

目次

OMT方法論の活用	資料説明
業務システム00化シナリオ	
業務システム00化オブジェクト図	
業務システム00化動的モデル	
業務システム00化状態遷移図1	
業務システム00化状態遷移図2	
業務システム00化機能モデル1	バージョン1
業務システム00化機能モデル1	バージョン2
業務システム00化機能モデル2	バージョン2
業務システム00化機能モデル1	バージョン3
業務システム00化機能モデル2	バージョン3
業務システム00化機能モデル3	バージョン3
業務システム00化機能モデル4	バージョン3
分析から設計への手順	
業務システム00化設計オブジェクトモデル1	
業務システム00化設計オブジェクトモデル2	
業務システム00化設計DFD	
受注登録プログラムイメージ	
業務パーツイメージ	
業務システム00化画面関連イメージ	

この章全体の対象となるシステムについて説明したものが「業務システム〇〇化シナリオ」である。この資料はOMTの記法とは直接関係なく分析の対象領域についてまとめたものである。関係するものも単純化し、顧客、〇〇会社、取引先の3つを四角で表している。〇〇会社内の長丸は業務処理を表しており、四角は資源を表している。出荷・納品の矢印がたくさん集まっているのは分納を表している。これらの図の下には業務の流れを簡単に箇条書している。まづはこのシナリオをある程度理解し対象としているシステムを把握することが必要である。

分析作業

オブジェクト指向の分析においてはオブジェクト図が中心的なダイアグラムとなる。（オブジェクトの抽出については「OMT分析1」を参照）従って多くの場合このオブジェクト図を作成することが最初の作業となる。

「業務システムの〇〇化オブジェクト図」は分析者が対象となるシステム領域をどのように見ているかを表したものであり、読み手も分析者の視点を意識しながらこのダイアグラムを見る必要がある。

例えば受注オブジェクトと受注明細は集約関係で結ばれている。これは「受注明細が存在しない受注はないと分析者は考えている」と見ればよい。また、受注と顧客の間は多対1の関係で結ばれているが、これは、受注から見れば一つの受注には一つの顧客しか対応していない関係であると読めるし、顧客から見ると一つの顧客は複数の受注と関係していると読める。このように一つ一つのオブジェクトの関係をみていくことで全体像が見えてくる。同時に属性として洗い出されているものを見るとオブジェクトが持つ情報が見えてくる。操作に関しては該当のオブジェクトや関係するオブジェクトの関連から、今回のシステムの対象となる機能を洗い出したものとして見ることができる。システムの対象となる世界は複雑なものであるが、オブジェクト図として書き表すことで、今回のシステム化の対象領域を明らかにし、その基本構造を示すことができる。

ある程度のオブジェクトが洗い出されたところで動的モデルを作成する。動的モデルはオブジェクト個々の起こり得る状態とその遷移を表現する状態遷移図や、事象の流れを時系列に追う事象トレース、オブジェクト間の事象の流れを表す事象フローを総称している。

「業務システム〇〇化動的モデル」が表現しているものは業務の大きな流れが、洗い出されたオブジェクトとどう関係しているかを表したものである。まず、シナリオを作成し業務の流れを整理する。その業務の流れとオブジェクトの時系列による関係を事象トレースで表し、その関係をオブジェクト間でまとめたものを事象フローとして表している。事象トレースや事象フローは最終的なドキュメントとして記述されるよりは、作成過程において非常に有用である。つまり、それを作成する過程においてオブジェクトや事象、状態、その関係等が洗い出されるからである。状態遷移図は必要な時にだけ作成するダイアグラムであり、今回対象になっている

システムなどでは、オブジェクトそのものにあまり変化がないので本来ならば作成しなくてもよいものである。ここではあくまでもサンプルとして作成した。状態遷移図は該当オブジェクトの起こり得る状態とその遷移を表している。例えば「業務システムOO化状態遷移図」の受注オブジェクトの状態遷移図は、受注には「受注済」「出荷中」「出荷済」の3つの状態があり、「受注済」状態は「出荷」事象により「出荷中」に遷移し、その遷移は「出荷完了」動作によって行なわれることが解る。また、同じ「出荷」事象でも「出荷中」状態から「出荷済」状態への遷移は「出荷予定全件出荷」（出荷予定のものが全て出荷された状態）の条件が満足する事象のときにしか遷移が行なわれない。この様に状態遷移図はいろいろな表現ができ、使い方によっては非常に有効な分析手法となる。

最後の機能モデルであるがこれはOMT方法論の中での位置付けがはっきりしていない。ここではOMTとは関係無く従来の構造化設計でのDFDの活用と同じ意味合いで使っている。DFDそのものの説明は他に譲として今回作成したモデルについて説明する。「業務システムOO化機能モデル」は同じものを3つ、バージョン1からバージョン3まで徐々に進化したものをそのまま載せている。バージョン1が一番最初に作成し一番粒度も粗い、このDFDの欠点は1枚の図の上に表現しているプロセスが多すぎることである。その欠点を解消し、受注出荷のプロセスを掘り下げて記述したのがバージョン2である。このバージョンの欠点はシステムとシステム外の境界線が曖昧なことである。この欠点を解消しプロセス間のバランスをよくしたのがバージョン3である。DFDの作成を通して実業務の流れを把握するには有用である。

以上のように3つのモデルを使い分析作業を行なっていくが、サンプルで示されているものはあくまでも結果であって、3つのモデルを作りながら、各々のモデルが改善されていくような繰返しの作業が分析作業である。モデルのどれをとっても一度で出来上がるモデルはなく試行錯誤の中で徐々にモデルが整備されていくものだと理解することが大事である。

設計作業

OMT方法論の場合分析作業から設計作業へ移る前にシステム設計というフェーズを用意している。このシステム設計は開発に当たっての資源や方式の基本的な方針を決定するフェーズである。

「分析から設計への手順」の上半分がその分析フェーズとオブジェクト設計フェーズの関連、その間にシステム設計フェーズが存在していることを表し、下半分は今回の開発で利用する「業務パーツ」という概念を説明している。

オブジェクト指向の分析設計においては対象領域の裏に潜む抽象的なデータ構造や共通的な手続きを洗い出すことが重要である。同時に対象領域の手続きや役割とは別に開発者サイドの設計方針の決定とその実現のためのオブジェクト指向技術を利用したメカニズムの導入が必要である。システム設計のフェーズはその開発の基本方針を決定するところであり、オブジェクト設計のフェーズはその実現のためのアルゴリズムを考えるフェーズである。

この業務パーツという考えは今回のサンプルのために考えられた概念であり、OMT方法論とは直接関係の無い話である。この業務パーツが必要であるという決定をシステム設計で行ない、オブジェクト設計においてその実現のためのメカニズムを考えることになる。ここではあくまでもこのサンプル上データベースの一貫性制御とアプリケーションからデータベースを隠ぺいするための枠組みとして業務パーツという概念を導入したことを表している。

「業務システムOO化オブジェクトモデル」は設計段階におけるオブジェクトモデルの一部である。OMT記法は分析から設計まで同じ記法で対応することができる方法論である。しかし、このことは分析で作成したオブジェクトモデルをそのまま設計で拡張していくことが可能である。ということを行っているわけではない。あくまでも同じ記法で扱えるということである。分析は「何を行なうか」whatが重要であり、設計では「どう実現するか」howが重要になる。同じオブジェクトモデルといっても分析と設計では視点が異なるため表現している意味が違ってくる。読み手もそのことを理解した上で読む必要がある。

設計段階でのDFDの役割はあまりないと思われるが「業務システムOO化設計DFD」ではマンマシンインターフェース・整合正チェック・受注登録等の特徴的な処理について代表例として作成したものである。業務系のシステムの場合プログラムレベルでの処理はさほど難しくないので代表的な処理についてDFDを作成しプロセス分割しておくことで用が足りる。OMT方法論上ではこのぐらいにしかDFDを使う必要性はないと考えられる。

オブジェクト指向言語での開発の利点として既存のクラスライブラリーを利用し効率的にプログラム開発ができることがあげられる。設計の段階においても設計者の頭の中では既存のクラスをどのように使うかを考えているものである。「受注登録プログラムイメージ」はそういう設計者の考えていることをイメージ化した資料である。いわゆるコレクションといわれるテーブル処理のクラスがライブラリーに既

に用意されている。そのテーブル処理を使って出荷、受注などの画面コントロールで共通に使えるテーブルを用意することを説明している。同時に受注、出荷のデータベースへのアクセスも共通化できるのではないかと考えていることを表している。

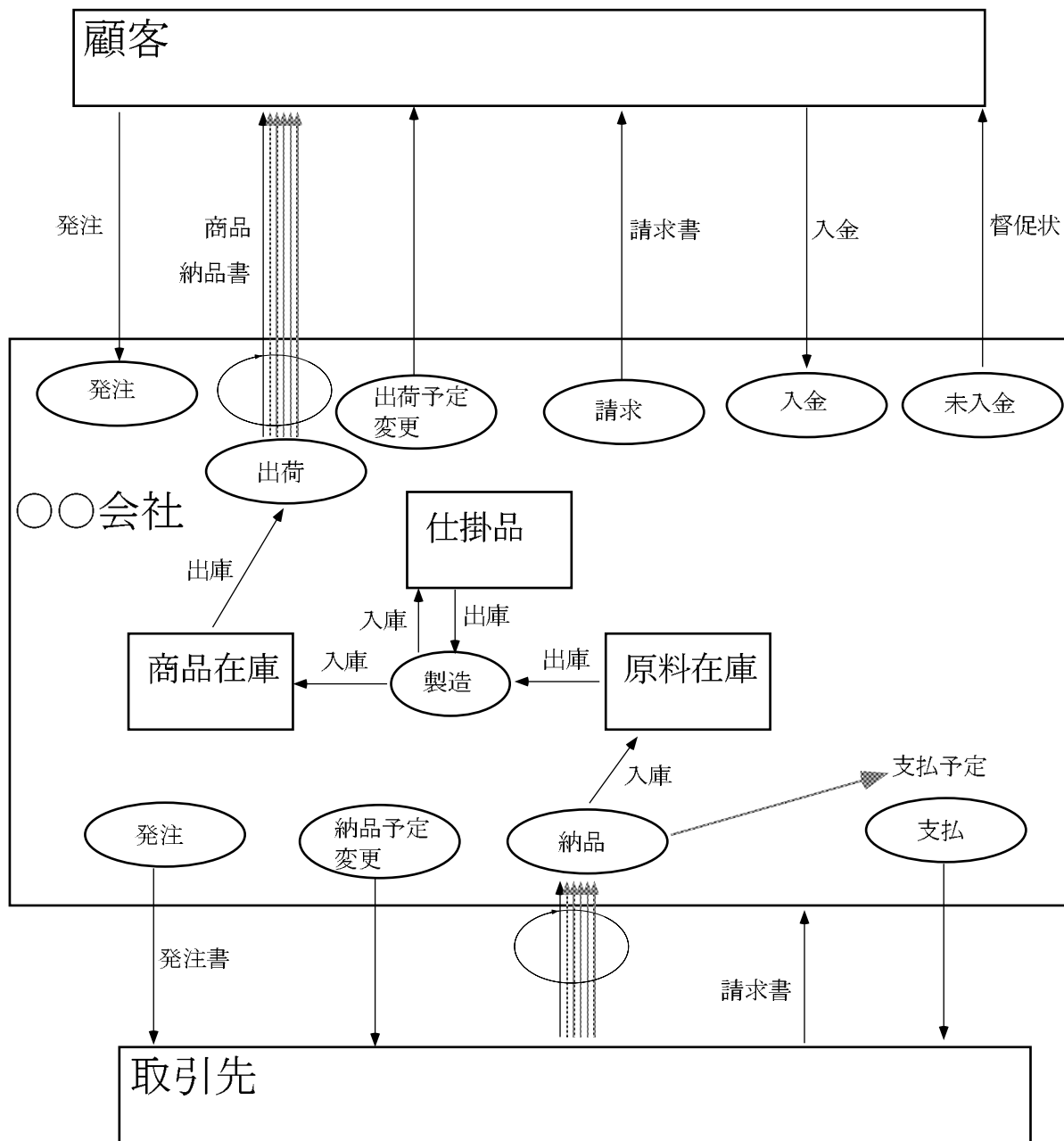
「分析から設計への手順」で説明されている業務パーツはシステム設計のフェーズで決定された方針を説明したものといえるが、「業務パーツイメージ」ではデータベースの一貫性制御（意味的な）の具体的なイメージを表したものである。上半分が一貫性制御のためのルーチン群で対象となるデータベース（下段）に対応する位置に書いてある。下段はデータベースの項目とそのリンク関係が書かれている。ここで重要なことはアプリケーションから見ると「出荷登録」「請求金額計算」「入金登録」などのように業務的な手続きが結果としてデータベースのアクセスにつながっており、アプリケーションから個々のデータベースについての知識や手続きを排除したことである。

大事なことはデータ構造（データベースを含めた大きな意味の）にアクセスする手続きはシステムとして決められたものを使用し、直接データベースを扱うようなことはしないようにしたことである。また、その実現に当たってはデータベース毎に単機能の手続きを用意しそれらを組み合わせることで実際のデータベースの更新を行なっていく構造にしたことである。この時「出荷登録」「請求金額合計計算」

「入金登録」などはこのシステムにおける部品として認識することができる。そしてこれらのパーツを実現するメカニズムをオブジェクト設計で検討する必要がある。

システムは必ず何らかの外部とのインターフェースを持っている。そして、設計段階においてはそのインターフェースをどう実現するかが重要なテーマになってくる。当然それはオブジェクト図にも影響を与える。その関係を図示したのが「業務〇〇化画面関連イメージ」である。例えば顧客一覧の画面は顧客オブジェクトに対応し、顧客明細画面は顧客明細オブジェクトに対応する。オブジェクト図上は顧客が顧客明細を持つ構造になっている。これはリレーショナルデータベースで考えられているシステムのときには顧客オブジェクトがどういう位置付けになるかが理解しにくいところではあるが、インターフェースから考えるとうまく対応関係の取れたオブジェクトに見えてくる。

設計作業はターゲットになるマシンや開発環境によって設計のやり方や手順が変わるが、OMTの方法論においても各モデルをその状況に応じてうまく使い分けて行く必要がある。また、ここに挙げたサンプル（OMT方法論の中で紹介された記述方を含め）だけでシステム開発の全行程を満足するするようなものを提示できていくわけではないので、今まで使用してきたいろいろな記述法を補完的に使っていく必要もある。



受注出荷

1. 顧客から受注がある。
2. 受注に基づき出荷を行う。
3. 受注時点で出荷予定も顧客と決める。
4. 月末に出荷した分をまとめて請求を行う。
5. 請求に対する入金を確認する。
6. 未入金のものに督促状を出す。

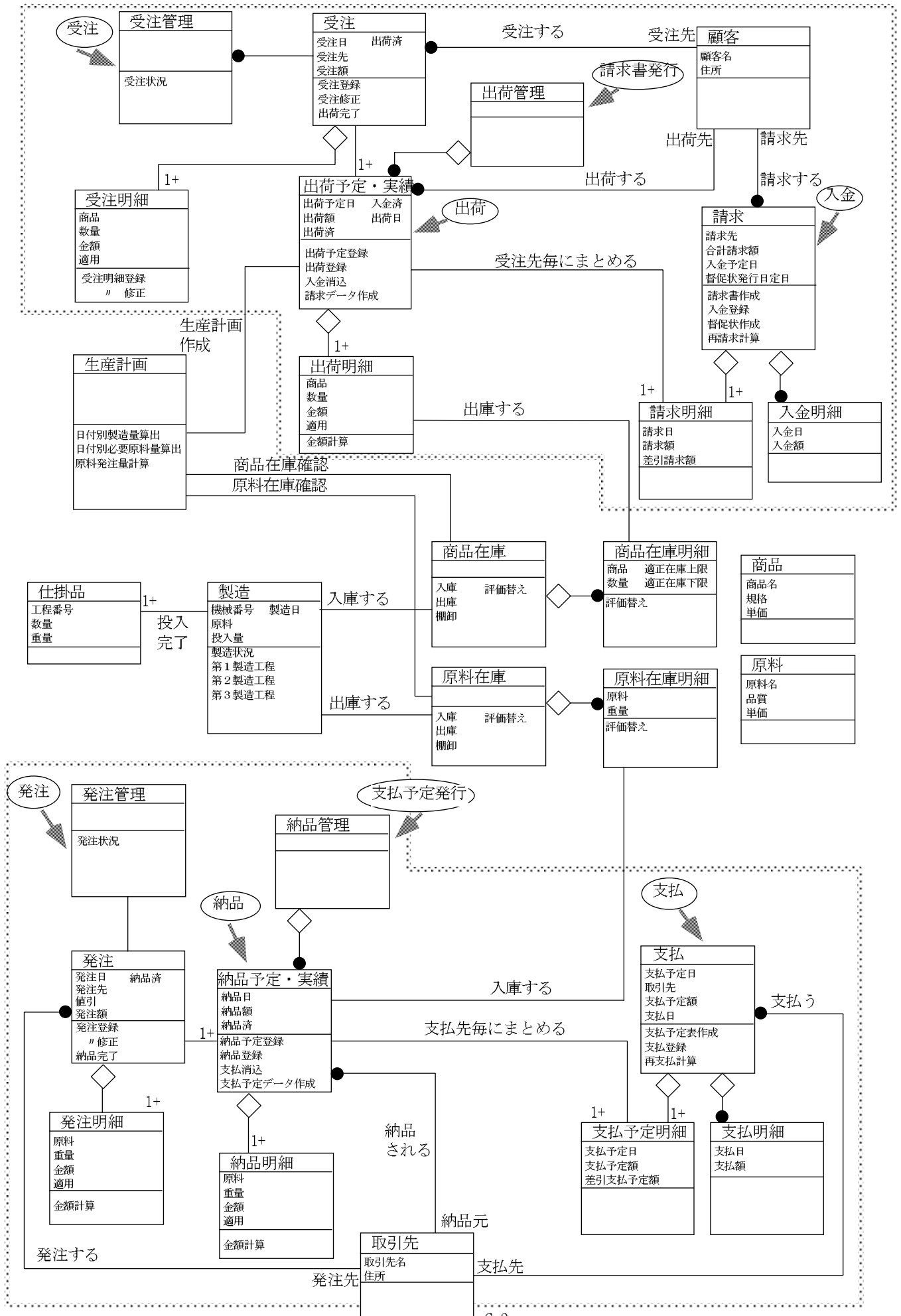
他社製品の販売(OEM)は考慮しない。

製造

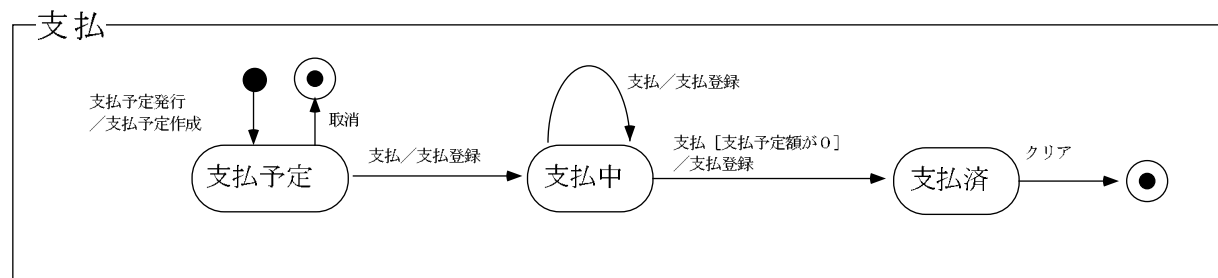
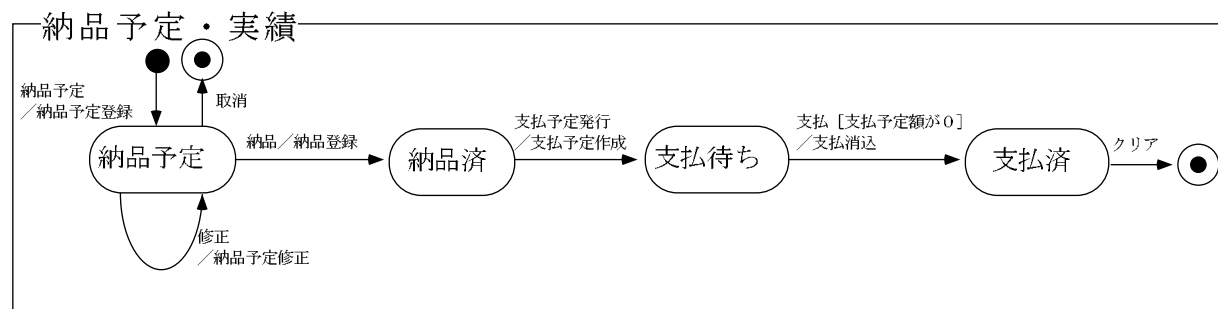
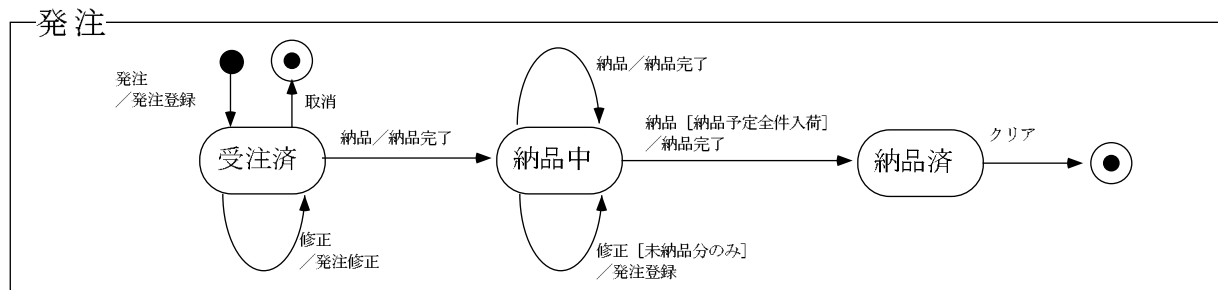
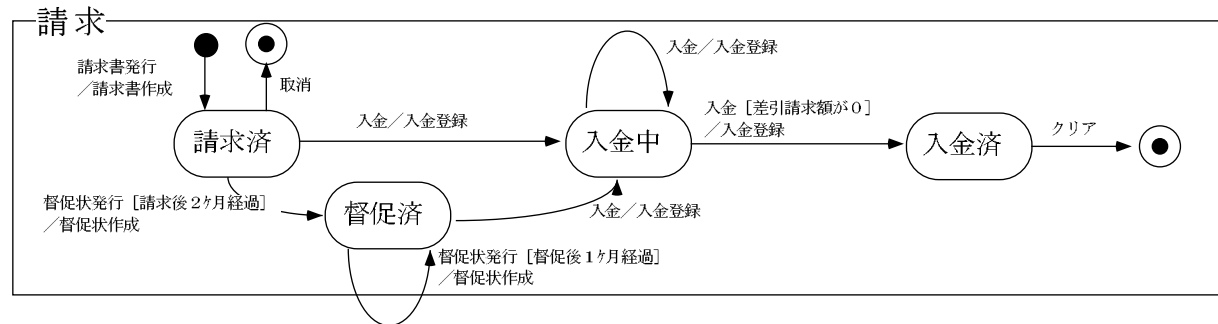
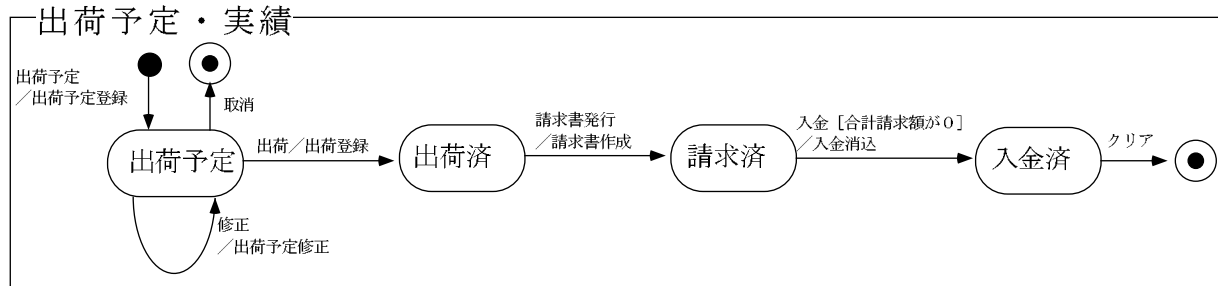
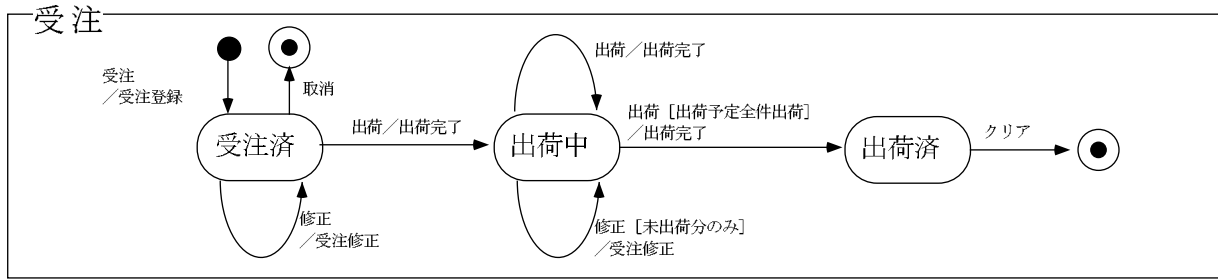
1. 生産計画は出荷状況と在庫状況より作成する。
2. 生産計画にもとづき製造を行う。
3. 製造は3回の工程をへて製品となる。

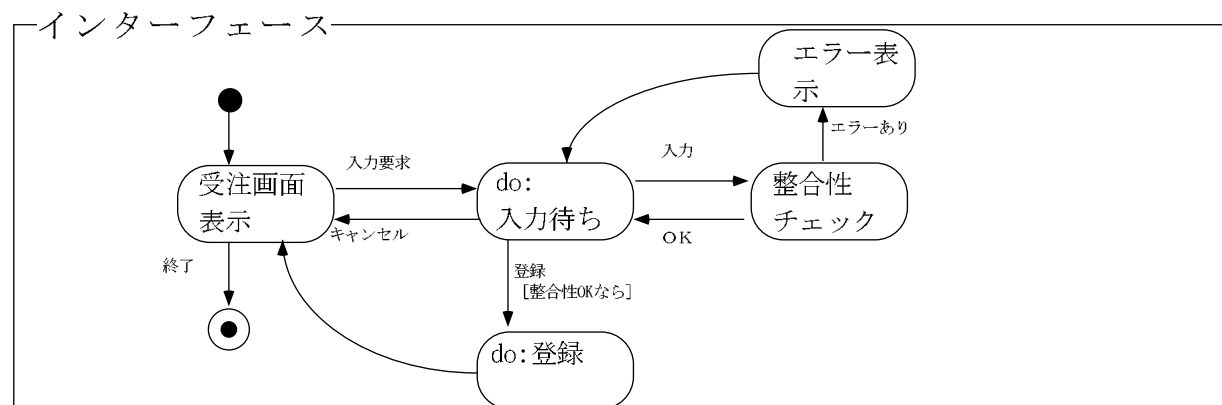
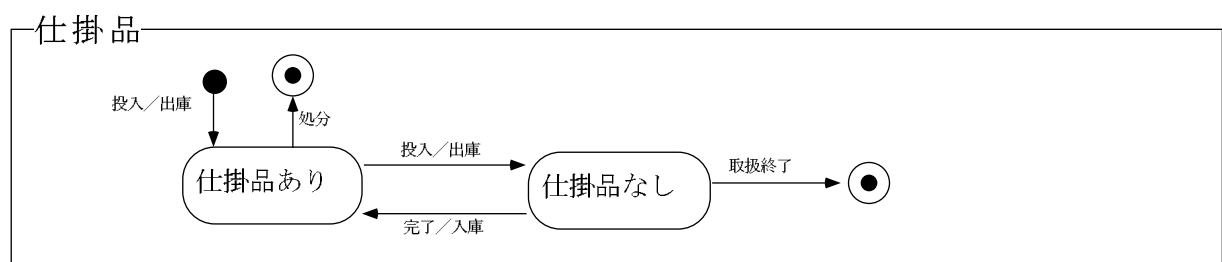
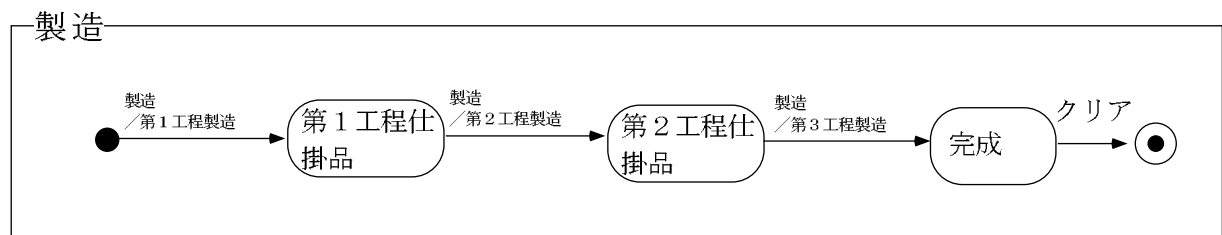
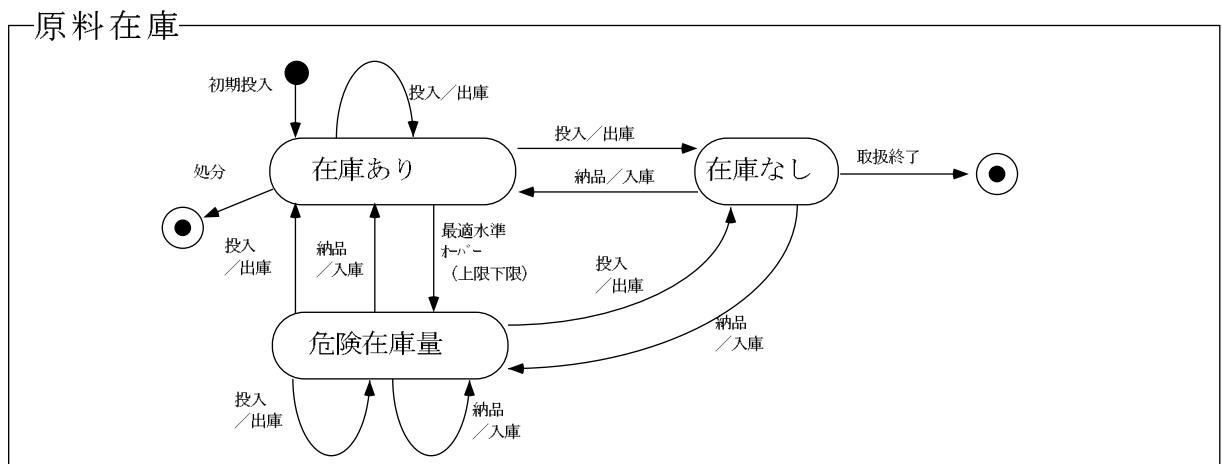
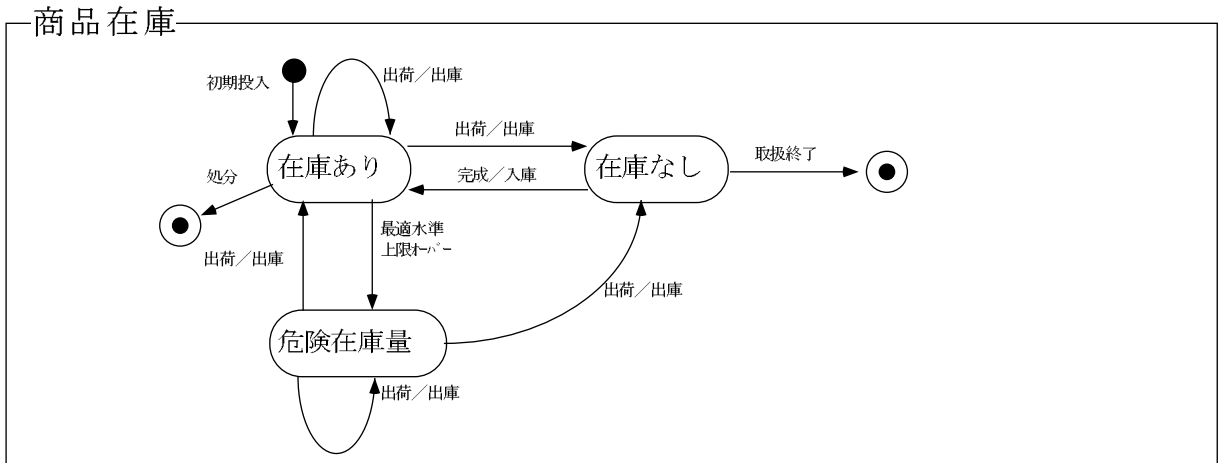
発注支払

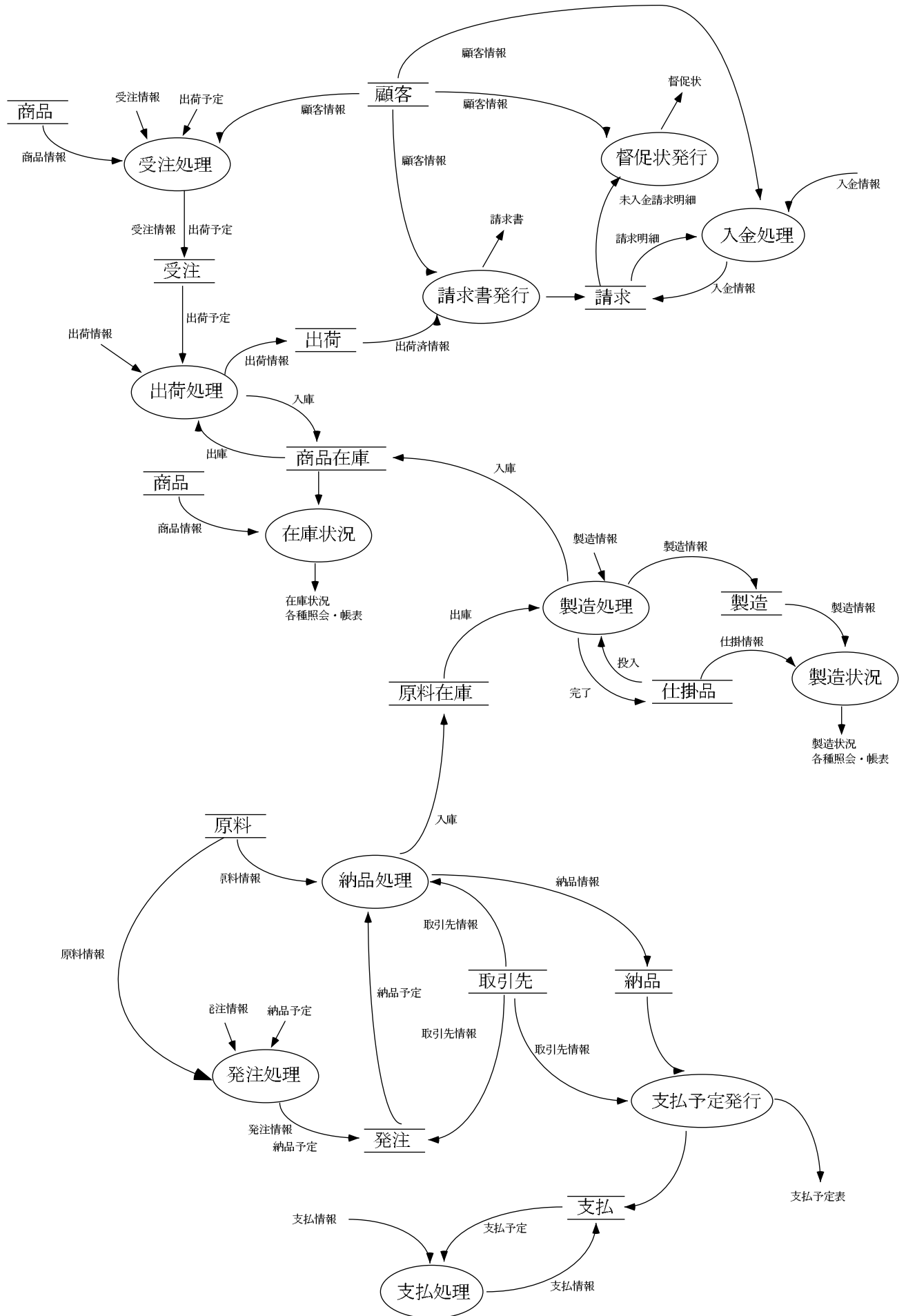
1. 在庫と受注状況から発注計画を作成し原料発注を行う。
2. 納品された原料を在庫する。
3. 発注時点で納品予定を取引先と決める。
4. 月末に納品された分をまとめて支払予定を作成。
5. 支払予定に対する支払を行う。

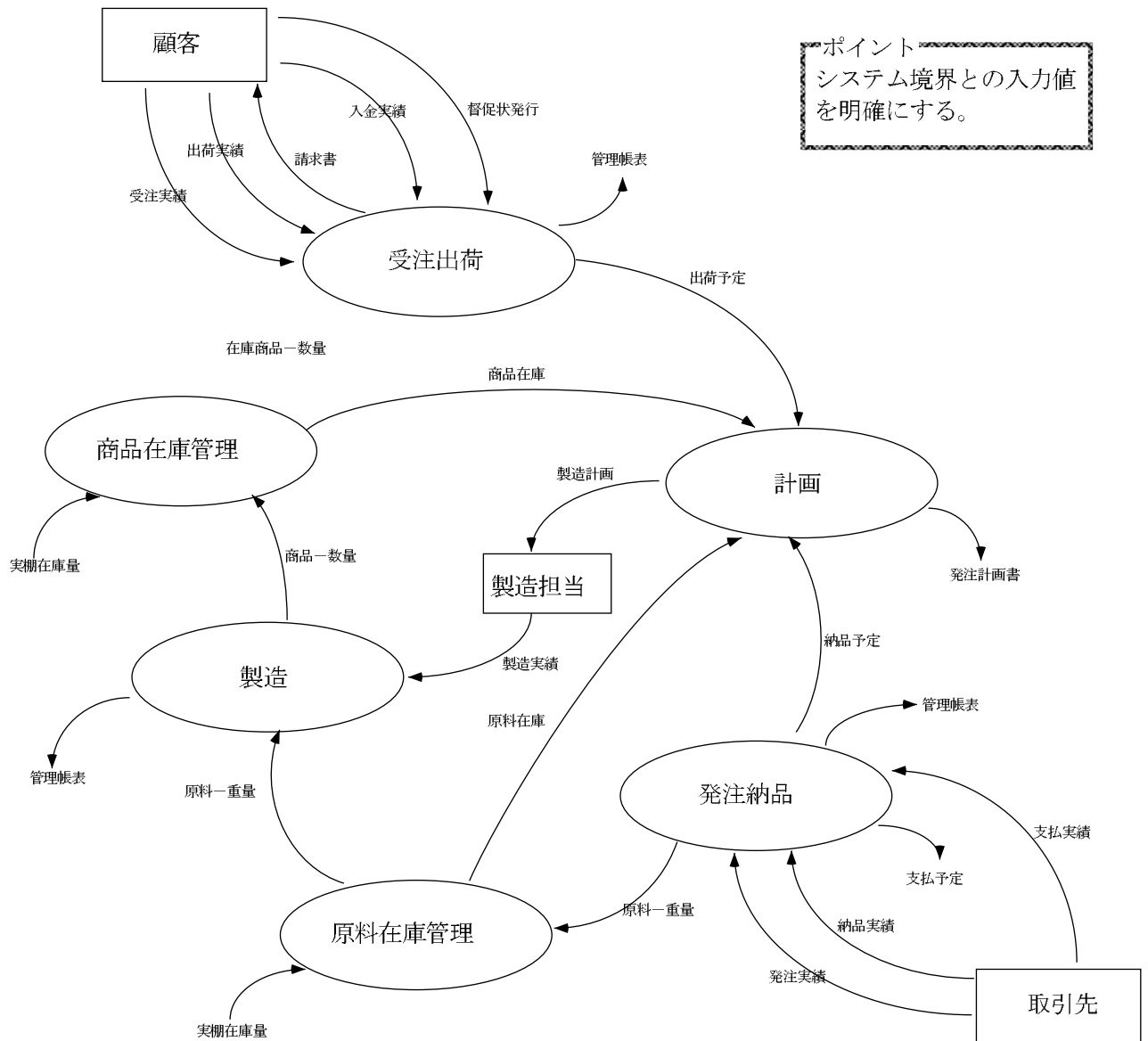


事象／動作

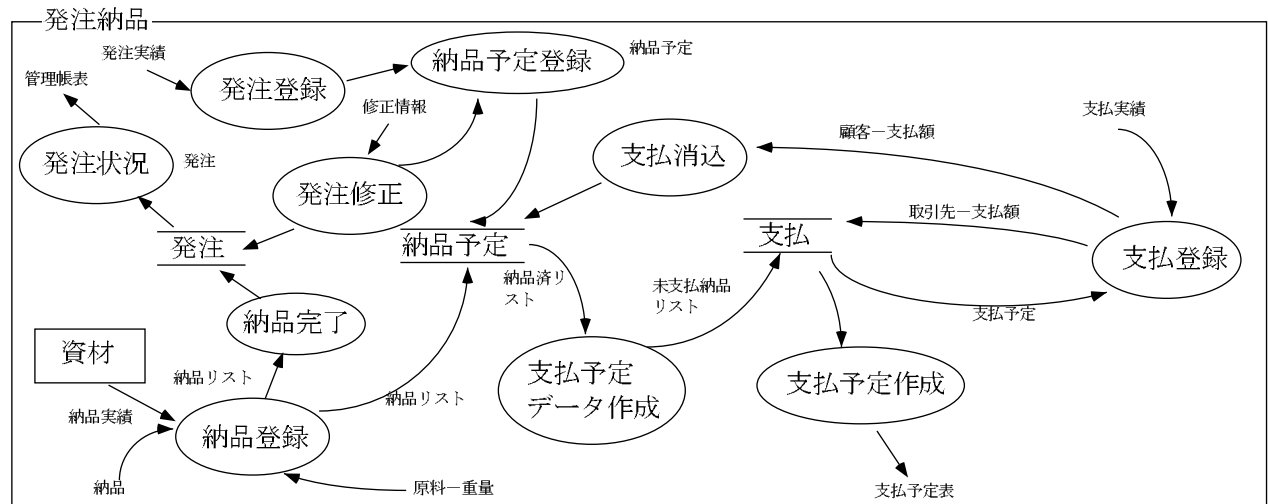
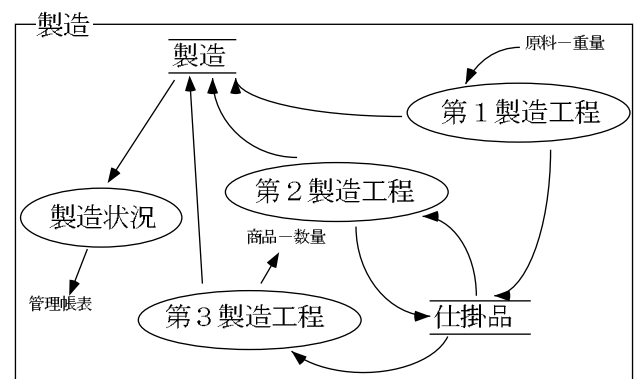
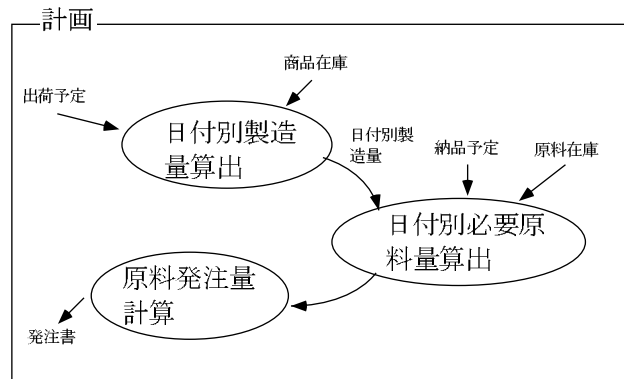
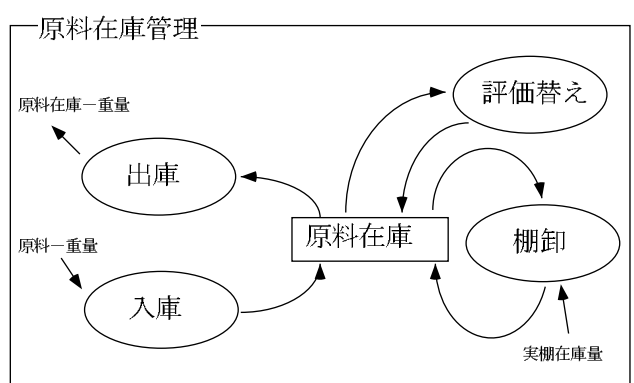
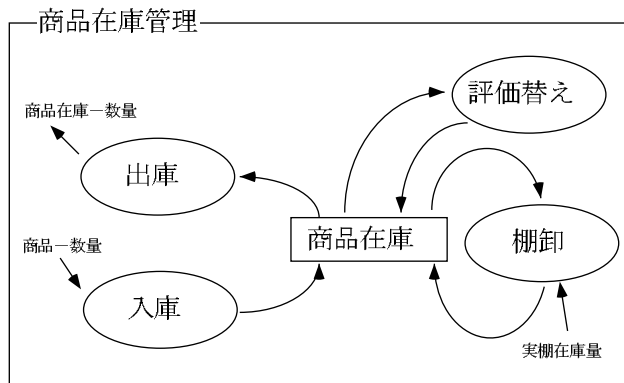
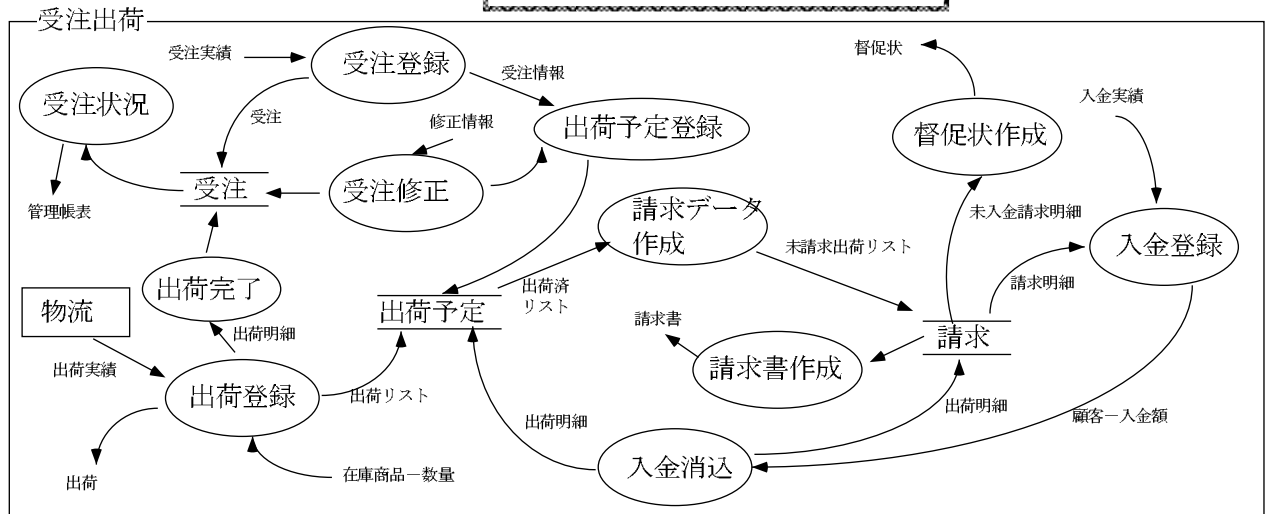


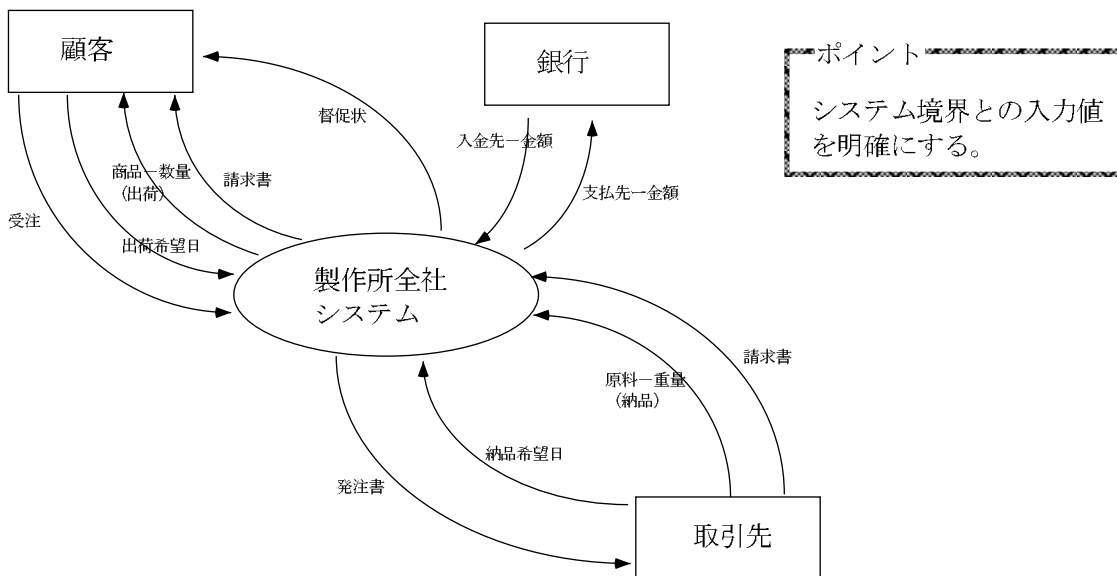




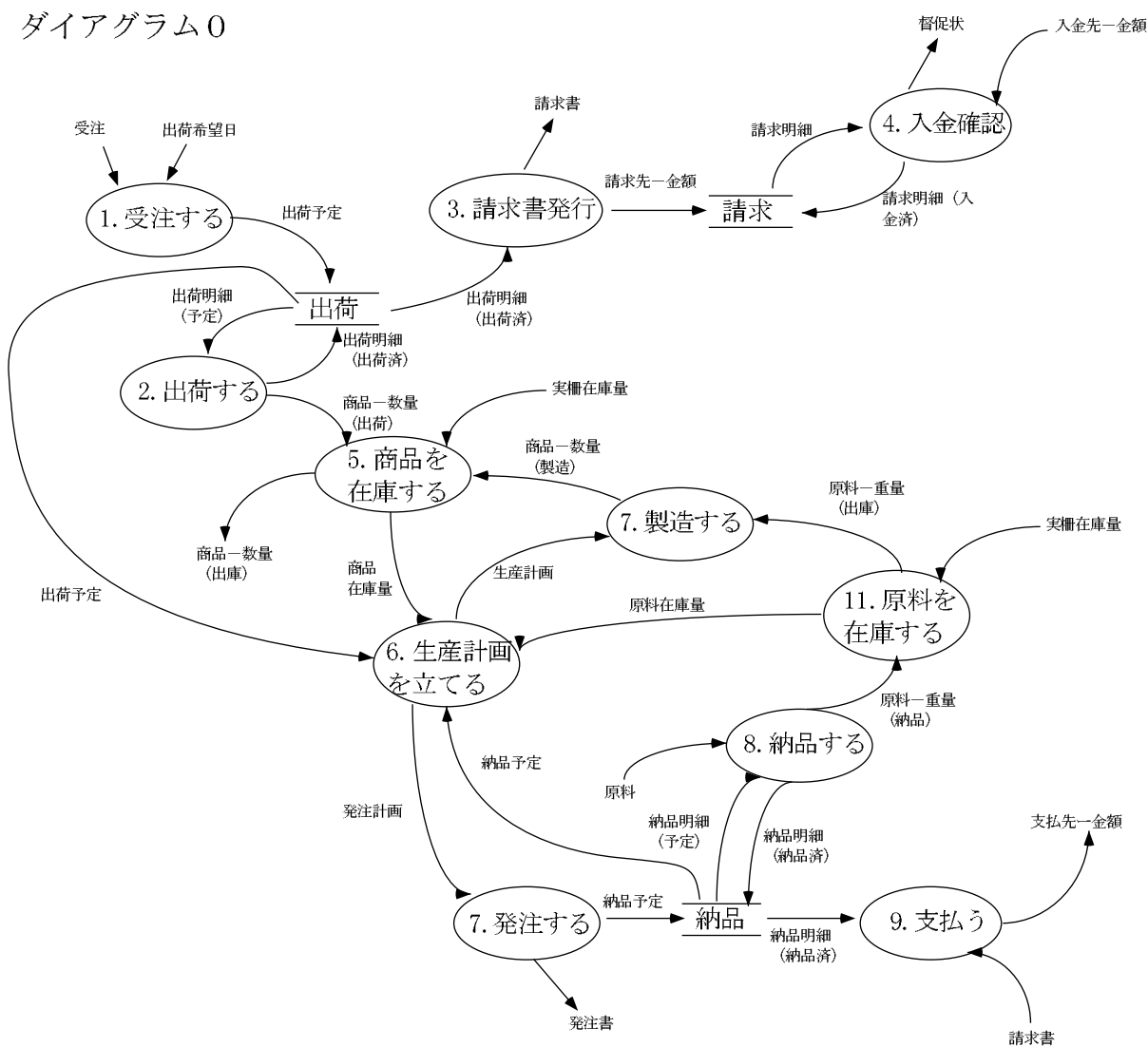


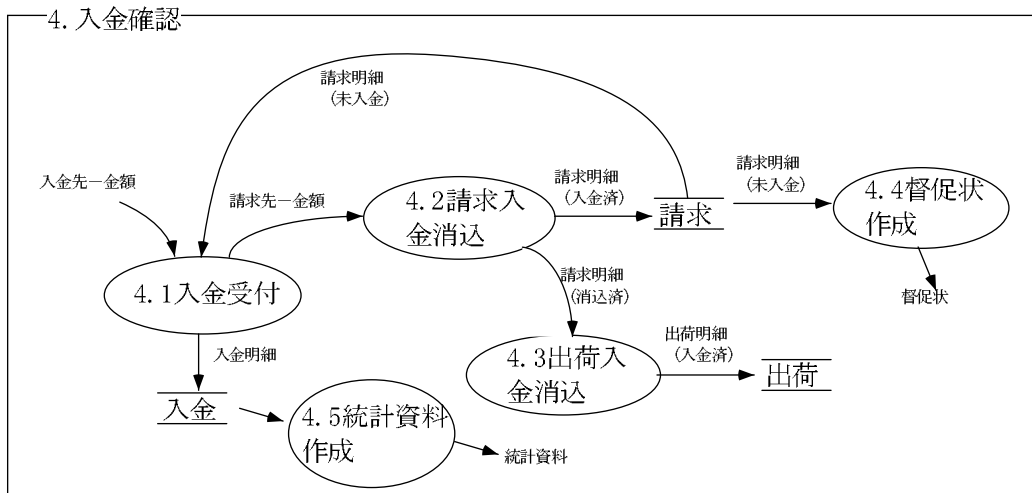
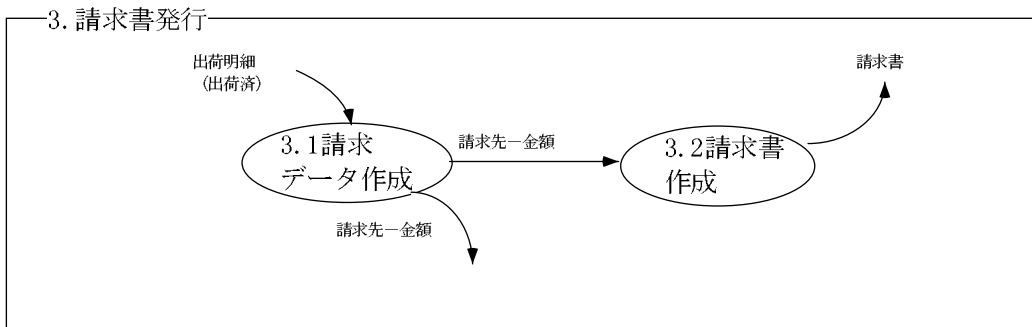
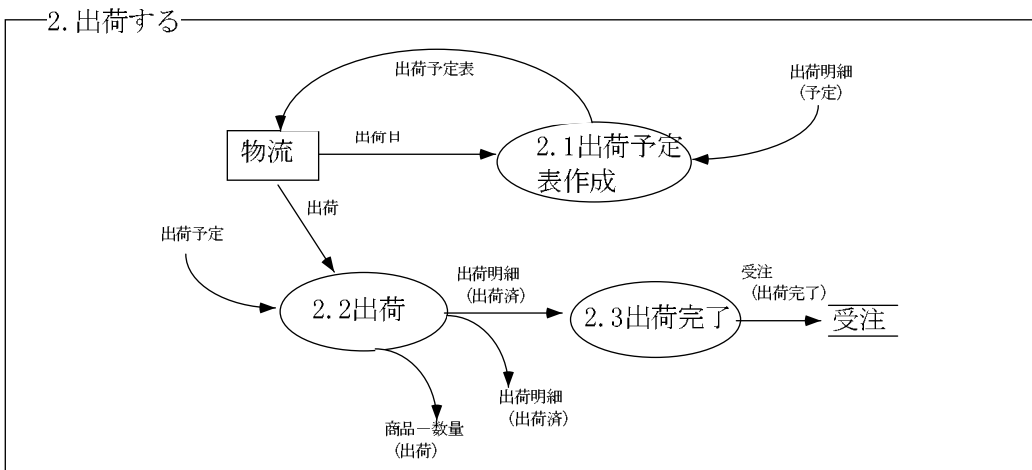
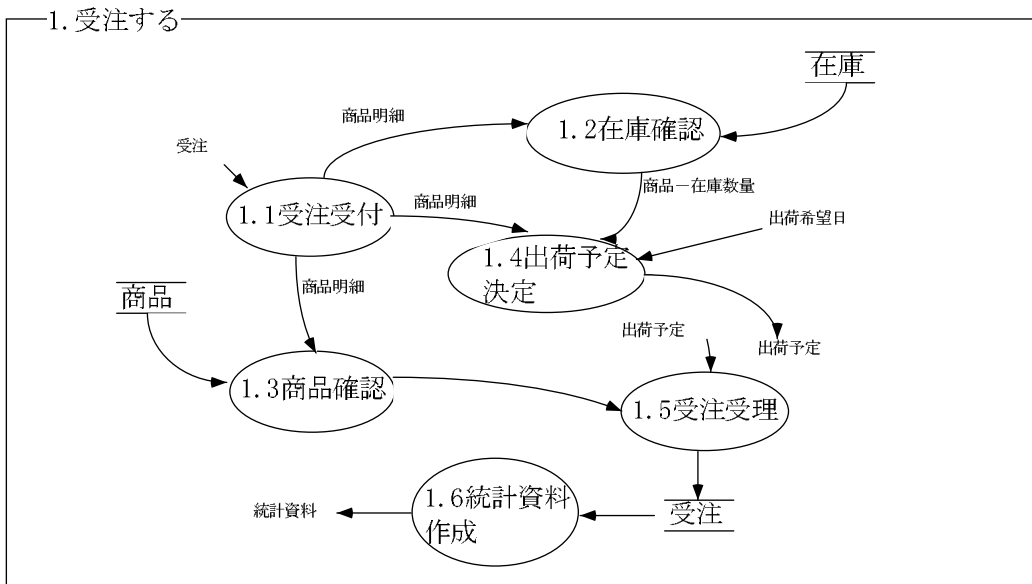
- ・機能(プロセス)の依存関係を把握
- ・プロセス・データストア・アクター-の関係性を把握
- ・全体としての計算経路の把握

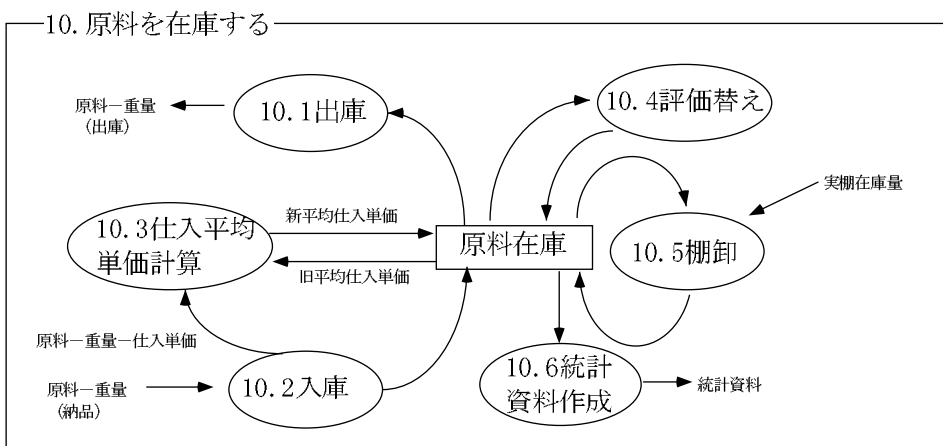
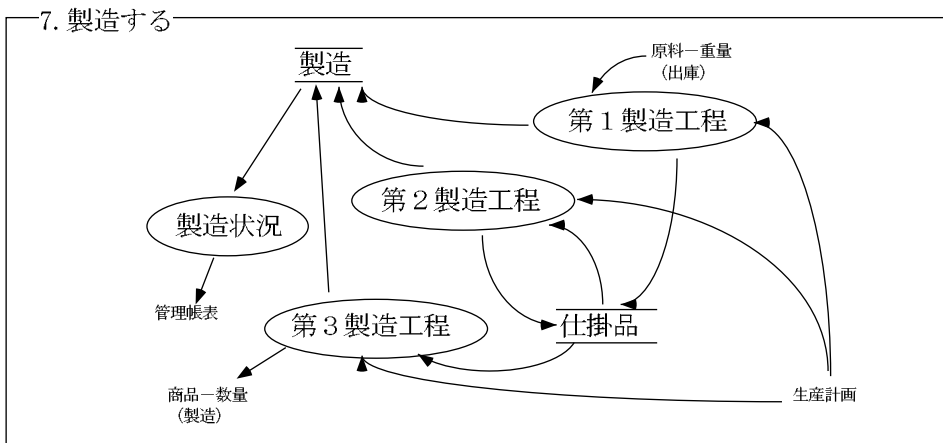
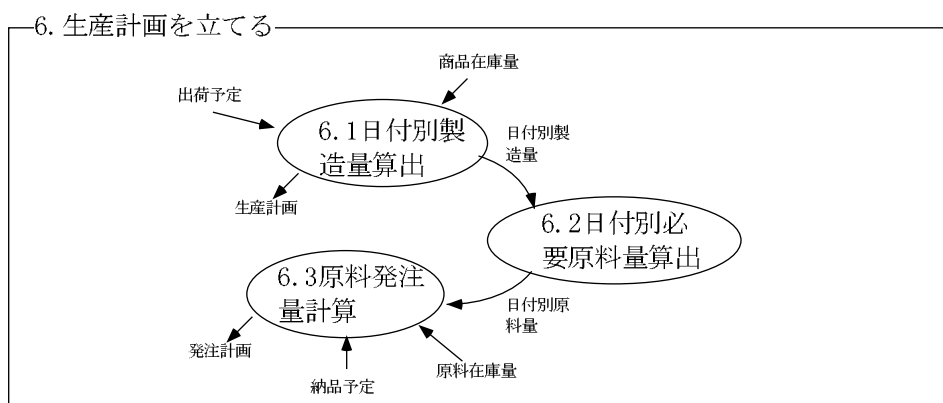
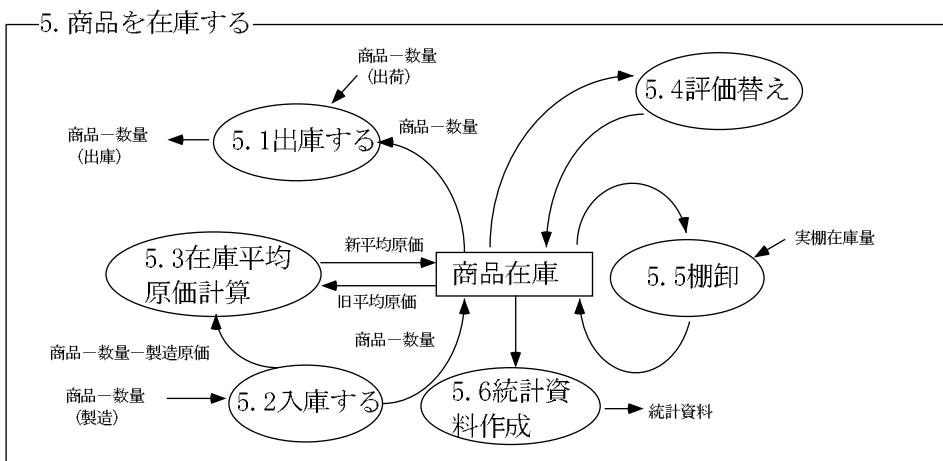


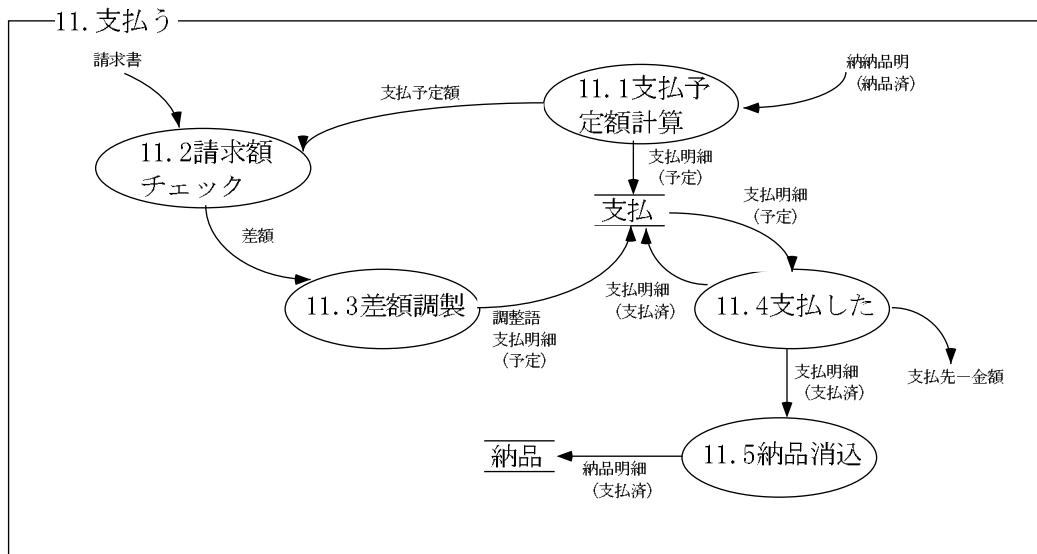
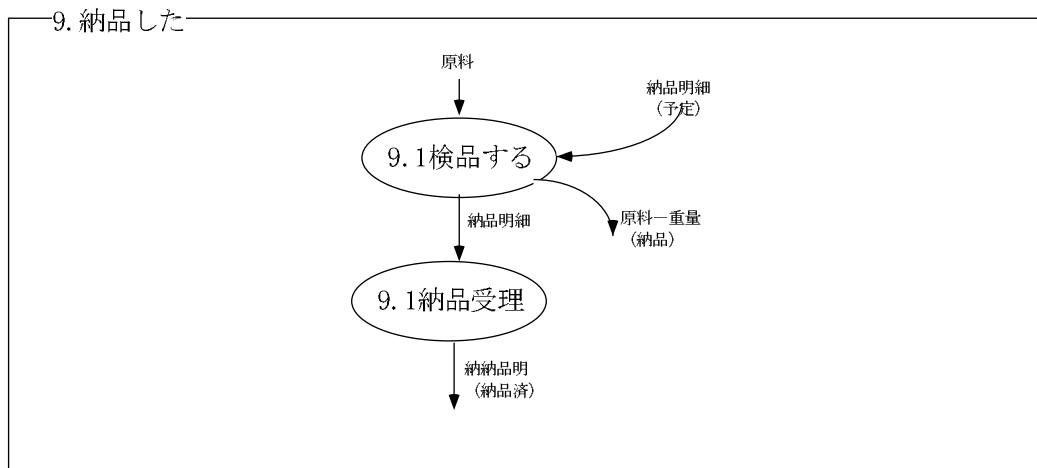
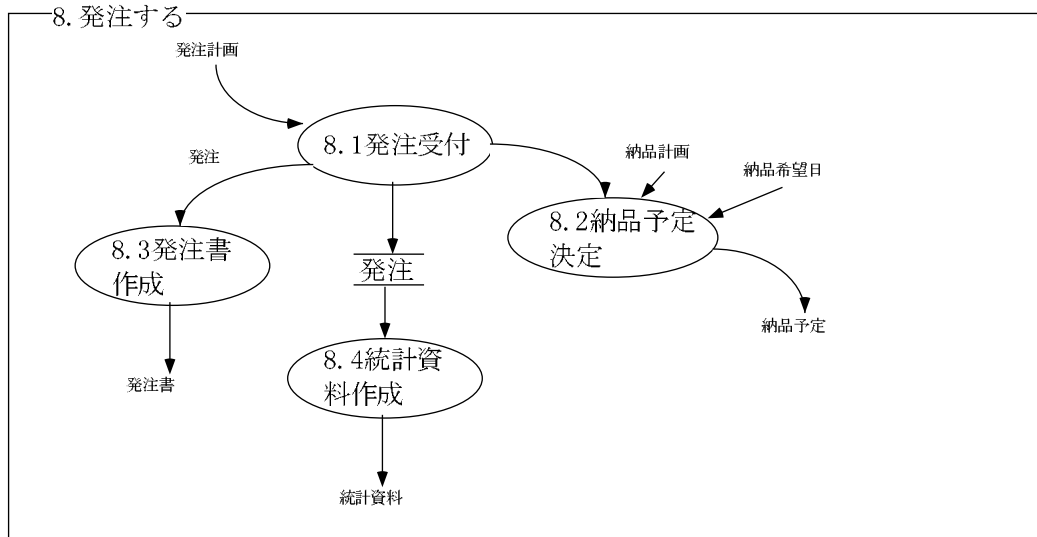


ダイアグラム0

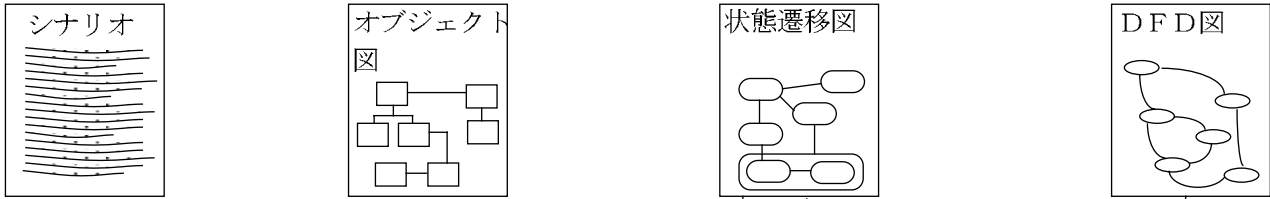








分析



システム設計

サブシステム分割
DB選定
開発環境

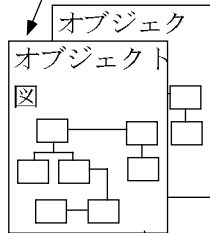
オブジェクト設計

DBオブジェクトの選定
ワークオブジェクトの抽出

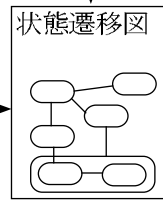
属性、操作の確定
関連オブジェクトの属性化

画面
帳表

画面、帳表のコントロール用オブジェクトを追加



新しいオブジェクトの状態遷移



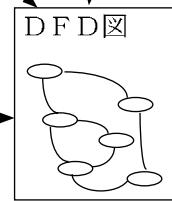
新しいオブジェクトでのデータフロー

状態遷移図とDFDは必要最小限度のものだけを作成する。

動作、活動の機能分解
一貫性制御関係を機能分割

各オブジェクトの動作、活動から機能を満足するフロー作成

画面
帳表



画面、帳表のコントロール用データフロー作成

業務パーツの作成

業務パーツとは

業務上のルール、手続き、状態を一まとりにした処理、ただし、業務パーツというのはその中でも、意味的にこれ以上分割できない単位（例：対象オブジェクトが一つの場合）をいう。つまり、ルールや手続きは当然変わり得るが、その中でも一つの機能として変わる可能性の少ない部分を、機能単位としたモジュールを業務パーツという。

例) 督促状は請求書発行後2ヶ月立っても入金のないもの

- ・請求日から現在までの日数
- ・請求データがある条件で抽出
- ・督促状の作成

出荷データは対応する入金が行われてから3ヶ月後に消す。

- ・入金済確認
- ・入金日から現在までの日数
- ・出荷データのクリア

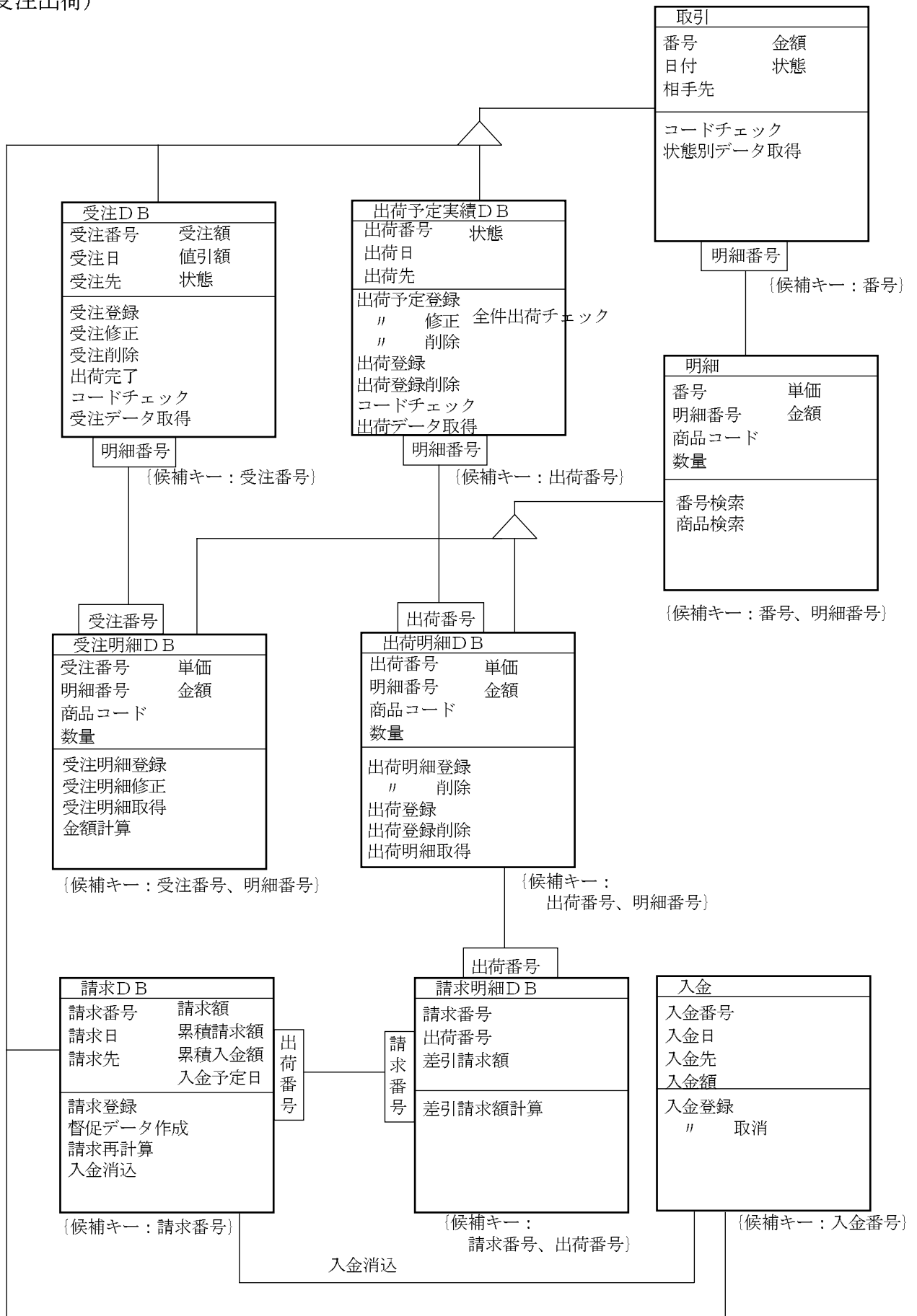
基本パーツ

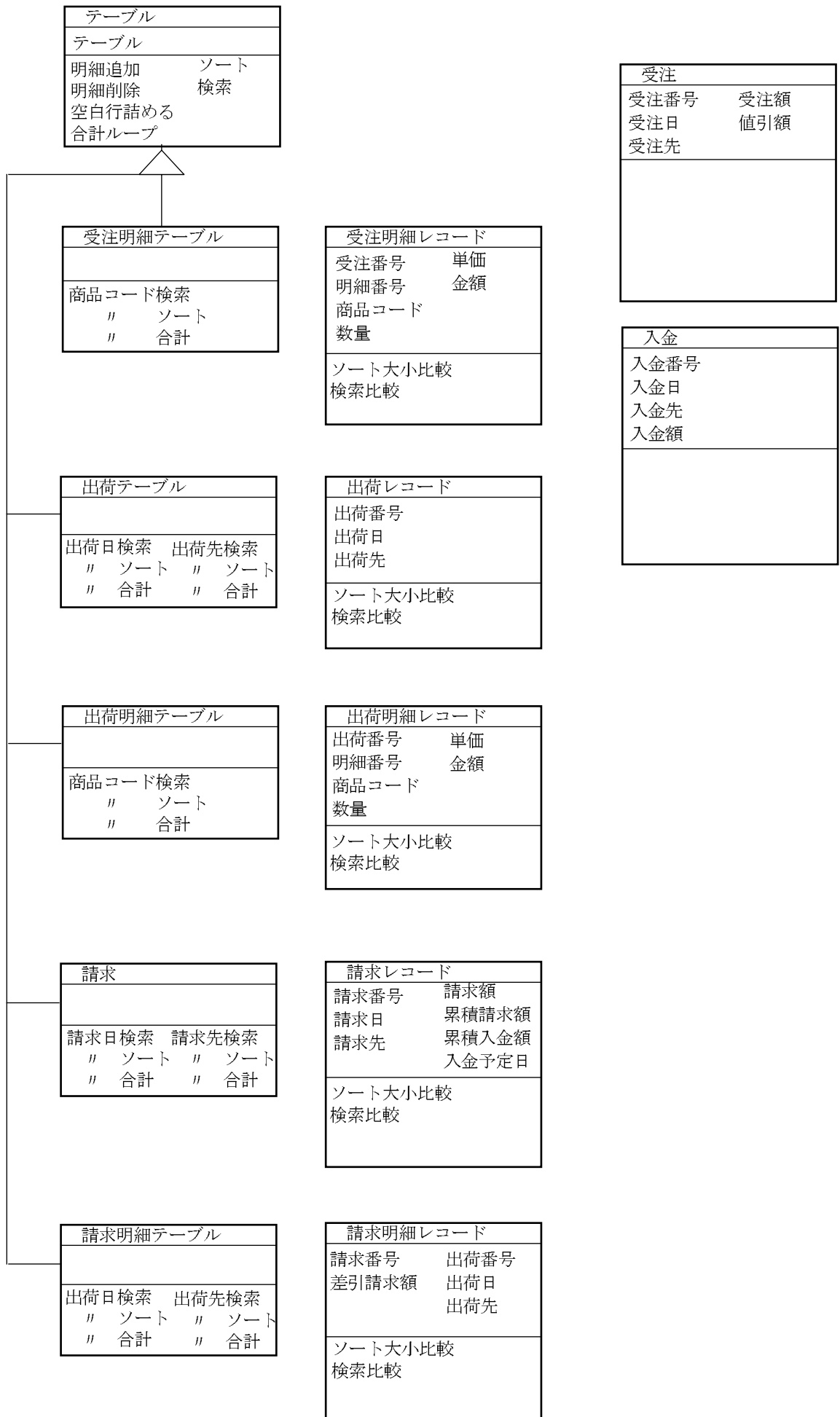
市販のクラスライブラリー
Collection date Debager

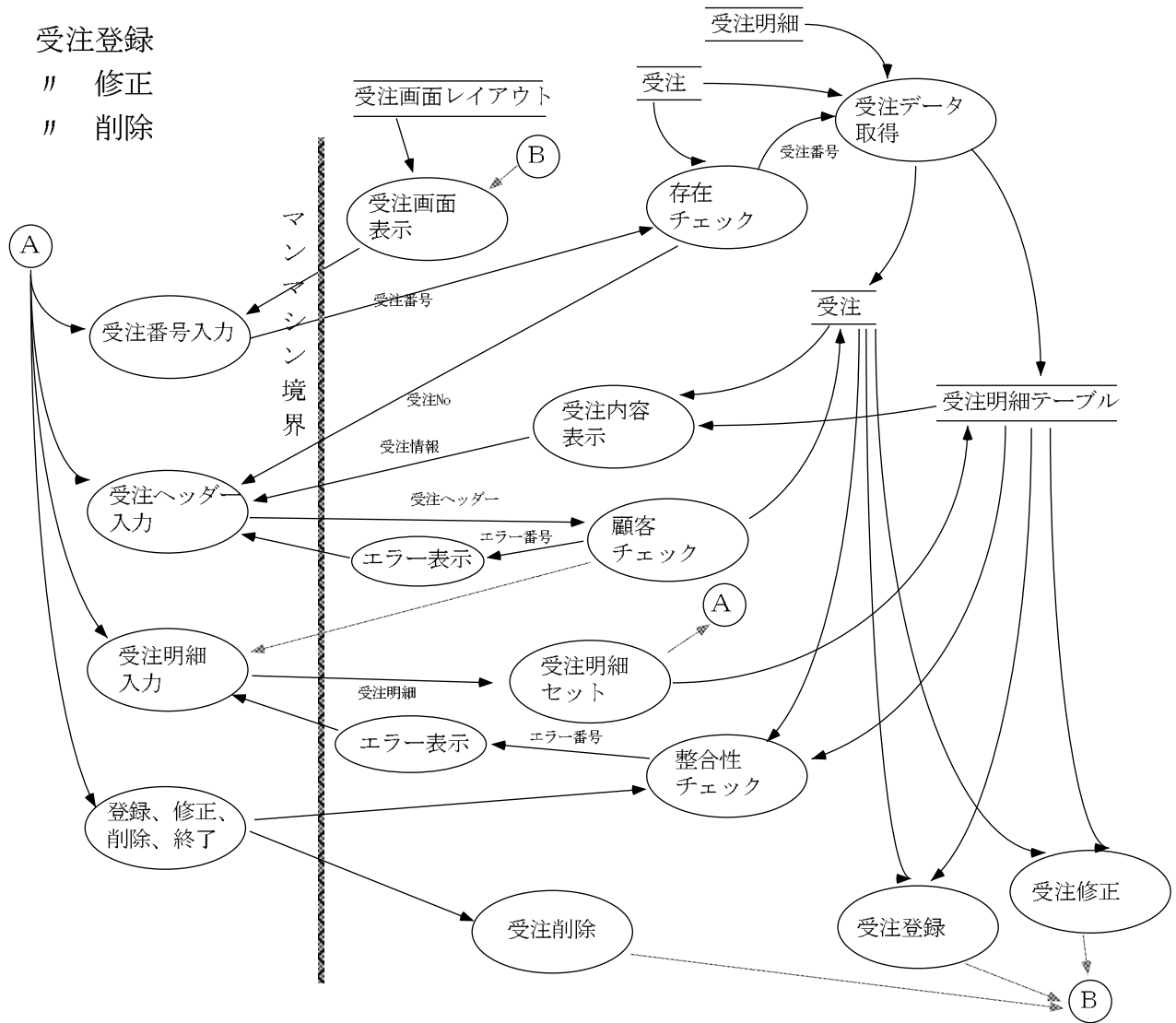
カレンダー
全社共通の処理

基本パーツを元に業務パーツを作る。ただし、基本パーツの使えないものは業務パーツを最初から作る。

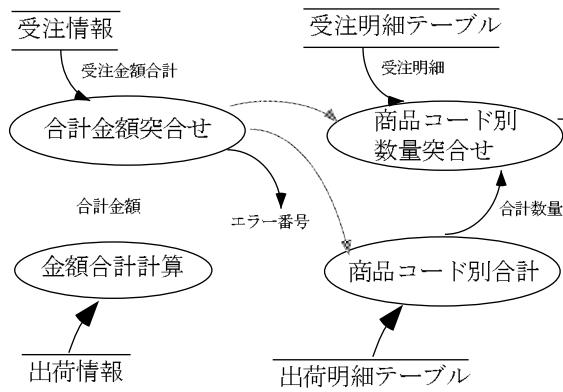
(受注出荷)



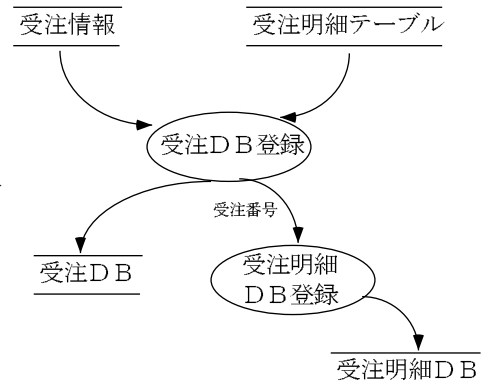


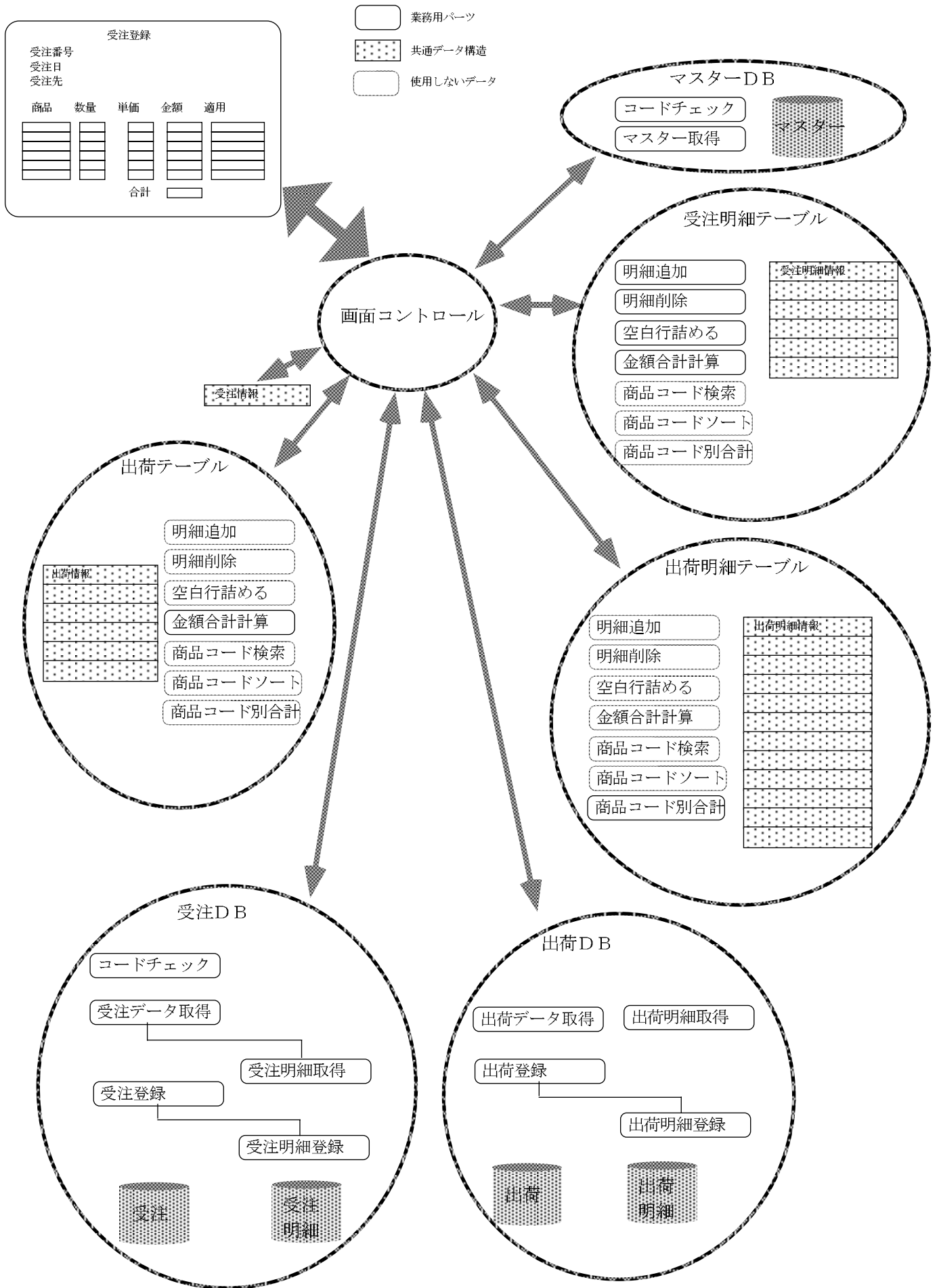


整合性チェック

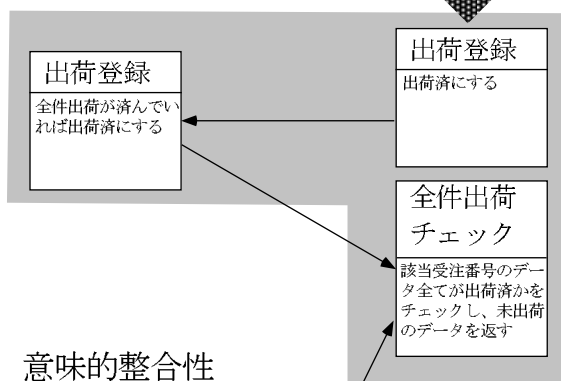


受注登録

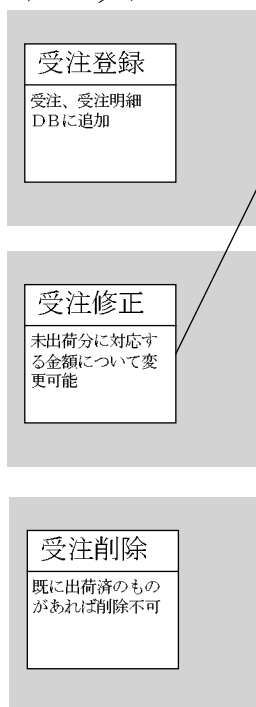




一貫性制御ルーチン



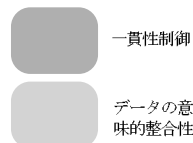
意味的整合性ルーチン



出荷事象に対する一貫性制御



各モジュールへの入口



補足

出荷テーブル

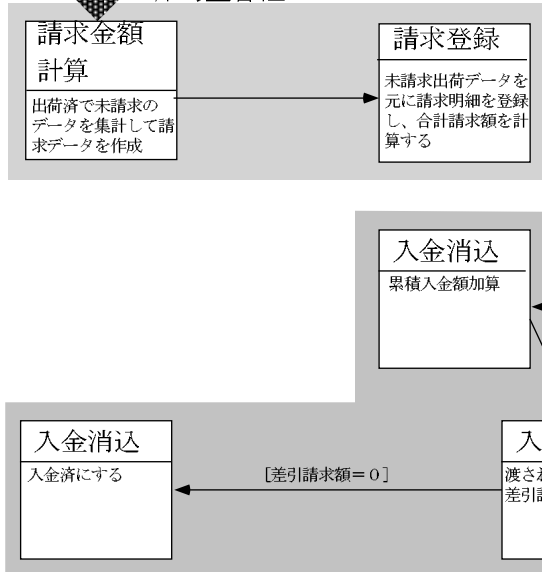
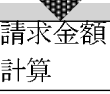
出荷データはいろいろなどからアクセスされる。それらが、リアル性を要求されるような場合であれば、DBアクセス低減のために、出荷データをテーブルに持ちそのテーブルに対して操作を行うようにすることもできる。テーブルに対する操作は事前に全てルーチンとして提供する。

ただし、以下の問題点が残る

- ・排他制御をどうとるか？
- ・行きすぎると返って難しくしてしまう。

操作ルーチン

請求事象に対する意味的整合性



入金事象に対する一貫性制御



データベース



代表的フィールド

- | | | | | | | |
|------|------|-------|------|-------|--------|-----|
| 受注番号 | 受注番号 | 出荷番号 | 出荷番号 | 請求番号 | 請求番号 | 請求先 |
| 受注日 | 商品 | 出荷予定日 | 商品 | 請求先 | 出荷番号 | 入金日 |
| 受注先 | 数量 | 出荷額 | 数量 | 合計請求額 | 出荷番号 | 入金額 |
| 受注額 | 金額 | 出荷済 | 金額 | 入金予定日 | 差引請求残高 | |
| 出荷済 | 適用 | 請求済 | 適用 | 累積入金額 | | |
| | | 入金済 | | | | |
| | | 受注番号 | | | | |

データリンクイメージ

上記データベースのリンク関係を表している

