情 報 シ ス テ ム 56－3
（1995．10．17）

3層クライアント/サーバアーキテクチャに適応したシステム設計手法の提案

高田 信一　畑 恵介　山本 修一郎

NTT ソフトウェア研究所

システムの機能の変更に柔軟に対応できるアプリケーションアーキテクチャとして3層クライアント/サーバアーキテクチャが注目されているが、その設計工程において従来の業務アプリケーションにみられる単一的な機能構成を適切に各層に配置しなければならない。

本稿では、要求仕様を入力として、手順に従って設計を推めることにより、各層に配置する機能を抽出する手法を提案する。

System Design Method based on 3-Tiered Client/Server Architecture

Shin-ichi Takata, Keisuke Hata, Shuichiro Yamamoto

NTT Software Labs.

As a flexible application architecture which allows a system to modify its functions, 3-tiered client/server architecture is noticed widely. However, it requires the adequate design which distributes functions into each tier from monolithic functional constructions which usual applications had.

This method provides design steps how to develop functional interfaces between 3 tiers from requirement specifications.

1　はじめに

ハードウェアとネットワークの飛躍的な性能向上を背景として、クライアント/サーバアーキテクチャによる情報処理システムが増えている。クライアント/サーバシステムのメリットとして

- GUI ベースのユーザオペレーションの提供
- アプリケーション開発期間の短縮
- ユーザ自身による要求の実現(ユーザプログラミング)

などが一般的に謳われており、現在ではそれをサポートするためのツールや環境が多数存在する。そのため、メインフレームによる伝統的な集中型情報システムの適応領域は次第に狭くなっている。

しかし、依然としてクライアント/サーバシステムには次のような問題がある。

業務拡張性 ビジネスフローの変更などをともなう業務仕様の拡張に対応したアプリケーションの変更が困難である。

システム拡張性 業務量の増加にともなうデータの増加や処理能力の増強が困難である。

接続性 他システムとの情報流通が困難である。

これらの問題の原因として、

- このようなクライアント/サーバシステムの大部分がアプリケーションプログラムの画面制御と業務機能とSQL文によるリモートデータベースアクセスが渾然一体となった構成をとっている(図 1)ため、わずかな仕様変更が全体に与える影響が大きい。また、プログラムの構成変更が必要なハード増設が必要となった時に、プログラムの全体に見直しが必要なことがある。

- システムを構成するツールがそのシステムの環境(DBMS, ハードウェア, OS など)に強く依存した仕様となっており、他の環境との親和性はよくない。

などが考えられる。

この問題に対して、John J. Donovan により提唱された"3層クライアント/サーバアーキテクチャ"[1],[2]が注目されている。このアーキテクチャではクライアント/サーバシステムの役割を以下に示す3つの層に分割している。

プレゼンテーション層 マンマシンインタフェースを提供する。

機能層 他の2層との接続、業務機能を提供する。

データ層 既存システムとの通信による情報交換も含む透過性のあるデータサービスを提供する。

図 2に示すように、この3つの層への分割は論理的な分割であり、これらの層がハードウェアなど物理的な配置を限定しているのではない。

このようなアプリケーションアーキテクチャをクライアント/サーバシステムに適用することにより、次のような効果が考えられる。

- 3つの層を構成上明確に分離し、層間に共通のインタフェースを設定することにより、システムを再利用性の高い独立の部品群として構成することができる。これらの部品を組み合わせてアプリケーションを作成することにより、改造が容易な再利用性の高いシステムを構築することが可能となる。[3]では多層のアプリケーションアーキテクチャを利用した再利用性に関する検討がなされている。

- 業務量増加や機能追加に際して、比較的容易にシステムを拡張できる。

- データ層に既存の外部システムとの通信機能を配置することにより、外部システムとの透過的接続が可能となる。メインフレームシステムからクライアント/サーバシステムへの柔軟なシステム更改が可能となる。

しかしながら、このアーキテクチャを適用したシステムを開発しようとしたときに、従来の単一的なアプリケーションプログラムを構成していた機能を、それぞれどの層の機能として、どのくらいのまとまりで切り出せばよいのか、という問題が争点となる。

本稿では、この問題を解決するための一つの方法として、手順に従い設計を行うことにより各層の機能を抽出していく手法を提案する。

2 "インタフェース"の導入

設計手法の検討にあたって、原典の3層アーキテクチャモデルでは詳細な議論が困難であるため、拡張されたアーキテクチャモデルを定める。3層アーキテクチャを適用することができるシステム環境はかなり幅が広く、それぞれ細部の仕様が異なるため、それらを総合的に議論することは困難である。ここで提示するモデルは、ある程度のトラヒックを持ったオンライン処理に利用されることを想定したものである。

図 3にそのモデルを示す。このモデルでは各層の間にインタフェースという概念を設けることに

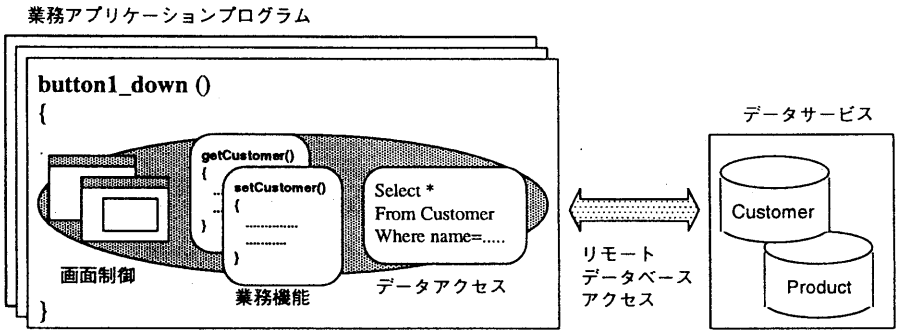

図 1: 典型的なクライアント / サーバアーキテクチャ

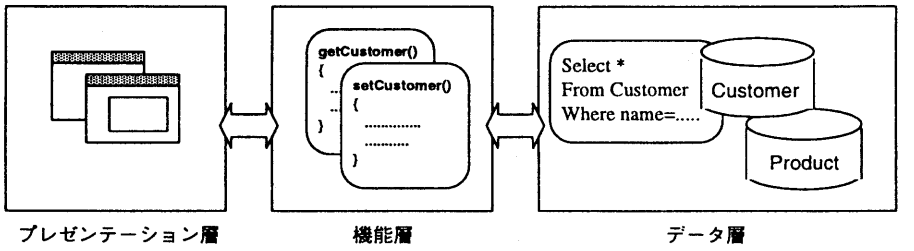

図 2: 3層クライアント / サーバアーキテクチャ

より、問題の焦点を明確にした。ここでいうインタフェースとは、上位層から下位層への要求と応答の対であり、以下のような属性を持つ。

識別子 要求情報であり、インタフェースを一意に識別するもの。識別子には機能が一意に対応する。

入力 要求情報であり、機能が必要とするデータ、機能の制御を指定するパラメタなどである。省略可能である。

処理結果 応答情報であり、機能の処理結果を判定する情報である。

出力 応答情報であり、機能の処理結果により得られた情報である。省略可能である。

また、プレゼンテーション層と機能層の間のインタフェースを **P-F** インタフェース、機能層とデータ層の間のインタフェースを **F-D** インタフェースと呼ぶ。

この言葉を使えばここで検討する問題を「与えられた要求仕様に対して、適当な P-F インタフェース、F-D インタフェースを抽出すること」と言い換えることができる。インタフェースは使用する環境により実装方法が異なるが、その違いは以下の設計手法に関する検討に影響を与えない。

3 設計前工程

ここではインタフェースを抽出する前に決定しておかなければならないことについて説明する。

3.1 業務分析

この作業は従来のシステム設計の分析作業とほとんど変わるところはない。この工程では、次のことを明確にする必要がある。

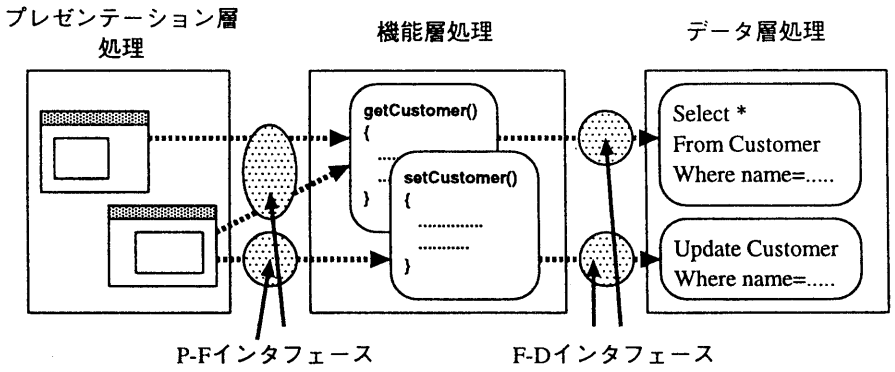

図 3: “インタフェース”

- システム化対象となる業務にはどのようなものがあるか
- それぞれの業務はどのような機能や制約を持つか
- それらの業務はどのように遷移していくのか
- それぞれの業務は何を入力として何を出力するのか
- それらの入出力情報はどこから来て、どこに行くのか

これらを業務フロー図として整理する。これらはすべて業務ロジックを決定する重要な要件であり、以下のすべての工程に利用される情報となる。

3.2 データ分析

この作業ではシステムで管理するデータを分析し、データ層の設計に反映する。この工程では、次のことを明確にする必要がある。

- それぞれの業務でどのような実体を管理するか
- それぞれの実体はどのような属性を持つか
- 実体間にはどのような関連があるか
- システムの一貫性を保証するどのような制約が、どの実体または実体間にあるか
- それぞれの実体間の主キーは何か
- 特に外部システムとの間で流通する情報には何があるか、どんな外部システムインタフェースが存在するか

この工程では実体関連図を作成する。

3.3 画面設計

画面設計は直接プレゼンテーション処理と P-F インタフェースの仕様に関わる設計情報を抽出する。画面設計はユーザの要望に影響するところが大きいので、設計と要望の確認を繰り返すことがある。

この工程で画面遷移を設計するが、ここでの画面遷移とは次の 2 つがあることを注意しておく。

- 異なる画面への切替え (通常の画面遷移)
- 同一の画面内でユーザの与えるイベントによるその画面の状態遷移

後者の例として、ユーザが一つの画面に対して段階的に情報を入力し、その都度ボタンをクリックしていくことにより、一つの画面で業務を完了するものがある。

画面と画面遷移の設計は次のような手順で行う。

1. 業務毎にその業務を構成する画面とそれらの画面遷移を設計する。

2. 全業務の画面の中で、複数の業務で使われる画面があれば統合して、新たな画面遷移を設計する。

3. 画面に識別子を付与する。

この工程では、次のことを明確にする必要がある。

- それぞれの業務について、設計した画面遷移に問題はないか
- それぞれの画面の入力情報、出力情報および機能は何か

この工程では画面遷移図、画面設計書を作成する。

4 インタフェース設計

ここで説明するインタフェース設計手法では、各層の処理を設計していく過程で、次第にインタフェースの全貌が明らかになっていく。

4.1 STEP0

前工程で設計した画面のうち、システムに対する要求を持つ画面、つまりP-Fインタフェースを持つ画面を抽出作業の対象とする。その中の一つの画面を選ぶ。以後の作業は各画面毎に適用して、P-F,F-Dインタフェースを抽出する。

4.2 STEP1

その画面に対する入力情報、出力情報および機能がすでにわかっているので、識別子を付与することにより、P-Fインタフェースは処理結果の種類を除いて決定する。このP-Fインタフェースを使用して、プレゼンテーション層の処理を設計する。この段階で詳細な処理を設計する必要はなく、全体の制御や機能に関係する処理を明確にすればよい。

4.3 STEP2

次にP-Fインタフェースに対する機能層処理に着目する。入力情報、出力情報および機能がP-Fインタフェースにより与えられているので、その条件を満たすように

- この処理でアクセスする実体
- その実体に対する要求

を抽出する。

4.4 STEP3

抽出した実体と要求毎に

- 入力情報
- 出力情報
- 機能

を明確にする。これがF-Dインタフェースとなる。

4.5 STEP4

F-Dインタフェースの機能を実現するデータ層処理を設計し、データベース、外部システムに対する問合せ処理をF-Dインタフェースごとに明確にする。これにより、F-Dインタフェースが処理結果として持つ“正常”以外の結果種別が判明する。

4.6 STEP5

明らかとなったF-Dインタフェースを利用し、機能層処理を設計し、機能が実現できていることを確かめる。F-Dインタフェースを実体へのアクセスとしたため、実体間の一貫性保証は機能層処理で行うこととなる。

機能層処理の設計過程において矛盾や不具合が発生して、F-Dインタフェースを考え直さなければならない場合には、STEP3にもどって新しいF-Dインタフェースを設定する。

機能層処理の明確化により、P-Fインタフェースが処理結果として持つ“正常”以外の結果種別が判明する。

4.7 STEP6

明らかとなったP-Fインタフェースの処理結果を利用し、プレゼンテーション処理を見直す。

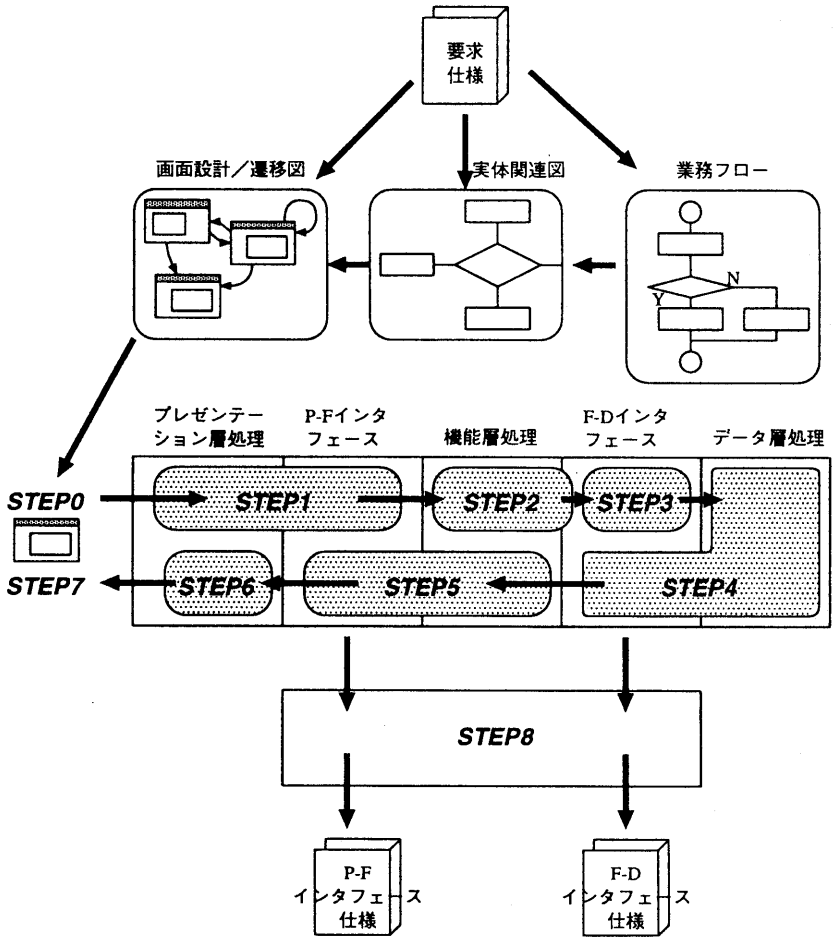

図 4: インタフェース設計手順

4.8 STEP7

最後に全体の処理を見直し、処理に問題がないか確認する。

4.9 STEP8

以上を全業務に行い、システムの持つ全インタフェースを抽出した。次に、入力情報、出力情報、機能のすべてが等しいインタフェースの識別子を統一し、インタフェースを統合する。

5 事例

ここでは商品流通業のためのシステムの中から受注登録処理にこの設計法を適用した事例を紹介する。

業務分析、データ分析および画面設計は既に完了しており、設計に必要な情報は整理されている。受注登録ウィンドウの設計情報のうち、層間インタフェース設計に必要な画面設計情報を図 5に示す。この中で、“一覧検索ボタン”というウィンドウの部品はクリックされることにより、別のウィンドウ“顧客一覧検索ポップアップ”を開き、そこでの検索結果を“顧客一覧リスト”に表示する。顧客一覧

の検索機能を別のウィンドウから呼び出される機能としたのは、この機能が多くのウィンドウから共通的に利用されるためである。このウィンドウで重要な部品は“登録ボタン”であり、クリックの背景で受注情報とその受注に対する明細情報が登録される。受注登録と明細登録は、どちらもそれ自体で一貫性を保証する単位となる、いいかえれば、どちらも単独で起動される可能性が業務フロー上あったため、2つの別のP-Fインタフェース候補とした。

前述の手順による効率的なインタフェース抽出作業を支援するためにシナリオフロー図を考案した。図6は受注登録の例である。図5の入力データおよび出力データの記載情報のうち、受注登録では受注伝票番号を除いた“受注”を入力として、“受注伝票番号”を出力とする。この条件により、P-Fインタフェース(PFIの欄)を決める。この時点では処理結果([OK],[ER]など)は明確になっていない。プレゼンテーション層の処理では入力をチェックし、P-Fインタフェース“受注登録”を呼び出し、伝票番号を表示するという処理を記述することができる。ここまでSTEP1を終了する。

続いてSTEP2に入るが、機能層処理では“受注”を1件追加することが自明である。しかしSTEP3において、“受注伝票番号”の扱いが問題となり、以下のような解が考えられる。

1. 与えられている“受注”の内容、現在のおよその時刻などから一意性のある“受注伝票番号”を機能層で作成し、F-Dインタフェースの入力とする。

2. F-Dインタフェースとして“受注”のみを入力とする。従って、データ層の処理で既に登録されている“受注”を検索し、未登録の“受注伝票番号”を払いだし、“受注”を登録する。

STEP3からSTEP5をそれぞれに適用した結果を評価すると、後者はシーケンシャルな“受注伝票番号”を得ることが可能だが、排他制御により、この登録処理がシステムのボトルネックとなることがわかった。ここでは最初のF-Dインタフェースを採用した。

図6中の処理結果、OK,FE,ER,DPはそれぞれ正常、致命的障害、エラー、データ重複を表現している。これらの詳細な内容は後の設計工程で明確にしていく。

STEP6では決定した処理結果に対して、ポップアップウィンドウを表示し、操作者に確認を求める処理を追加し、プレゼンテーション層の処理を完成した。

この例のように、F-Dインタフェースの設計で作業者の知識により設計結果に不統一が発生する可能性がある。この不統一を解消するために、最終的にSTEP8のレビュにより不統一を吸収する作業が必須となる。

6 データ指向アプローチとの比較

ここで提示した設計法は、ユーザの要求に合致したプレゼンテーションの設計からデータ層とのインタフェースを設計するトップダウン的なユーザ要求指向アプローチである。一方で、ボトムアップ的な手法であるデータ指向アプローチも有力な設計法として考えられる。ここで提案した3層インタフェース設計法にデータ指向アプローチを適用すると、F-Dインタフェース設計の後でP-Fインタフェースを設計することとなる。このアプローチにおいても、プレゼンテーションとデータのギャップを埋めるために、機能層処理の設計はプレゼンテーション層処理とデータ層処理の設計後に行なうのが適当である。

両手法の利点と欠点はそれぞれ相対するものである。例えば、ユーザ要求指向アプローチではユーザ要求を満たすようにインタフェースを設計するので、ユーザ要求の達成度は高い。データ指向アプローチではユーザ要求の実現のために、新たにF-Dインタフェースを追加しなくてはならないことがある。また、管理するデータに対する詳細な条件が定まっていない場合でもインタフェースを設計し、データに対する条件を抽出することができるが、データ設計に対する影響はある。

一方、データ指向アプローチでは既存システムのデータベースを利用する場合に、実現可能な機能を

提供することが容易である。ユーザ要求指向アプローチではF-Dインタフェース設計時になって始めて実現可能なインタフェースが既存システムに用意されていないことに気がつくことがある。また、要求仕様がはっきりしていない場合にもインタフェースを設計し、ユーザに提供可能な機能を準備することができる。しかしながら、要求仕様に対する割り切りが必要な可能性がある。

以上を整理すると、本手法では要求が明確になっているシステムを新規に設計する場合に有効であり、データ指向アプローチは既存システムをデータ層として利用する場合、または現行システムを改造する場合に有効である。

本稿では要求仕様が明確になっていることを前提にしているためユーザ要求指向アプローチをとったが、既存システムを利用する場合のためにデータ指向アプローチを部分的に採用する必要がある。その方法については今後の課題としたいが、両者の組合せにより、あまりにも繁雑な手法とならないように注意したい。

7 課題

その他、次のような今後の課題が挙げられる。

手順の効率化 例えば、代表業務について先行的にインタフェースを設計し、その結果を他の業務に反映することにより手順を効率化する手順が望まれる。

後続工程への接続 各層処理やインタフェースの設計情報をその後の設計工程に連続的に利用していく方法の検討が必要である。

改造時の手順 ここでは新規設計の手順を説明したが、既存3層アーキテクチャシステムの改造時の手順の検討が必要である。

ツール化 シナリオフロー図は現在紙と鉛筆を使用しており、工程途中何度も修正が必要となった。この作業を効率化するために、手順に沿った設計をサポートするツールが望まれる。

8 おわりに

3層クライアント/サーバアーキテクチャを適用したシステムの設計情報を抽出する方法を提案した。近年、日本でも3層アーキテクチャが注目されてきたが、それにともない3層アーキテクチャシステムの構築に関わるさまざまな技術が要求されている。

今後、先に掲げた課題を検討するとともに詳細な手順や判断基準を整理して、実験的な3層アーキテクチャシステムの開発と改造を行い、手法の有効性やシステム変更に対する柔軟性を評価する予定である。

謝辞

本検討を通して多大な協力を頂きました伊藤 路夫 情報システム本部部門長および平野 健一 グループリーダ、ならびに日頃ご指導頂く、細谷 僚一 ソフトウェア研究所長および長野 宏宣 部長に深謝いたします。

参考文献

[1] John J. Donovan. : *"Business Re-engineering with Information Technology"*, Prentice Hall. ISBN 0-13-125907-5.

[2] 拝原 正人 : "クライアント/サーバによる分散処理の動向 *(その 2)*, クライアント/サーバの実際と今後", NTT 技術ジャーナル 1994 年 12 月.

[3] 武宮博, 菊池一彦, 白沢智輝, 山岸信寛, 樋地正浩, 布川博士, 宮崎正俊 : "サービス統合化アーキテクチャの提案", 電子情報通信学会技術研究報告, OFS 94-36.

ウィンドウ名：　受注管理　受注登録　　作成者

SeqNo	イベント／操作				入力データ			出力データ			
	widget	Action	条件	種別	widget	データ項目	条件	widget	データ項目	条件	
1	一覧検索ボタン	s-click		ウィンドウ				顧客一覧リスト	顧客コード		
									顧客名		
									電話番号		
2	顧客一覧リスト	s-click		ウィンドウ	顧客一覧リスト	顧客コード		顧客コードテキスト	顧客コード		
						顧客名		顧客名テキスト	顧客名		
								配送先テキスト	住所		
								住所テキスト	住所		
3	明細作成ボタン	s-click		ウィンドウ				明細一覧リスト	商品コード		
									商品名		
									数量		
									販売単価		
									金額		
									受注明細番号		
								合計金額テキスト	合計金額		
5	登録ボタン	s-click		トランザクション	受注日テキスト	受注日		伝票番号テキスト	受注伝票番号		受注登録
					顧客コードテキスト	顧客コード					明細登録
					顧客担当者テキスト	顧客担当者名					
					担当者コードテキスト	販売担当者社員					
						コード					
					配送先テキスト	配送先					
					納品希望日テキスト	配送希望日					
					合計金額テキスト	合計金額					
					明細一覧リスト	商品コード					
					〃	販売単価					
					〃	数量					
					〃	金額					
						受注明細番号					
6	終了ボタン	s-click		ウィンドウ							

図 5: 受注登録画面設計情報

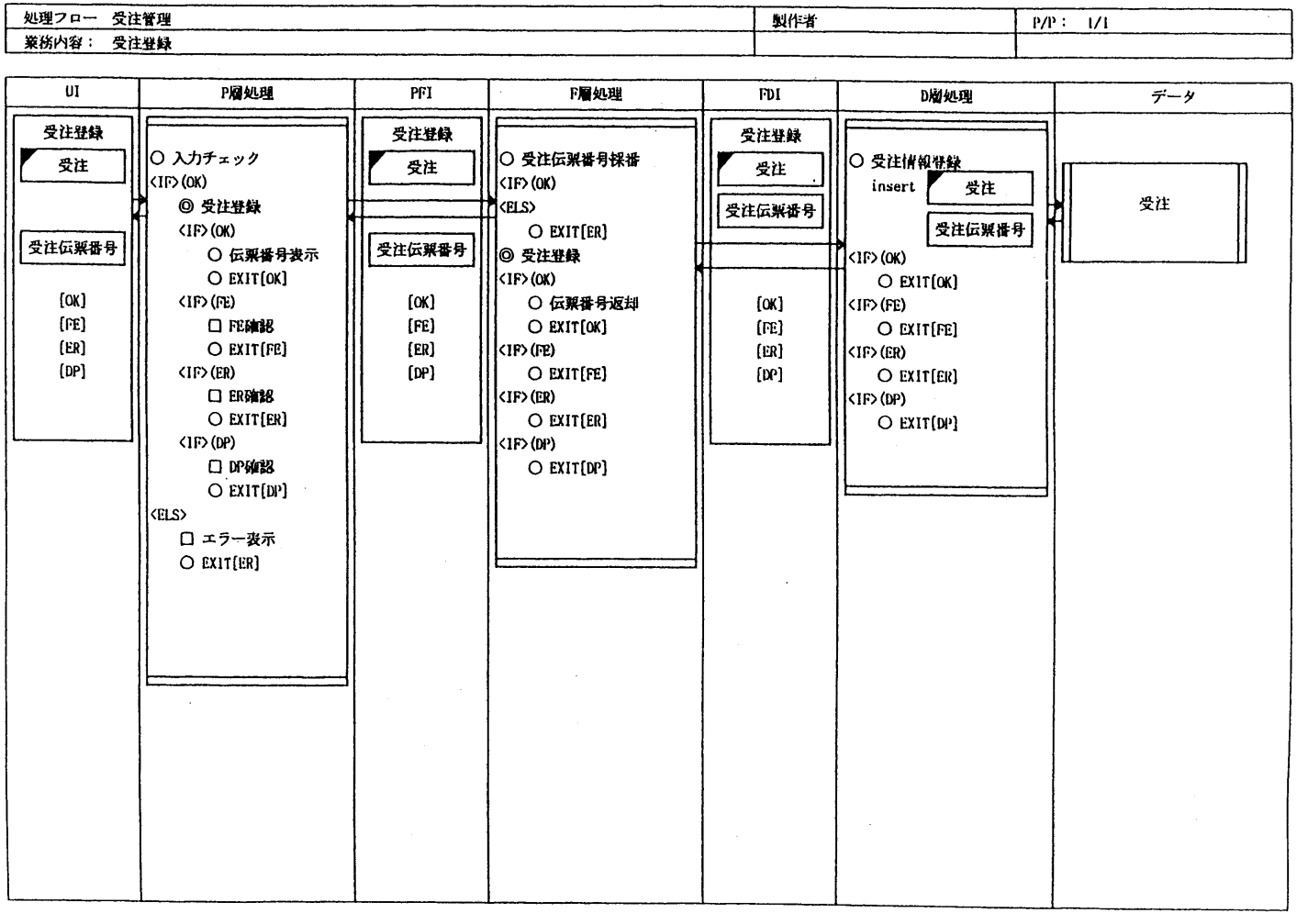

図 6: 受注登録シナリオフロー図