用な手法として、次の2つの方法論を扱います。

- UP（UMLを含む）
- 構造化分析・設計法（コンテキストダイアグラム、DFD）



（1．8）Unified Process(UP)とUML

■UPは、次の3つを大きな特徴とする開発プロセスです。
- ユースケース(use case)駆動
- アーキテクチャ中心
- 反復的でインクリメンタル

■ここでは、それぞれの内容の詳細については触れません。UPは様々な側面（モデル）を持ちますが、本スライドでは話題を絞って扱っていきます。

■UMLは、UPの考え方に従って規定されています。

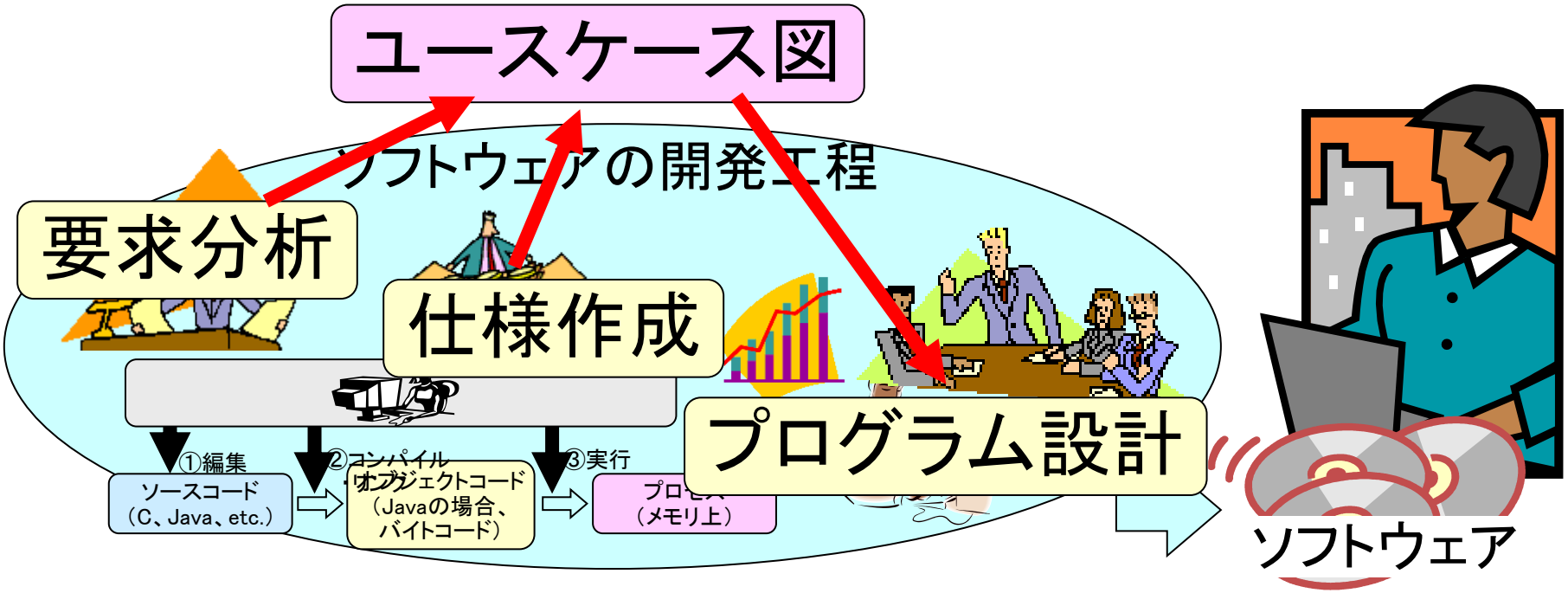
■UP/UMLは、ソフトウェア工学にて指導的な立場にある3名の研究者(Ivar Jacobson, Grady Booch, James Rumbaugh)が、オブジェクト指向モデリングの世界に標準を設けようという考えから、互いの手法を統一（Unify）しようとした結果誕生しました。

■1997年、UMLはOMG（Object Management Group）より標準勧告を受け、現在ではオブジェクト指向モデリング記述言語の標準として、広まりつつあります。

（2）ユースケース図

■ユースケース図の作成は、要求分析、および、仕様作成の工程にて作成されます。

■UPでは、全ての工程にて中心となる図として位置づけられています。



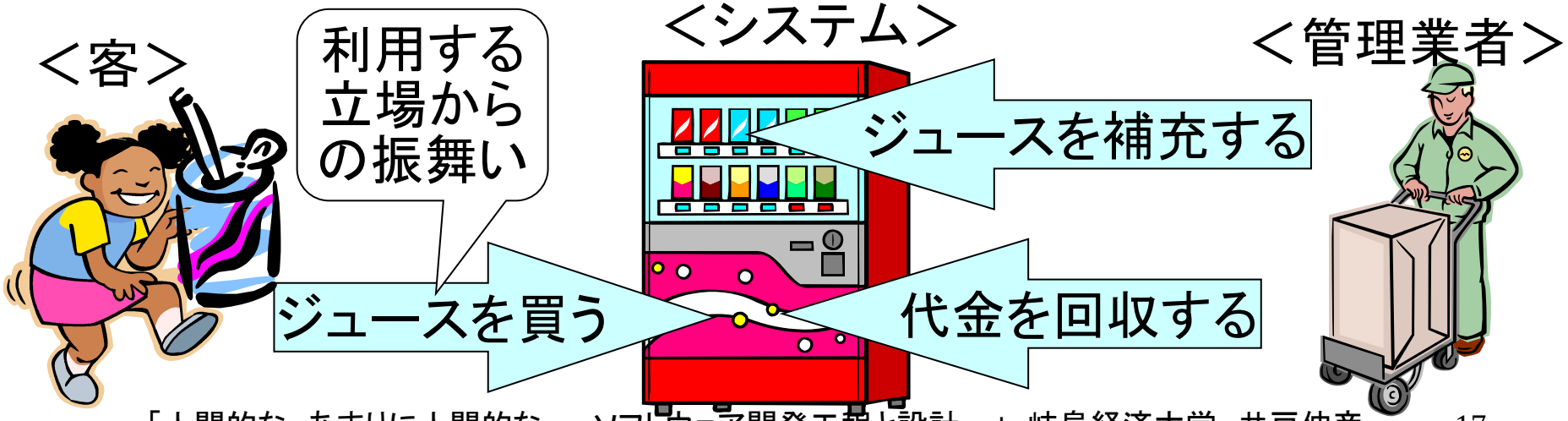
（2．1）ユースケース

■ユースケース(use case)とは？

- ユースケースとは、注目するシステムが期待されている動作・振舞いを、利用者が行為する立場から表現したものです。

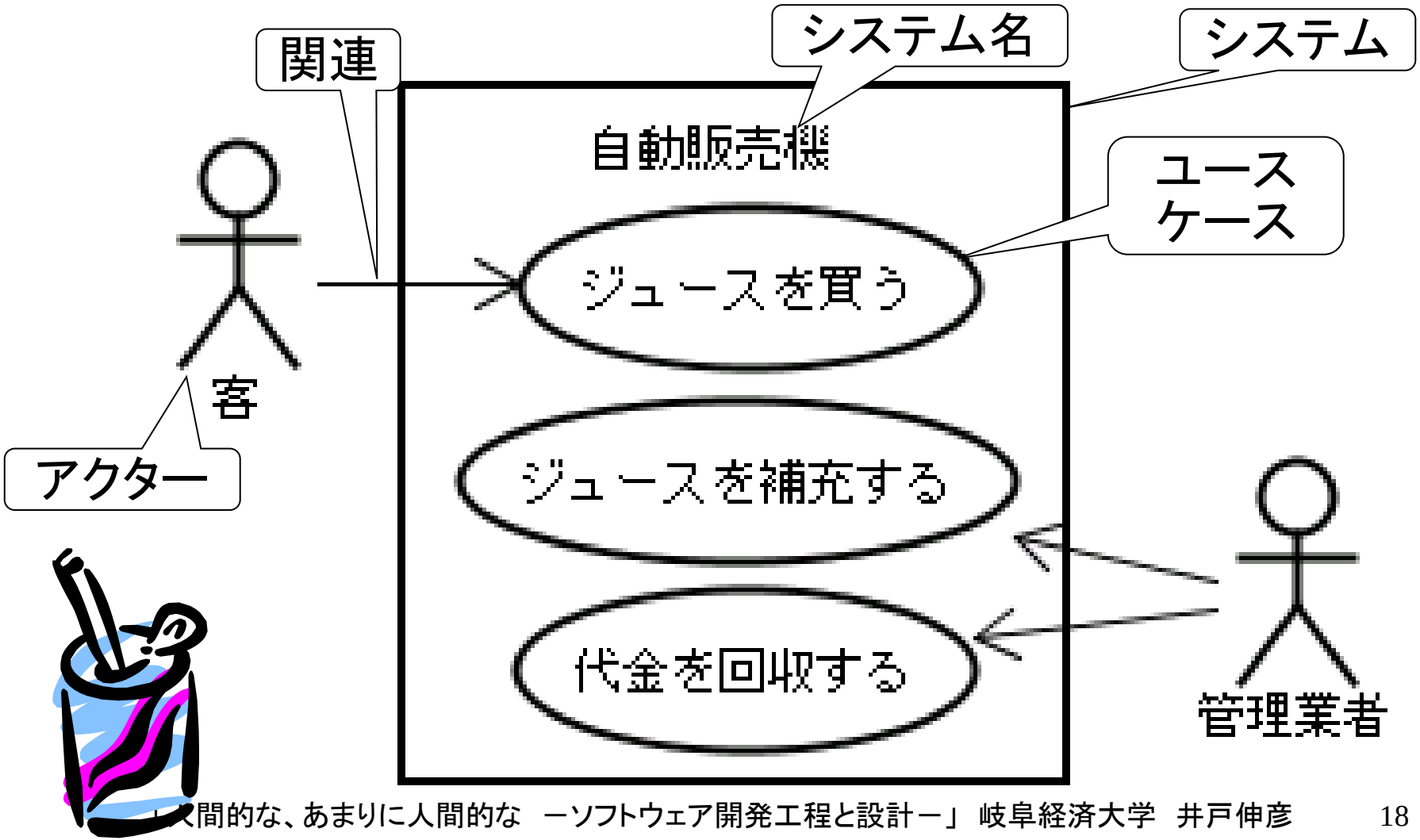
■例：自動販売機

- システムとして自動販売機を考えると、これに対して“客”は「ジュースを買う」という行為を行うことを期待しています。
- これに対して“管理業者”は、「ジュースを補充する」、「代金を回収する」といった行為を行うことを期待しています。



（2．2．1）ユースケース図で記す

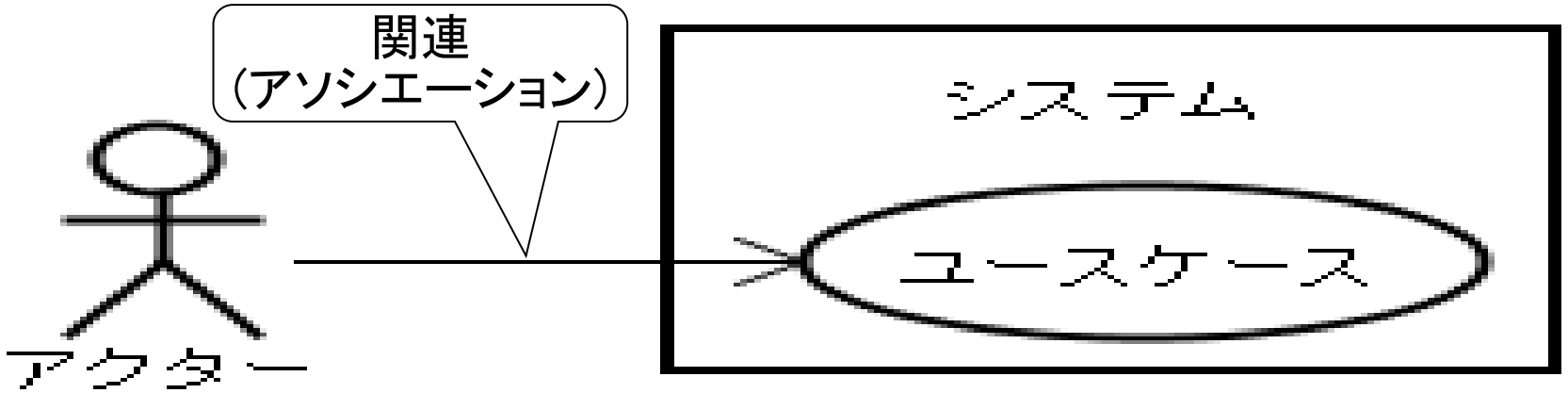
■UMLのユースケース図では、前スライド(2.1)の自動販売機のユースケースを、次のように記します。



（2．2．2）記法（アクター、ユースケース、関連）

■ユースケース図の一般的な形を整理します。

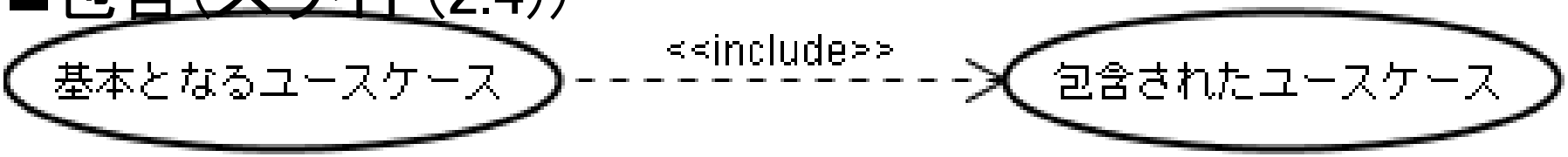
- システムを四角で表し、システム名を記します（これは解りやすさのために記すものであり、必須ではありません）。
- ユースケースは、システムの四角の中に楕円で記します。
- 利用者にあたるものを“アクター”と呼び、システムの四角の外に記します。
- アクターが関連付けられるユースケースへは、矢印を記します。
- アクターは人とは限りません。他のシステムが注目するシステムを利用する場合などは、それもアクターとなります。



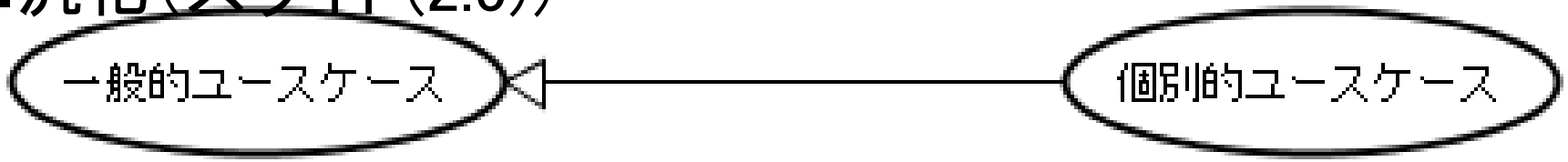
（2．3）ユースケース図の組織化

■次のスライド以下、(2.4)～(2.5)では、ユースケース図をさらに詳しくしていく方法のひとつめとして、包含・汎化・拡張の3つを見ていきます。

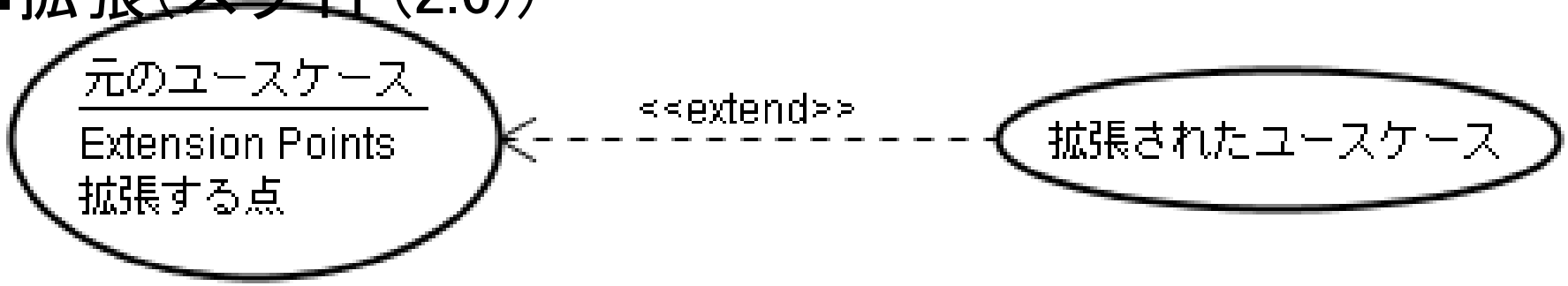
■包含（スライド(2.4)）



■汎化（スライド(2.5)）



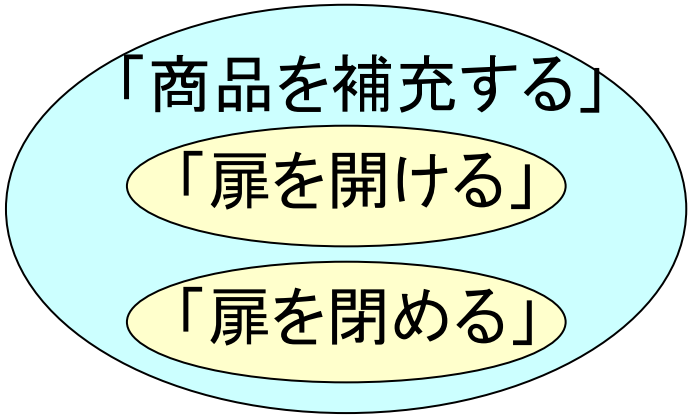
■拡張（スライド(2.6)）



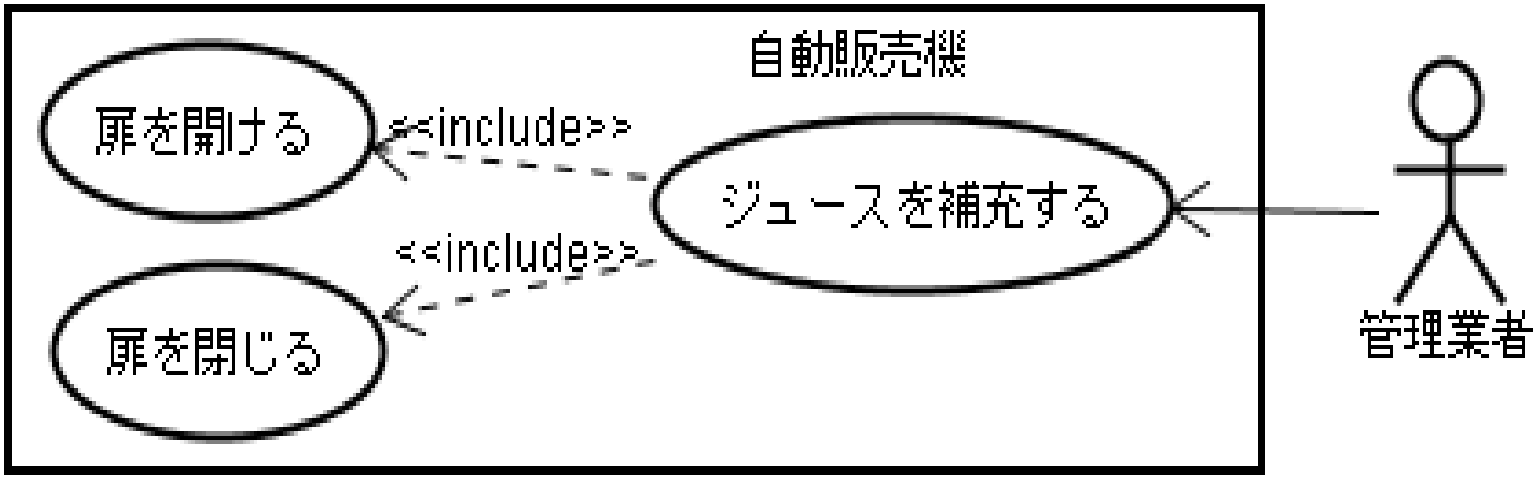
（2．4．1）ユースケースの組織化　－包含－

■自動販売機の例にて、「商品を補充する」というユースケースは、「扉を開ける」と「扉を閉じる」というユースケースを部分として含んでいます。

■このような場合、次のようにユースケース図を記します。



「商品を補充する」は、「扉を開ける」と「扉を閉める」を含む

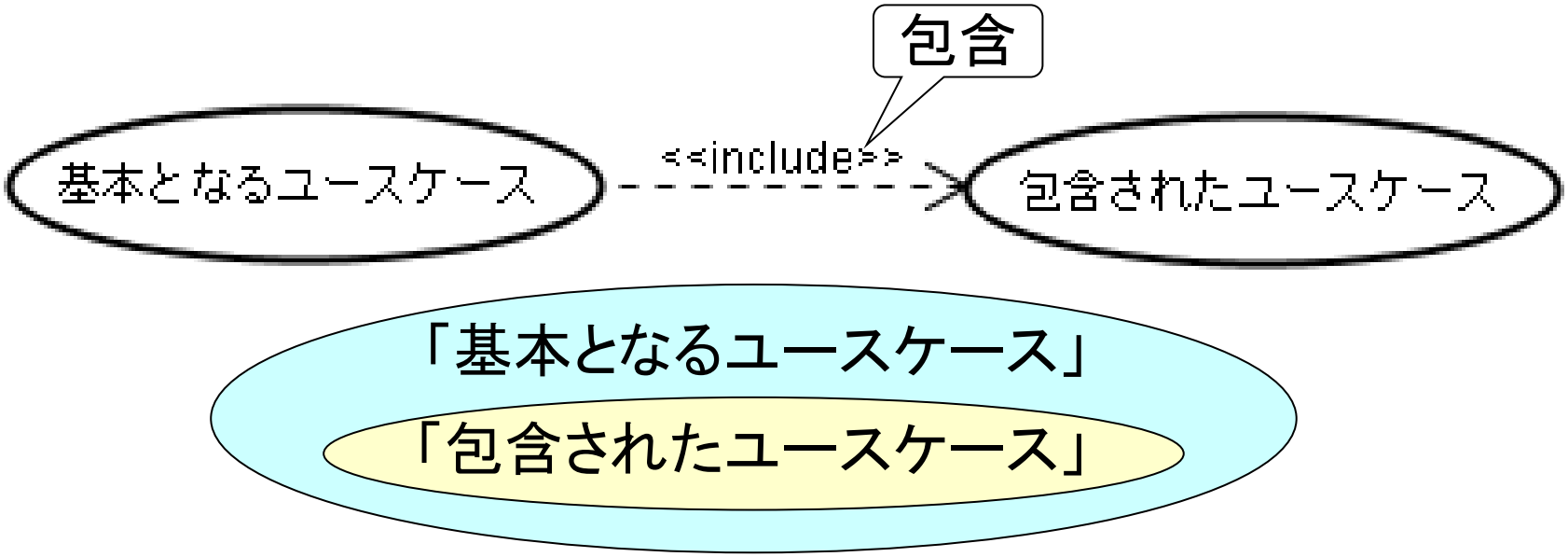


（2．4．2）包含の記法

■このように、あるユースケースが、別のユースケースを含んでいる場合、この関係を“包含”と言います。

■包含は、”<<include>>” (含む)というラベルを付けた破線矢印を用いて記します。

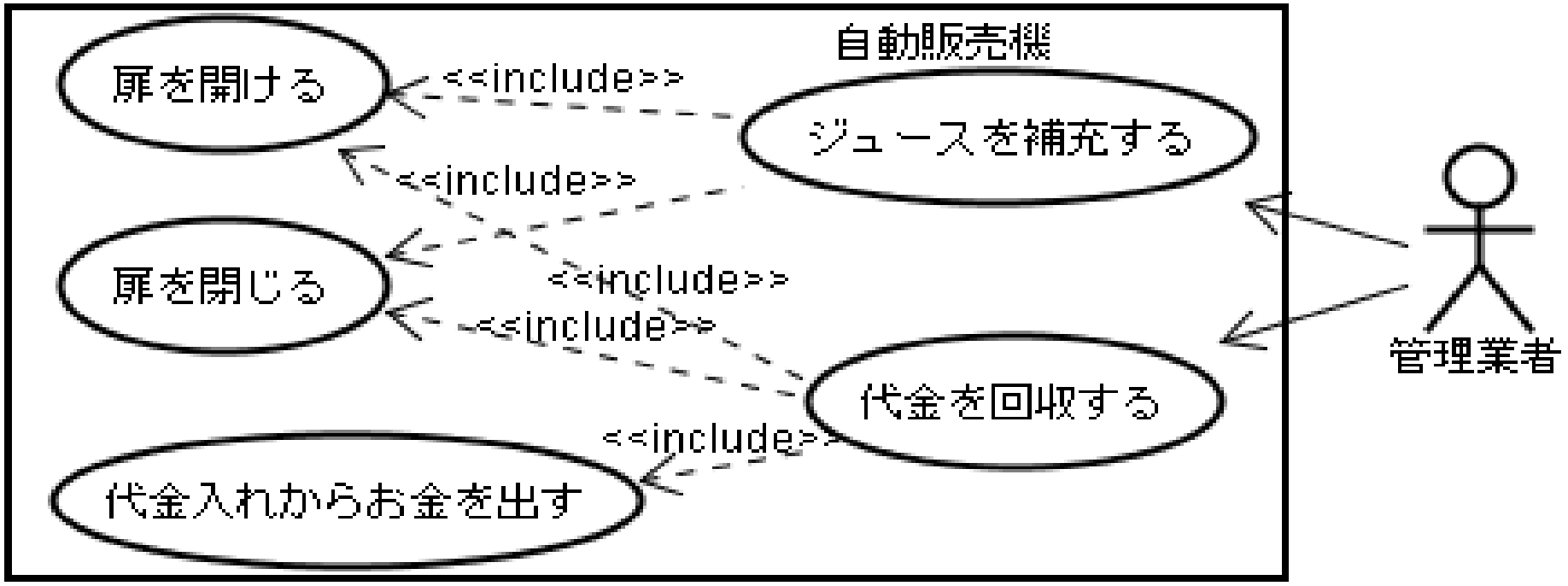
■矢印の方向は、「基本となるユースケース」から、「包含されたユースケース」に向かいます。



（2．4．3）包含を使うことの利点

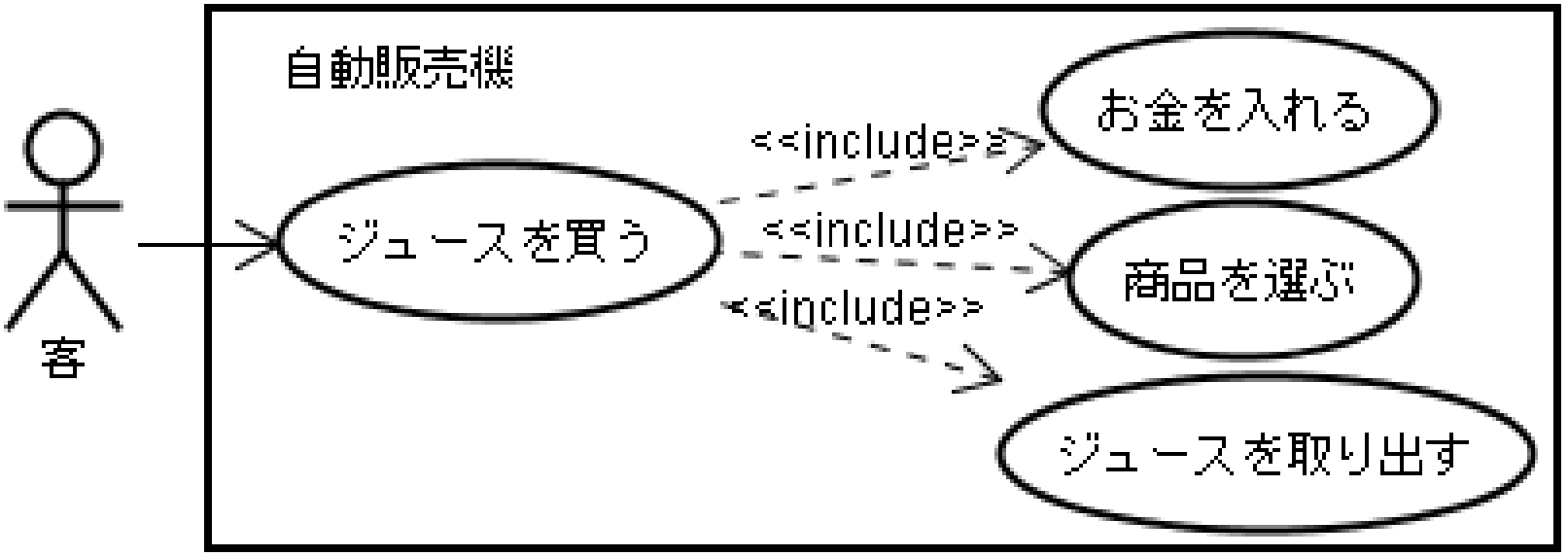
■包含を使ってユースケース図を作成することにより、他のユースケースにも出てくる部分的なユースケースを導き出すことが便利になります。

■下図のユースケース図では、扉の開閉のユースケースは、「代金を回収する」のユースケースにも含有されています。



（2．5．1）ユースケース図の組織化　－汎化－

■「ジュースを買う」というユースケースは、次の3つのユースケースを“包含”しています。

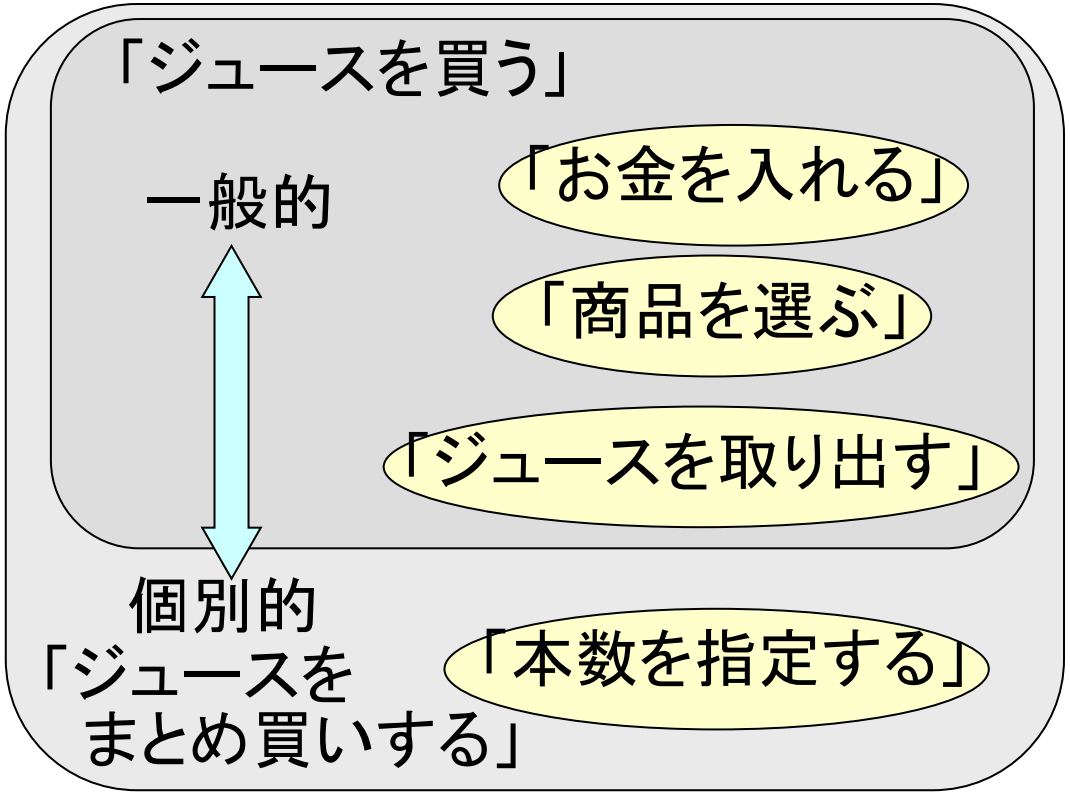


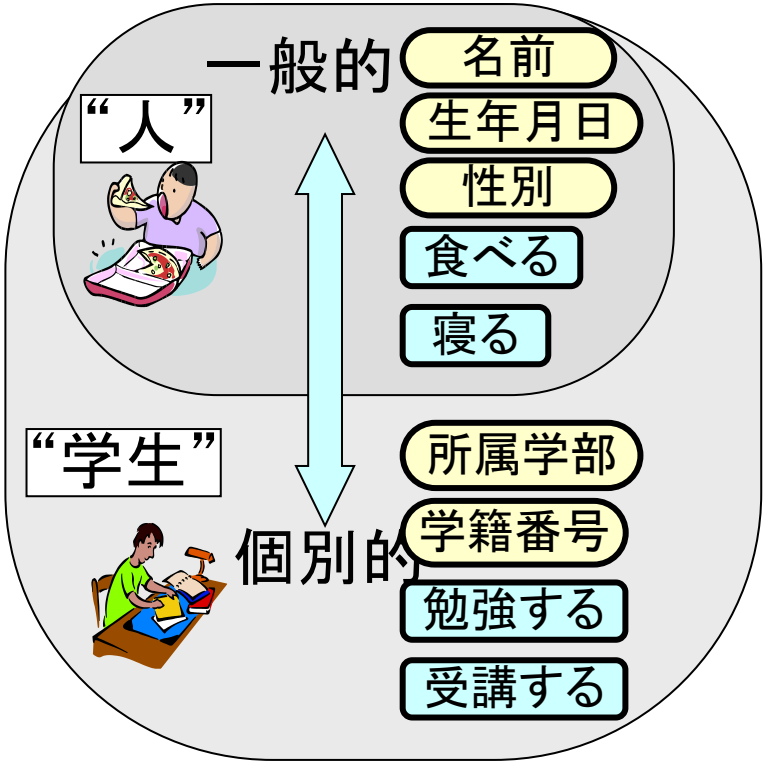
■今考える自動販売機には、「まとめ買い」の機能があるとします。すなわち、同じ商品を本数を指定して買うことが出来るとします。

（2．5．2）汎化の意味

■「ジュースを買う」というユースケースに、余分なユースケース「本数を指定する」を付け加えると、「ジュースをまとめ買い」というユースケースになります。

■これは、クラス図で見た継承(汎化)の関係と同じです。

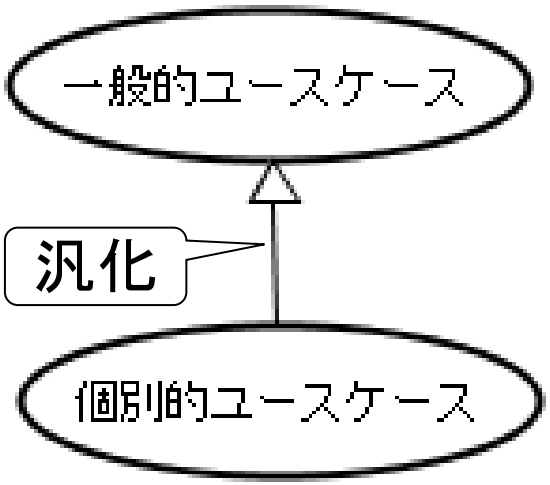


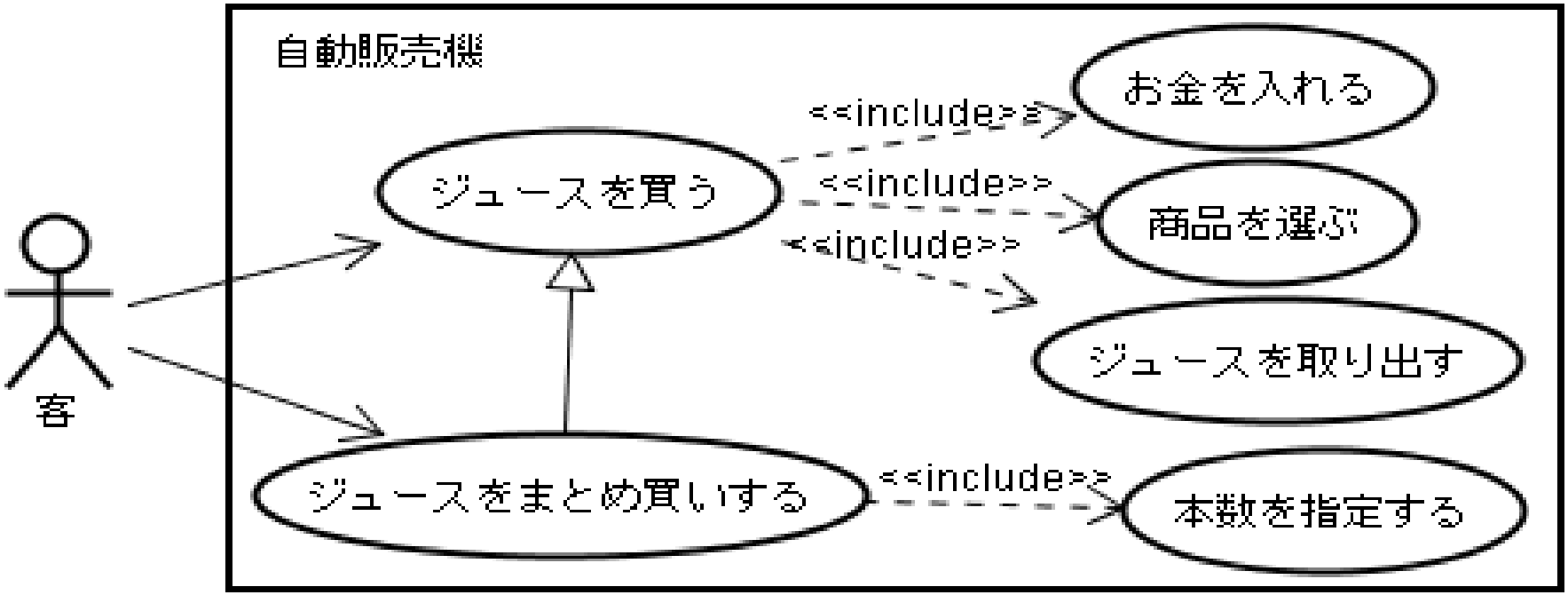


（2．5．3）汎化の記法

■ユースケース図でも、汎化（継承）の関係は、クラス図と同じく、△の矢印で記します。

■汎化を用いると、「まとめ買い」のユースケース図は、次のようになります。





（２．６．１）ユースケース図の組織化　－拡張－

■今、自販機についてつり銭のことを考えていませんでした。つり銭のことを考えるとき、“つり銭切れ”という事態が想定されます。

■“つり銭切れ”では、つぎのようなことになります。

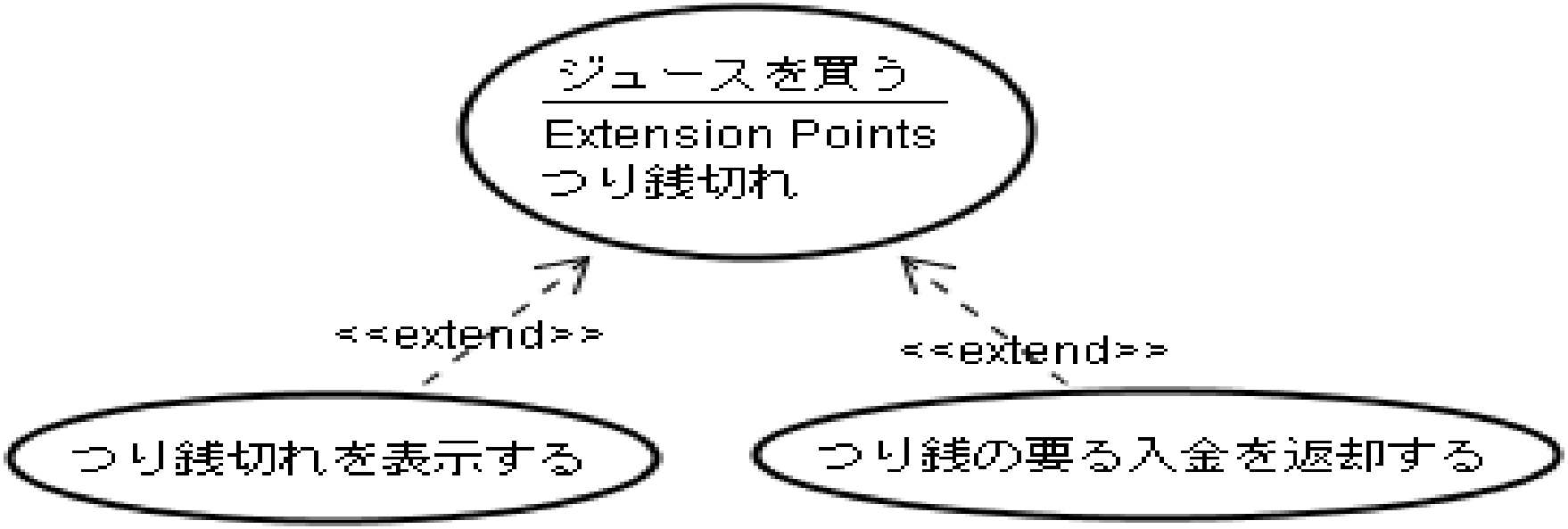
- つり銭の要らない販売は行う。
- つり銭切れの表示を出す
- つり銭の要る入金を、返却する。

（2．6．2）拡張の意味

■前スライドの“つり銭切れ”の状況でも、基本は「ジュースを買う」というユースケースは通じます。

■“つり銭切れ”という特定の状況下だけで発生する、「つり銭切れを表示する」と「つり銭の要る入金を返却する」という2つのユースケースを加えてやれば良いわけです。これを“拡張”と呼びます。

■これを表すユースケース図は、次のようになります。



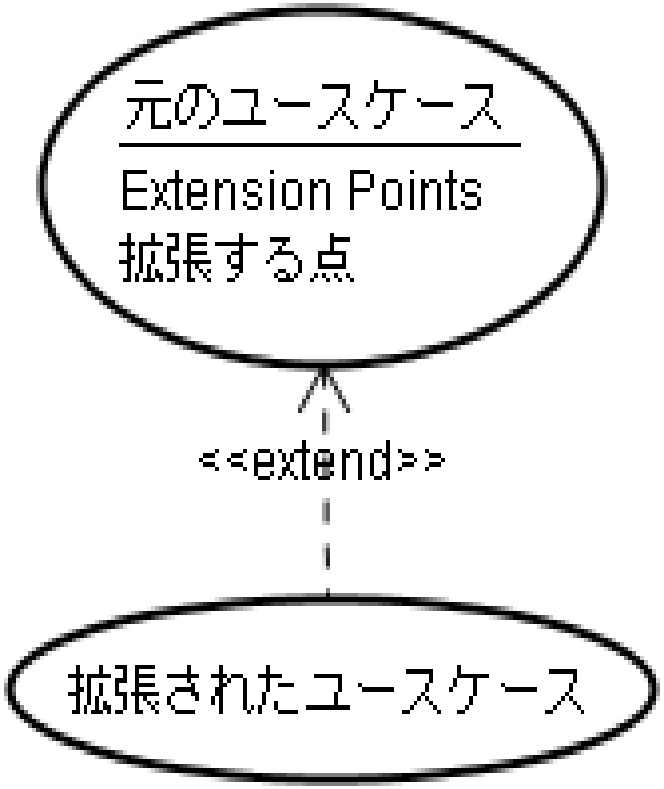
（２．６．３）拡張の記法

■拡張は、”<<extend>>”（拡張する）というラベルを付けた破線矢印を用いて記します。

■矢印の方向は、「拡張されたユースケース」から、「元のユースケース」に向かいます。

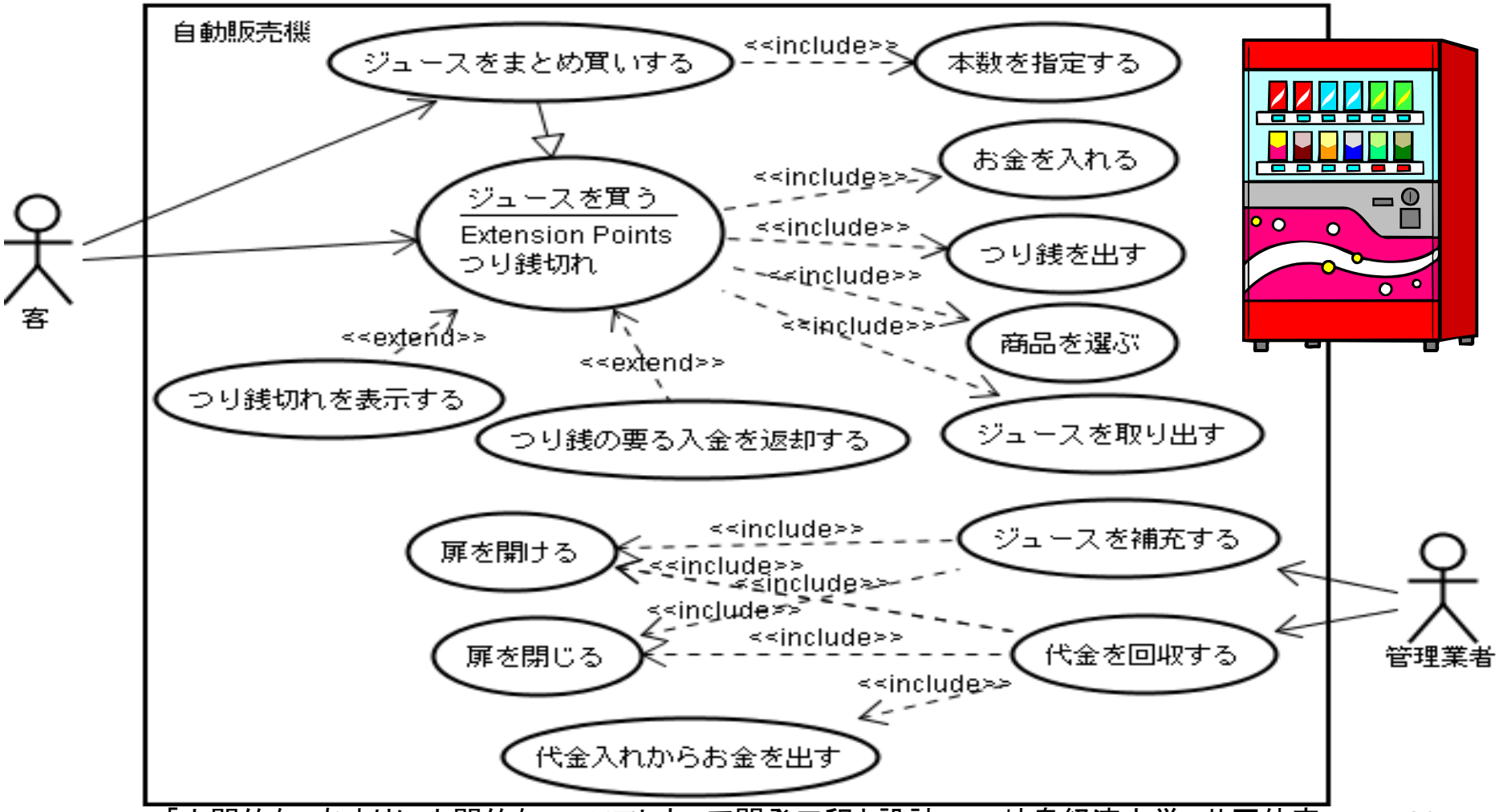
■「元のユースケース」には、「拡張されたユースケース」が発生する状況（ポイント）を、“Extended Point”として記します。

■“拡張”という言葉は、クラス図では“汎化”、“継承”の関係でも用いたため、紛らわしいことに注意してください。



（2．7）組織化されたユースケース図

■以上、(2.3)～(2.6)の組織化の技法を加えて、自動販売機のユースケース図は、次のようになります。



（2．8）ユースケース図の重要性

■前スライド(2.7)を見て、自動販売機のソフトウェアのイメージが湧いてきたでしょうか？

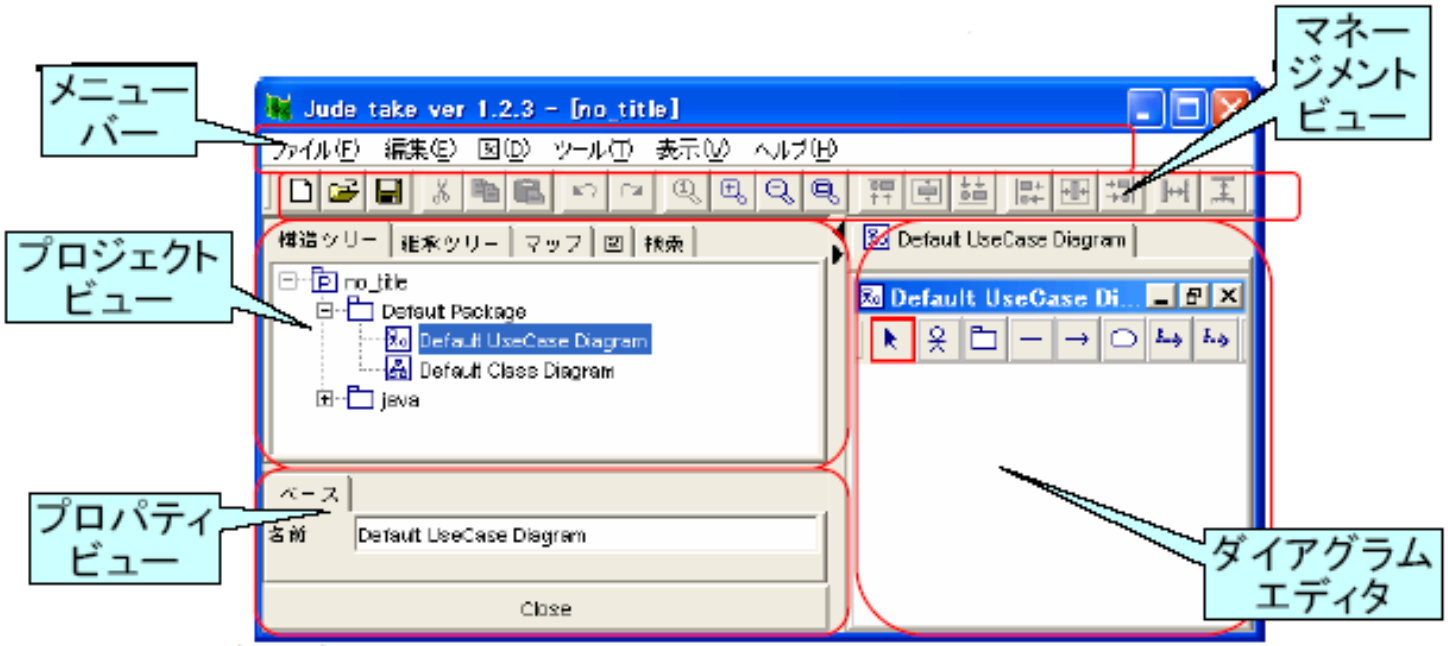
■ソフトウェア開発は、当然、「作りたいものを作る」のではなく、「要求されているものを作る」ことになります。その要求が何であるかは、いずれの工程でも明確にしておかなければ、道を間違えることになります（“木を見て森を見ぬ”状況に陥ります）。

■ユースケース図は、「要求されているもの」を視覚的に文書化する点で、非常に重要です。

■UPが、“ユースケース駆動”を柱としている点は、ここにあります。

（2．9）Judeでのユースケース図の作図

■オブジェクト指向プログラミングの学習で用いた、Judeを引き続き使用します。



■メニューバー：各種作業のメニュー

■マネージメント・ビュー：Jude全体を扱う操作のボタン。

■プロジェクト・ビュー：上部のタブを切り替えることで、さまざまな方法でプロジェクトの構成を示します。ここで選んだ図をダイアグラム・エディタ上で操作することが出来ます。

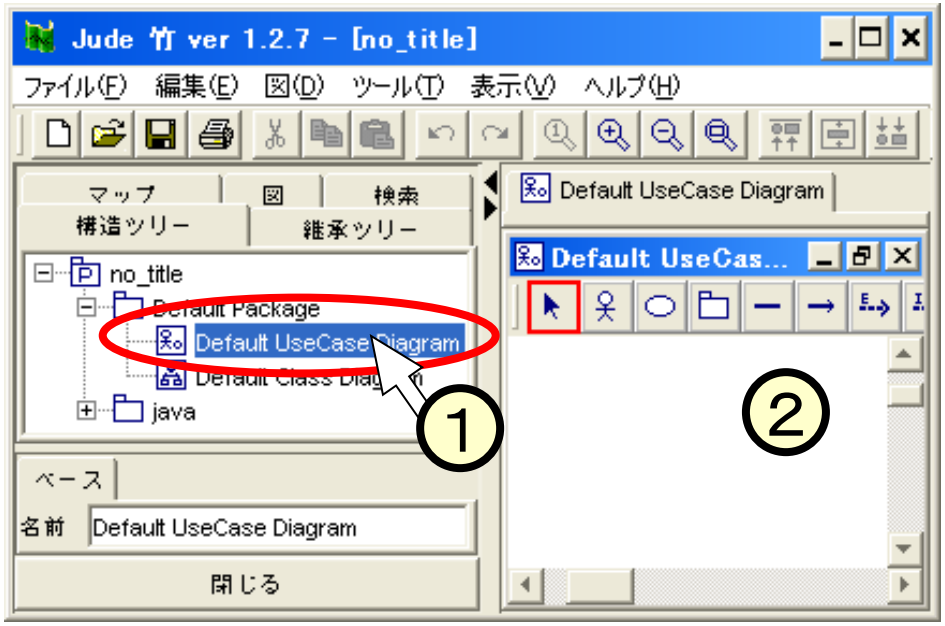
■ダイアグラム・エディタ：ここでダイアグラム（図）を作成する。

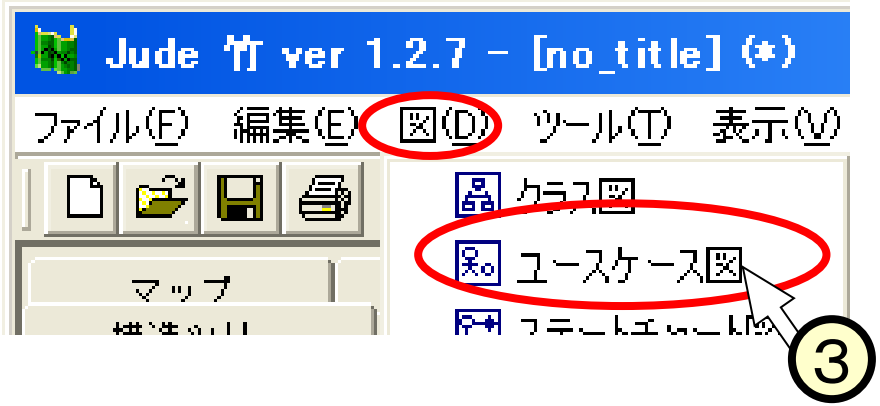
■プロパティ・ビュー：ダイアグラム（図）中の要素への操作を行います。

（2．9．1）新規ユースケース図を開く

■プロジェクトビュー中の”Default UseCase Diagram” をダブルクリック（①）すると、ダイアログエディタ（②）にて図の作成が可能になります。

■新たなユースケース図を作成する場合には、[図]–[ユースケース図]をクリック（③）します。





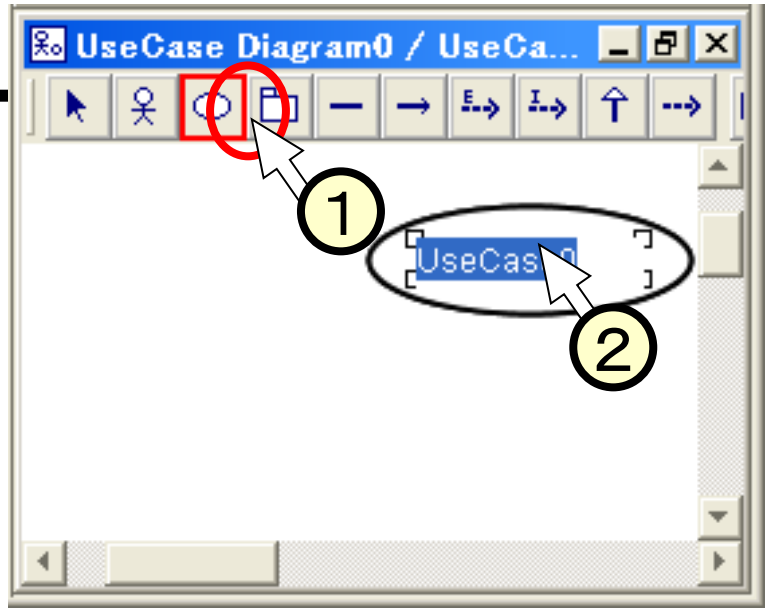
(2.9.2)ユースケースの配置

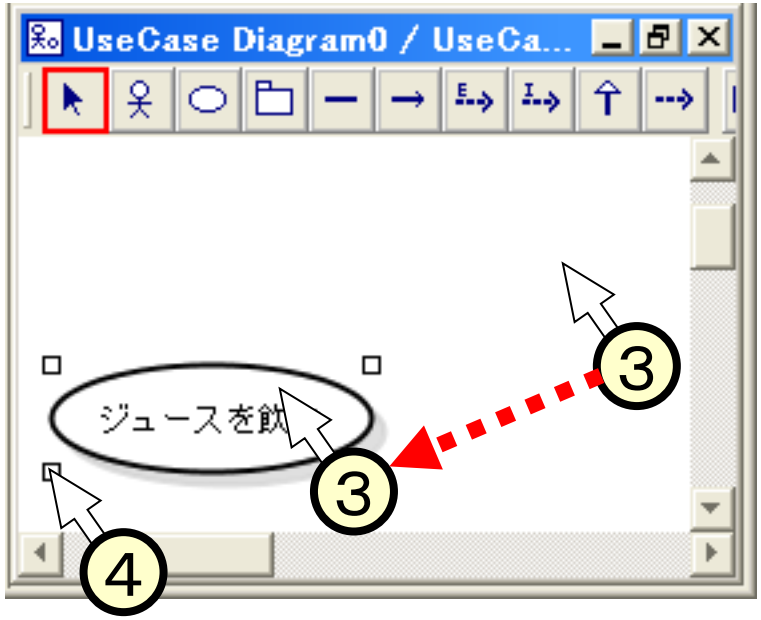
■ツールバーの[ユースケース]をクリック(①)して、編集画面上をクリック(②)します。

■ユースケースが、名前の入力待ちの形で配置されます。ここで名前の入力が可能ですが、後から名前の欄をダブルクリックして入力することも出来ます。

■ドラッグ(③)することで、移動させます。また、要素の四隅のつまみをドラッグ(④)して、大きさを調整します。

■フォーカスされた状態(＝四隅につまみが表示)で[Delete]キー押下すると、要素は削除されます。

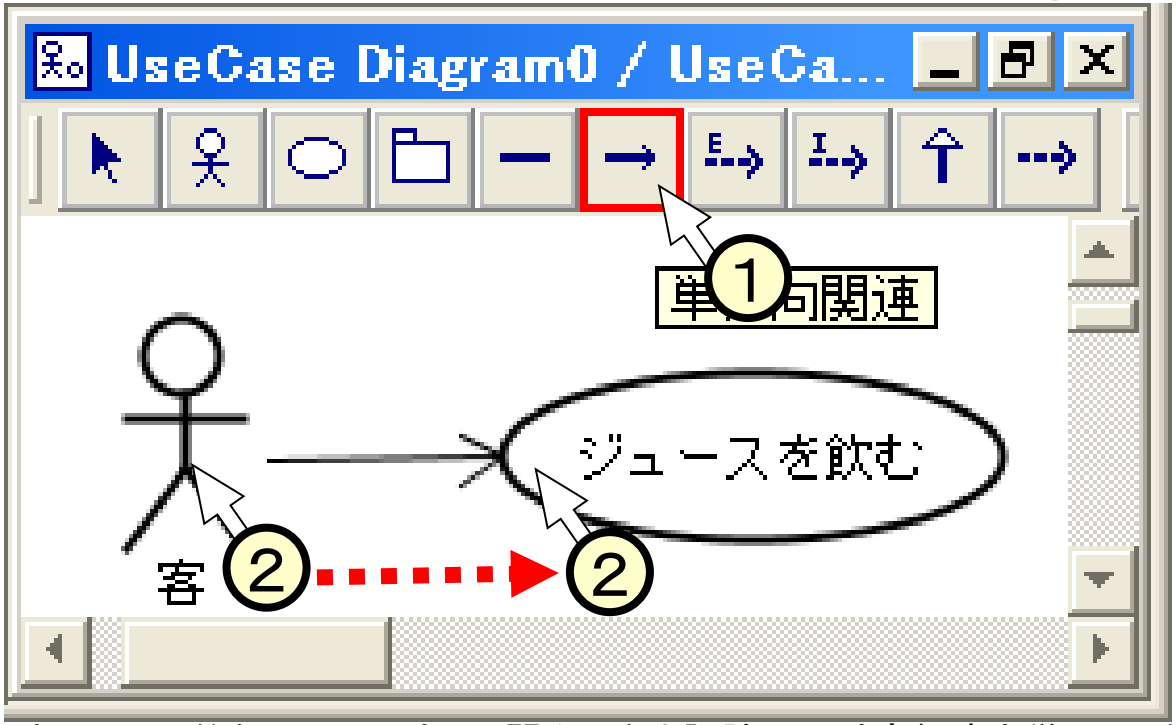




（2．9．3）関連(アソシエーション)の配置

■アクターも、ユースケースと同様の手順で配置できます。アクターからユースケースに関連を配置します。

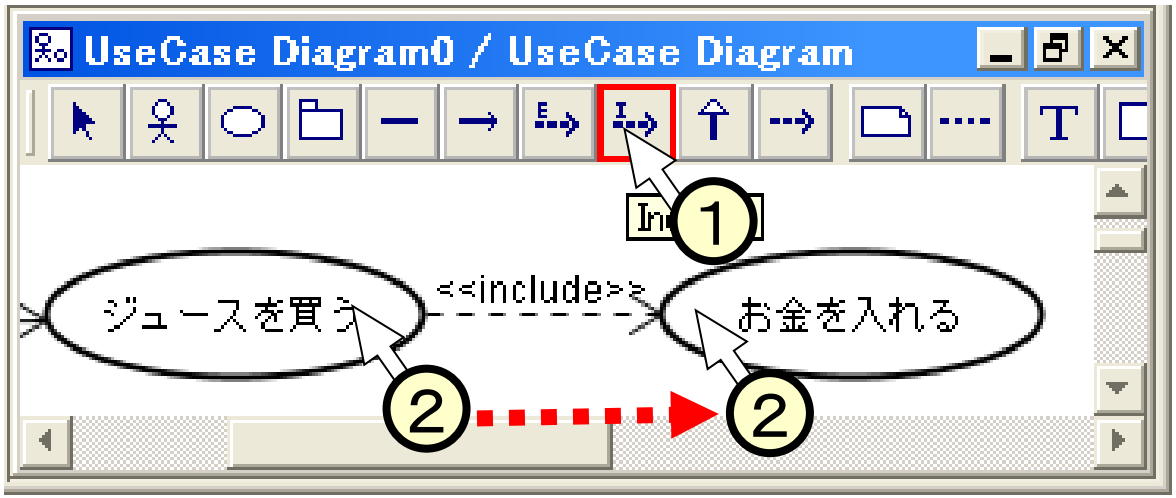
■ツールバーの[単方向関連]をクリックし(①)、アソシエーションの元側（客アクター）からアソシエーションの先側（ユースケース）へと、ドラッグします（②）。



（2．9．4）包含・汎化・拡張の配置

■ユースケースからユースケースへの包含を作図します。（汎化・拡張もほぼ同様です）。

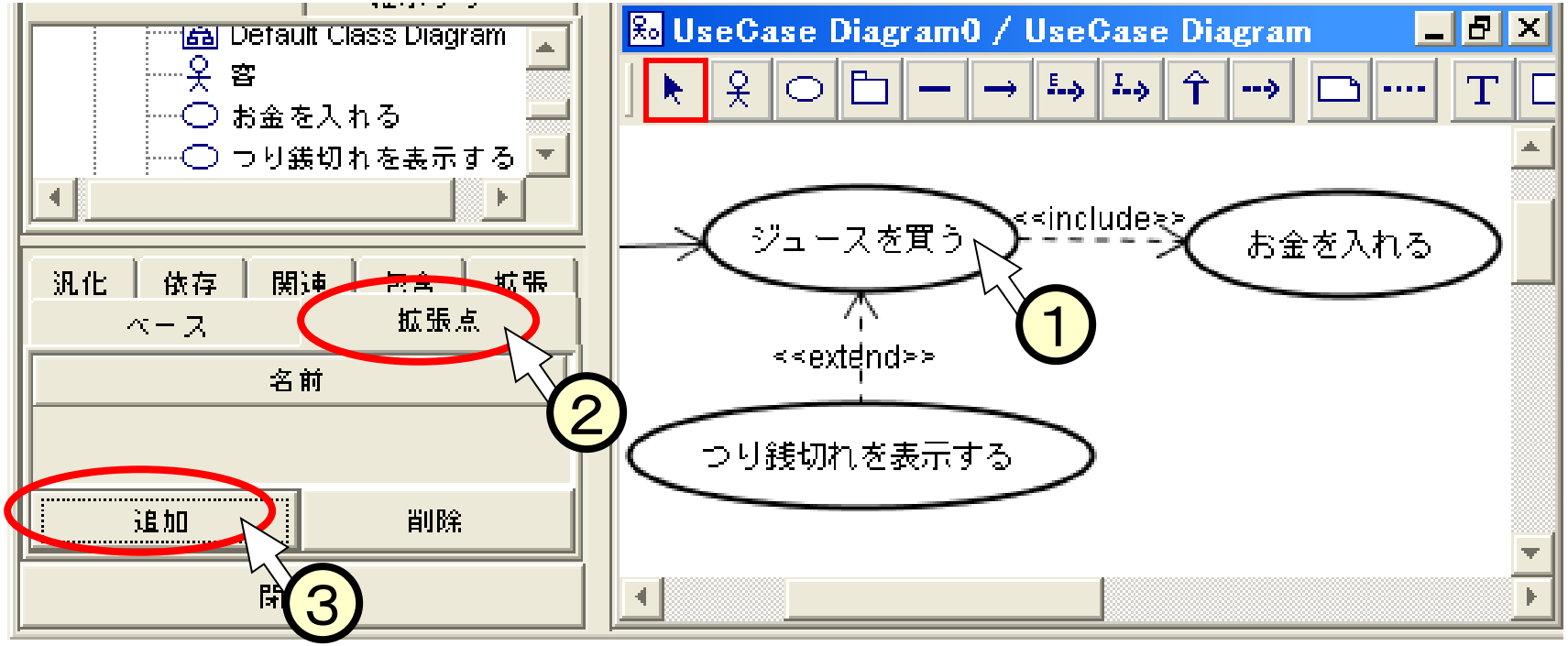
■ツールバーの[include]をクリックし（①）、ユースケースの元側（図では“ジュースを買う”）からユースケースの先側（図では“お金を入れる”）へと、ドラッグします（②）。



（2．9．5）拡張点の配置　－1－

■拡張点を追加するユースケース図（下図では“ジュースを買う”）をクリック（①）してフォーカスします。

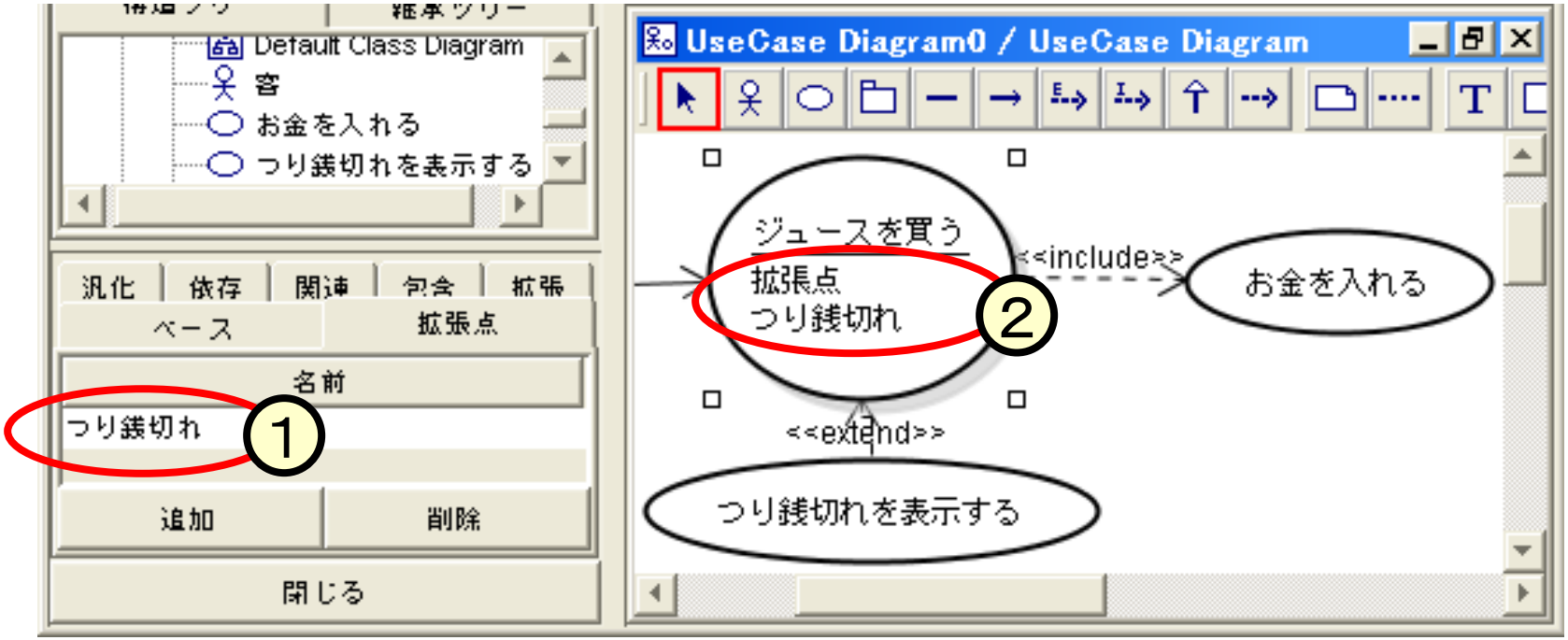
■プロパティビューにて、[拡張点]タグをクリック（②）し、追加ボタンをクリック（③）します。



（2．9．6）拡張点の配置　－2－

■名前の欄に、各頂点の名称（下図では“つり銭切れ”）を入力（①）します。

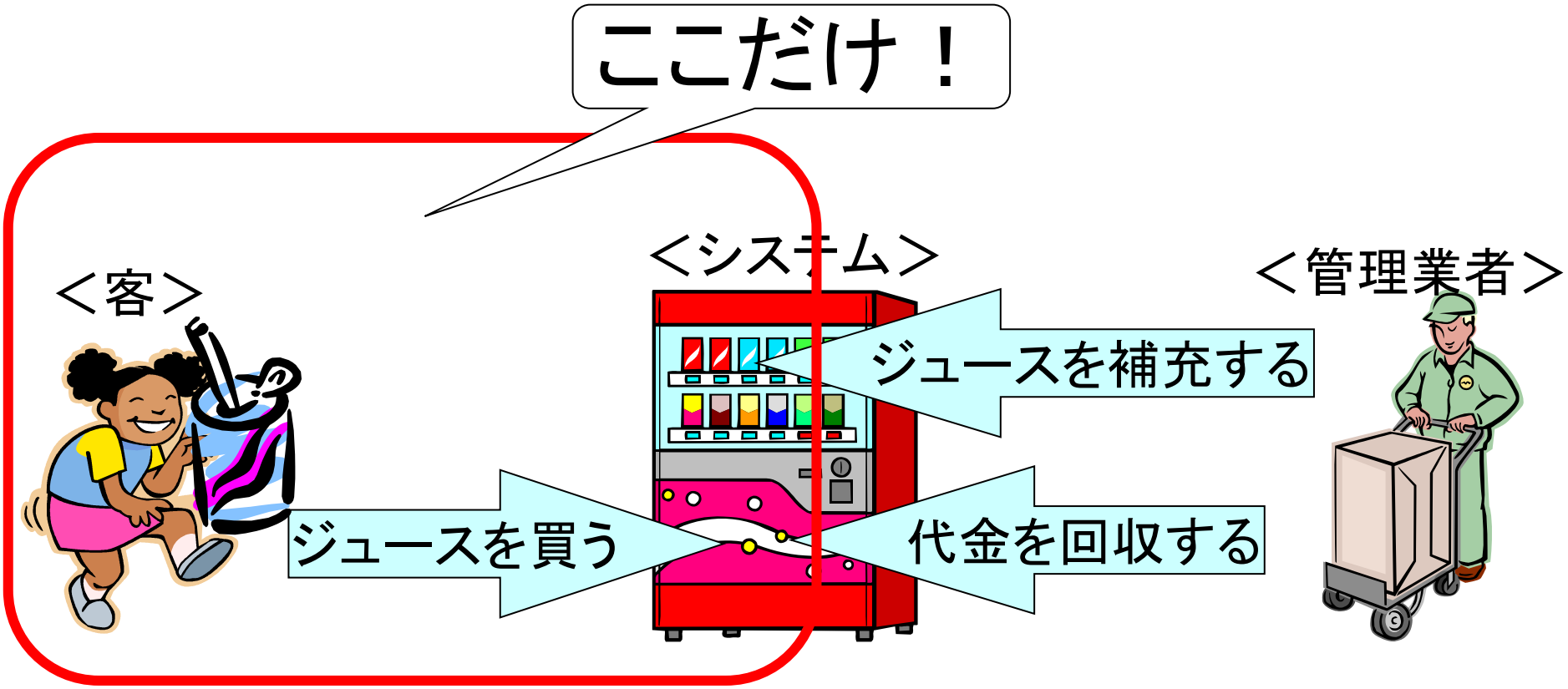
■ユースケースに各頂点が表示（②）されます。



（2．10）演習1：課題1

■スライド(2.7)のユースケース図を作成してください。

- ただし、客アクターに関する部分だけで結構です。
- 管理業者アクターに関わる部分は省略してください。

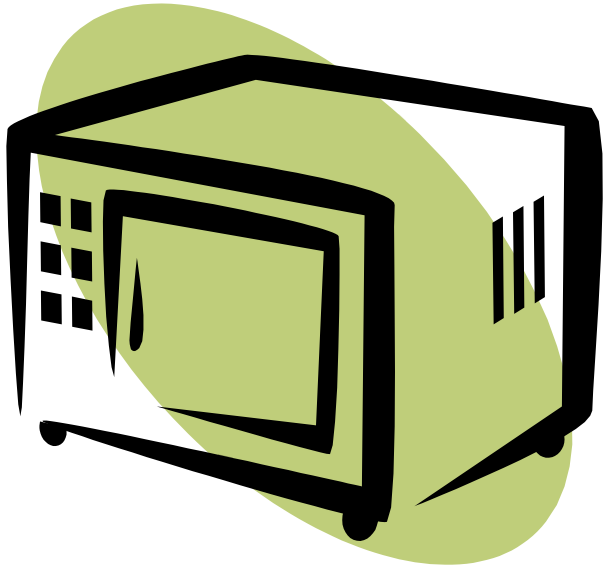


（2．11）演習2：課題2

■オーブンレンジのユースケース図を描いてみてください。

- オーブンレンジは、オーブンと電子レンジとの機能を持ち、選択することが出来ます。

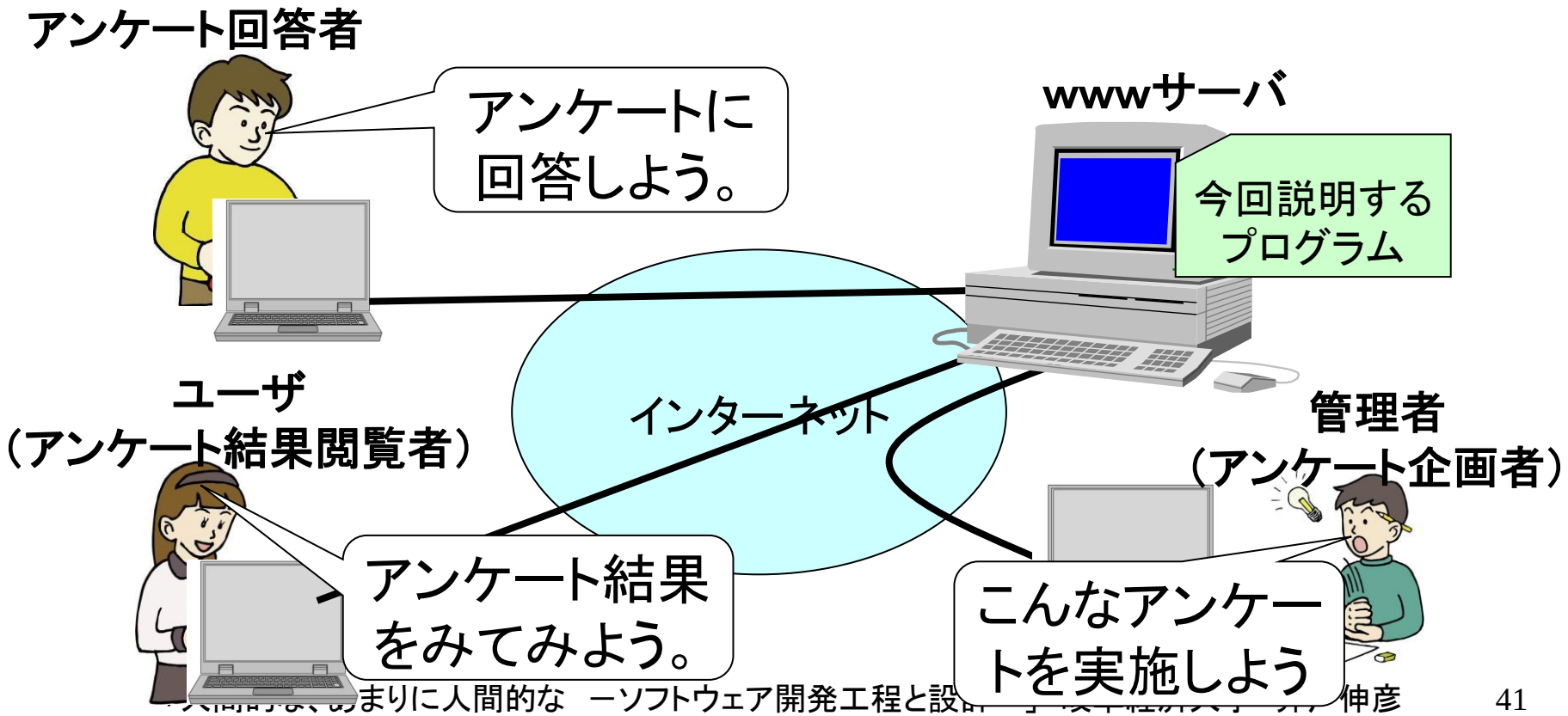
■機能については、自分で考えてください。簡単なものでも結構です。



（2．12）演習3：課題3

■次のようなシステムを考え、ユースケース図を作成してください。

（この問題はやや難しいので、出来る方のみで結構です。）



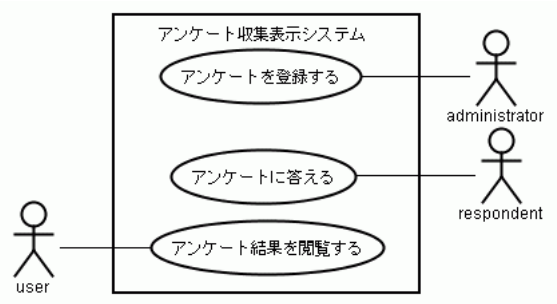
（3）シナリオとイベントフロー

■シナリオ

- ユースケースの具体例を書いたもの。

■イベントフロー

- アクターとユースケースが互いに作用してどのように動くかを網羅的に記述したもの。



より明確に

シナリオ　　**イベントフロー**

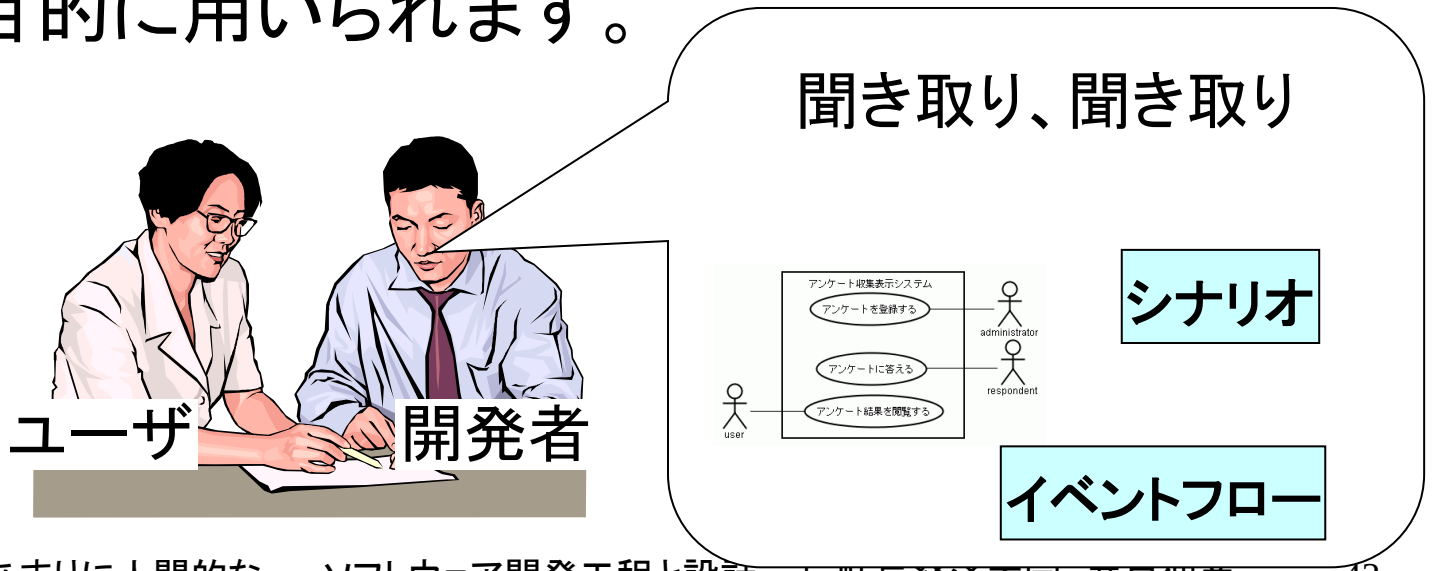
例として具体的に　⟷　もれなく総合的に

（3．1）ユーザからの聞き取り

■システム開発者がユーザの要求を把握することは、後になってシステムがユーザの望むでなかったというような事態を招かないために、非常に重要です。

■ユーザから望むシステムの概要を的確に把握するためには、開発者はユーザから十分な聞き取りを行う必要があります。

■シナリオとイベントフローは、ユースケース図と合わせて、この目的に用いられます。



（３．２）アンケートシステム

■例として、課題3にあったアンケートシステムを扱います。

■次のURL（学内専用）にアクセスしてみてください。

（井戸のサイト中、本授業のページにリンクがあります。）

- アンケート回答者サイト
  - http://server-ido.gifu-keizai.ac.jp/strutsex/respondentLogin.do
- アンケート閲覧者サイト
  - http://server-ido.gifu-keizai.ac.jp/strutsex/menu.do

■回答者のユーザ名・パスワードは次のとおりです。

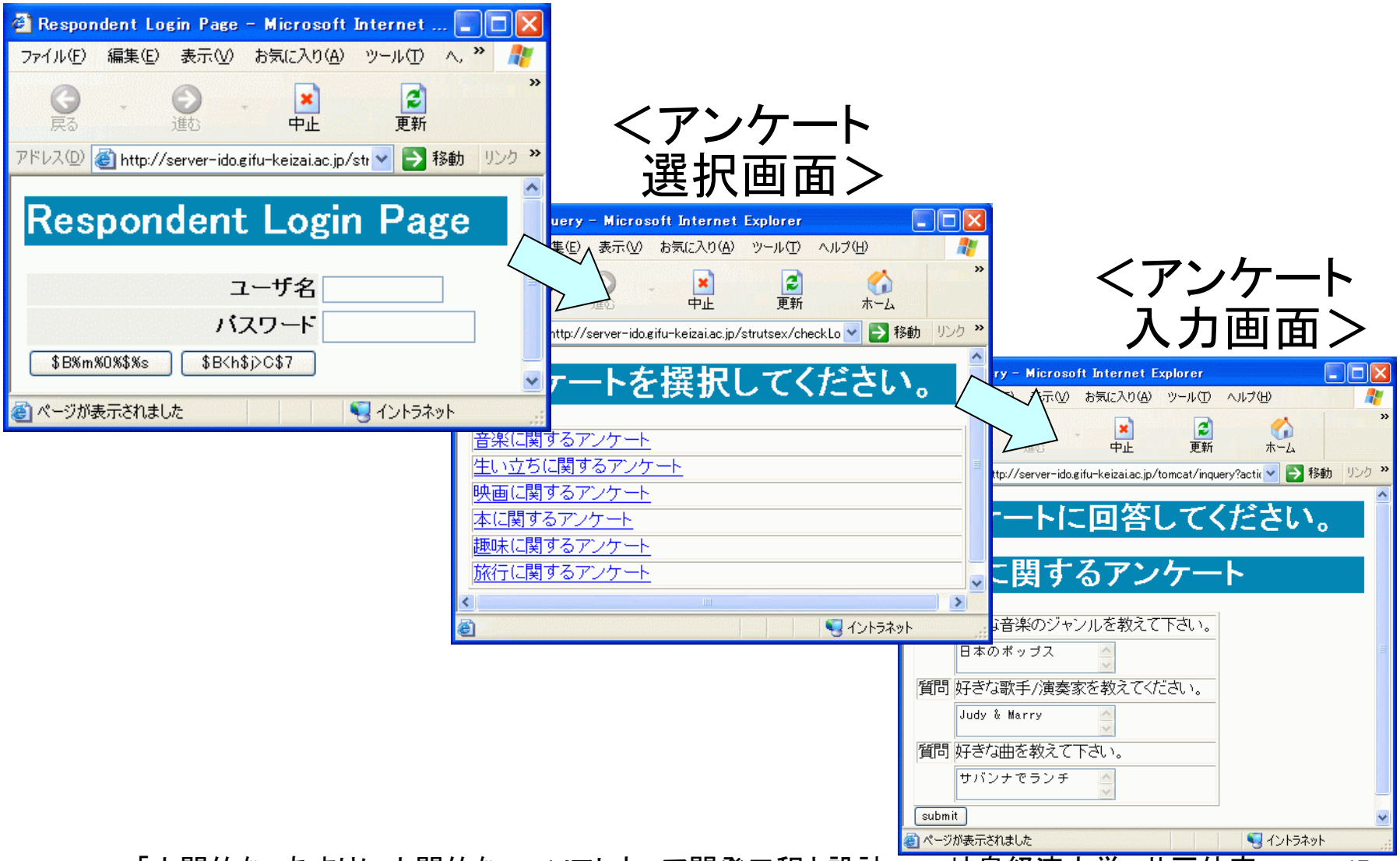
- ユーザ名は、“a”、“b”、“c”、....、“z”
- それぞれのパスワードは、“aa”、“bb”、“cc”、....、“zz”

（3．2．1）画面構成　－回答者－

＜ログイン画面＞

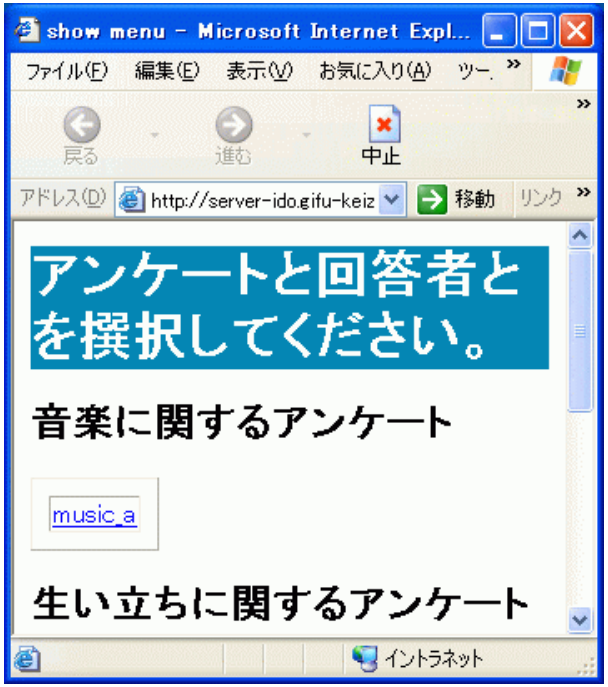
＜アンケート
選択画面＞

＜アンケート
入力画面＞

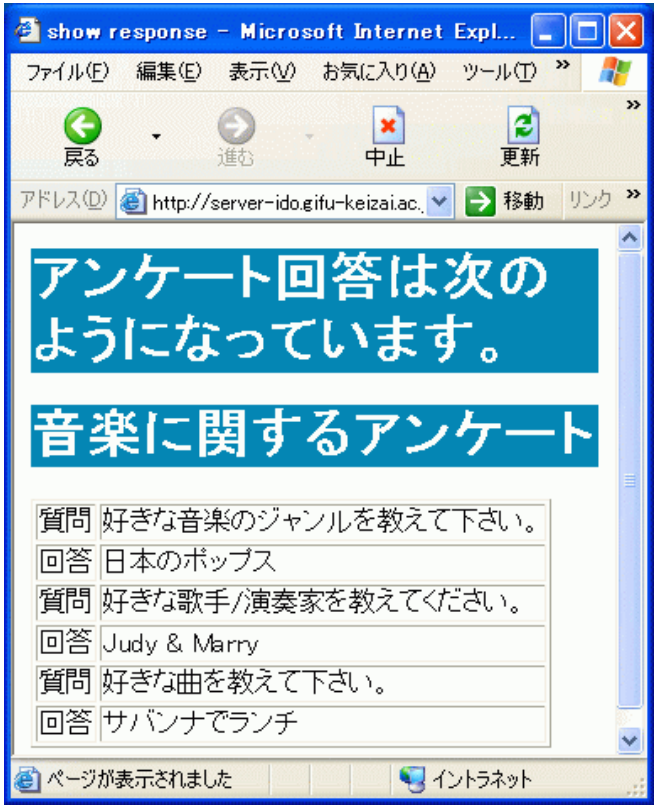


（3．2．2）画面構成　－閲覧者－

＜アンケート・
回答者選択画面＞



＜アンケート閲覧画面＞



（３．３．１）シナリオの例（アンケートシステム）

■アンケート回答者は、次のような使い方をすることが解ったと思います。

唐津くんは、アンケート回答サイトにアクセスします。
システムは、ログイン画面を表示します。
唐津くんは、ユーザ名とパスワードを入力します。
システムは、アンケート（リンク）一覧を表示します。
唐津君は、回答するアンケートを選択してクリックします。
システムは、アンケート回答入力画面を表示します。
唐津君は、アンケートに回答し、送信ボタンをクリックします。
システムは、アンケート回答を保存し、アンケート回答画面を表示します。

■これが、シナリオになります。

（３．３．２）シナリオの特徴

■シナリオは、①主語を明示的に書く、②アクターの立場から記述する、③システムの中身について意識しないという点に注意して書いていきます。

主語を明示的に書く

システムの中身について意識しない

アクターの立場から記述する

＜客＞

利用者は、硬貨を投入します。

システムは、購入可能な商品のランプを点灯させます。

利用者は、購入する商品を選択してボタンを押下します。

システムは、商品とつり銭とを出します。

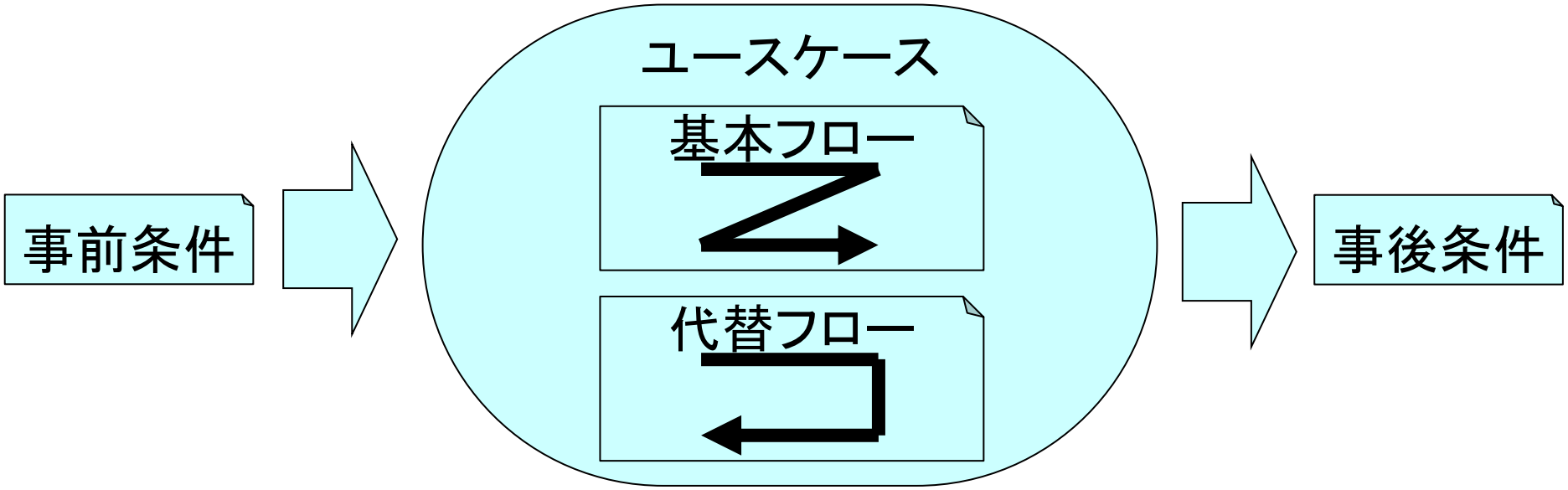
＜システム＞

ジュースを買う

（３．４．１）イベントフロー

■イベントフローでは、次の項目を記します。

- 事前条件：ユースケースを起動する上で、その前に完了しておくべき前提条件。
- 基本フロー：もっとも基本的な流れ。
- 代替フロー：基本フローと異なる別の流れ。
- 事後条件：ユースケースが終了した直後のシステムの状態。



（3.4.2）イベントフォローの例(アンケートシステム)

１．事前条件

なし

２．基本フロー

（１）ユーザは、アンケート回答サイトにアクセスする。
（２）システムは、ログイン画面を表示する。
（３）ユーザは、ユーザ名とパスワードを入力する。
（４）システムは、ユーザ名とパスワードとをチェックする。（E-1）
（５）システムは、アンケート（リンク）一覧を表示する。
（６）ユーザは、回答するアンケートを選択する。
（７）システムは、アンケート回答入力画面を表示する。
（８）ユーザは、アンケートに回答し、送信ボタンをクリックする。
（９）システムは、アンケート回答を保存し、アンケート回答画面を表示する。

３．代替フロー

E-1：（４）でアカウントとパスワードとが間違っていれば、警告メッセージを出して（２）に戻る

４．事後条件

（８）（９）の動作を連続して行うことが出来る。

（３．５）ユースケース記述

■ユースケース記述とは、「ユーザとシステムにどのようなやりとりがあるか」について記した文書であり、シナリオとイベントフローとを含みます。

■形式は様々ですが、次に一例を示します。

＜形式＞

ユースケース名
　ユースケースの概要
　イベントフロー
　　事前条件
　　基本フロー
　　代替フロー
　　事後条件
　シナリオ

＜例：アンケートシステム＞

＜アンケートに答える＞
１　概要
このユースケースは、アンケート回答者が、用意されているアンケートを選択して、回答を行うことが出来るようにする。

２．１　事前条件（略、スライド（３．４．２）参照）
２．２　基本フロー（同上）
２．３　代替フロー（同上）
２．４　事後条件（同上）

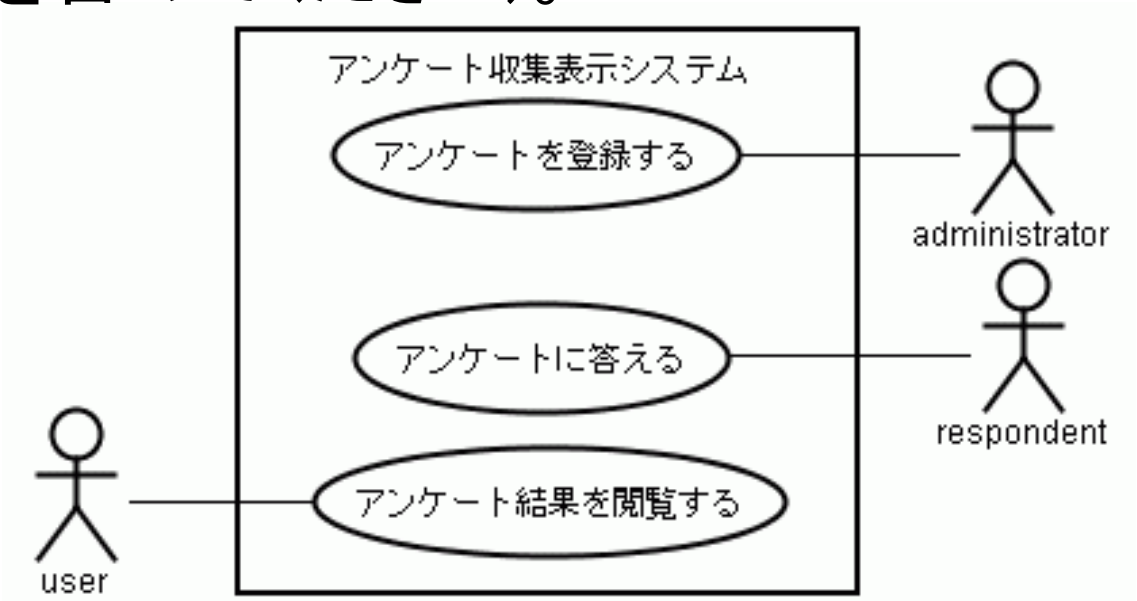
３　シナリオ（略、スライド（３．３．１）参照）

（３．６）演習４：課題４、５

■アンケートシステムについて、次のユースケース記述を作成してください。

- （課題４）アンケート結果を閲覧する
- （課題５）アンケートを登録する

■メモ帳にて作成し、井戸のネットワークドライブにアップしてください(ファイル名には学籍番号と“kadai4(もしくは5)”を含めてください)。



（３．７．１）演習５：課題６、７

■次の文章を読んで、ユースケース図（課題６）、ユースケース記述（課題７）を作成してください。

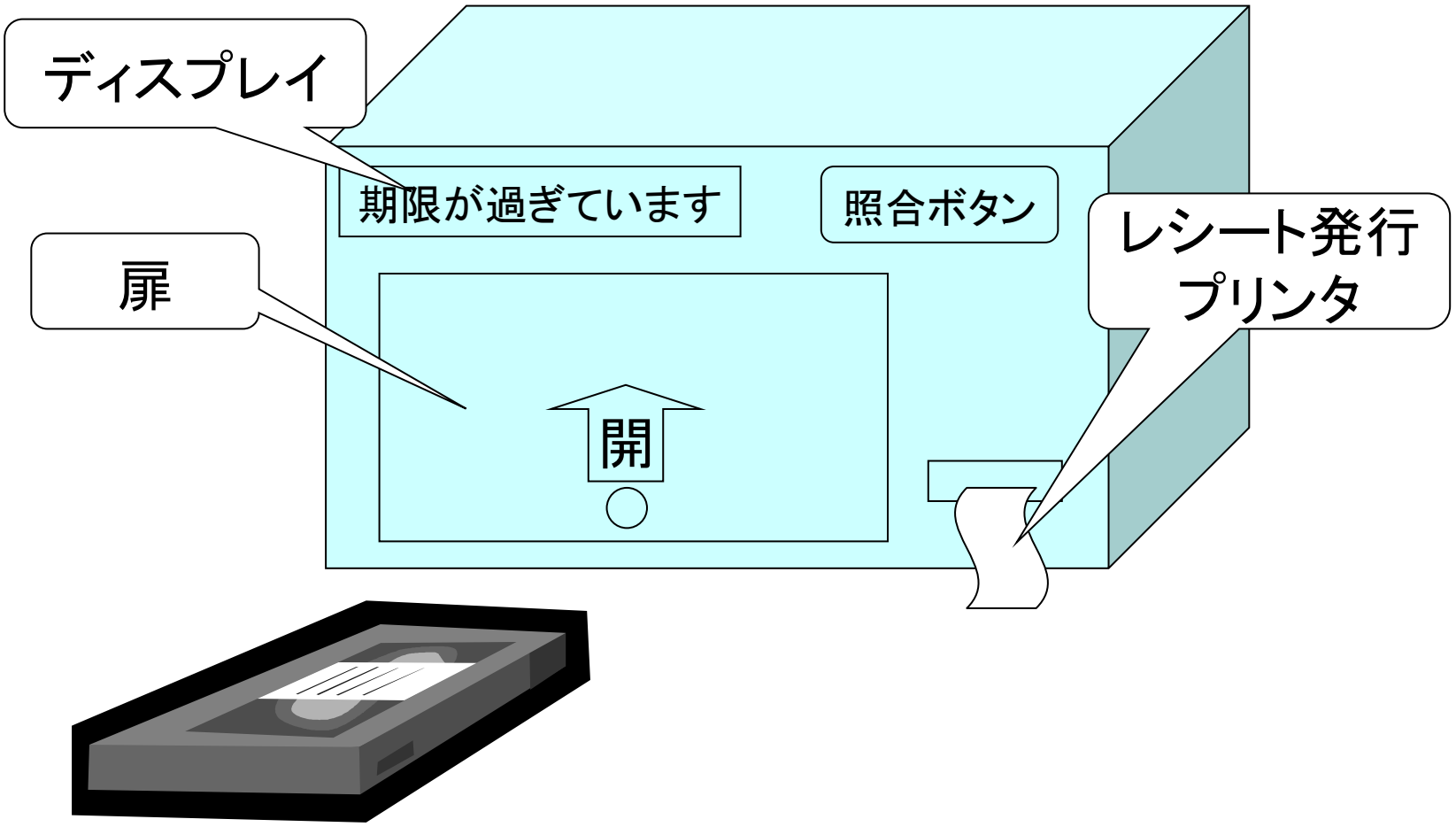
- （ビデオレンタル店経営者）ビデオの返却を無人で行いたい。
- （システム開発者）どのような装置を考えていますか？
- （経営者）返却ボックスを設置する。
- （開発者）どのような装置ですか？
- （経営者）ビデオを入れて照合ボタンを押すと、ビデオ上のバーコードを読み取ってレンタル品を確認し、返却ボックスの底が開いてレンタル品を回収する。
- （開発者）返却ボックスには、照合ボタン、バーコードリーダが備わっていて、底が開く仕組みがある訳ですね。
- （経営者）そのとおり。
- （開発者）貸出期限切れのビデオはどうしますか？
- （経営者）それは受け取らない。（⇒次スライドへ）

（３．７．２）演習５（続き）

- （開発者）どうするのですか？
- （経営者）返却ボックス上のディスプレイに、受付不可の表示を出す。
- （開発者）返却ボックスにはディスプレイがついているのですね。
- （経営者）Yes.　その他に、返却ボックスにはレシート発行プリンタもついている。
- （開発者）それは何に使うのですか？
- （経営者）正常に返却が終わったとき、返却を行ったことを証明する“返却レシート”を、利用者に発行する。
- （開発者）バーコードを読み取った後、返却ボックスの底が開く前に利用者がビデオを盗み去る恐れはありませんか？
- （経営者）返却ボックスには扉があって、システム側から鍵を掛ける機能がある。照合ボタンを押してから、底が開くまでは、鍵を掛けて、ビデオが取り出せないようにする。

（3．7．3）演習5:ヒント

■次のような返却ボックスをイメージしてください。



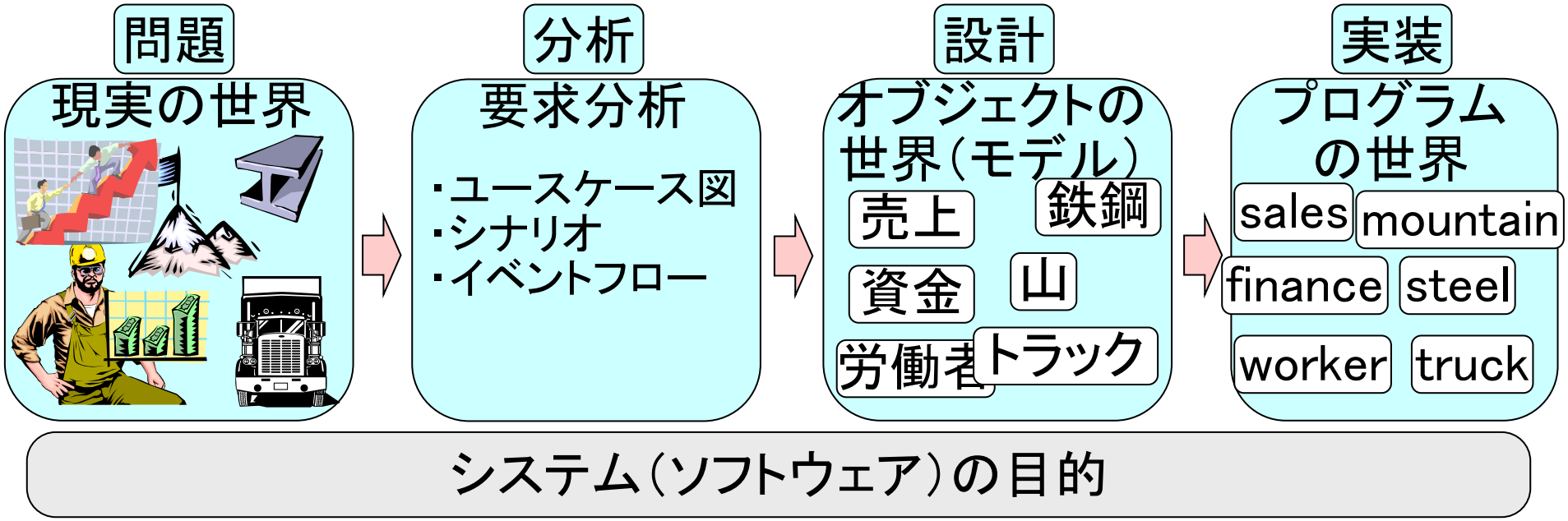
（３．７．４）演習５：課題８

■無人ビデオ返却装置について、あったら良いと思う機能を付け加えて、ユースケース図、ユースケース記述を作成してください。

（4）要求分析から設計へ

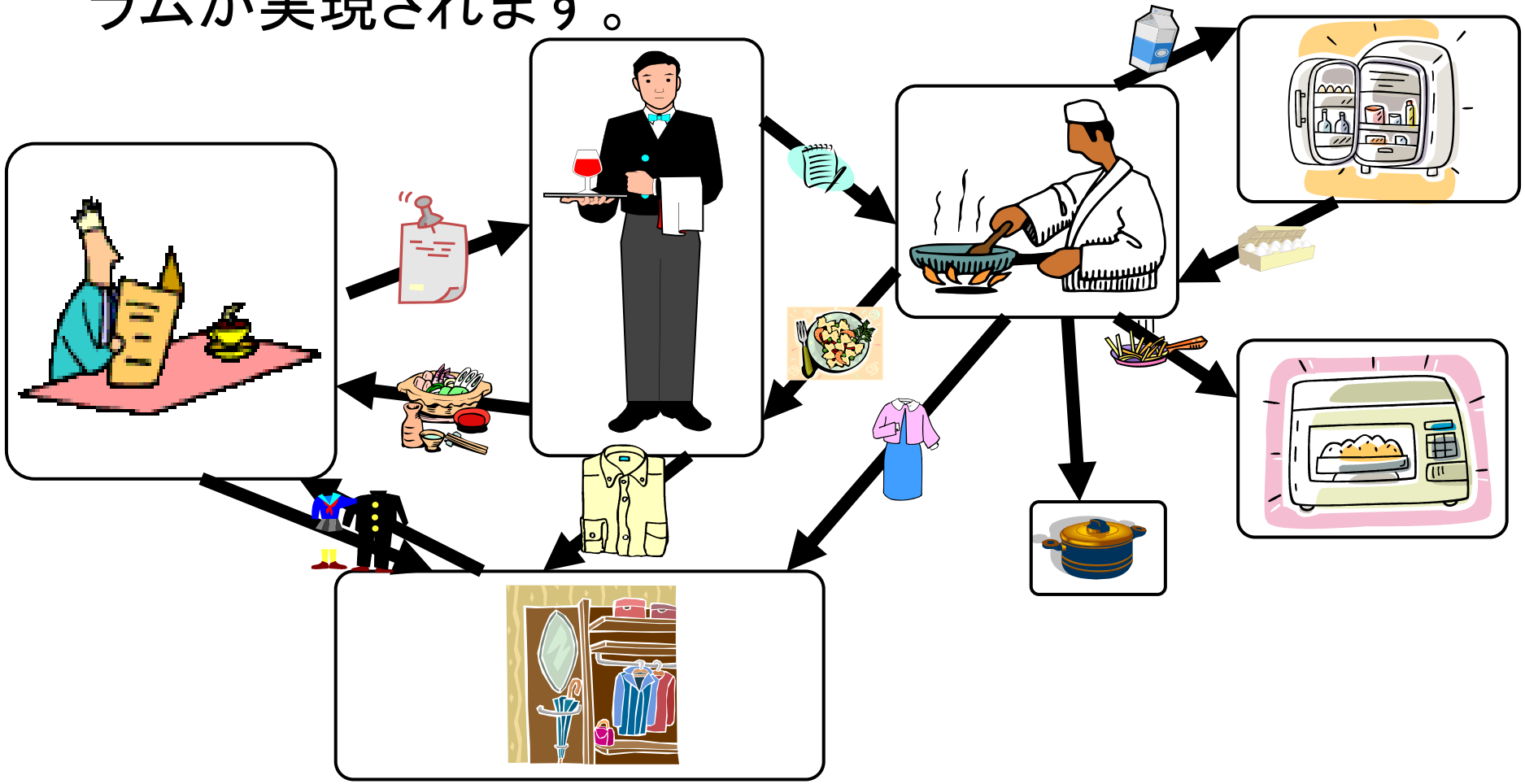
■ユースケース図等を用いて行った要求分析に基づき、モデルの作成（すなわち設計）を行うことになります。

■オブジェクト指向のところで学んだ内容につながっていきます。その様子を整理していきます。



（4．1）オブジェクトの世界

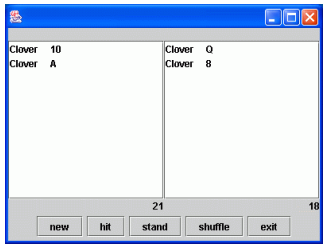
■シナリオ・イベントフローで表された内容が、さまざまな“もの”の間の動き（＝クラス間のうごき）として、プログラムが実現されます。

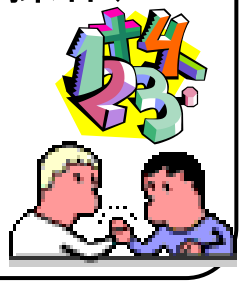


(4．2)クラスの抽出

■イベントフローやシナリオに登場し、システムを構成するために役割を果たしている物体・概念を、クラスとして考えていきます。

ウインドウ(Window)クラス
・操作を行う画面のウインドウ
・ゲーム(カード配布や得点、勝ち負けを操作)





プレイヤー(player)クラス
・競技を行う人

ディーラー(dealer)クラス
・プレイヤーと競って競技を行う人

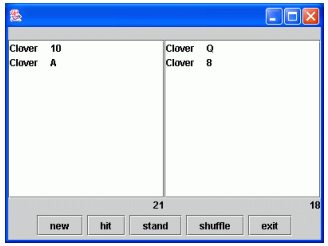
カード(card)クラス
・１枚のカード

カードデッキ(card deck)クラス
・競技者に配るカードを束ねたもの

（4．3）クラスの属性

■クラスの性格づけにより、おおよその属性（データ、変数）が決まってきます。

ウインドウ(Window)クラス



画面表示のデータ

プレイヤー(player)クラス

所持カード一覧

ディーラー(dealer)クラス

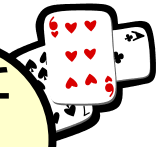
所持カード一覧

カード(card)クラス

マーク

数字

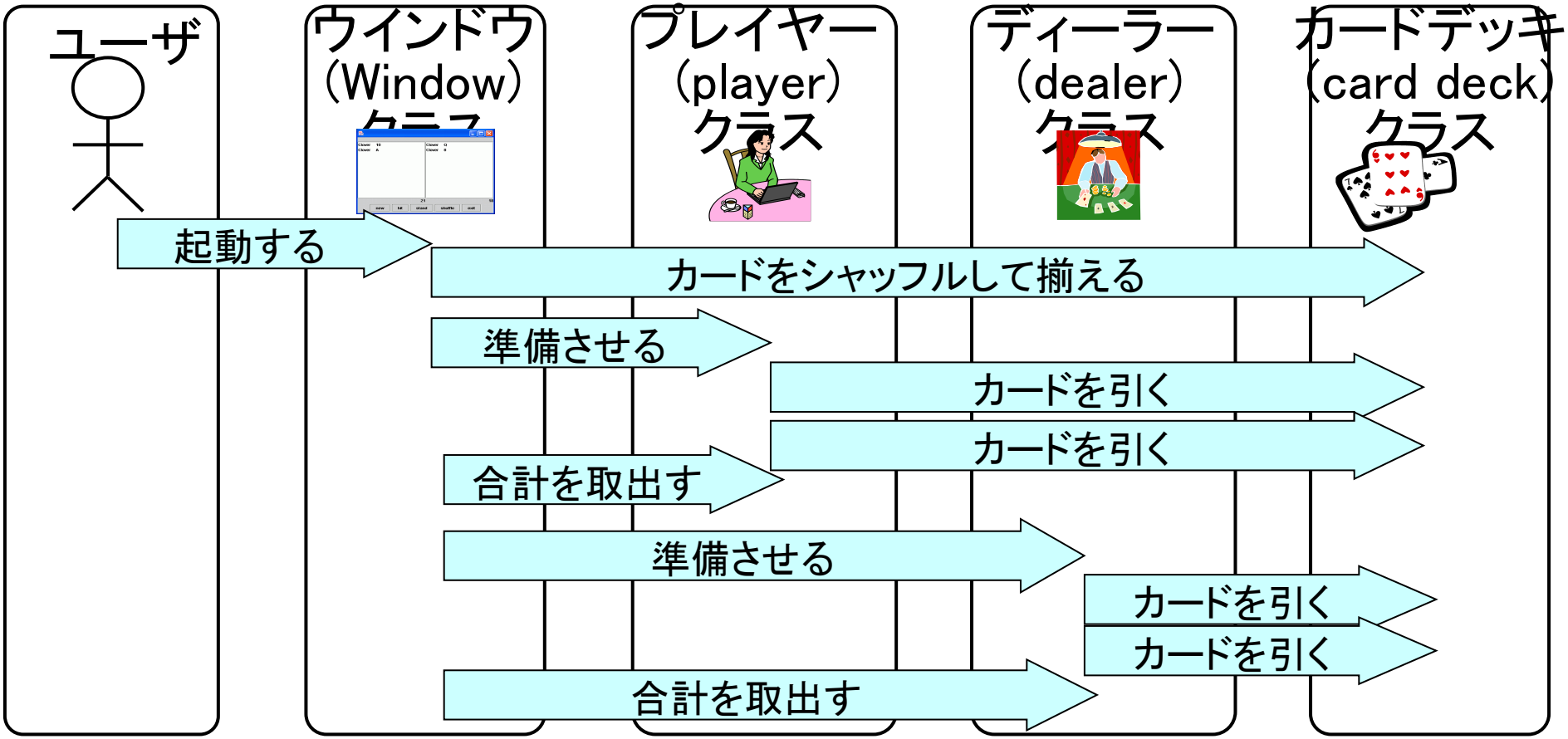
カードデッキ(card deck)クラス



積み上げたカードの一覧

（4．4）クラス間の動き

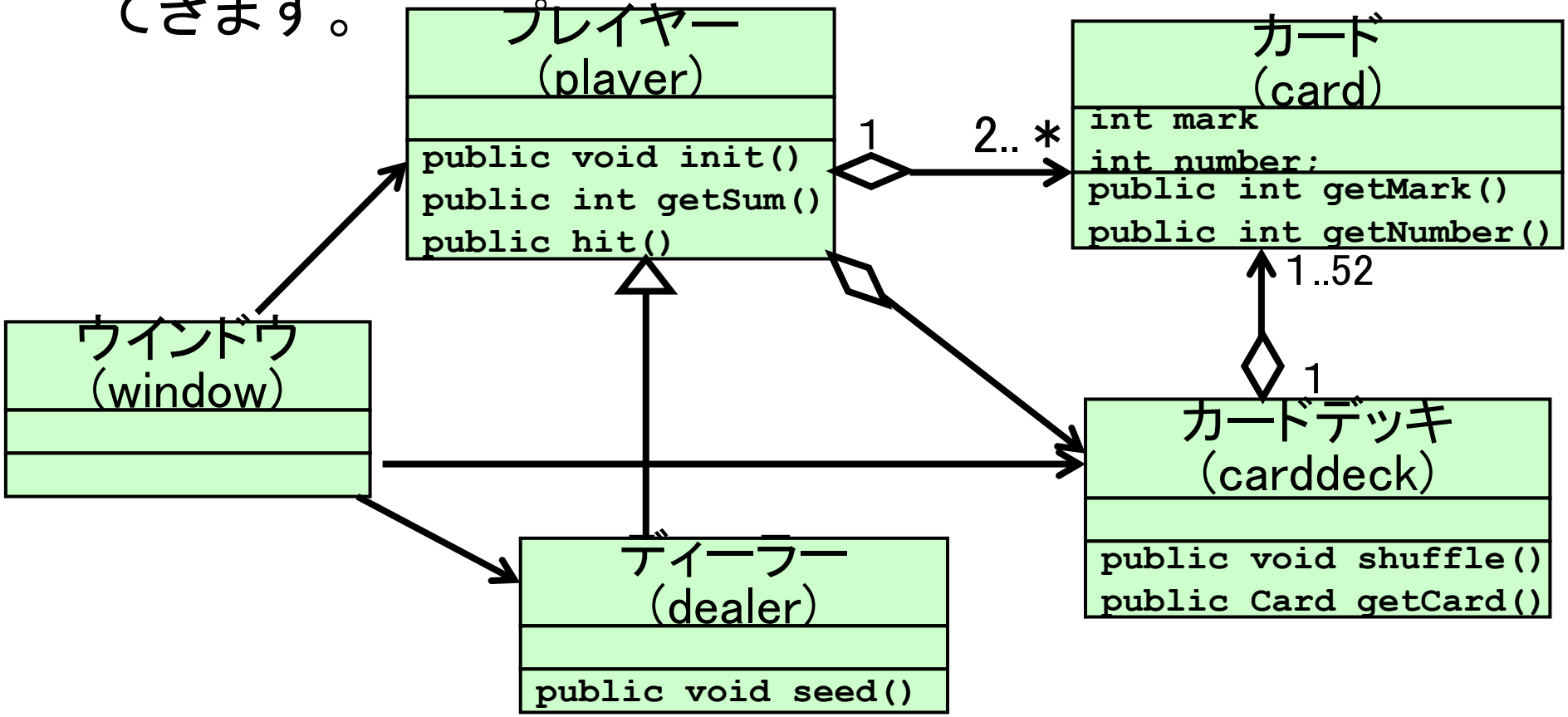
■イベントフローやシナリオに記された動作が、どのようなクラス間の動きにより実現されるのかを、シーケンス図でまとめます。



（4．5）クラス図

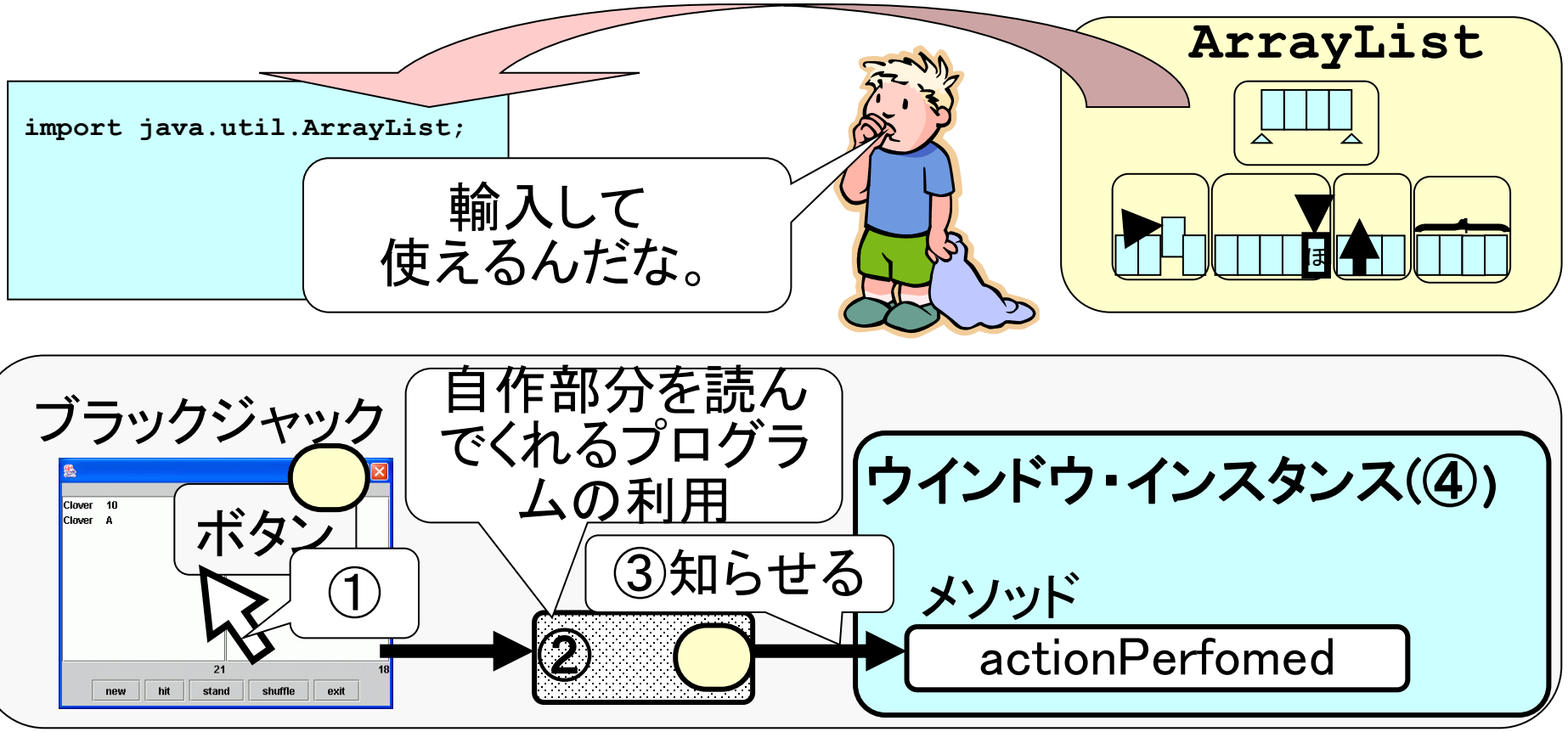
■シーケンス図により、①クラスがどのように振舞うか（メソッド）が見えてきます。

■②クラス間の関係がどのようになっているかも決まってきます。



（4．6）プログラムの再利用

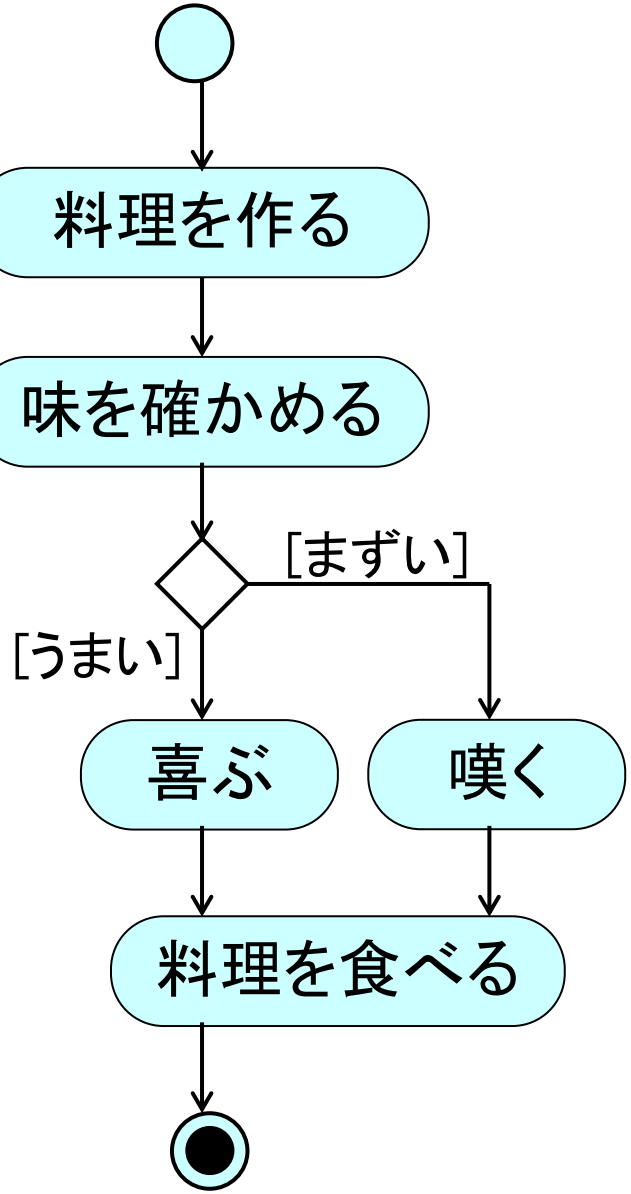
■クラスの構成の中では、広範囲に適用するものから、プログラムの部分に適用するものまで、さまざまなレベルで他人の作ったクラスを利用します。



（4．7）アクティビティ図

■アクティビティ図は、設計の複数の段階で使用します。

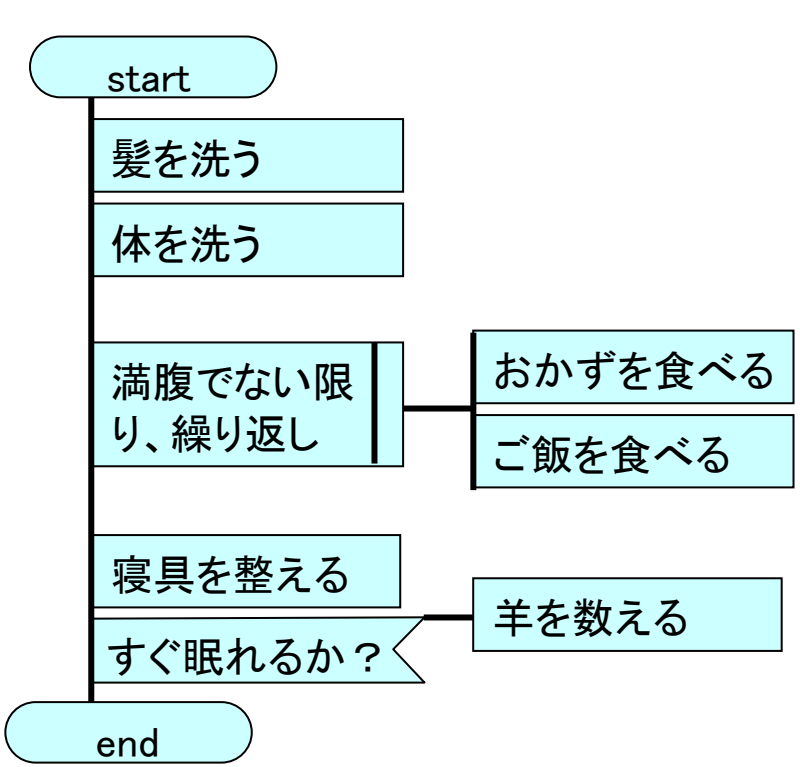
- ビジネスモデリング
  - 仕事の流れを理解するために、アクティビティ図を描きます。
- ユースケースモデリング
  - イベントフローを記述する場合にも用います。
- 実装時のドキュメント
  - プログラムの処理ロジックを記す場合もあります。

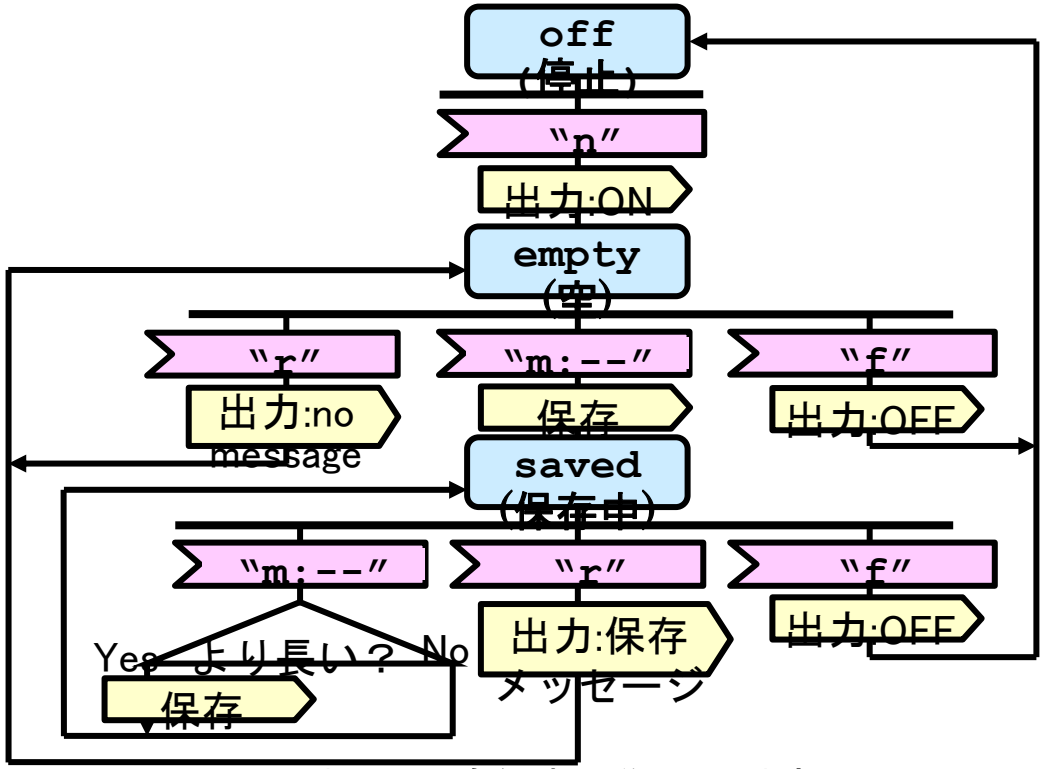


（4．8）構造化プログラミング、状態遷移

■メソッドをコーディングするレベルでは、構造化プログラミングによる整理が必要です。

■「以前の状態により次の操作が決まる」場合には、状態遷移図での整理が必要となります。





（４．９）演習６：課題９

■演習５のビデオの返却装置について、オブジェクトを抽出してください。

■ごく簡単な、シーケンス図を描いてみてください。

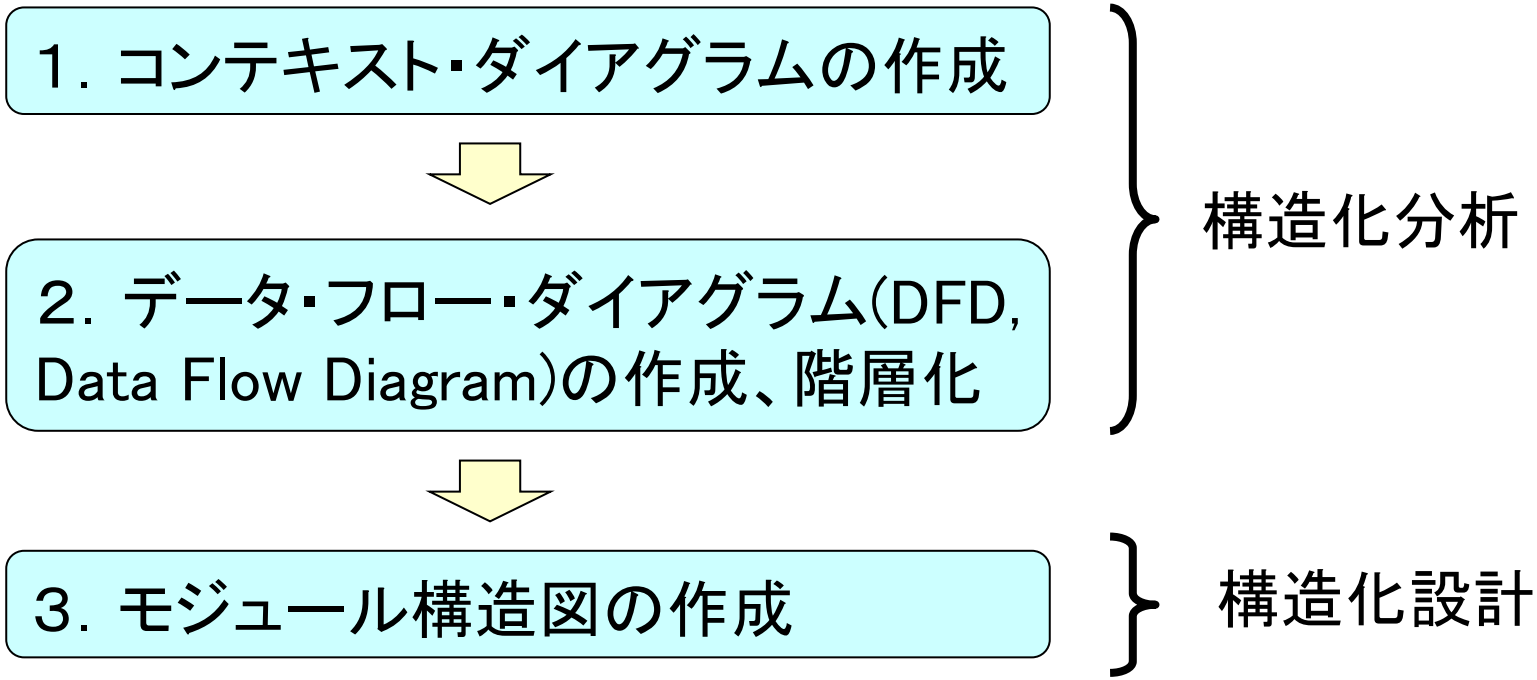
5章目次

- ■（5）構造化分析・設計
- ■（5．1）例題
- ■（5．2）図書館には何が求められているか？（DFD：コンテキストダイアグラム）
  - (5.2.1)DFD
  - (5.2.2)データディクショナリ
  - (5.2.3)他のデータディクショナリ
  - (5.2.4)まとめ
  - (5.2.5)演習7：課題10
- ■（5．3）内部の処理を考える（DFD：ダイアグラム0）
  - (5.3.1)どのように詳しくしていくか？
  - データ
    - (5.3.2.1)予算に関するデータ
    - (5.3.2.2)発注に関するデータ
  - (5.3.3)データ・ストア
  - プロセスを考えていく
    - (5.3.4.1)購入依頼書を入力する
    - (5.3.4.2～4)購入依頼書を受理する
    - (5.3.4.5)発注票を印刷する
    - (5.3.4.6)購入依頼・発注関係のDFD
  - (5.3.5)ダイアグラム0
  - (5.3.6)2つの図の関係
- ■（5．4）プロセスをさらに詳細化する（：ダイアグラム2、2．4）
  - (5.4.1)ダイアグラム2
  - (5.4.2)演習8：課題11
  - (5.4.3)プロセス2．4をさらに詳細化する。
  - (5.4.4)ダイアグラム2．4
  - (5.4.5)ミニ仕様書
  - (5.4.6)全体DFD
- ■（5．5）DFDを何に使うのか？
  - (5.5.1)構造図
  - (5.5.2)構造図を見て気付くこと
  - (5.5.3)DFDの意味
- ■（5．6）演習9：課題12

（5）構造化分析・設計

■構造化分析・設計（SSA/AD,Structured Systems Analysis and Structured Design）

- 事務処理プログラムを作る際によく使われる手法。
- おおまかな作業手順は次のとおり。

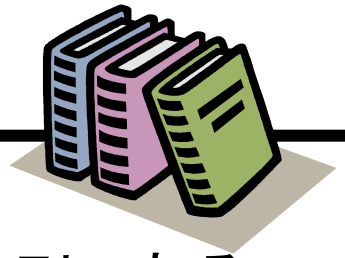


（５．１）例題

■大学付属図書館における図書購入問題（梶谷84より）

- [1]大学付属図書館では、各研究室からの図書購入依頼に応じて図書を発注、納品の受け入れなどの業務を行う。
- [2.1]図書館は、各研究室から図書の購入依頼書を受け取る。
- [2.2]購入依頼書には、書名（著者名、発行所等を含む）、価格、研究室名が記入されている。
- [2.3]購入依頼書を受け取ったら、記入漏れがないことを確認し、その研究室の予算でその図書が購入出来るかどうかを調べる。
- [2.4]これは、研究室の予算残高と、支出予定額（発注したがまだ納品されていない図書の価格の合計）から調べることが出来る。
- [3.1]購入出来ることが確かめられたら、購入依頼書から発注票とその控えを作り、発注票は書店へ送る。
- [3.2]発注票には、書名、価格、発注者名、発注番号、発注日が記入される。

（５．１　つづき）例題



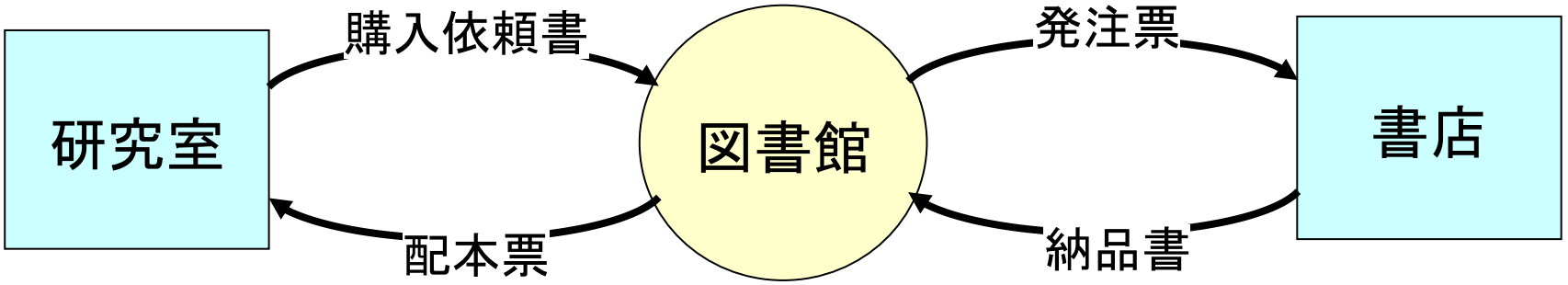
- [3.3]発注者名には研究室名が記入される。
- [3.4]発注番号は、発注票を識別するための記号列である。
- [3.5]同時に、発注した研究室の支出予定額を増やしておく。
- [4.1]書店から図書が納品書と共に届けられたら、納品書に記載された発注番号から発注票の控えを調べ、その図書が本当に発注されたものかどうかを確認し、そうであれば受け入れる。
- [4.2]納品書には発注番号の他に、書名と図書の実際の価格が記載されている。
- [4.3]受け入れの際には、その図書を購入する研究室の予算残高と支出予定額を変更する。図書は配本票とともに研究室へ届けられる。
- [4.4]配本票には、書名、実際の価格、研究室名が記載されている。

（５．２）図書館には何が求められているか？

■スライド(5.1)に示した文章には様々なことが書かれていますが、“図書館の仕事”と外部との関係について言えば、次のようにまとめられます。

- 図書館は、研究室より“購入依頼書”を受け取る。
- 図書館は、研究室へ“配本票”を送る。
- 図書館は、書店へ“発注票”を送る
- 図書館は、書店より“納品書”を受け取る。

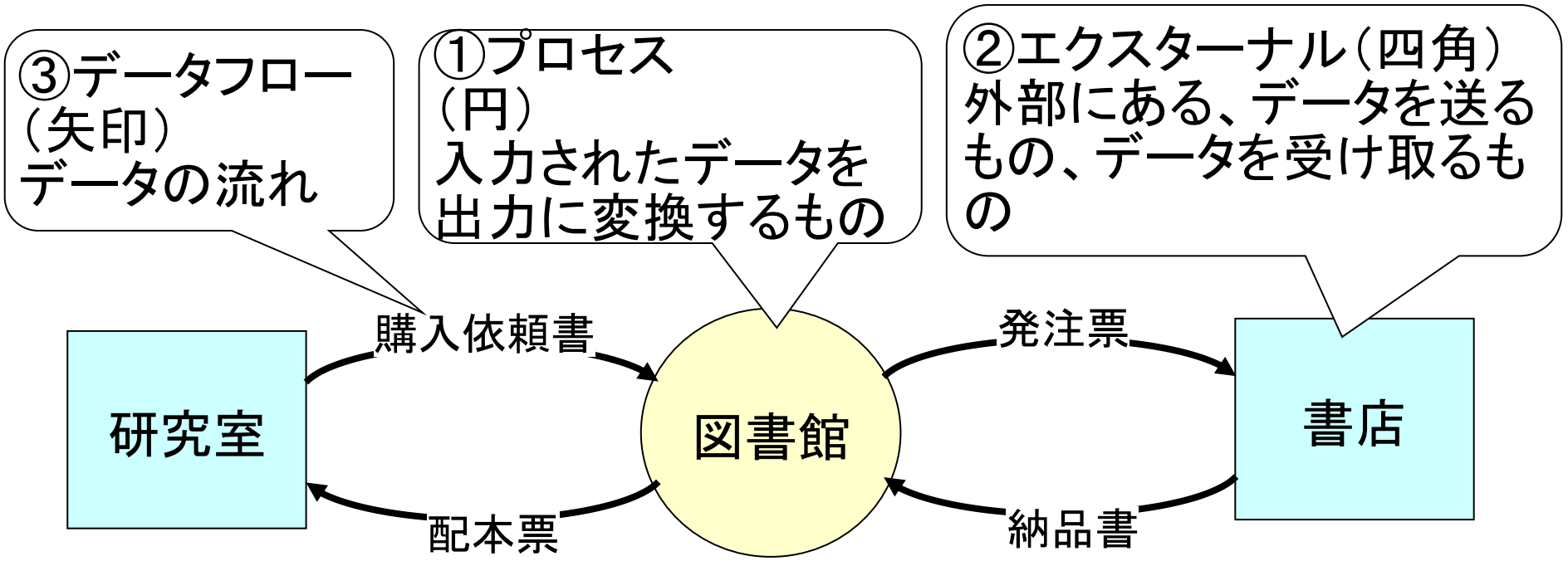
■上記の事項を図にすると、次のようになります。図に描くことで、スライド(5.1)の文意が明確になります。



（5.2.1）DFD(Data Flow Diagram)

■スライド(5.2)のような図を、DFD(データ・フロー・ダイアログ) と言います。特に、この図のように外部とのやり取りを示し、システムが何をやろうとするのかを示すDFDを、コンテキスト・ダイアグラム(Context Diagram)と言います。

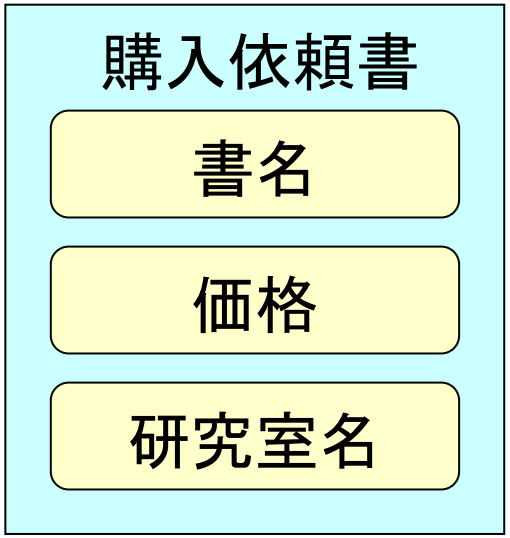
■図に用いられている要素は、次の通りです。



（5．2．2）データ・ディクショナリ(Data Dictionary)

■スライド(5.1)には、次の記述があります。

- [3.2]「購入依頼書」には、「書名」（著者名、発行所等を含む）、「価格」、「研究室名」が記入されている。



■スライド(5.2)のコンテキスト・ダイアグラム中、データフローとして記されている“購入依頼書”は、その内容（＝データ構造）をさらに詳しく決めることが出来ます。これを次のように表し、データ・ディクショナリと呼びます。

購入依頼書 ＝ 書名 ＋ 価格 ＋ 研究室名

（５．２．３）他のデータ・ディクショナリ

■同様に、スライド(5.1)の各文から、データディクショナリを書いてみます。

- [3.2]発注票には、書名、価格、発注者名、発注番号、発注日が記入される。

発注票 ＝ 書名 ＋ 価格 ＋ 発注者名 + 発注番号 + 発注日

- [3.3]発注者名には研究室名が記入される。

発注者名 ＝ 研究室名

- [4.2]納品書には発注番号の他に、書名と図書の実際の価格が記載されている。

納品書 ＝ 発注番号 ＋ 書名 ＋ 実価格

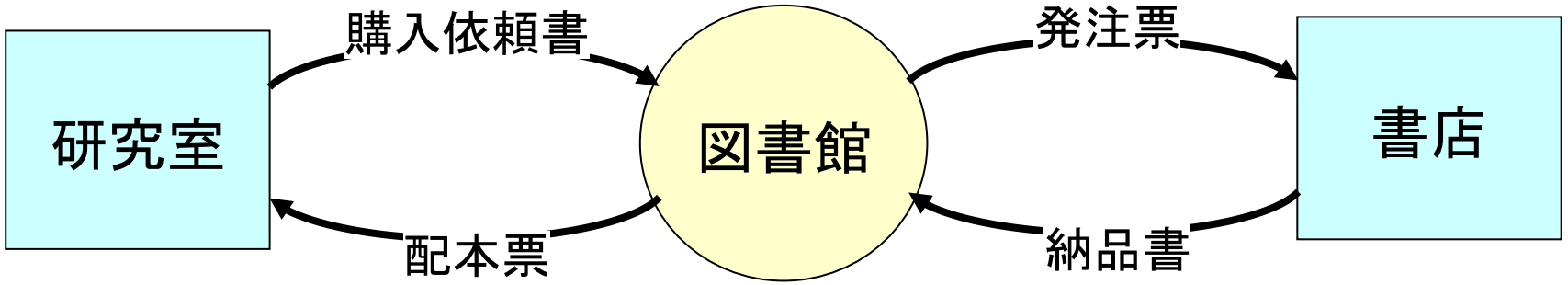
- [4.4]配本票には、書名、実際の価格、研究室名が記載されている。

配本票 ＝ 書名 ＋ 実価格 ＋ 研究室名

（5．2．4）まとめ

■以上の記述をまとめると、“図書館に求められていること”は、次のようにまとめられます。スライド(5.1)に比べて、たいぶ見通しが良くなりました。

＜DFD（コンテキスト・ダイアグラム）＞



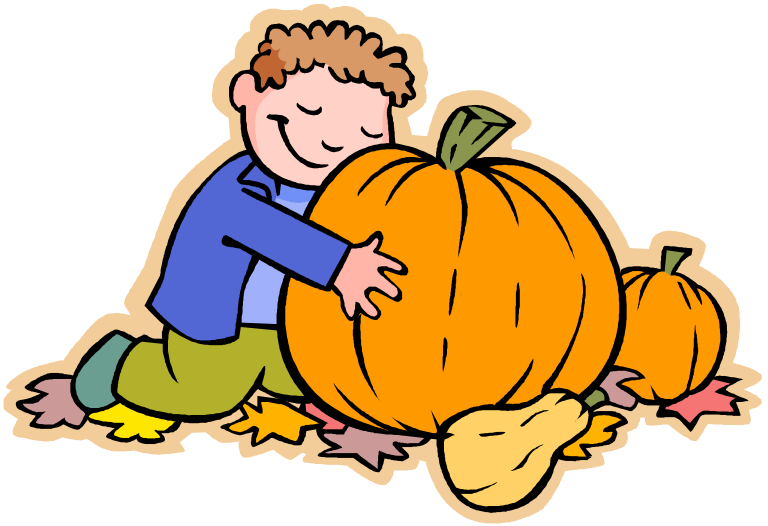
＜データ・ディクショナリ＞

購入依頼書 ＝ 書名 ＋ 価格 ＋ 研究室名
発注票 ＝ 書名 ＋ 価格 ＋ 発注者名 + 発注番号 + 発注日
発注者名 ＝ 研究室名
納品書 ＝ 発注番号 ＋ 書名 ＋ 実価格
配本票 ＝ 書名 ＋ 実価格 ＋ 研究室名

（5．2．5）演習7：課題10

■前スライドのコンテキスト・ダイアグラムを作図してください。

- 作図では、POPKINというツールを持ちます。
- POPKINの使い方については、スライド（6）を参照してください。

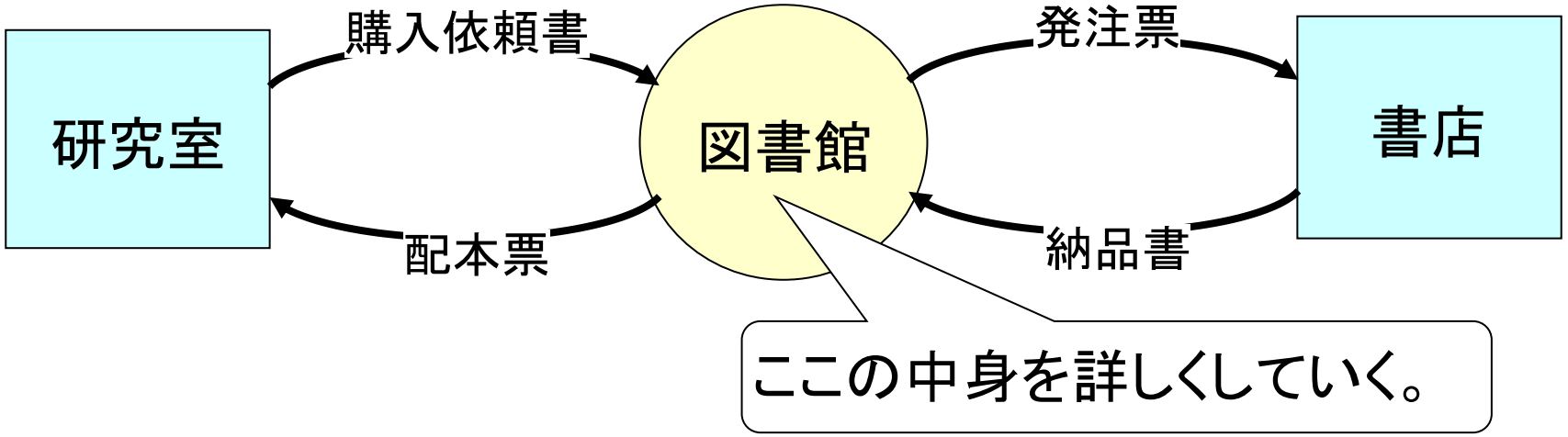


（５．３）内部の処理を考える

■スライド(5.2.4)の図には、例題(スライド(5.1))の全ての内容が示されている訳ではありません。

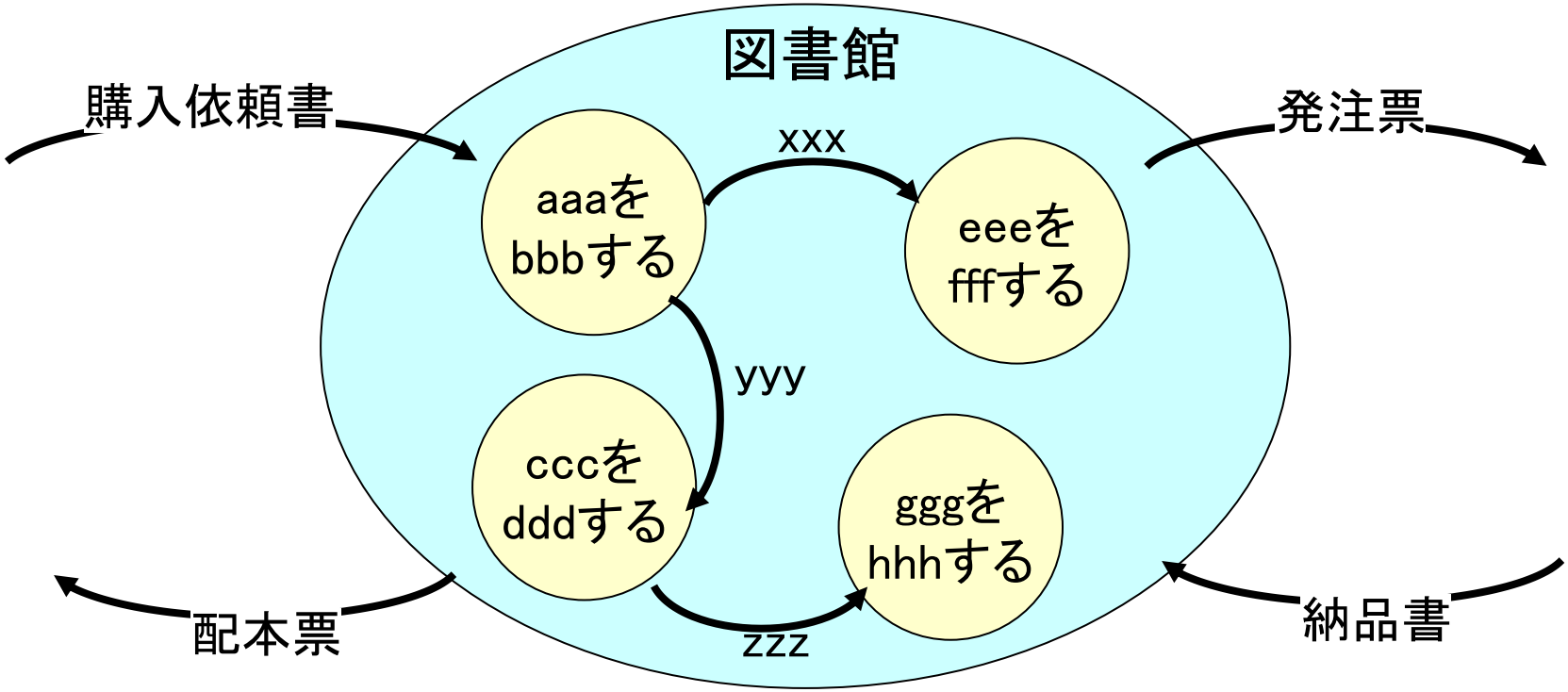
■また、具体的にどのような手順で処理をすれば良いかは、スライド(5.1)の記述以外に決めなければならない（設計しなければならない）部分もあります。

■そこで、図書館のプロセスをもっと詳しく考えていくことにします。



（５．３．１）どのように詳しくしていくか？

■図書館のプロセスの中に、データ・フローのやり取りを行う一連のプロセスを考えていきます。



■DFDでの新たな要素として、データ・ストアも扱います。

■まず、図書館の例題にて、どのようなデータがあるかを見ていきます。

（５．３．２．１）予算に関するデータ

■例題の文を見ると、予算に関するデータを持つ必要があることが分かります。

- [2.4]これは、研究室の予算残高と、支出予定額（発注したがまだ納品されていない図書の価格の合計）から調べることが出来る。

■すわなち、右のようなイメージのデータが必要です。

研究室名	予算残高	支出予定額
井戸研究室	¥30,000	¥5,000
中山研究室	¥40,000	¥2,000
:	:	:

■データ・ディクショナリにて表すと、次のようになります。

研究費予算データ ＝ {研究室名 ＋ 予算残高 ＋ 支出予定額}

“{ X }”は、Xが繰り返されることを示します。

（5．3．2．2）発注に関するデータ

■同様に、発注の控えも、持っておく必要があります。

- [4.1]書店から図書が納品書と共に届けられたら、納品書に記載された発注番号から発注票の控えを調べ、その図書が本当に発注されたものかどうかを確認し、そうであれば受け入れる。

■発注票のデータ・ディクショナリは、次のとおりです。

発注票 ＝ 書名 ＋ 価格 ＋ 発注者名 + 発注番号 + 発注日

■この発注票の控えですから、次のようになります。

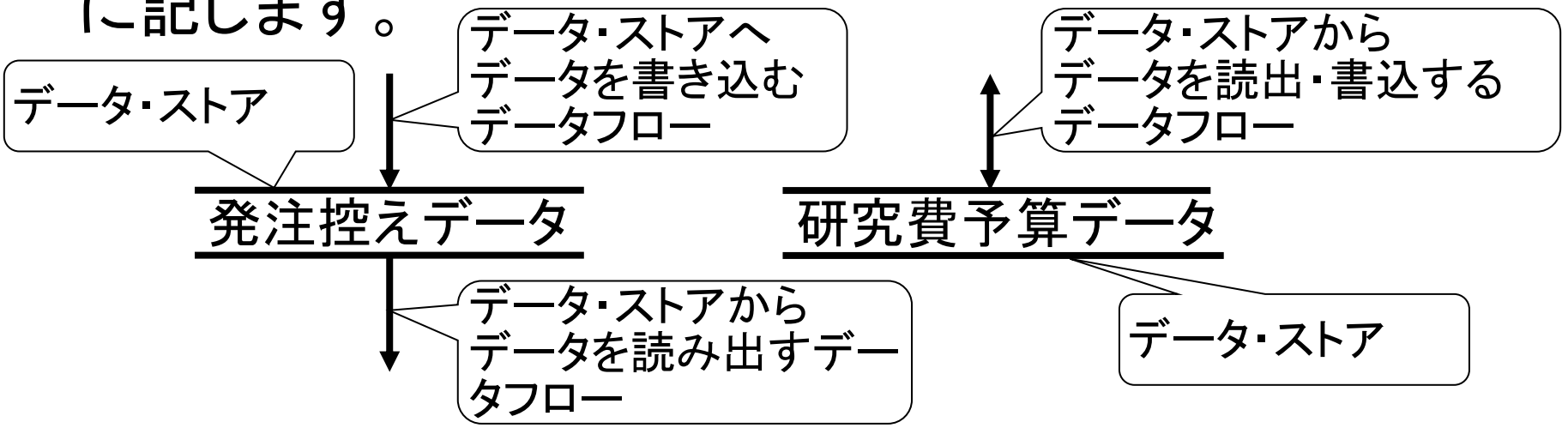
＜データ・ディクショナリ＞ 発注控え = {発注票}

※　右表のようなイメージになります。

書名	価格	発注者	発注番号	発注日
Java入門	¥3,000	井戸研究室	1001	2004/1/19
C++入門	¥2,000	中山研究室	1002	2004/1/22
:	:	:		

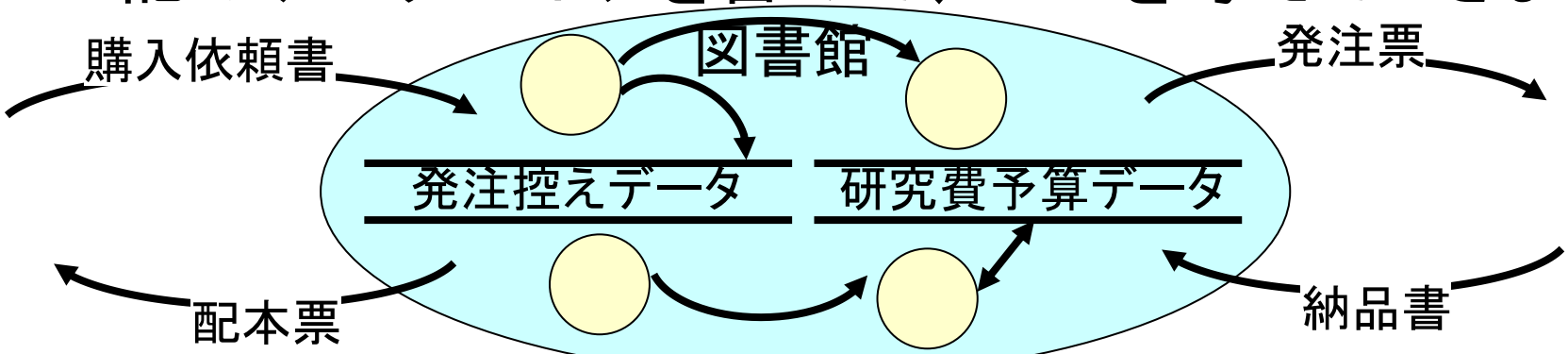
（５．３．３）データ・ストア

■スライド(5.3.2)に記したデータは、DFDの中で次のように記します。



■データ・ストアに関わる矢印には、名前を書きません。

■上記のデータ・ストアを含めて、DFDを考えていきます。



（５．３．４．１）購入依頼書を入力する

■図書館が受け取る「購入依頼書」は、用紙に書かれたものなので、図書館では、まずこれを計算機システムにデータ入力が必要になります。

■これをDFDでは、次のように記します。

■「購入依頼レコード」は「購入依頼書」と同じ内容ですから、データ・ディクショナリは、次のようになります。

購入依頼レコード = 購入依頼書

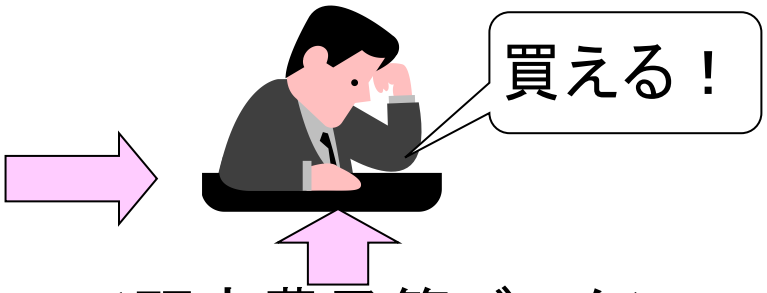
（5．3．4．2）購入依頼書を受理する―1―

■例題(スライド(5.1))には、次のように記されています。

- [2.3]....、その研究室の予算でその図書が購入出来るかどうかを調べる。

＜購入依頼レコード＞

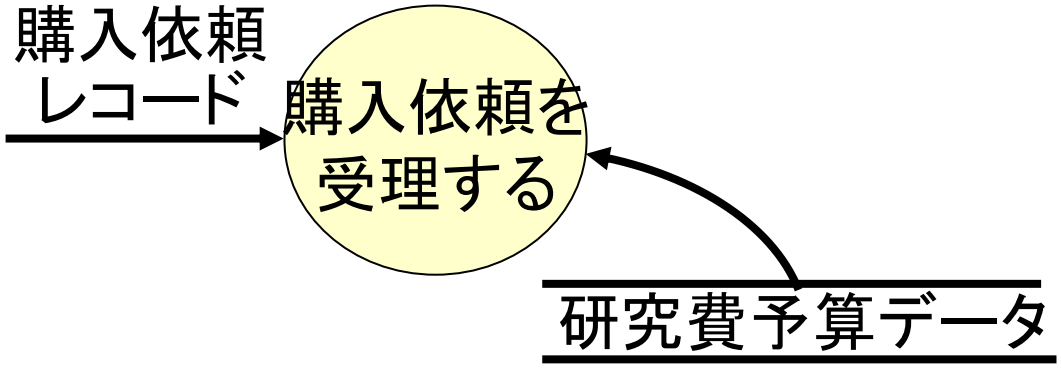
書名	価格	研究室名
Java入門	¥3,000	井戸研究室



＜研究費予算データ＞

研究室名	予算残高	支出予定額
井戸研究室	¥30,000	¥5,000

■これを、DFDでは次のように記します。



（5．3．4．3）購入依頼書を受理する―2―

■購入依頼を受け付けると、研究費予算データは更新されることになります。

- [3.5]同時に、発注した研究室の支出予定額を増やしておく。

＜購入依頼レコード＞

書名	価格	研究室名
Java入門	¥3,000	井戸研究室

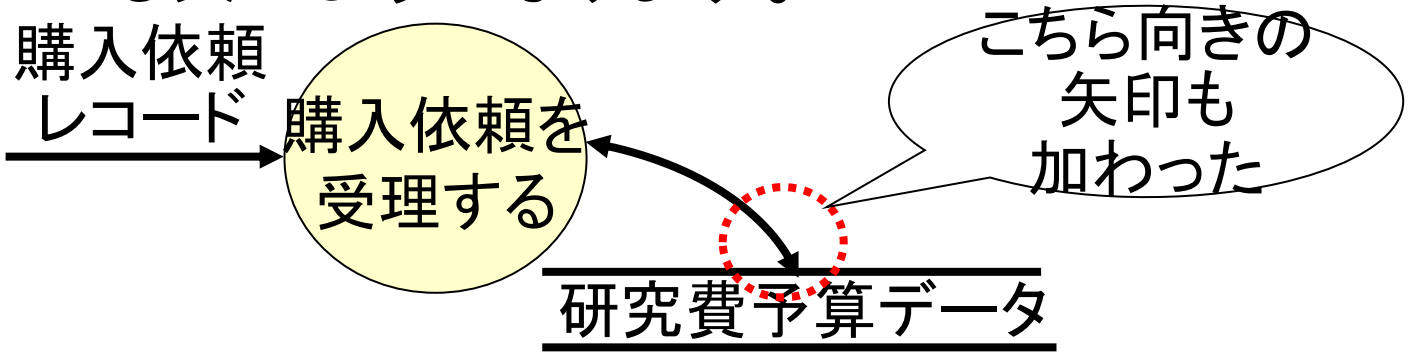
5,000円に3,000円足して、8,000円になるな。

＜研究費予算データ＞

研究室名	予算残高	支出予定額
井戸研究室	¥30,000	¥5,000

研究室名	予算残高	支出予定額
井戸研究室	¥30,000	¥8,000

■従って、DFDも次のようになります。



（5．3．4．4）購入依頼書を受理する―3―

■例題(スライド(5.1))には、次のようも記されています。

- [3.1]購入出来ることが確かめられたら、購入依頼書から発注票とその控えを作り、発注票は書店へ送る。

＜購入依頼レコード＞

書名	価格	研究室名
Java入門	¥3,000	井戸研究室

＜発注レコード＞

書名	価格	発注者	発注番号	発注日
Java入門	¥3,000	井戸研究室	1001	2004/1/19

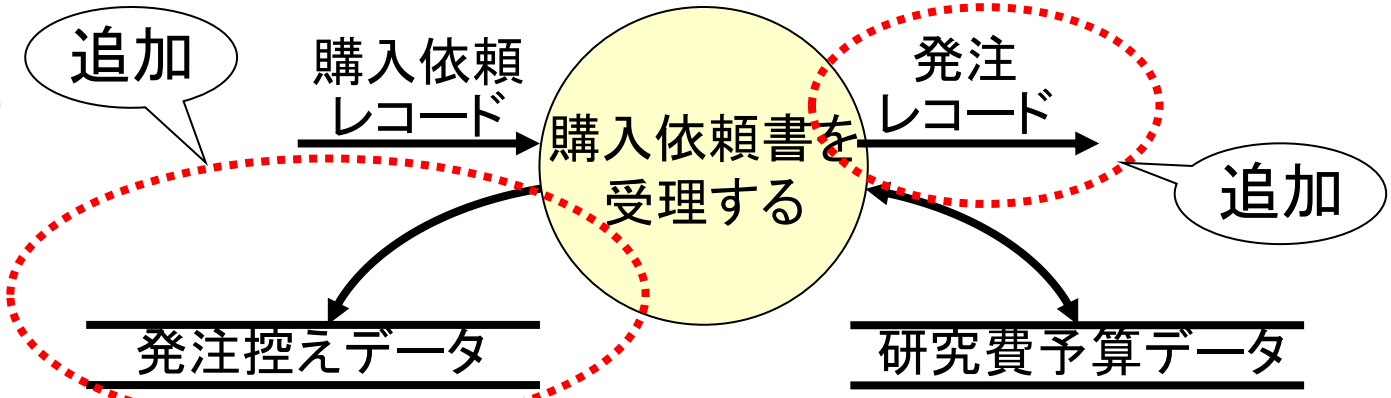
＜研究費予算データ＞

研究室名	予算残高	支出予定額
井戸研究室	¥30,000	¥8,000

＜発注控えデータ＞

書名	価格	発注者	発注番号	発注日
Java入門	¥3,000	井戸研究室	1001	2004/1/19

■DFDは、右のようになります。

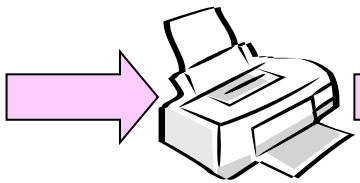


（５．３．４．５）発注票を印刷する

■書店に送る、「発注票」は、用紙に印刷されたものなので、図書館では、まず発注レコードよりこれを印刷します。

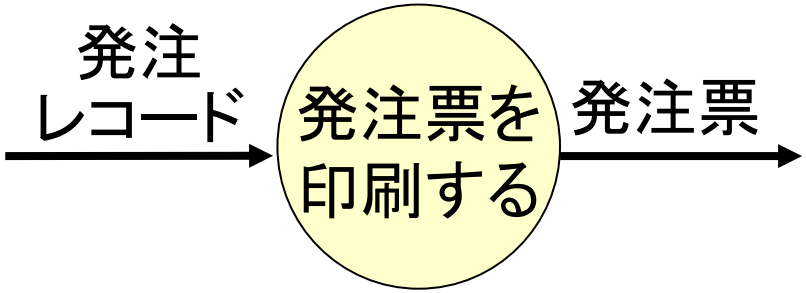
＜発注レコード＞

書名	価格	発注者		発注日
Java入門	¥3,000	井戸研究室	1001	2004/1/19



＜発注票＞

■これをDFDでは、次のように記します。

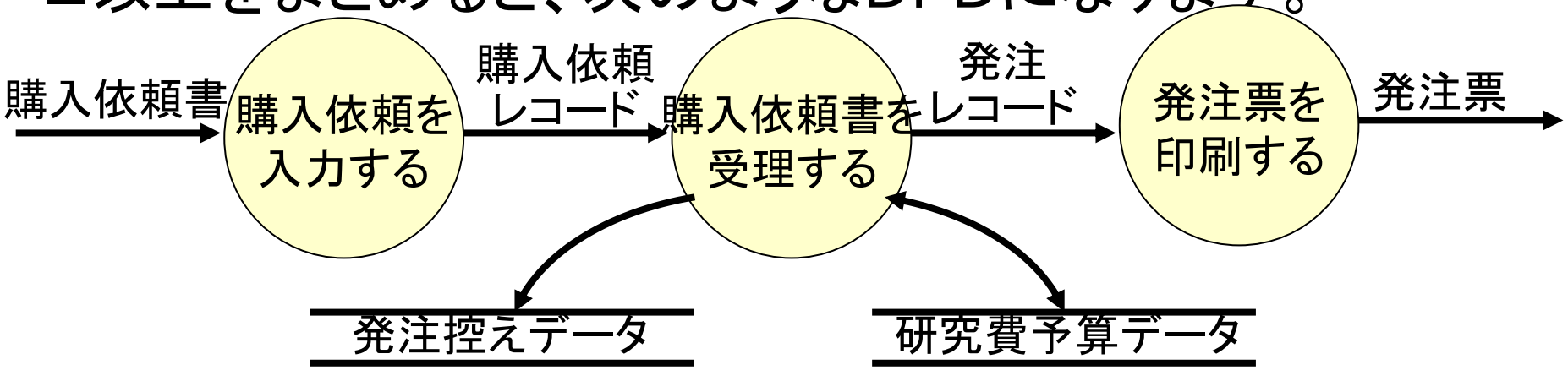


■「発注レコード」は「発注票」と同じ内容ですから、データ・ディクショナリは、次のようになります。

発注レコード = 発注依頼書

（5．3．4．6）購入依頼・発注関係のDFD

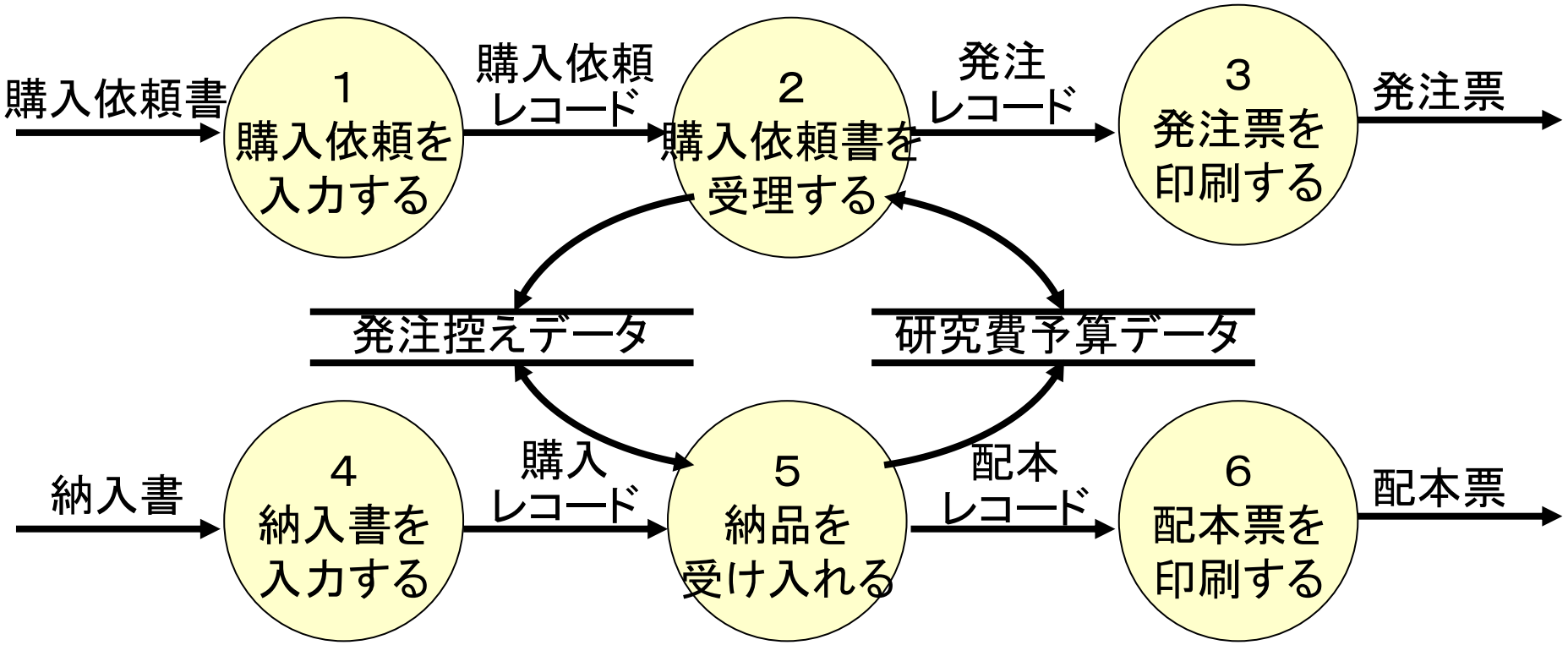
■以上をまとめると、次のようなDFDになります。



購入依頼書 ＝ 書名 ＋ 価格 ＋ 研究室名
発注票 ＝ 書名 ＋ 価格 ＋ 発注者名 + 発注番号 + 発注日
発注者名 ＝ 研究室名
発注控えファイル = {発注票}
研究室予算ファイル = {研究室名 + 予算残高 + 支出予定額}
購入依頼レコード = 購入依頼書
発注レコード = 発注票

■納入・配本側も同じように考えていくと、次スライドのようなDFDを得ます。これを、DFDの“ダイアグラム0”と呼びます。プロセスには、番号を振っています。

（5．3．5）図書館のDFD（ダイアグラム0）



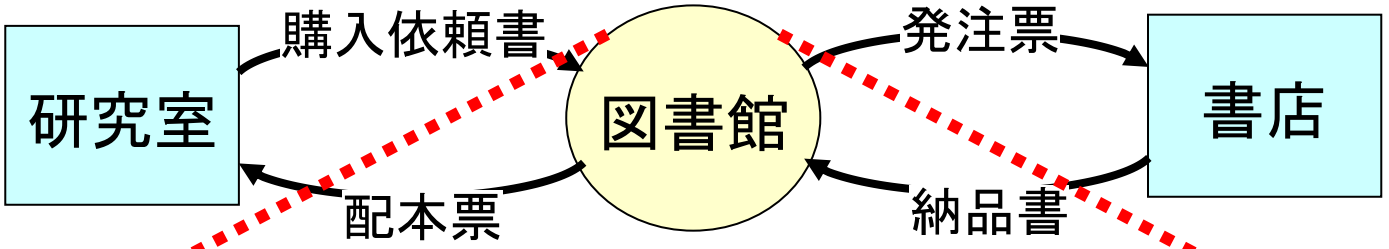
購入依頼書 ＝ 書名 ＋ 価格 ＋ 研究室名
発注票 ＝ 書名 ＋ 価格 ＋ 発注者名 + 発注番号 + 発注日
発注者名 ＝ 研究室名
納品書 ＝ 発注番号 ＋ 書名 ＋ 実価格
配本票 ＝ 書名 ＋ 実価格 ＋ 研究室名
発注控えファイル = {発注票}
研究室予算ファイル = {研究室名 + 予算残高 + 支出予定額}

購入依頼レコード = 購入依頼書
発注レコード = 発注票
配本レコード = 配本票
納入レコード = 発注番号 + 実価格

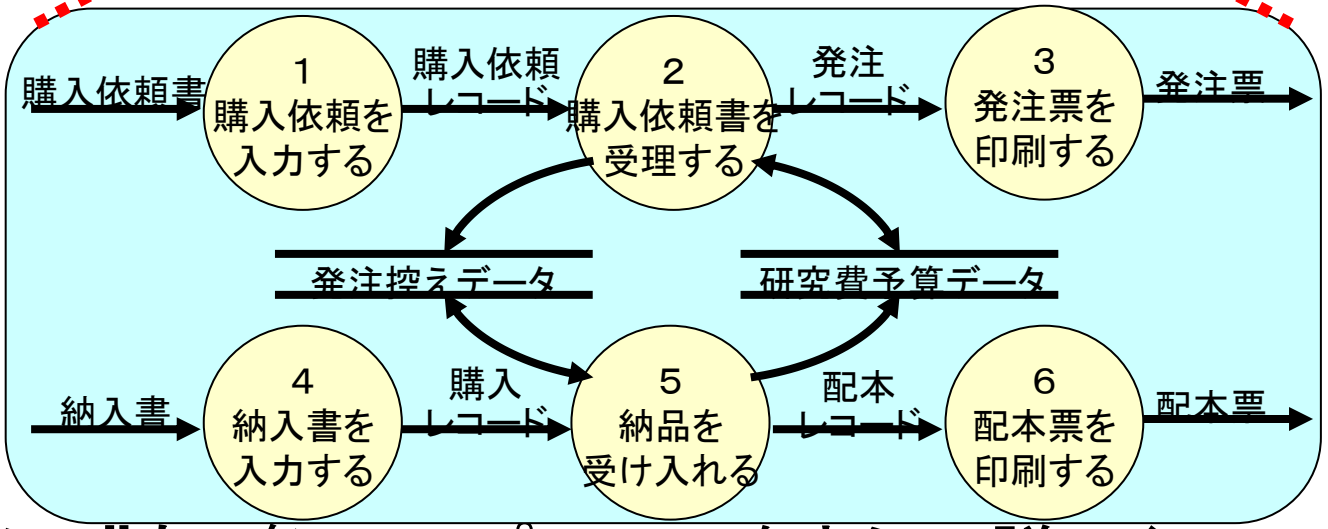
（5．3．6）2つの図の関係

■スライド(5.3.5)のDFDは、スライド(5.2.4)のDFDの図書館のプロセスを詳しくしたものです。

(5.2.4)
DFD
コンテキスト・
ダイアグラム）



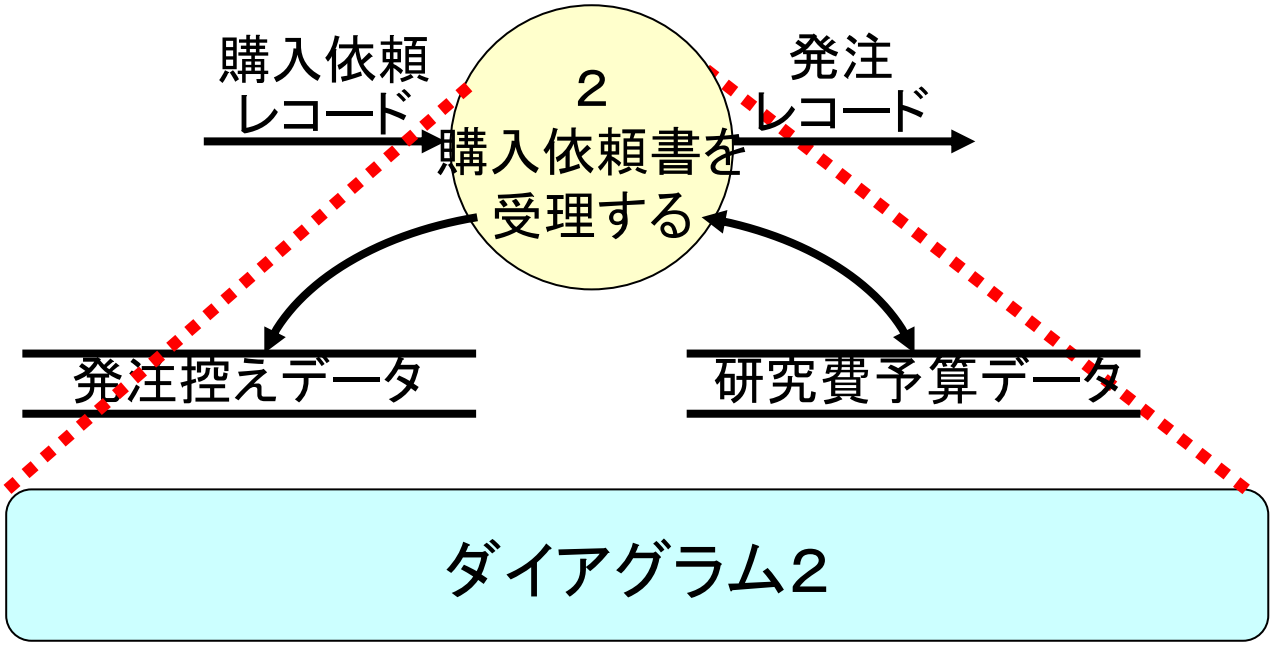
(5.3.5)
DFD,
ダイアグラム0



■“ダイアグラム0”も、各々のプロセスをさらに詳しくしていくことが可能です。

（5．4）プロセスをさらに詳細化する

■“ダイアグラム0”中の、「2．購入依頼書を受理する」というプロセスをさらに詳細化していきます。

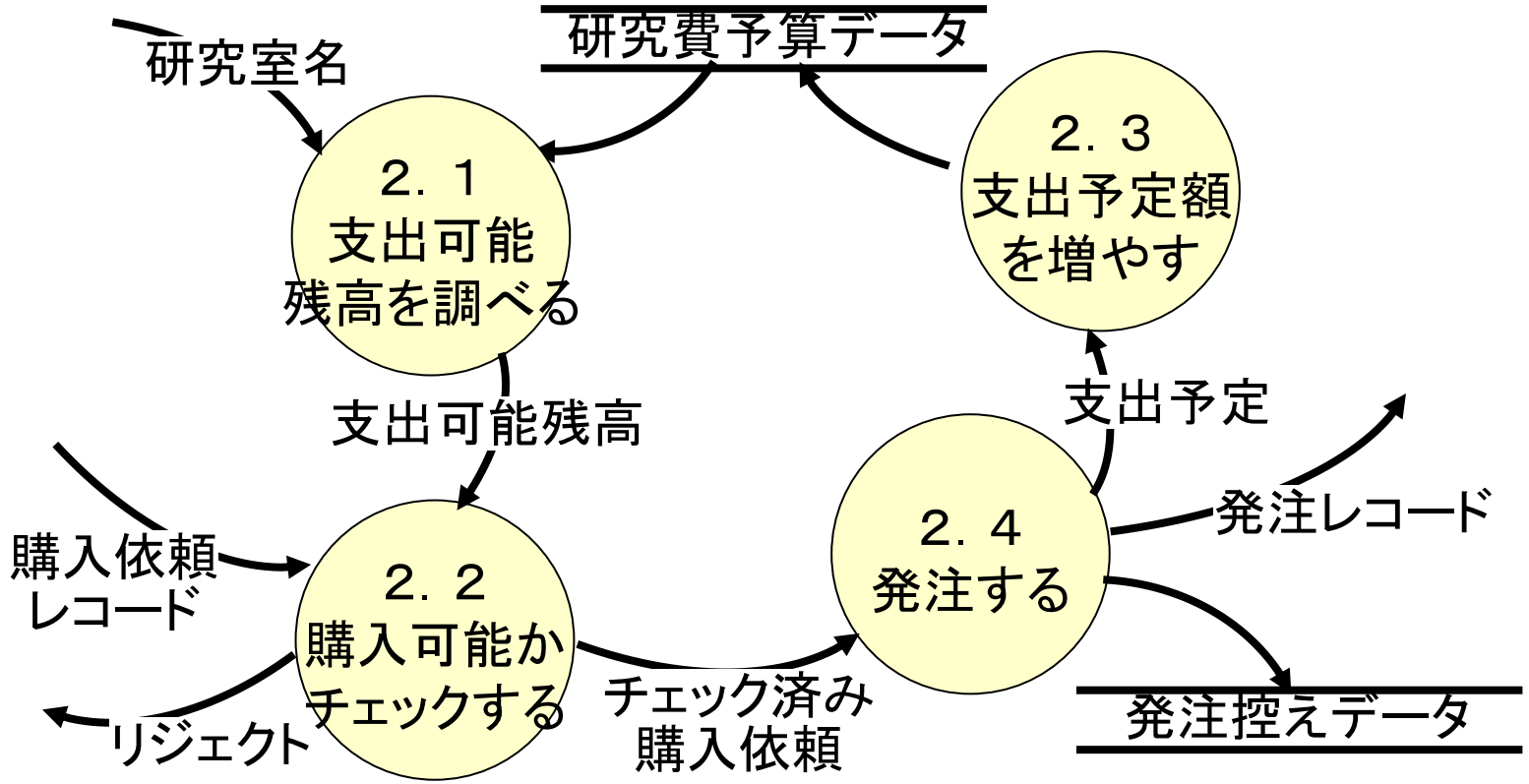


■“ダイアグラム0”中で、番号“2”が振られたプロセスの詳細化したDFDであることから、“ダイアグラム2”と呼びます。

（５．４．１）ダイアグラム２

■図中、「支出予定」、「チェック済み購入依頼」などの新たなデータ・フローが描かれています。

■図中、「リジェクト」は、重要な処理とは見なさず、“ダイアグラム０”では描いていませんでした。

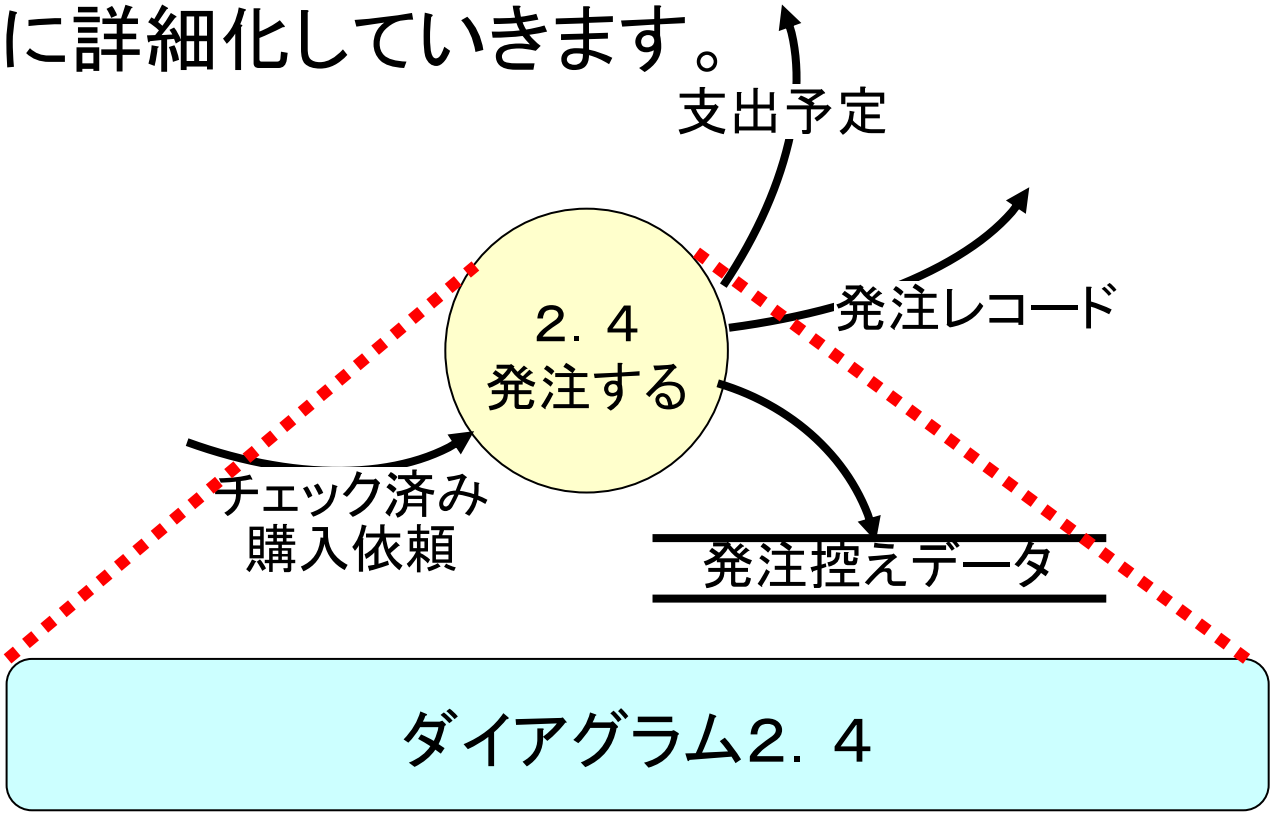


（５．４．２）演習８：課題１１

■「支出予定」、「チェック済み購入依頼」のデータフローの、データ・ディクショナリを書いてください。

（５．４．３）“プロセス２．４”をさらに詳細化する

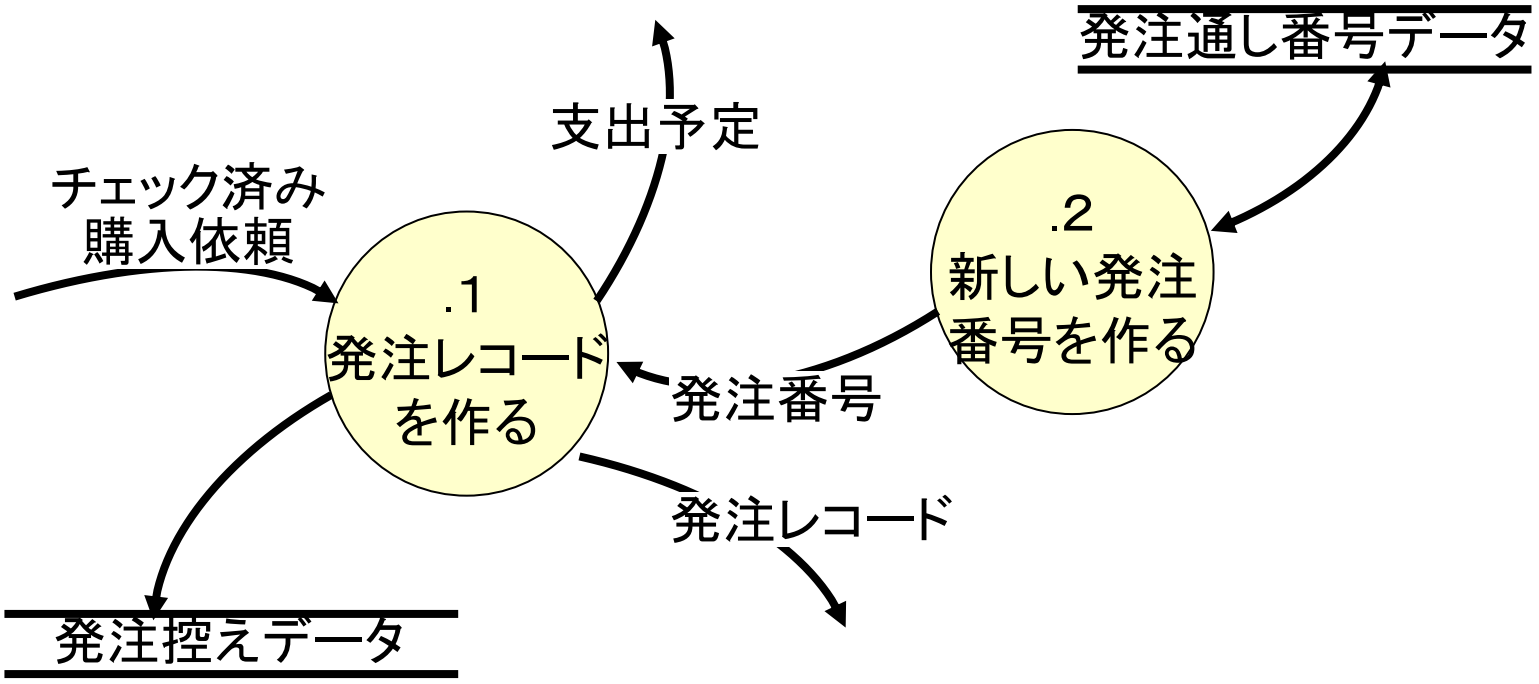
■“ダイアグラム2”中の、「２．４　発注する」というプロセスをさらに詳細化していきます。



■前と同じ流儀で、作成するDFDを“ダイアグラム２．４”と呼びます。

（５．４．４）ダイアグラム２．４

■「発注通し番号データ」は、“ダイアグラム２．４”のみで現れるデータ・ストアです。このようなデータを、“内部データ”と呼びます。



（５．４．５）ミニ仕様書

■これ以上詳細化が必要なくなった場合、そのダイアグラムに現れるプロセスを基本機能と呼びます。

■各基本機能について、何を行うかを記したものを、“ミニ仕様書”と呼びます。

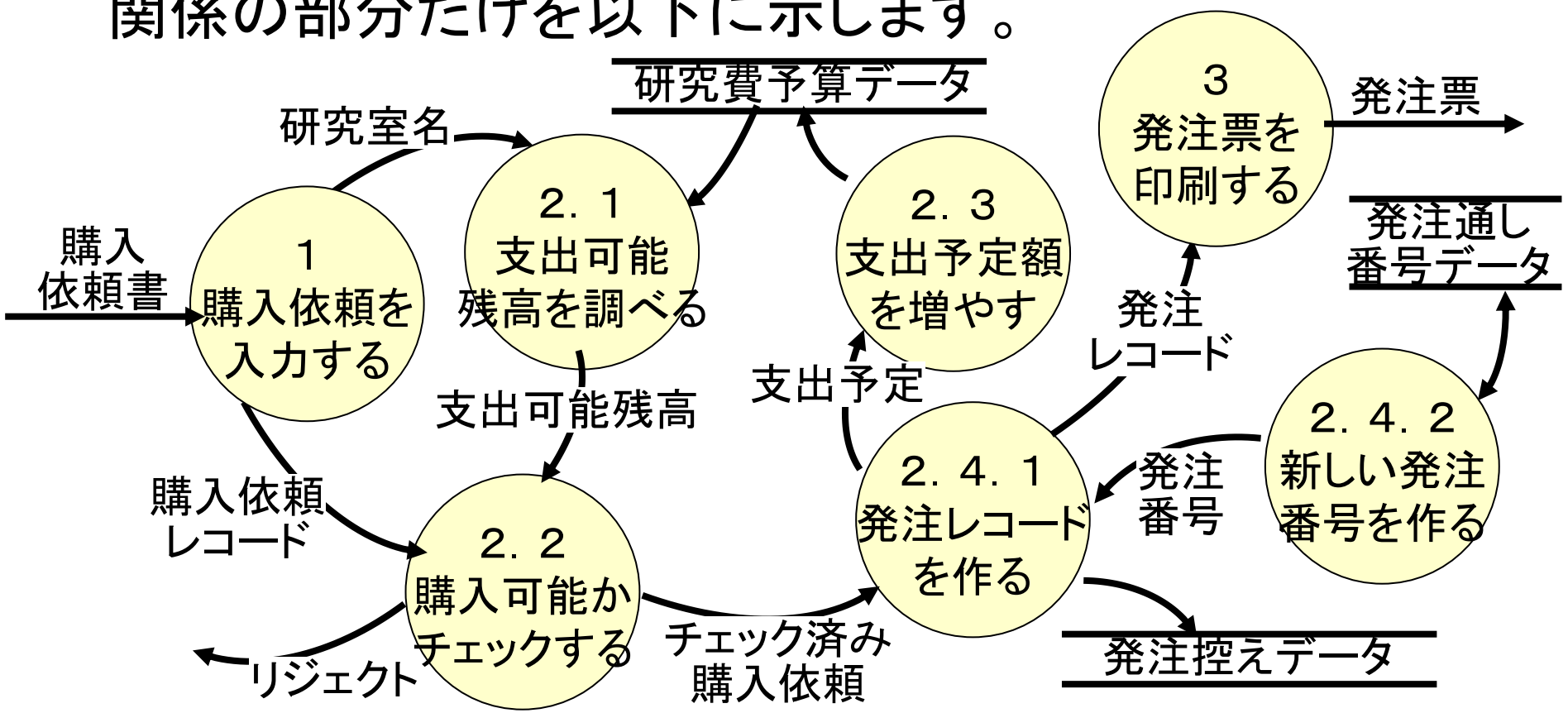
プロセス：２．４．１
プロセス名：発注レコードを作る

入力された「チェック済み購入依頼」について以下の動作を行う。
「発注番号」を得る。
「発注日」を得る。／＊マシンから＊／
「発注者名」と「価格」から「支出予定」を作成する。
「書名」、「価格」、「発注者名」、「発注番号」、「発注日」から
　　「発注レコード」を作製する。
「発注控えデータ」に「発注レコード」を書き込む。
「支出予定」、「発注レコード」を出力する。

（５．４．６）全体DFD

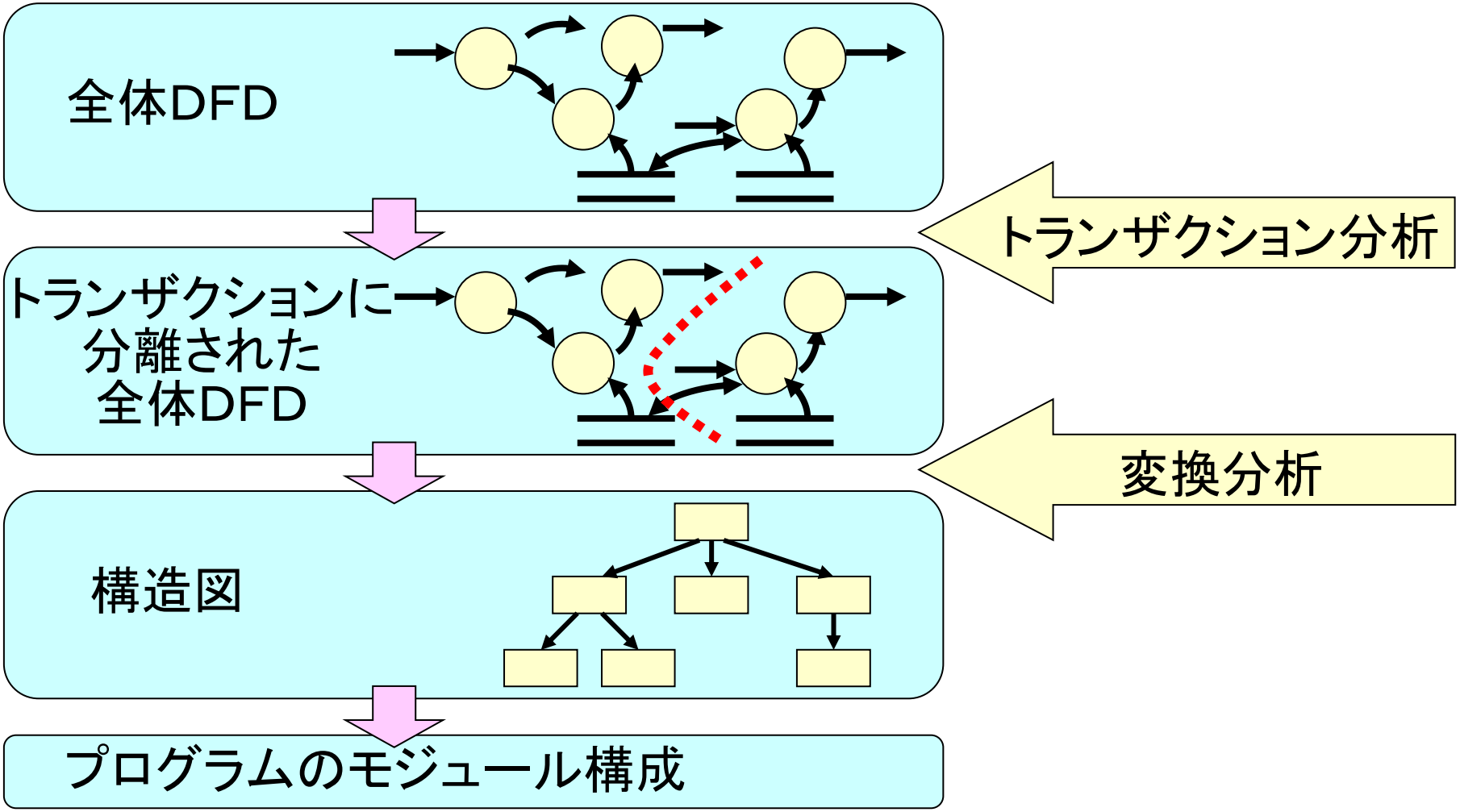
■詳細化により作成したダイアグラムを、ダイアグラム0に埋め込んだものを、全体DFDといいます。

■全体DFDのうち、（スペースの関係で）購入依頼・発注関係の部分だけを以下に示します。



（5．5）DFDを何に使うのか？

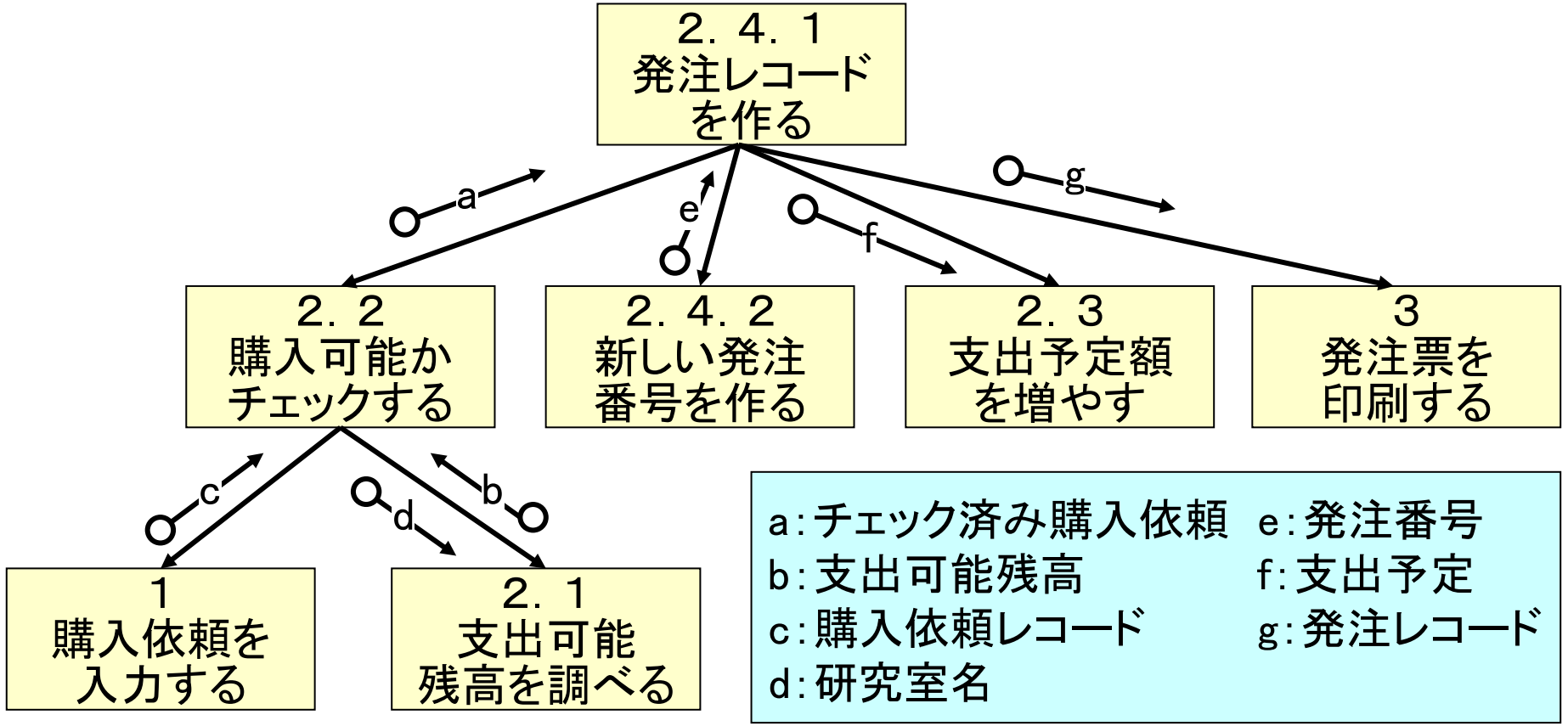
■構造化分析・設計（SSA/AD）では、全体DFDを作成した後、次のような手順でプログラムの作成に至ります。



（５．５．１）構造図

■購入依頼の処理部分の構造図を示します。

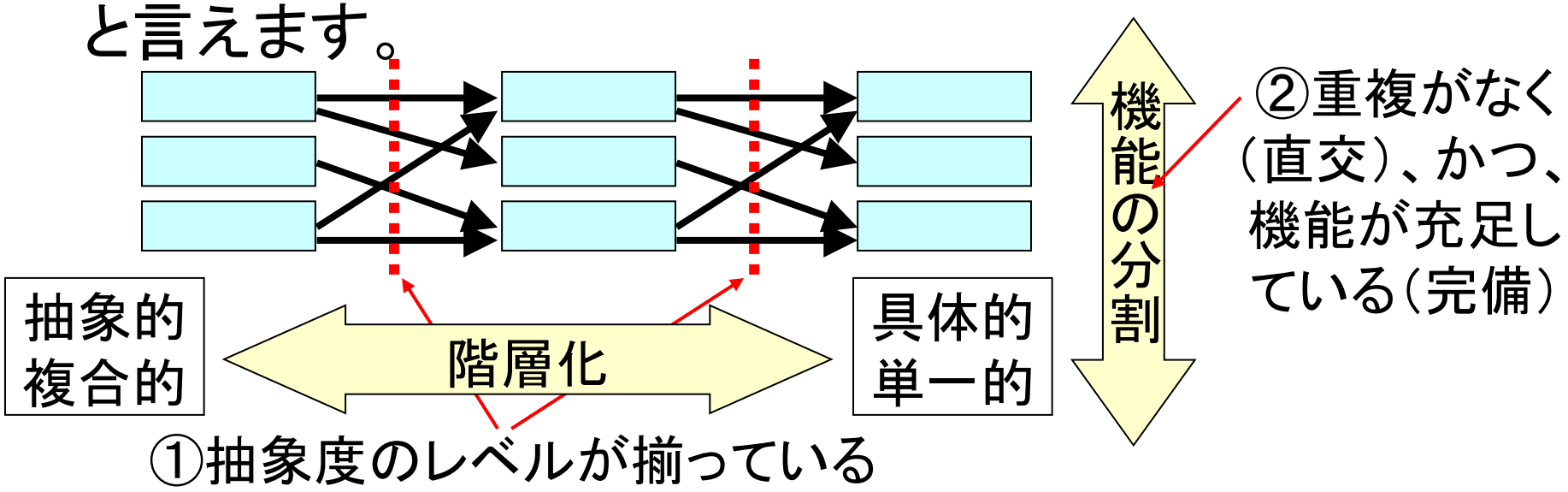
■ここでは、構造図の内容については説明しませんが、何となく雰囲気が分かるのではないでしょうか。



（5．5．2）構造図を見て気が付くこと

■前スライドの構造図を見て、何か気が付くことはありませんか？

■SSA/ADの構造図は、“階層構成・機能分割”的であると言えます。



■従って、構造図から導き出されるプログラムは、“オブジェクト指向”とは、相容れない性質のもとのなります。

（５．５．３）DFDの意味

- 講師は、「これからの時代は、“オブジェクト指向”の頭でプログラミングをすべき」と考えています。
- 従って、全体DFDよりも先のSSA/ADの手順（“トランザクション分析”、“変換分析”）は、勉強する必要性は低いと考えています。
- しかしながら、今まで見てきたように、DFD自体はシステムの動きをデータの流れに沿って理解する上で、大きな助けになるものです。
- UMLを用いたオブジェクト指向設計においても、必要に応じてDFDを援用することは、良いことであると考えます。実際の設計現場でも行われているようです。

（５．６）演習９：課題１２

- 次の文章を読んで、倉庫事務所のコンテキストダイアグラムとデータフローダイアグラム0とを作成せよ（これを提出してもらいます）。
- 家具販売会社の倉庫事務所では、買い付け担当者からの入庫依頼を受けて家具を倉庫に搬入し、販売店からの出庫依頼を受けて家具を倉庫から搬出する業務を行っている。
- 倉庫事務所は、買い付け担当者から入庫依頼書を受け取る。入庫依頼書には、品名、型番、個数、日付が記入されている。入庫依頼書を受け取ったら、記入漏れがないことを確認し、倉庫内に収納可能であるかを調べる。これは、倉庫の使用・予約状況から調べることが出来る。

（５．６　つづき）演習９：課題１２

■入庫できることが確かめられたら、入庫承諾書を発行する。入庫承諾書には、品名、型番、個数、日付、倉庫部屋番号が記入されている。

■倉庫事務所は、販売店からの出庫依頼書を受け取る。出庫依頼書には、品名、型番、個数、日付が記入されている。出庫依頼書を受け取ったら、記入漏れがないことを確認し、倉庫内に在庫があり、出庫可能であるかを調べる。これは、在庫一覧から調べることが出来る。

■出庫できることが確かめられたら、運送会社に配送依頼書を発行する。配送依頼書には、品名、型番、個数、日付、倉庫番号が記入されている。

（6．1）POPKINのインストール

■一旦、ログアウトをおこなって、“岐阜経済大学管理”のユーザで入り直してください（やり方は説明します）。

■井戸のネットワークドライブから、次のフォルダをコピーしてください（コピー先はMyDocumentsでOKです）。

- Popkin

■上記フォルダを開き、“POPDEMO”のアイコンをダブルクリックしてください（①）。

（6．2）POPKINの起動

■次のフォルダを開いてください。
- C:¥POPDEMO

■“SAM”のアイコンをダブルクリック（①）してください。

■POPKINの画面が現れ（②）、その中にユーザIDを問うダイアログが表示されます（③）。適当な名前（自分の名前）を入力して、[確認]をクリック（④）してください。

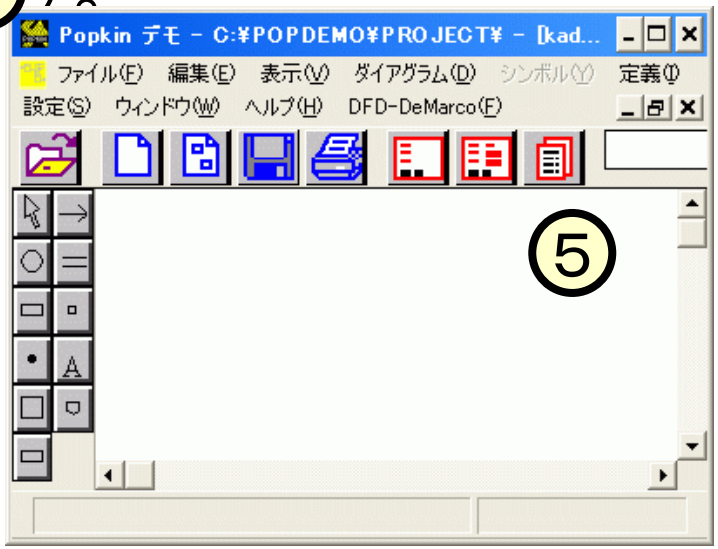
■ユーザIDは、次回のために覚えておいてください。

（6．3）DFDの作成

■メニューから、[ダイアログ]–[新規作成]をクリックします（①）。

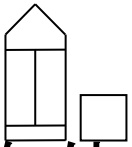
■「新規ダイアグラム」のダイアログの中で、[名前]を記入し（②）、[DFD]にて[DFD(Yourdon/DeMarco)]をクリック（③）し、確認をクリック（④）します。

■作画が可能な画面となります（⑤）。



（6．4）エクスターナルシンボルの追加

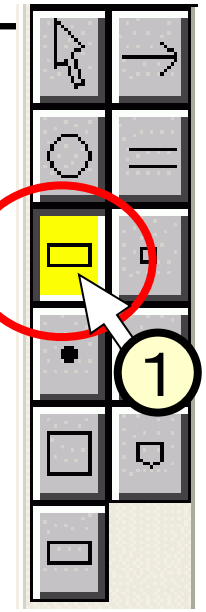
■画面左のシンボルボタン群の中から、エクスターナルボタンをクリック（①）します。

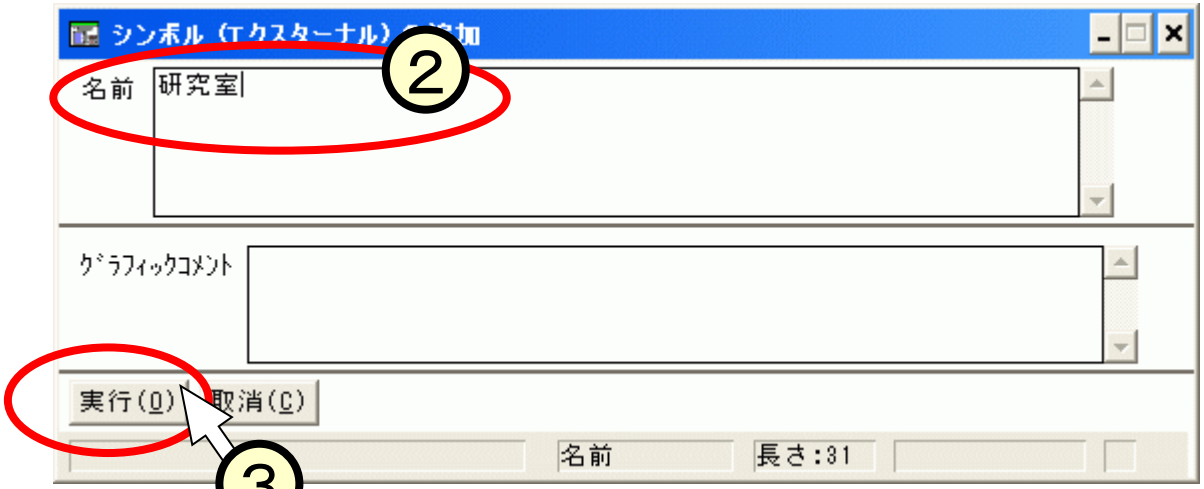
■マウスポイントが、  の形になります。

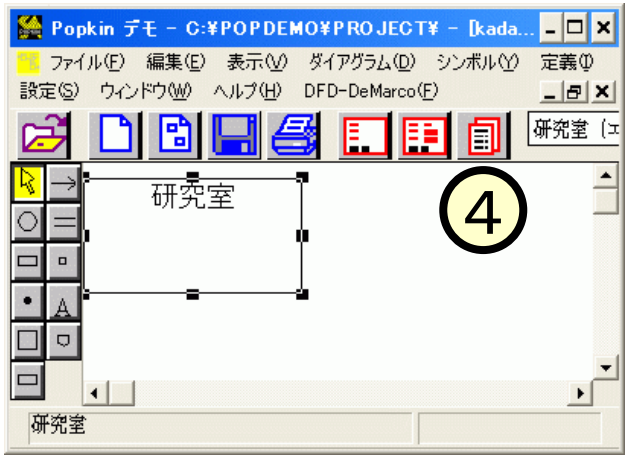
■作画領域中、適当な位置をクリックします。

■「シンボルの追加」ダイアログにて、[名前] を入力し（②）、[実行]をクリック（③）します。

■エクスターナルが作画領域に現れます（④）。





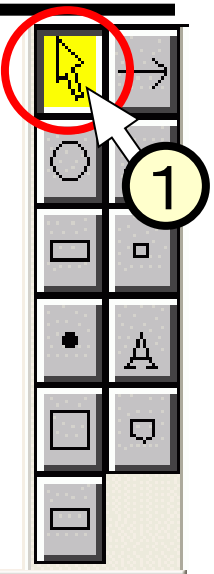


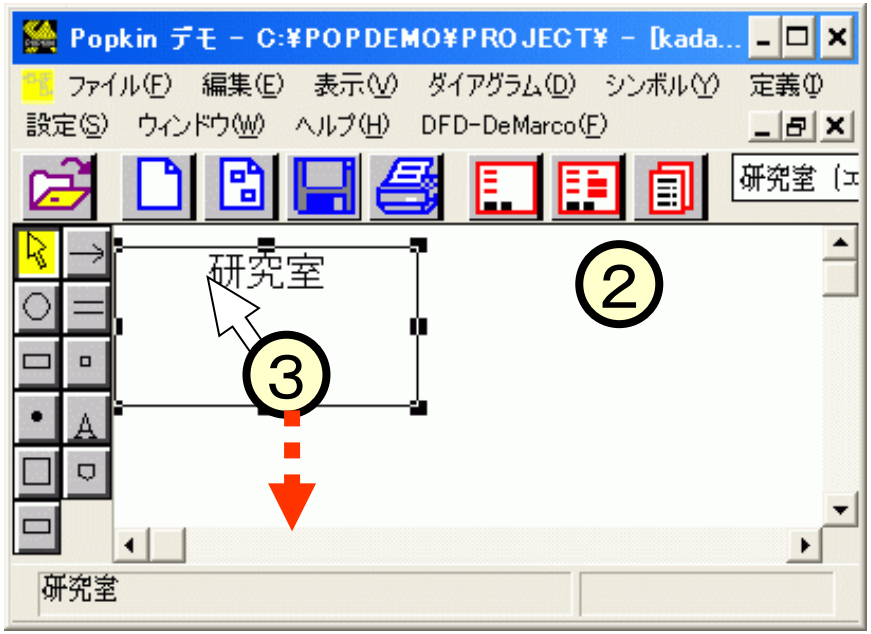
（6．5）シンボルの移動

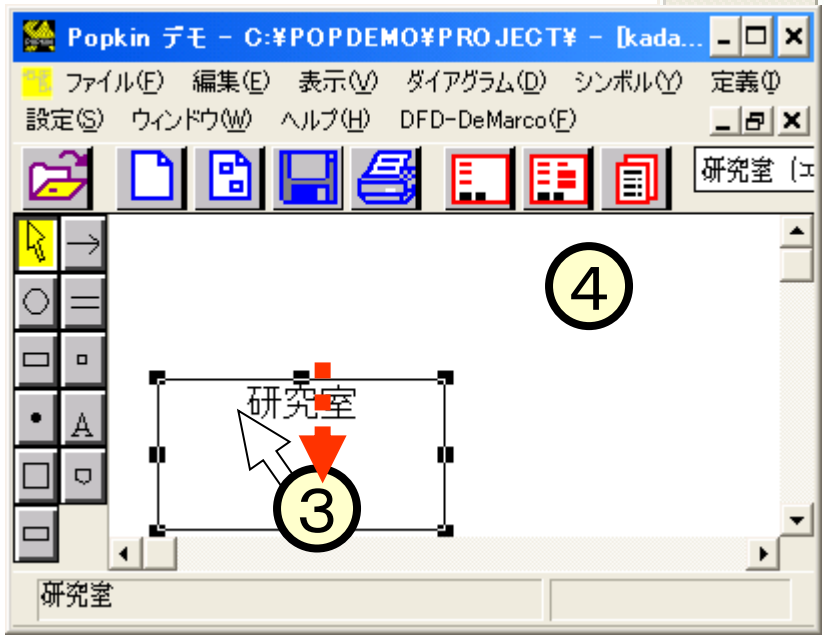
■矢印のボタンをクリックします（①）。

■シンボルをクリックして、選択された状態（角と辺につまみ“■”が表示されます）とします（②）。

■ドラッグする（③）ことで、シンボルが移動します（④）。



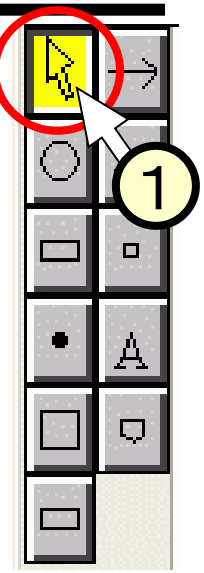


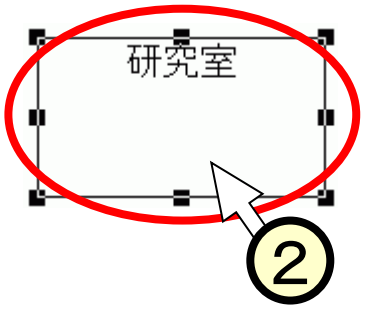


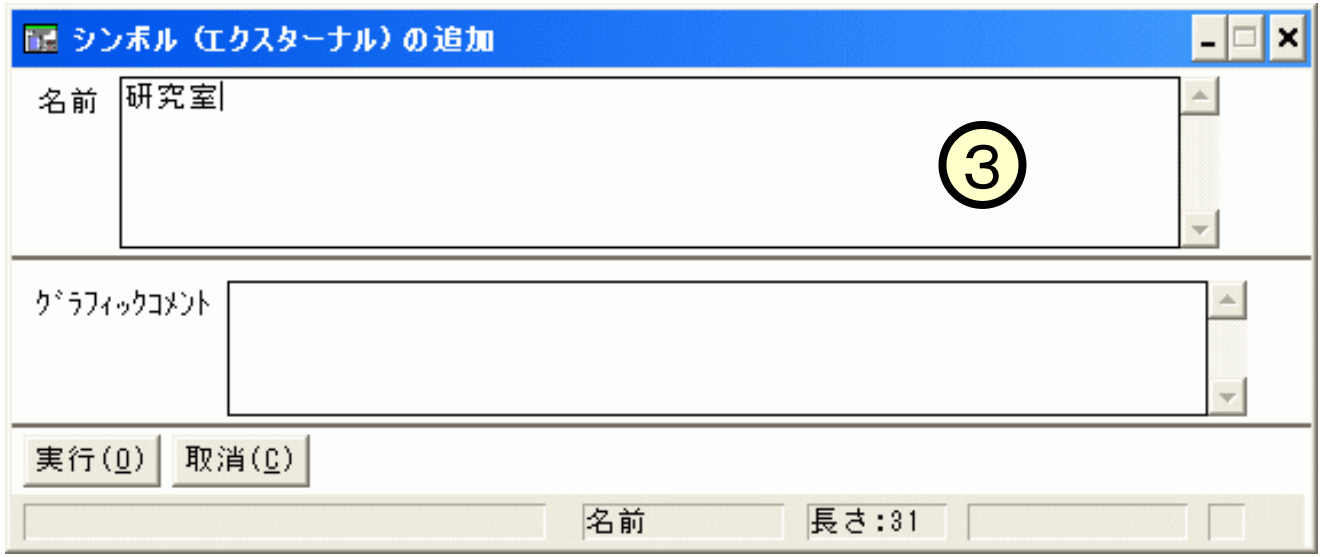
（6．6）シンボルの編集

■矢印のボタンをクリックします（①）。

■シンボルをダブルクリックすると（②）、「シンボルの追加ダイアログ」が現れ（③）、名前等の編集が可能となります。





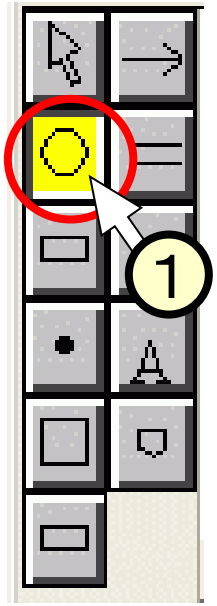


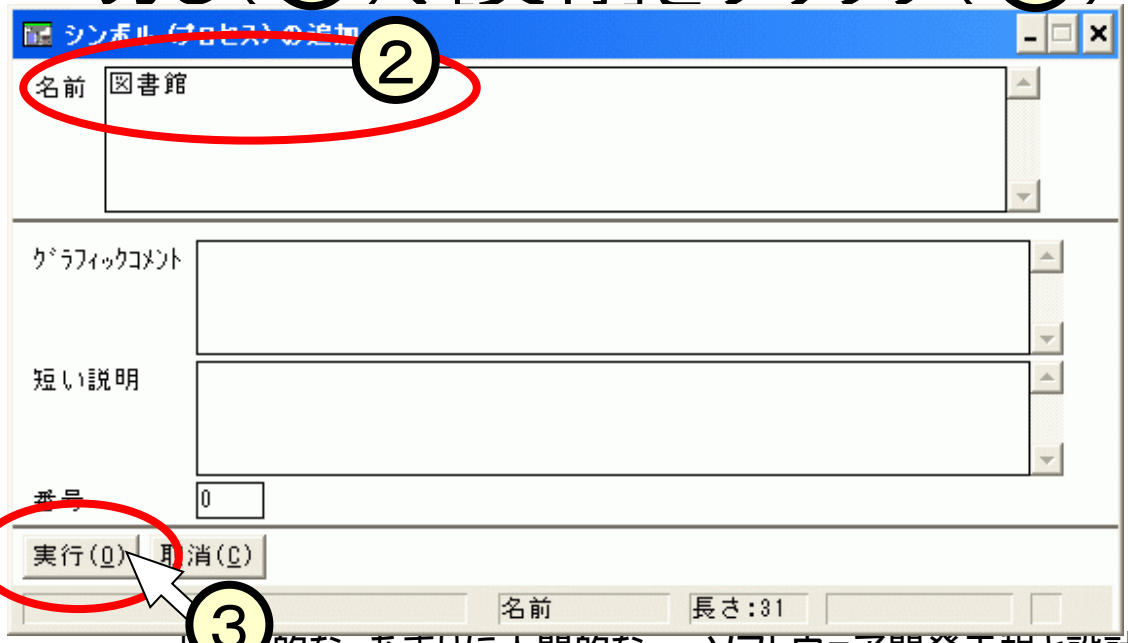
（6．7）プロセスシンボル要素の追加

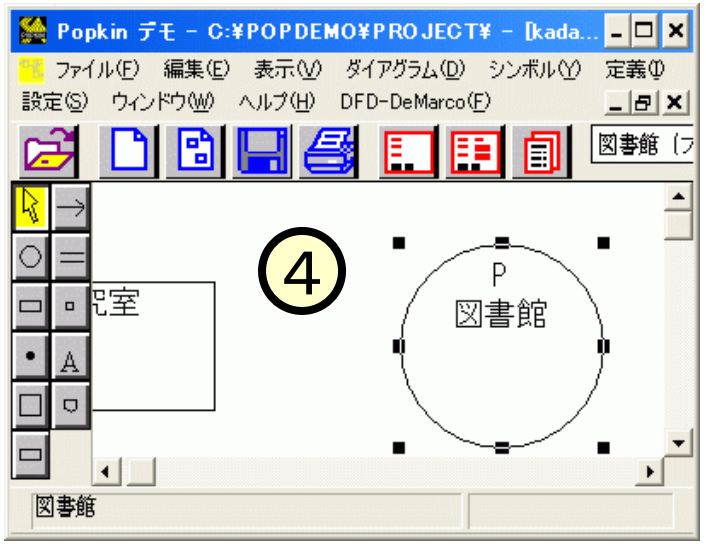
■プロセスシンボル（④）は、エクスターナルとほぼ同様に追加出来ます。

■画面左のシンボルボタン群の中から、エクスターナルボタンをクリック（①）します。

■「シンボルの追加」ダイアログにて、[名前] を入力し（②）、[実行]をクリック（③）します。





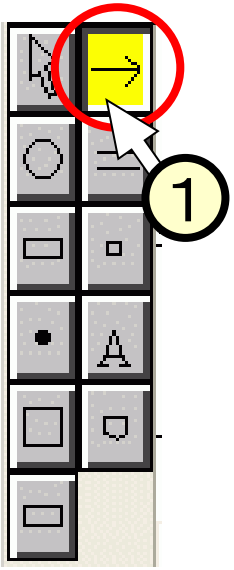


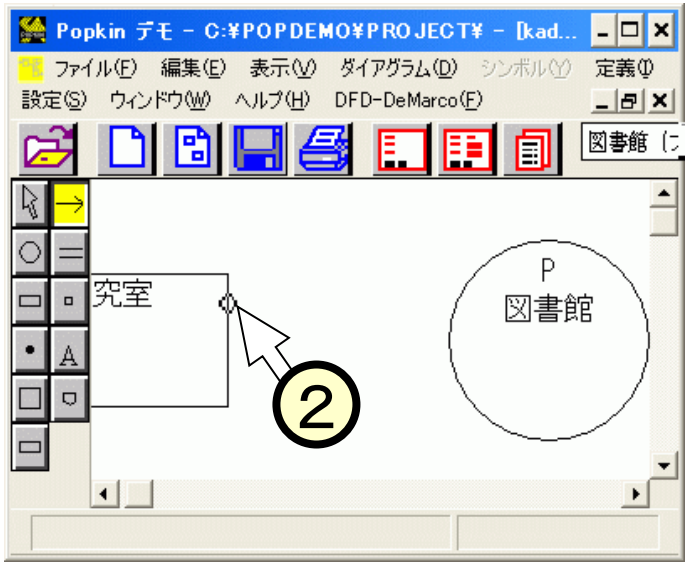
（6．8．1）データフローの追加ーその1ー

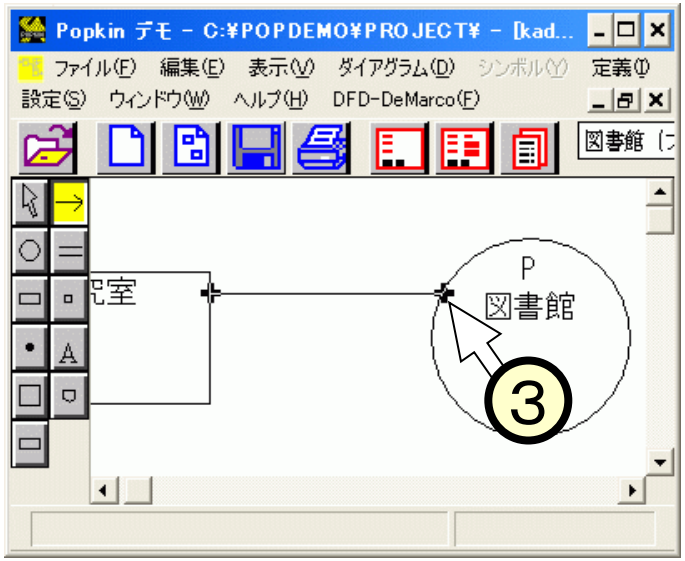
■データフローのボタンをクリックします（①）。

■矢印の元となるシンボルをクリックします（②）。

■矢印の先となるシンボルをクリックします（③）。







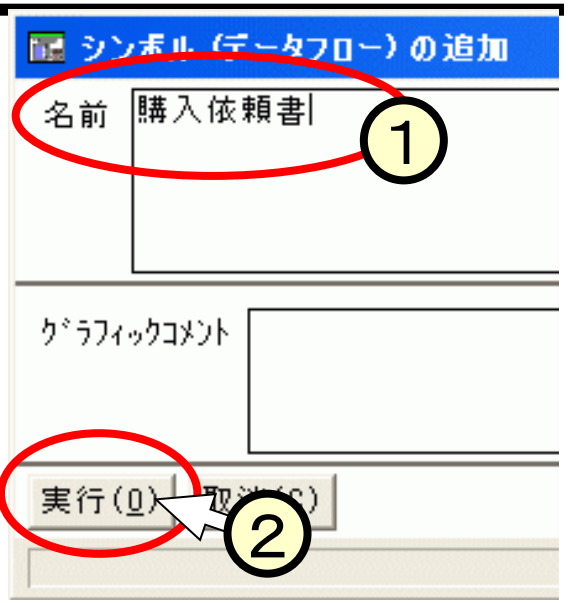
（6．8．2）データフローの追加ーその2ー

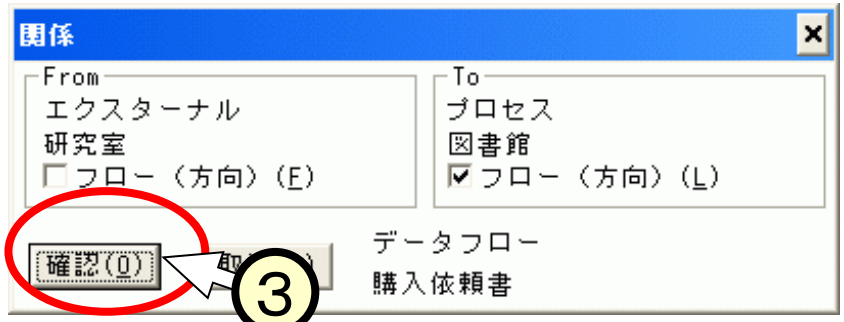
■「シンボルの追加」ダイアログで、[名前]を入力し（①）、[実行]をクリックします（②）。

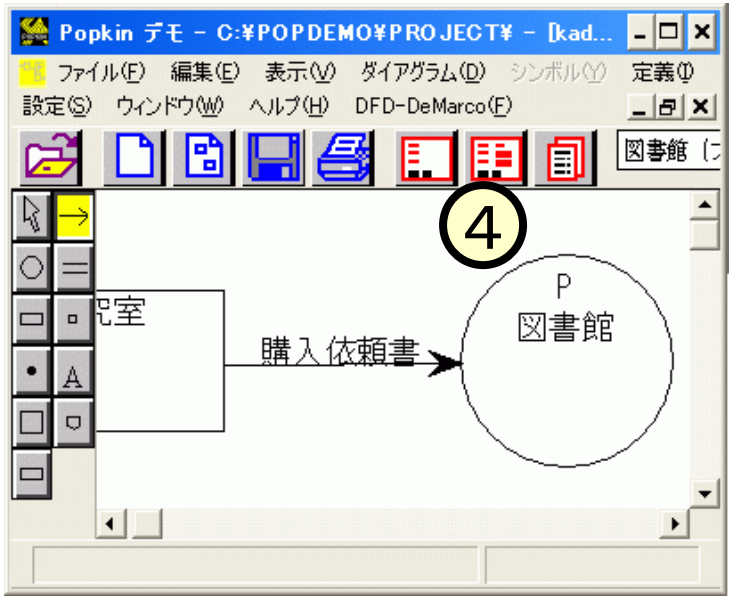
■「関係」ダイアログで、[確認]をクリックします（③）。

■データフローが描画されます（④）。

■シンボルと同様に、ダブルクリックにより名前の変更が行えます。







（6．9）ファイルのセーブ／オープン

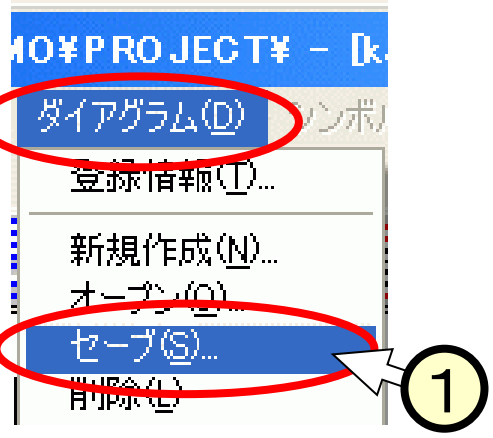
■＜＜セーブ＞＞メニューから、[ダイアグラム]–[セーブ]をクリックします（①）。

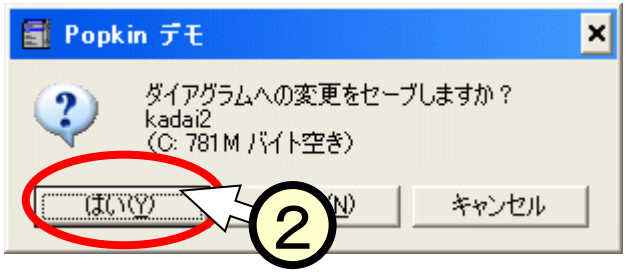
■確認ダイアログにて、[はい]をクリックします（②）。

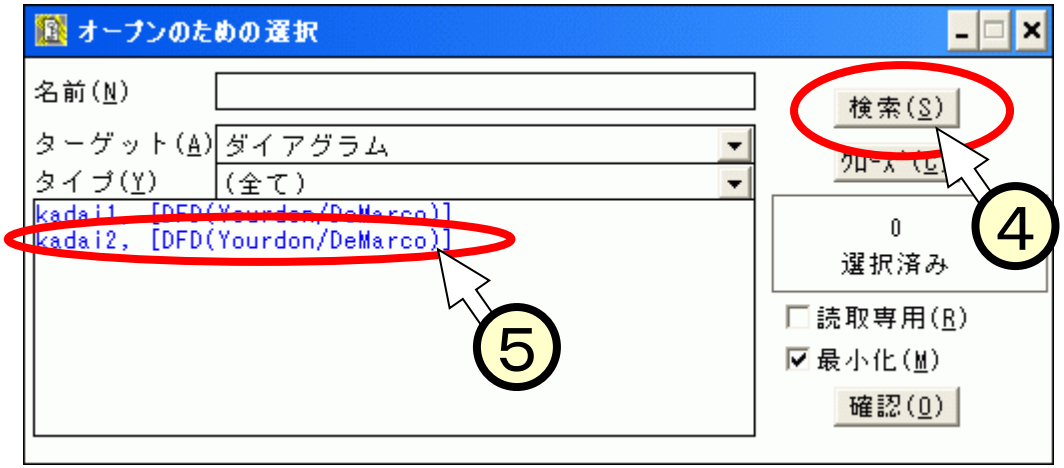
■＜＜オープン＞＞：メニューから、[ダイアグラム]–[オープン]をクリックします（③）。

■「オーブンのための選択」ダイアログにて、[検索]をクリックします（④）。

■現れたファイルの中から、開くファイルをダブルクリック（⑤）します。







（６．１０．１）提出ファイルの作成

■提出ファイルは、次の手順で作成します。

- Popkinの画面上で、スクリーンショットを取る。
- アクセサリからペイントを開き、スクリーンショットを貼り付ける。
- ペイントにて、gif形式でファイルをセーブする。
  - ファイル名は、としてください。
    - 学籍番号_kadai12con.gif
    - 学籍番号_kadai12diag0.gif
- ファイルを井戸のネットワークフォルダにコピーする。

（6．10．2）手順

■POPKINを開いて、作成した図が見えるように、ウインドウを適当な大きさに調整します。

■POPKINがアクティブになっている状態で、[Alt]+[Print Screen]を押下します（①）。

- [Alt]+[Print Screen]
  - [Alt]キーを押しながら[Print Screen]キーを押すことを意味します。
- [Print Screen]
  - キーボードの右上の方にあります）を押下します。

■ペイントをアクティブにし、[編集]−[貼り付け]（もしくは、[Ctrl]+[v]）をクリックします（②）。

■これをGIFとして保存すれば（③）、提出用の画面イメージの画像ファイルが得られます（④）。