Prepress 공정에서 포스트스크립트와 PDF 제어방법 비교에 관한 연구

양보규†, 오성상*, 유건룡**, 이의수

동국대학교 지역연고산업진흥 사업단, *신구대학 정보미디어학부,

**한국폴리텍대학 인쇄정보미디어과

(2007년 10월 10일 접수, 2007년 11월 15일 최종 수정본 접수)

The Study on the Comparison Method in Controlling PostScript and PDF in Prepress Process

Bo-Kyu Yang†, Sung-Sang Oh, Keun-Ryong Yoo**, Euy-Soo Lee*

Regional Innovation System, Dongguk University,

*Dept. of Printing Information Media, Shingu College,

**Dept. of Printing and Information Media, Korea Polytechnic College

(Received 10 October 2007, in final from 15 November 2007)

Abstract

The working environment of DTP tends to increase from Macintosh to MS Windows in Korea because of being developed exclusively cross-platformed applications. Base of this change is not related to spread of using PDF. Recently, it's cause of falling into dilemma in Prepress. It's due that operators in Prepress don't have correctly understanding about differences and features between Postscript and PDF.

This paper tried to search a way to solve problems via changing circumstance in operating Postscript and PDF.

1. 서 론

사용자가 작성한 데이터를 인쇄 공정에 적합하도록 재구성하는 단계로써 Prepress 공정은 일찍부터 IT기술을 적극적으로 차용하여 발전하였다. 이는 출판 산업에서는 DTP(DeskTop Publishing)의 등장 및 발전을 가져왔으며, 인쇄 산업에서는 필름 출력기(CTF, Computer to Film, Image setter)의 출현과 보급으로 구현되었다.[1)]

이러한 발전의 근간에는 미국 어도비사(Adobe, Co.)에서 1984년 발표한 포스트스크립트(postscript)가 자리 잡고 있다. 포스트스크립트는 DTP 애플리케이션에서 작성한 데이터와 Prepress 공정을 연결하는 데이터의 표준이다. 또한 포스트스크립트는 페이지 기술 언어(Page Description Language)이며 자체로 하나의 기능을 하기 보다는 전용 해석기(interpreter)의 규격에 맞게 각종 애플리케이션에서 생성한 데이터를 특정 작업에 대해 표준화하기 위한 미들웨어(middle-ware)의 성격을 갖는다. 어도비사는 1999년 포스트스크립트 레벨 3을 발표한 이후로 더 이상의 업그레이드는 중단한 상태이며 그 대신 PDF(Portable Document Format) 규격을 제안하고 있다.[2~5)]

PDF 규격은 포스트스크립트의 장점을 유지한 채, 가독성 및 기능을 확장한 것으로 기존의 포스트스크립트 기반의 Prepress 공정을 급격하게 대체하고 있다. 특히 PDF 중심의 워크플로 구축 시, PDF 파일의 헤더 확장 영역을 이용하여 후(後)공정의 관리에 필요한 정보를 임의로 삽입할 수 있어 CIP4 구축이 용이하다는 장점이 있다.[6~7)]

하지만 국내의 DTP 및 Prepress 환경은 PDF 기반의 워크플로에 적합한 서체 보급의 미비, CTP(Computer To Plate)와 같은 디지털 기반의 장비 보급의 미비 등으로 현재까지 포스트스크립트 기반의 DTP 환경과 Prepress 관련 업무를 진행하고 있다. 하지만 DTP용 애플리케이션들은 PDF에서 확장된 기능을 적극 수용함으로써 사용자가 작성한 문서와 포스트스크립트 중심의 기존 Prepress 공정에서의 데이터 규격 불일치로 인한 문제가 발생하고 있는 상황이다.[8)]

따라서 본 연구에서는 포스트스크립트와 PDF, 두 포맷의 차이와 특성 등에 대해 살펴보고 데이터 특성에 따른 Prepress 공정에서의 데이터 제어와 관련한 내용을 기술하고자 한다.

2. 본 론

2-1. 포스트스크립트 기반의 Prepress 공정 개요

최근 윈도즈 기반의 애플리케이션에서 작성한 데이터에 대한 인쇄요구 수요는 상당히 늘고 있는 추세이지만 대부분의 경우, 매킨토시 환경에서 작성한 문서 혹은 데이터

를 대상으로 업무를 진행하고 있다. 이는 DTP 보급 초창기부터 보급되어 온 상업용 서체의 보편적인 사용과 무관하지 않다. 일반적으로 DTP용 서체는, 매킨토시와 같은 클라이언트 PC에서는 화면 디스플레이용 저해상도 서체를 사용하여 데이터를 작성하고 고해상도 출력시에 데이터에 포함되어 있는 서체는 RIP(Raster Image Processing) 엔진에 저장되어 있는 고해상도 서체로 대체(assign)되어 출력하는 구조를 갖고 있다. 이러한 구조는 매킨토시 운영체제의 특성과 포스트스크립트 라이선스에 대한 권리 보호에 기인한 것이지만 사용자에게는 컴퓨터 처리 능력의 한계와 더불어 합리적인 구조로 이해되었다.

여러 애플리케이션에서 각각 작성한 데이터는 프린터 드라이버를 통해 포스트스크립트 파일로 변환, 전송한다. 이 때 이후 진행할 공정의 필요에 따라 클라이언트 PC에 파일로 저장하거나 RIP 서버에 설정해 놓은 가상 프린터 [일반적으로 특정 폴더, 이 폴더는 서버에서는 핫 폴더(Hot Folder) 혹은 잡 폴더(Job Folder)로 인식하지만 클라이언트 PC에서는 운영체제에 등록해놓은 프린터 드라이버로 인식] 로 인쇄한다. 즉, 클라이언트 PC에서는 인쇄 명령을 실행한 것이지만 RIP 서버에서는 포스트스크립트 파일로 인식하며 EOF(End Of File)을 감지한 후에는 미리 사용자가 정의해 놓은 후속 작업을 자동으로 진행한다.

이렇게 생성된 파일은 운영체제에 관계없이 파일 식별을 위한 파일명과는 별도의 확장자(extension)인 '.ps'를 자동으로 추가한다. 이 확장자는 매킨토시 운영체제에서도 마찬가지로 적용된다. 매킨토시 운영체제에서의 파일 관리는 각각의 파일 앞부분에 정의되어 있는 리소스(header resource)를 통해 파일의 속성을 감지하기 때문에 확장자가 필요 없지만 윈도즈나 유닉스 계열에서는 확장자를 통해 파일의 속성을 인식하기 때문이다.

포스트스크립트 파일의 저장 위치가 달라지는 것은 대부분 터잡기(imposition) 작업을 수행하는 프로그램 때문이다. 터잡기 프로그램의 운영 환경에 따라 공정 수행의 효율성을 위한 파일 저장 위치를 달리 한다. 터잡기 작업을 클라이언트 PC에서 수행할 경우 새로이 생성된 포스트스크립트 파일(터잡기 작업을 수행하면서 새로이 생성한 파일)을 RIP 서버로 전송한다. 반면 터잡기 작업을 RIP 서버에서 수행한 경우, RIP 서버 모듈에 의한 자동 데이터 마이그레이션 과정을 거쳐 터잡기 작업에서 사용한 포스트스크립트 파일을 삭제하거나 별도의 터잡기용 레이아웃 파일을 새로이 생성하기도 한다.

RIP 서버에서는 CPSI(Configurable PostScript Interpreter) 모듈을 통해 이러한 과정을 거친 포스트스크립트 파일을 해석한다. CPSI 모듈은 포스트스크립트 규격의 확장으로써, 특정 리소스에 대한 정책을 설정한다. 즉, 포스트스크립트 파일의 헤더 분석을 통한 문서 전체에 적용할 트래핑(trapping) 설정, OPI(Open Prepress Interface) 사용 유무 등에 대한 정책, 일부 리소스에 대한 오버라이트(overwrite) 혹은 넉아웃(knockout) 처리 기준 등을 분석한다. 이 값은 사용자의 필요에 따라 포스트스크립트 파일 헤더의

내용 혹은 CPSI 모듈에서 새로이 정의한 내용을 우선할 것인지를 결정할 수 있다.

RIP 서버에서는 최종적으로 이러한 과정을 마친 포스트스크립트 파일을 대상으로 인쇄 공정에 적합한 망점으로 구현한 CMYK(cyan, magenta, yellow, black) 각각의 색상 분판 정보를 분리한 4개의 1비트 TIFF(Tag Image File Format)을 생성한다. 만일 기본 4색 이외의 특정한 색이 추가되어 있을 때에는 별도 색판을 인쇄하기 위한 별도의 1비트 TIFF 파일을 추가로 생성한다. 이들 파일들이 각각의 필름 혹은 CTP 판재에 노광하기 위해 쓰인다.

반면 RIP 서버에 연결되어 있는 컬러 플로터(Color Plotter)에 적용할 때에는 RIP 서버에서는 이들 파일의 컬러 데이터를 기준으로 하나의 파일로 재구성하여 출력한다.

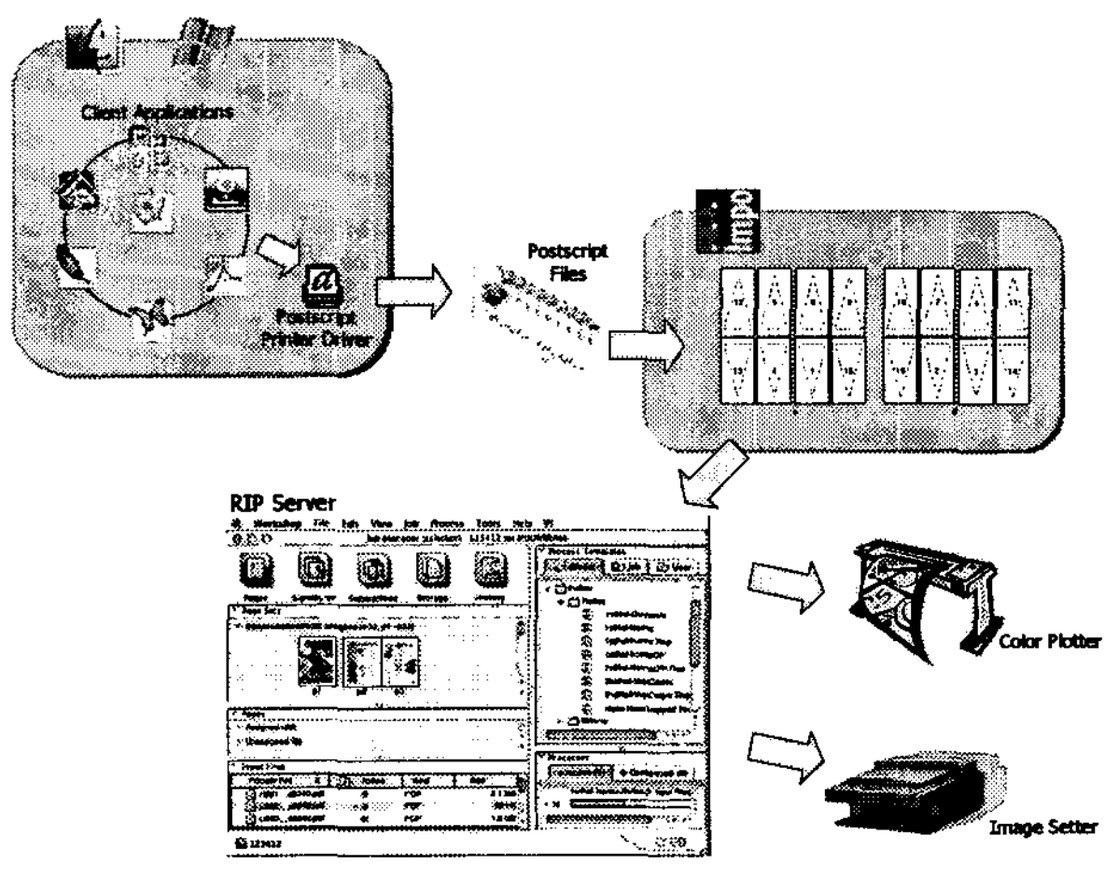

Fig. 1. Process of PostScript based.

2-2. PDF와 포스트스크립트의 비교

PDF 파일은 1992년 어도비사에서 처음 제안한 이후 현재는 문서 보관용, 고해상도 출력용, 프레젠테이션 등으로 그 용도를 넓히고 있다. 포스트스크립트가 인쇄 등과 같은 특정한 영역에서 쓰이는 데에 반해, PDF 규격의 용도가 다각화된 데에는 아래와 같은 요소들 때문이다.

먼저, PDF는 트루타입 서체(true type font)를 사용함으로써 화면 디스플레이와 고해상도 출력 관련 요구 모두를 충족시킬 수 있다. 포스트스크립트는 전용 해석기를 통해 가시화시켜도(rastering) 서체 리소스에 대해서는 구현할 수가 없다. 포스트스크립트용 서체는 래스터라이징 연산시 필요한 기준 좌표(anchor point)와 각 노드(node)의 좌표값 및 각 노드 연결에 필요한 정보(직선 혹은 곡선에 대한 정의로 곡선인 경우, 노드의

좌우측 연결 곡선의 세부 속성 정의에 필요한 control point 속성을 추가로 정의)만이 있을 뿐이다. 또한 이를 해석하기 위해서는 포스트스크립트 파일 내에 정의된 해당 서체의 포스트스크립트 파일이 별도로 존재하여야 하며, 이 파일은 해석기가 동작하는 환경 내에 반드시 정의되어 있어야 한다. 즉, 포스트스크립트 파일 내에는 서체 자체와 관련한 리소스가 포함되어 있지 않다. 화면용 서체(screen font)란 포스트스크립트 서체를 화면 해상도에 맞게(매킨토시의 경우, 72 dpi) 비트맵 서체(bitmap font)로 추출한 후, 각 글자(numeric character)의 코드 값을 매핑(mapping)한 것에 불과하다. 때문에 엄밀한 의미에서의 WISWIG(What You See Is What You Get)을 구현한 것은 아니다.

트루타입 서체는 애플사(Apple, Co.)가 어도비사의 포스트스크립트에 대한 대항으로 제안한 규격이다. 대표적인 특징은 아웃라인(outline) 속성을 갖고 있어 화면 디스플레이와 같이 비교적 저해상도(96 dpi 이하)에서도 정확한 표현이 가능하며, 고해상도로의 이용 시에도 같은 결과를 얻을 수 있다. 즉, 포스트스크립트와 같이 장치 독립적인 속성을 갖고 있다. 이러한 속성 때문에 서체를 문서 내에 저장(embedding)하는 것이 가능하며, 이를 적극 활용하고 있는 것이 PDF다. 즉, 기존의 비트맵 이미지와 아웃라인 이미지, 서체를 모두 수용할 수 있는 가시화 가능한 문서 포맷이 등장한 것이다.

둘째, PDF 규격은 다양한 해상도, 속성의 이미지를 지원할 수 있다. 포스트스크립트 규격에는 비트맵 이미지에 대한 리소스는 원시 데이터(raw data, 사용자가 작성한 특정 애플리케이션을 사용하여 그 내용을 확인할 수 있는 데이터)의 속성을 그대로 유지한다. 즉, 포스트스크립트 파일 내에 포함되어 있는 이미지가 300 dpi의 해상도를 갖는다면 래스터라이징 과정에서 그 값을 그대로 사용할 뿐이다. 이에 대한 다운샘플링이나 업스케일링은 해석기에서 진행한다.

반면 PDF는 원시 데이터에 대해 미리 정의해 놓은 설정(job option)을 통해 화면 디스플레이와 RIP 프로세스에 적합한 정의를 규정할 수 있다. 이 설정 값을 통해 이미지에 대한 다운샘플링, 표현 색상 기준(RGB 혹은 CMYK) 등을 규정한다. 따라서 사용 목적에 따른 컬러 테이블을 적절하게 설정함으로써 매체에 적합한 색상 표현이 가능하다.

부가적으로 각각의 리소스에 적합한 손실(loss) 혹은 무손실(lossless) 압축 알고리즘을 적용함으로써 새로이 생성한 PDF의 데이터 크기를 적절하게 줄일 수 있다. 포스트스크립트 규격에서는 정식으로 지원하는 압축 알고리즘은 없으며 일부 상용 RIP에 따라 기능 확장 개념으로써, ZIP과 CCTII 계열 압축 알고리즘을 지원한다.

셋째, PDF 파일을 생성할 때 헤더에 사용 목적에 적합한 다양한 부가 정보를 첨부(attach)할 수 있다. 이러한 기능을 이용하여 인쇄 공정을 포괄할 수 있는 CIP4를 유지하기 위한 JDF(Job Definition Format) 규격을 각각의 PDF 파일 내에 부가한다.

DTP 전용으로 특화된 몇몇 애플리케이션들은 운영체제에 설치한 프린터 드라이버와는 별도의 PPD(Postscript Print Definition) 파일을 사용한다. PPD 파일은 각 애플리케

이션에서 처리하는 각각의 리소스의 처리 방법에 대한 기준을 선언하는데 이 정보는 포스트스크립트 파일 작성시 파일 전체 내용의 선언에 대한 헤더의 일부로 사용된다. 이 때 인쇄의 각 공정을 진행하기 위한 세부 지시 사항 등과 같은 내용은 포스트스크립트 파일에 추가할 수 없다. 포스트스크립트 헤더에 정의되지 않는 값이기 때문이다. 따라서 각 공정을 진행하는데 필요한 세부지시 사항 등은 별도의 ERP(Enterprise Resource Planning) 등을 구축하여 연동되도록 구축하여야 한다. 반면 PDF는 헤더에 사용자 정의(User Defined) 영역이 충분히 할당되어 있어 XML 기반의 JDF를 포함할 수 있다.

또한 PDF는 PPD를 사용하지 않는 MS-Word나 한글에서 작성한 데이터에 대해서도 PPD를 적용한 수준의 세부적인 리소스 제어에 대한 규칙을 설정할 수 있다. 이 작업은 PDF를 생성하는 애플리케이션인 Distiller에 마련되어 있는 옵션을 통해 수행한다.

2-3. PDF 출력의 문제점

최근까지 인쇄업계에서는 PDF를 가시화된 포스트스크립트(Visible Postscript) 개념으로 이해하고 사용하였다. 이와 같은 개념이 정립된 데에는 PDF 파일을 생성할 때, 원시 데이터에서 포스트스크립트 파일을 생성하고 Distiller나 PDF Writer를 통해 PDF 파일을 생성하였기 때문이다. 다만, 매킨토시 환경에서 작성한 데이터의 경우에는 서체의 특성 때문에 클라이언트 PC에서는 Acrobat이나 Distiller를 이용하여 PDF 파일을 생성할 수 없다. 국내에 보급되어 있는 매킨토시용 서체는 트루타입 형식이라 하여도 스케일링(scaling)과 관련한 정보만 담고 있을 뿐 기존의 포스트스크립트 서체에서와 마찬가지로 RIP 서버에서 서체 매핑 방식으로 운영되기 때문이다. 이 때문에 PDF 파일 생성시, 원시 데이터에 포함되어 있는 서체는 PDF 파일 내에 저장할 수 없다.

인쇄용으로 적합한 PDF 규격은 버전 1.3과 호환되는 PDF/X-1a이다. PDF/X-1a는 포스트스크립트 레벨 2 수준의 호환성을 유지하고 있어 RIP에서의 처리를 위해 다시 포스트스크립트 파일로 변환하여도 큰 문제가 없다.

하지만 MS-PowerPoint나 Adobe Illustrator 8 이후의 버전에서 구현할 수 있는 기능 중, 투명 효과(transparency effect), 3 층 이상의 다중 레이어(multi-layer), 복잡한 계조 효과(rainbow gradient effect) 등과 같은, 기존의 포스트스크립트 규격에는 없는 리소스를 처리하는 과정에서 많은 문제를 야기하였다.

특히 투명 효과는 대표적인 예로, 기존의 CPSI 기반의 RIP 서버에서는 둘 이상의 오브젝트나 리소스가 겹친 부분의 투명 효과를 처리하기 위해 겹친 부분을 평면화(flatten)시킨 다음, 별도의 오브젝트로 분리하고 트래핑 값을 0으로 설정하여 기존의 오브젝트와 결합(combine)하는 방법을 취하고 있다.(Fuji 사의 Celebrant RIP, Kodak 사의 Brisque RIP 등) 하지만 이 방법도 아웃라인 속성의 오브젝트로만 구성되어 있을 때에만 제대로 구현했을 뿐, 비트맵 이미지 리소스가 투명 효과 영역에 포함되어 있을

때에는 결과를 예측할 수 없다. 일례로 Haloquin RIP에서는 해당 영역을 포함하는 흰 사각 영역으로 나타나도록 처리하며 Heidelberg 사의 Meta Dimension RIP에서는 해당 페이지 전체에 걸쳐 투명 효과를 적용하여 뿌옇게 보이도록 처리한다.

이와 같은 문제는 PDF의 확장된 기능을 포스트스크립트 규격에 맞도록 강제하는 과정에서 각 업체마다 처리하는 루틴(routine)이 다르기 때문에 일어나는 현상이다. 즉, 기본적인 RIP 엔진은 어도비 사의 CPSI 모듈을 사용하지만(Haloquin RIP은 예외) PDF 규격을 처리하기 위한 확장 기능 부분과 이의 처리 방법은 제품마다 다르다.

또한, 어도비사는 SDK(Software Develope Kit)를 통해 PDF 생성 툴에 대한 정보를 공개하여 상당히 많은 수의 서드-파티(Third-Party) 제품들을 확보하는 정책을 취하였다. 이로 인하여 사용자가 작성한 원시 데이터를 어떠한 PDF 생성기(generator)에서 생성하였느냐에 따라 RIP 엔진에서의 처리가 원활하거나 오류를 발생하는 상황도 발생하고 있다.

일반적인 서드-파티 제품에서는 PDF 파일의 데이터를 최소화하기 위해 페이지 레이아웃에 대한 여백 부분 삭제, 중첩 리소스에 대한 상위 레이어 우선 저장 등의 방법을 사용한다. 이러한 처리는 터잡기 작업 시의 여백 영역 확보(bleeding) 불가, 트래핑 옵션의 변경 불가, 서체 임베딩 정보 오류로 인한 글자 변경 불가능 등의 문제가 발생한다.

2007년에 이르러서야 어도비사는 Fig. 2와 같이 기존의 CPSI 모듈과는 별개로 PDF를 처리하기 위한 PDF Print Engine이라는 모듈을 보급하기 시작하였다. Fig. 3과 같이 PDF Print Engine은 개별 RIP 프로세스에서는 클라이언트 PC로부터 전송되는 데이터가 PDF 파일일 때 이를 포스트스크립트로 변환하지 않고 곧바로 래스터 작업을 수행한다. 즉, 포스트스크립트 파일일 때에는 기존의 CPSI 모듈이, PDF 파일일 때에는 PDF Print Engine이 동작한다.

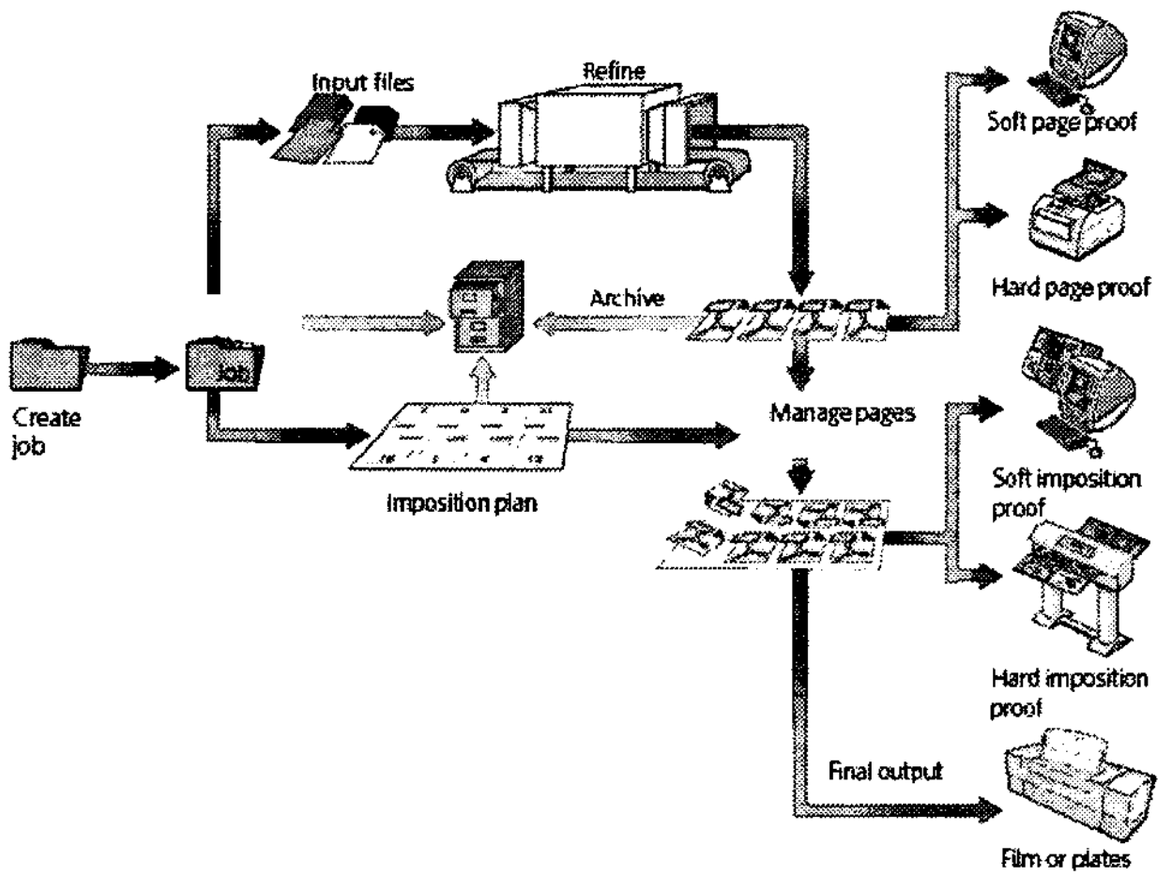

Fig. 2. The work flow of RIP server in PDF Print Engine.

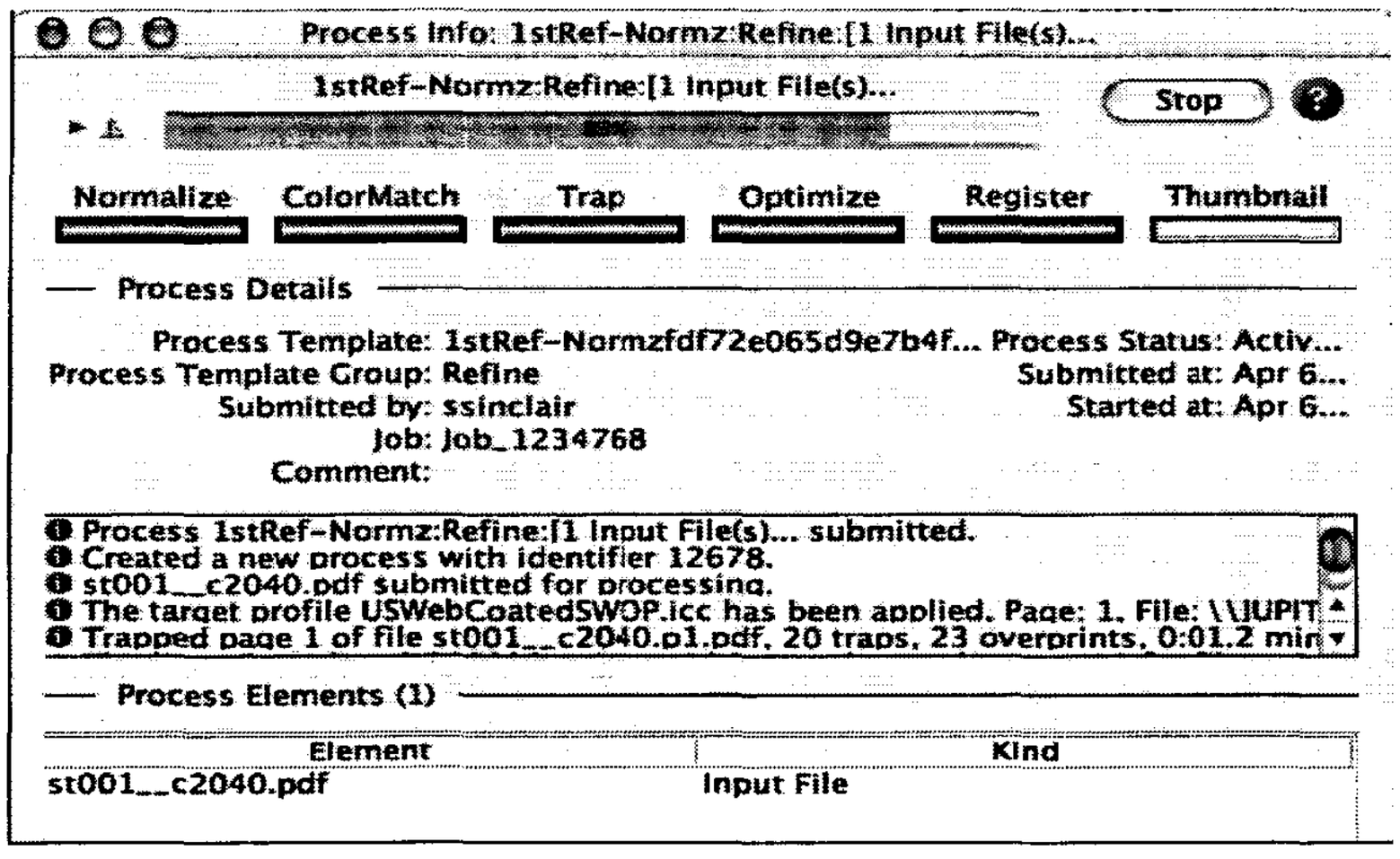

Fig. 3. The process monitering working.

PDF Print Engine을 통해 작성한 PDF 파일은 단순히 파일 내에 포함되어 있는 각종 리소스를 인쇄 공정에 적합하도록 평면화하여 분판(separation)하는데 그치지 않고 JDF 정보를 PDF 파일 내에 포함하여 인쇄를 수행하는 각 공정을 원활하게 진행하기 위한 정보를 포함하고 있다. 이는 PDF Print Engine의 성능을 효과적으로 사용하기 위해서는 인쇄 공정 전체에 걸쳐 CIP4에 기초한 통합 워크플로가 구축되어야 함을 의미한다.

3. 결 론

인쇄산업에서의 PDF 이용은 많은 장점을 갖는다. 최종 인쇄 결과를 미리 시뮬레이션하기 위한 도구로써의 사용만으로도 기존 공정에 비해 비용과 시간 면에서도 큰 이점이 있다. 또한 Acrobat을 통한 작업자 간 의견 소통의 통로로 이용하여도 인쇄 공정을 훨씬 매끄럽게 진행할 수 있다.

특히 포스트스크립트 기반에서는 구현하기 어려웠던 OSMU(One Source Multi-Use) 정책을 간단히 구현할 수 있다는 점은 매우 큰 장점이다. 출력 장비(prepress device)에 따라 생성해 놓은 포스트스크립트 파일을 재구성(re-RIP)하여야 하는 데 반해 PDF는 출력 방향을 바꾸는(re-direct) 것으로 작업을 완료할 수 있기 때문이다.

하지만 인쇄산업 현장에서는 PDF를 작업의 핵심 데이터로 인식하기보다는 포스트스크립트의 보조 포맷 정도로 인식하는 것 또한 사실이다. 이러한 상황은 국내의 낙후된 서체 사용 환경과 소프트웨어 저작권에 대한 인식 미비 등과 무관하지 않다. 특히 국내

의 서체 사용 환경의 후진성은 인쇄 산업의 발전을 가로막는 큰 장애가 되고 있으므로 이러한 문제점을 해결하기 위한 많은 노력과 연구가 필요할 것이라 사료된다.

이미 CTP 보급이 포화 상태에 이른 인쇄 선진국에서의 인쇄 공정은 DTP 단계에서부터 제품 출하 단계까지 디지털 공정에 의거한 워크플로를 구축하여 높은 생산성을 담보하고 있다. 그에 반해, 국내는 아직까지 미진한 CTP 보급률과 더불어 인쇄 공정 전체를 포괄할 수 있는 공정 관리 시스템의 보급은 미미한 수준이지만, 필요성에 따라 점점 증가할 것이라 예측하고 있다.

2004년 드루파 전시 이후 출시되고 있는 신제품의 경향은 기존의 포스트스크립트 중심의 프로세스에서 PDF 기반의 통합 공정 관리 워크플로로 변화하고 있으며, 이후 출시되고 있는 인쇄 관련 장비들 또한 개별 장비로써의 기능에 부가하여 공정을 유지하기 위한 모듈로써의 기능을 강화하고 있다.

참 고 문 헌

1) 오성상 외 2명, 화상제판, 성안당, pp. 274~325 (2001).
2) Adobe System, Inc., "Portable Document Format(PDF) Reference Manual, Ver.1.2, November 27 (1996).
3) 오세웅 외 2명, Digital Graphic Arts, BGI, pp. 108~109 (2006).
4) Patrice M. Dunn, "Open Process Integration Data Formats & Digital Workflow Issues", TAGA, pp. 272~286 (1997).
5) Kipphan, H., :Digital Multicolor Printing and Computer to Technologies; Evolution or Revolution in the Graphic Arts Industry", TAGA Proceedings, Vol. 1, pp. 635~654 (1995).
6) Kurt Schlapfer and Erwin Widmer, "Specification for Prepress Digital Data Exchange", TAGA, pp. 145~161 (1998).
7) Helmut Kipphan, Handbook of Print Media, Springer, Berlin, pp. 923~936 (2001).
8) Robert R. Buckly, "A Framework for Digital Data Workflow in a Graphic Arts System" TAGA, pp. 337~348 (1997).