ソフトウェア工学 101－5
（1994．11．18）

分散オブジェクトによるプリンタ・サーバ

渡辺 治彦　　大平 剛

日本アイ・ビー・エム株式会社 東京基礎研究所

UNIX上での印刷処理の問題点は，エラー・リカバリや印刷制御などの管理・運用機能が不十分であること，常にプリンタを意識した印刷を行なわねばならないことである．我々はこれらの問題点を解決するとともに，システムの拡張性，再利用性を高めるために，オブジェクト指向技術を用いてプリンタ・サーバを設計し，実現をした．このプリンタ・サーバはクラス階層の継承関係を利用することにより，多機能で，かつ様々な印刷に対応するシステムとなった．本稿では，この分散処理システム上のプリンタ・サーバの設計と実現について述べ，その有効性について考察する．

A Printer Server with Distributed Objects.

Haruhiko Watanabe and Tsuyoshi Ohhira

IBM Research, Tokyo Research Laboratory

On UNUX systems, there are two problems that Printing management function is insufficient and printing is always aware of printers. We design and implement a printer server used Object oriented techinique to solove those problems and to increase extensibility and reusability. This system have many functions and support various printings because it make use of inheritance. This Paper discribe a printer server on distributed Processing systems, and consider its effect.

1 はじめに

UNIX ワークステーションやパーソナル・コンピュータのハードウェア技術，ネットワーク技術や分散システム技術の発展により，コンピュータ・システムが1台の汎用コンピュータと中心をするシステムから複数台の UNIX ワークステーションやパーソナル・コンピュータのネットワークによるシステムに移行している．このような分散処理システムへ移行するにつれ，従来，汎用コンピュータで処理されていた業務の大部分は分散処理システム上で稼働するようになってきている．

一方，UNIX ワークステーションやパーソナル・コンピュータを中心とする分散処理システムではまだ対応できていない機能や不十分な機能がある．その代表的な機能の1つに印刷処理機能がある [2]．従来の汎用コンピュータと比較して，UNIX あるいはパーソナル・コンピュータのオペレーティング・システムが提供する印刷処理機能は，印刷制御機能やエラー・リカバリなどの点において不十分であり，実際の業務上の要求を満たすには至っていない．

本稿では，UNIX での印刷処理機能の問題点を解決するプリンタ・サーバを，拡張性・再利用性という観点からオブジェクト指向技術を用いて設計・実現する．また3つの適用例を示し，プリンタ・サーバの有効性について述べる．

2 印刷処理

本節では，UNIX での印刷の方法，印刷制御の方法について述べ，UNIX の印刷処理が抱えている問題点を抽出する．そして，印刷処理に対する要求について考える．

2.1 UNIX の印刷処理

一般に，UNIX 上で印刷を行なう際には lpr コマンドを使用する．文書だけでなく図形を印刷する際にも，機種が異なるプリンタへ印刷する際にも lpr コマンドを使用する．しかし，lpr コマンドは単に指定されたファイルの内容を指定されたプリンタ (スプーラ) へ送る機能しかなく，図1に示すようにユーザがあらかじめ出力先のプリンタ機種を考慮し，出力先のプリンタが印刷可能な形式へフォーマット変換をしなくてはならない，つまり，印刷をしようとするユーザは常に出力先のプリンタ機種と出力先のプリンタが受け付けるフォーマットへの変換方法を知らなくてはならない．

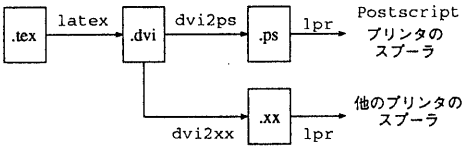

図 1: UNIX の印刷手順 (LaTeX の場合)

印刷処理を行なうアプリケーション・プログラムも同様に，出力先プリンタを意識する必要がある．アプリケーション・プログラムが印刷処理を行なうには，アプリケーション・プログラムの内部で，出力先プリンタの受け付けるフォーマットへ変換する必要がある．従って，アプリケーション・プログラム内部には，出力先のプリンタが印刷可能であるようなフォーマットに変換するルーチン，つまり出力先デバイスに依存したコードを埋め込む必要がある．もし印刷先として新たなプリンタ機種が導入された際には，図2に示すように，アプリケーション・プログラムの変更が必要となる．またこの変更はアプリケーション・プログラムごとに必要であり，システムが大規模になればなるほど，印刷処理を行なうアプリケーション・プログラムが多ければ多いほどアプリケーション・プログラマの負担も大きくなる．

プリンタやネットワークの障害などのトラブルにより，印刷されなかった場合を考えると，UNIX は印刷におけるエラー・リカバリの機能を提供しておらず再度印刷を実行する必要がある．

このように UNIX では，極めて簡単な印刷処理機能しか提供しておらず，実際の業務で使用するには，不備な点が目立つ．特に障害発生時のリカバリ処理や，印刷ステータスの確認などの運用・管理面での機能が不十分で

ある.

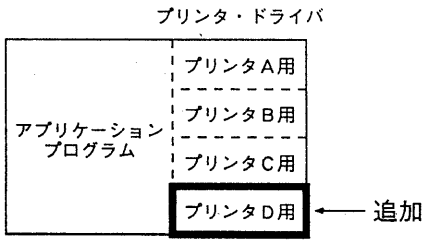

図 2: 新規プリンタ導入によるアプリケーション・プログラムの変更

UNIX での印刷処理の問題点をまとめると次の 2 つに絞られる.

UNIX の印刷処理の問題点

- 出力先プリンタを意識する必要がある
- 運用・管理機能が不足している

2.2 印刷処理に対する要求

実際の業務で印刷処理で要求される機能には, まず第一に障害発生時のエラー・リカバリ機能や印刷状況が確認できるといった運用・管理機能が挙げられる. また出力先のプリンタを意識せずに印刷可能にする機能やフォーム・オーバレイ, テキスト・フォーマットなどの高機能印刷の要求もある.

ネットワークを使用した分散処理システムでは, 出力先のプリンタの変更, 多機種に渡るプリンタのサポート, 印刷負荷の分散などが必要である [2].

このような印刷に対する要求をまとめると以下のようになる.

エラー・リカバリ機能 プリンタやネットワークなどの障害により印刷処理が正常に終了せず, 印刷がされなかった際に, 再度, 印刷処理を実行できる機能である. またマシン自体が障害により停止した場合, 停止前の状態を保持し, その時点に復旧できる機能も含まれる.

印刷制御機能 印刷優先順位の変更, 印刷の取り消し, 出力先の変更, 再印刷などの印刷処理を制御する機能である.

高度印刷機能 帳票の枠とその帳票に印刷するデータを印刷時にオーバレイする機能, テキストをフォーマットして印刷する機能, あるいは図形印刷機能などである.

負荷分散 印刷処理は, 他の処理に比べ時間がかかることが多く, 高速印刷の機能も要求される. 高速印刷に関しては, プリンタの印刷能力に依存するところが大きいが, 印刷処理の負荷分散機能により改善される.

プリンタ・ドライバ ネットワークを使用した分散処理システムでは, オープン化, マルチベンダ化が進んでおり, それに伴い印刷処理に関しても様々なプリンタのサポートが要求されている. そこで従来のように出力先のプリンタを意識して印刷するのではなく, プリンタ・ドライバを使用する印刷機能が要求されている.

3 分散オブジェクトによるプリンタ・サーバ

本節では, 印刷処理に必要な機能を満たし, UNIX での印刷処理が抱えている問題点を解決するようなプリンタ・サーバの設計と実現について述べる.

3.1 概念

本研究で取り上げるプリンタ・サーバの概念を図 3に示す.

プリンタ・サーバは, 通常の印刷処理においてユーザあるいはアプリケーション・プログラムからの印刷要求を受け, フォーム・オーバレイ処理などとともにプリンタに依存したフォーマットへの変更をし, 印刷をするシステムである.

また印刷要求を一括して引き受けて印刷処理を管理し, 障害に対するリカバリ機能を提供する.

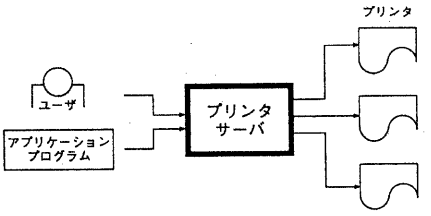

図 3: プリンタ・サーバの概念

3.2 機能

今回，実現するプリンタ・サーバの持つ機能を以下に示す．

- エラー・リカバリ機能

  プリンタ障害，ネットワーク障害時の再印刷機能とマシン・ダウン時の復旧を可能にする機能

- 印刷制御機能

  印刷取り消し，出力先変更，再印刷，印刷履歴の保持機能

- 高度印刷機能

  フォーム・オーバレイなどの高度な印刷処理に対応する仕組み

- 負荷分散

  印刷処理時の負荷分散を可能にする仕組み

- プリンタ・ドライバ

  印刷の際，プリンタに依存したフォーマットに変換できる機能

3.3 設計

3.2節で示した機能を実現するために，図4に示すようにプリンタ・サーバを5つの要素で構成する．

Entry 印刷処理に必要なデータである．印刷そのものに必要なデータと印刷制御に必要なデータが含まれている．

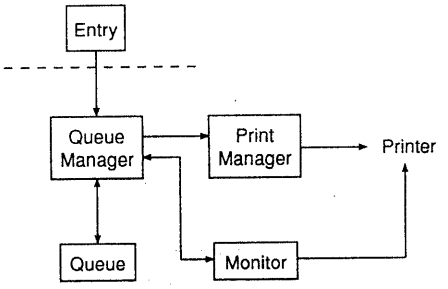

図 4: プリンタ・サーバの構成要素

QueueManager 印刷のエントリ管理を行なう要素である．キューの管理と印刷実行の要求処理，エラー・リカバリ機能，印刷制御機能を行なう．

Queue Entry を格納するキューである．各種の障害に対して対応する仕組みを持つ．

PrintManager 印刷を実行する要素である．フォーム・オーバレイなどの高度印刷機能と出力先のプリンタに合わせたフォーマット変換のプリンタ・ドライバ機能を受け持つ．

Monitor 印刷制御の実行を要求する．印刷取り消しや再印刷などの印刷制御処理を QueueManager に要求し，またプリンタの稼働状態も確認する．

このように，プリンタ・サーバを機能と役割に応じて，5つの構成要素に分けたが，Entry や PrintManager などは，印刷対象や印刷先のプリンタが代われば，それにともない変化する．

プリンタ・サーバでは多様な印刷とプリンタに依存しない印刷方法を提供することから，これら変化する構成要素を統一的に扱えるような考え方と方法を導入する必要がある．今回，このような考え方に関して，オブジェクト指向技術を採用する．

図4に示した5つの要素をオブジェクト指向技術のクラスとして構成した．その結果を OMT(Obejct Modeling Technique) [1] を用いて図5に示す．

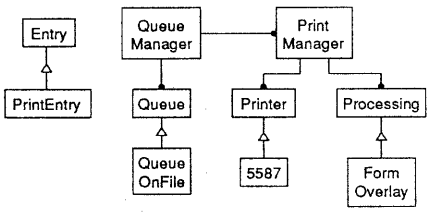

図 5: クラス構成

図5に示すように，プリンタ・サーバの機能を満たすために，新たに Print と Processing というクラスを導入している．Entry，Queue を合わせサブクラスを持つクラスが 4 つある．これらに共通していえるのは抽象クラスであるということ，それぞれがプリンタ・サーバ上で違いのある複数の情報や処理を統一的に扱おうとしていることである．

Entry クラスでは，印刷エントリの情報を統一的に扱おうとしている．先に印刷エントリの情報として，印刷そのものに必要なデータと印刷制御のために必要なデータの 2 種類があると述べた．印刷そのものに必要なデータは，印刷対象やフォーム・オーバレイなどの印刷処理が異なれば，それに伴い異なる．一方、印刷制御に必要なデータは印刷対象や印刷処理が異なってもデータ構造は同じである．

このようなデータ構造を統一的に扱うために，印刷制御用データを持つ Entry クラス (抽象クラス) と印刷対象などにより変化するデータを持つクラス (具象クラス) を考え，継承関係を持たせている．つまり図 6に示すように，帳票印刷，テキスト印刷，図形印刷を行なうための印刷エントリは，Entry クラスのサブクラスとして実現が可能である．さらに QueueManger など印刷エントリを扱うクラスは，Entry のサブクラスがなにであろうと，Entry クラスを介して，印刷エントリを統一的に扱うことができるようになる．

他のクラスに関しても同様のことがいえる．以下に Entry，Queue，Print，Processing についてまとめる．

Entry 印刷エントリの違いを吸収する．印刷エントリの詳細を意識することなく統一的に扱うことができる．

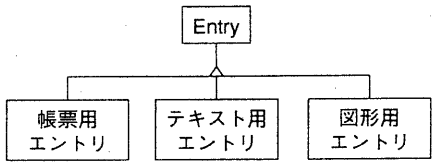

図 6: Enrty クラスとサブクラス

Queue キューを物理的なデバイスから分離する．キューがファイル上に構築されるか，あるいはファイル上に構築されるかを意識することなくキュー管理・操作を行なうことが可能になる．

Printer 物理的なプリンタの違いを吸収する．どのようなプリンタに出力するということを意識することなく，印刷処理を行なうことが可能になる．

Processing 印刷処理の違いを吸収する．フォーム・オーバレイなどどのような印刷処理が行なわれようとしているか意識することなく印刷を行うことが可能になる．

3.4 実現

プリンタ・サーバを実現するにあたり，負荷分散を考慮する必要がある．プリンタ・サーバを 1 つのプロセスで実現しようとすると以下に示すような懸念が発生する．

- プリンタ・サーバの稼働するマシンが過負荷になる可能性がある
- ある処理が他の処理の実行に大きな影響を与える可能性がある

  フォーム・オーバレイなどの時間がかかる処理を行なっている間は，印刷制御，印刷要求の受理など他の処理が実行されない．

そこで，プリンタ・サーバをキュー管理プロセス，印刷プロセス、そしてモニタプロセ

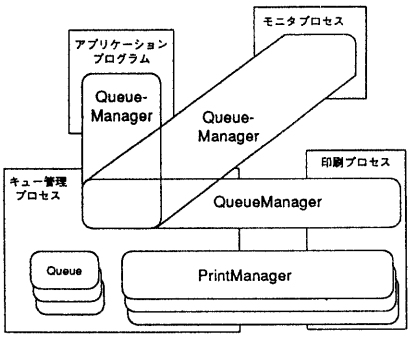

図 7: 実行中のオブジェクト

スの 3 種類のプロセスで構成し，各プロセス内のオブジェクトを図 7に示す．

図 7では，QueueManager と PrintManager が複数のプロセス間に跨って存在しているが，このオブジェクトを図 8に示すようにそれぞれを Requester と Server の 2 つに分け，各オブジェクト間を RPC(Remote Procedure Call) で通信を行なう．このようにすることで，各プロセス間を RPC で通信すること同じになる．

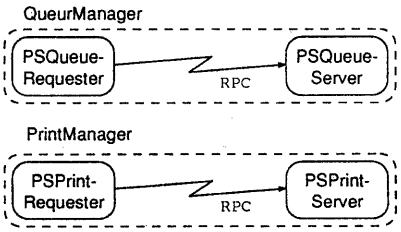

図 8: Requester と Server

以上により，プリンタ・サーバのプロセス構成を図 9に示す．

各プロセスの役割は以下の通りである．

キュー管理プロセス 印刷エントリ・キューを管理するとともに実際の印刷処理を行なう印刷プロセスの管理も行なう．このプロセスはアプリケーション・プログラムからの印刷要求を受け付け，データを適切な印刷プロセスへ割り当てる．またこのプロセスにより障害時のエラー・リカバリや再印刷，印刷取り消しなどの印刷制御が可能になる．

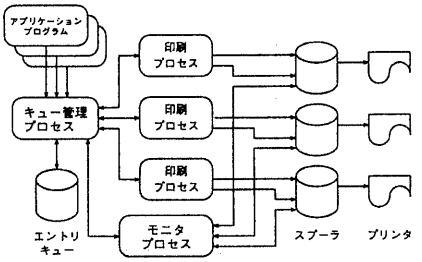

図 9: プロセス構成

印刷プロセス 実際の印刷処理を行なうプロセスである．このプロセスはフォーム・オーバレイなどの印刷処理を行なうとともに出力先プリンタに依存したデータへ変換する．この印刷プロセスは，プリンタ・ドライバの役割をし，アプリケーション・プログラムからデバイス依存モジュールを分離する．またこのプロセスは，キュー管理プロセスと同一マシン上で稼働する必要はない．ネットワークで継れた他のマシン上で稼働させることができ，印刷処理の負荷分散が可能となる 。

モニタプロセス 印刷状況，出力先の状況確認を行なうプロセスである．このプロセスからキュー管理プログラムへの印刷状況の確認，再印刷，印刷取り消しなどの印刷制御を要求することができる．また，出力先の状況確認を行なうことができる．

実現にあたり分散環境に対応するために RPC を使用したので，図 5に示すクラス構成とは異なるクラス構成となった．実現時のクラス構成を図 10に示す．

4 評価

4.1 特長

本研究で作成したプリンタ・サーバは，運用・管理面でのエラー・リカバリ機能，印刷

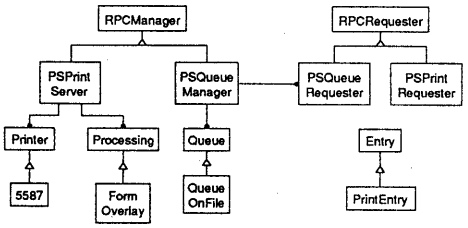

図 10: 実現時のクラス構成

制御機能を始めとして，要求される印刷処理機能はほぼ満たしていると言える．

このプリンタ・サーバでは，印刷処理を印刷プロセスとしてキュー管理とは別のプロセスで実現することにより，プリンタに依存する部分を分離している．つまり図 11に示すように，アプリケーション・プログラムは出力先のプリンタ機種を意識する必要はなく，印刷エントリと定められた形式のデータを渡すことで印刷を可能にする．したがって，アプリケーション・プログラムは出力先のプリンタ機種が変更されたり，新しいプリンタ機種が導入されたとしても変更する必要はない．また印刷プロセスをネットワークで接続されている他のマシン上で稼働させることにより，印刷負荷の分散に対応している．

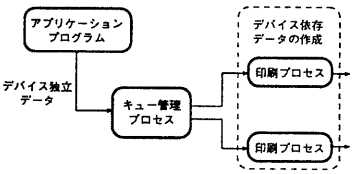

図 11: 印刷プロセスの働き

オブジェクト指向技術を利用し，クラス階層のサブクラスをうまく利用することで，プリンタ・サーバ自体の拡張性・再利用性が高まったといえる．例えば，単なるテキスト印刷しかできなかったプリンタ・サーバに帳票オーバレイなどの印刷処理を加える場合，印刷の処理の記述を Processing クラスのサブクラスとして定義し，帳票オーバレイの印刷時に必要な情報を Entry クラスのサブクラスとして定義すれば良い．

また新しいプリンタを導入する場合は，プリンタが受け付けるフォーマットへの変換処理など物理的なプリンタに対する記述を Printer クラスのサブクラスに定義すれば良い．

このように，オブジェクト指向技術を利用したことにより，追加や変更すべきところが局所化され，メンテナンスやカスタマイズが容易になるといえる．

4.2 適用例

プリンタ・サーバの適用例として，3 つの例を挙げる．印刷処理に対するそれぞれの要求は以下の通りである．

A 社 エラー・リカバリ
　印刷制御

B 社 帳票オーバレイ印刷
　エラー・リカバリ
　印刷制御

C 社 エラー・リカバリ
　地図印刷
　専用回線を利用した遠隔地印刷
　印刷制御

A 社の場合，アプリケーション・プログラムから送られたテキスト・データをそのままプリンタへ出力する．B 社の場合，アプリケーション・プログラムからテキスト・データを受けとり，印刷プロセスにおいて印刷エントリで指示された帳票とオーバレイを行なう．C 社の場合，図形データを印刷する．その際に LAN 上のプリンタだけではなく，専用回線を通して遠隔地のプリンタへ印刷を行なう．

上記 3 つの例では，印刷対象や処理が異なるので，印刷エントリである PrintEntry クラスの内容が異なる．以下にそれぞれの印刷エントリの内容を示す．

	A 社	B 社	C 社
Entry	出力先 受付日時 完了日時 印刷状況	出力先 受付日時 完了日時 印刷状況	出力先 受付日時 完了日時 印刷状況
PrintEntry		帳票 ID 用紙	地図名 中心地点

印刷エントリの内容が異なっても，その違いをEntryクラスのサブクラスとして実現することにより，他のクラスからは，Entryクラスのインタフェースを使用することができるので，QueueManageなどのクラスを変更する必要はない．またプログラマは印刷エントリの違いだけを記述し，Entryクラスのサブクラスにすれば良い．

またこの例の場合，テキスト印刷，帳票オーバレイ印刷，図形印刷と印刷処理そのものが異なるので，それぞれProcessingクラスのサブクラスを変更もしくは追加する．そして，使用するプリンタ機種が異なる場合，Printerクラスのサブクラスを定義する必要があるし，同じプリンタを使用する場合，何もする必要はない．

このようにプリンタ・サーバでは，印刷対象，印刷処理，プリンタ機種が異なっても極めて少ない変更で対応できる．

5 まとめ

本研究では，オブジェクト指向技術を使用したプリンタ・サーバを作成した．このプリンタ・サーバは，従来，UNIX上での印刷機能に不足していた管理・運用上の機能を持ち，ネットワークなどの分散システムにも対応している．

オブジェクト指向技術，特にクラス階層の継承関係を利用することにより，プリンタ・サーバの拡張性・再利用性が高まり，印刷対象，印刷処理，プリンタ機種といった様々な違いをわずかな変更で対応できるようになった．

今後の課題としては，今回実現時にRPCを使用することによって，図5に示したクラス階層を図10のように変更をした．しかし実際には，RPCさえも隠れたクラス階層のほうが望ましい．つまり，図5に示すクラス階層が理想形であると思われる．

従って，今後はそのクラス階層でのプリンタ・サーバについて考察を深め，実現をすることにより印刷に関するフレームワークを目指していきたい．

具体的には，OMG (Object Management Group) のCORBAに準拠し，IBMが提供するSOM (System Object Model)[3, 4, 5] を採用することで解決が計られる．RPCをSOMの構成要素の1つであるDSOM(Distributed SOM)[5] で置き換えることにより，RPCなどの通信を隠すことが可能である．またSOMのPersistence Frameworkによりキュー自体の扱いがより汎用的になり，エラー・リカバリに関する処理が容易になると考えられる．

参考文献

[1] J. RUMBAUGH *et al* (羽生田栄一監訳): オブジェクト指向方法論 OMT, トッパン (Jul. 1992).

[2] UNIX の大量印刷 高速印刷と分散印刷の環境整備進む, 日経オープンシステム, 13(Apl. 1994), 129–136.

[3] IBM's System Object Model (SOM) : Making Reuse a Reality, First Class, OMG, (Oct. 1994)

[4] SOMobject ツールキット ユーザーズ・ガイド, 第1版, 日本アイ・ビー・エム, (Oct. 1993).

[5] Experience with SOMobjects : Distributed System Object Model(DSOM), IBM Corp., (Nov. 1993).