特輯

전자출판의 발전상 및 해결 과제

李 壽 淵
光云大學校 電子計算機工學科 敎授

I. 서 론

최근들어 레이저 프린터, 고해상도 디스플레이, 포인팅 디바이스, 윈도우 환경, 고성능 워크스테이션의 기술발전과 보급에 힘입어 이러한 환경을 이용하여 문서를 편리하게 작성하고 고품질의 문서를 출력할 수 있는 전자출판 시스템에 대한 요구가 급증하고 있다.

전자출판 시스템은 기존의 문서처리 단계 즉, 문서의 입력, 편집, 교정과 출력의 각 단계가 분리되어 처리되던 것과는 달리 이들 각 단계를 관리하고 결과를 통합하여 완성된 문서를 종이 또는 화면에 출력하는 메커니즘을 말하며, 다른 시스템과의 문서정보 교환을 포함하고 있다.

한편 전자문서는 이와같이 출력 매체상에 하드카피된 정적인 문서를 말하기도 하지만 문서가 참조되기만 할 때 참조되는 대상 페이지를 디스플레이상에 동적으로 표현하는 것의 의미를 동시에 가진다. 텍스트를 중심으로 처리되어 오던 이러한 전자문서는 멀티미디어 기술의 발전으로 이미지, 오디오, 동화상등의 여러 메디아 정보를 문서내에 포함하는 멀티미디어/하이퍼미디어 문서로 발전하고 있으며 이들 멀티미디어 객체들을 통합된 문서처리환경에서 표현하고 처리, 전송하기 위한 연구가 활발히 진행되고 있다.

본고에서는 전자출판 시스템의 발전을 토대로한 시스템들의 특징과 WYSIWYG 환경에서 전자출판 시스템의 핵심적 구성요소 및 처리방식을 소개하며, 문서교환을 위한 국제표준화 활동, 멀티미디어 문서를 논하고 차후 문서처리 시스템에서 해결하여야 할 문제에 대해서 기술한다.

II. 문서처리 방식의 발전

초창기에 개발된 문서처리 시스템들은 사용할 수 있는 장치들이 펀치카드, 타이프라이터 수준의 장치로 거의 한정되었고 대부분의 포맷팅 기능들은 기계어 수준의 명령들로 제공되었으며 문서의 페이지, 라인에 관련된 객체만을 처리하고 몇몇 명령들만이 단어, 문장, 단락과 같은 문서의 논리적인 내용을 다룰 수 있었다. 대표적인 시스템으로는 RUNOFF와 FORMAT 시스템을 들 수 있는데 이들은 포맷팅 명령을 지정하는 양식에서 크게 구별되었고 이러한 방식은 다음 세대의 시스템에 영향을 주었다.

1964년 MIT에서 개발된 RUNOFF 시스템은 TYPSET라는 분리된 에디터를 사용하여 텍스트와 포맷팅 명령들로 표현되는 문서를 작성하였고 포맷터의 기능은 문서의 물리적인 형태(format)를 직접 다루는 18개의 포맷팅 명령을 갖는 정도로 한정되었으며 문서의 논리적인 내용은 다루지 않았다.

포맷팅 명령을 텍스트로 부터 구분하기 위하여 명령모드와 텍스트 모드의 두가지의 입력 모드를 위한 구분되는 라인를 두었는데, 명령을 입력하기 위하여 '.'으로 시작하는 특별한 명령 라인을 갖음으로서 텍스트 라인과 구분하였다. 그러나 이 시스템의 비융통성, 한정된 기능, 출력되는 페이지의 객체를 직접 제어하기 위한 포맷팅 명령들을 갖는 시스템의 단점에도 불구하고 텍스트와 명령 라인의 분리된 입력의 두 모드를 두어 명령과 텍스트를 구분을 시키는 방식은 그 후의 다른 시스템들에 크게 영향을 주었다.

1960년대 후반에 발표된 FORMAT은 텍스트 에디터

가 제공되었지만 에디팅 기능은 논리적으로 구분되어 전형적인 batch 처리 환경을 갖었다. 포맷팅 명령들은 실제 출력될 페이지의 물리적 객체를 직접 조작하는 명령들이었고 character, phrase, paragraph level의 3가지 형식의 명령들로 구성되었다. 포맷팅 명령은 텍스트 속에 지정되는(embedding)방식으로 이들을 구분하기 위해서 미리 예약된 escape 문자를 사용하였는데 이 방식은 그 후의 시스템에서 일반화되었다. 또한 단락 명령(paragraph commands)과 같이 논리적 객체를 다룰 수 있도록 한 점과 writer's workbench 특징을 둔 것등은(예를들면 문서에서 사용되어진 단어들을 알파벳 순으로 목록을 만드는 것)이 시스템이 다른 시스템에 중요한 영향을 준 요소이다.

1960년대 말에서 1970년대 초에는 제 1세대의 구조화된 포맷터가 개발되어 졌는데 이들 시스템들은 RUNOFF, FORMAT과 같은 포맷터에서 명령의 지정방식과 writer's workbench 특징에 강하게 영향을 받아 텍스트 속에 low-level 명령을 지정하는 방식은 아직까지 남아 있지만 매크로 정의기능, 컴퓨터 언어에서 block structuring 개념등의 도입등을 통해 고수준의 명령을 정의할 수 있게 하고 이것을 입력 문서에 적용하여 문서를 구조화 시키는 것이 가능하게 되었다. Writer's workbench 특징은 더욱 강화되어 문서를 더 편리하게 작성할 수 있도록 하였는데 예를들면 절(section)에 자동으로 번호를 붙이고 목차와 색인이 포맷팅 과정동안 만들어지고 각주는 적절히 배치되고 번호가 붙여진다. 이러한 범주에 속하는 대표적인 시스템으로는 PUB와 순수 NROFF시스템을 들 수 있다.

PUB는 1971년 스텐포드대학에서 개발이 시작 되었는데 명령 형식은 RUNOFF의 영향을 많이 받았다. PUB의 명령들은 FORMAT과 똑같은 종류의 저수준의 명령을 다루었지만 column, 각주, 절등과 같은 고수준의 객체도 제공되었다. 절은 자동으로 번호가 붙여질 수 있었고 목차를 생성할 수도 있었다. 특히 ALGOL언어의 block structuring 개념을 도입하여 입력되는 문서의 부분들은 BEGIN/END로 묶여지는 블럭들로 그룹되어 질 수 있으며 그 블럭안에서 매크로와 변수들이 정의되고 그 값의 범위는 블럭 안에서만 유효하다. PUB의 중요한 공헌은 포맷팅 명령에 프로그래밍 언어의 구조 개념을 도입함으로서 포맷팅 명령의 확장할 수 있는 가능성을 제시해 주었으며 광범위하게 강력한 writer's workbench 툴을 개발함으로서 사용자가 문서를 편리하게 작성할 수 있도록 한 점이다.

순수 NROFF시스템은 1970년대 중반 초에 Bell연구소에서 개발되었다. 포맷팅 명령의 기능 정도와 프로그래밍 언어 특징을 가지는 것은 이전 시스템으로 부터 영향을 받은 것이며 실제로 이 시스템의 원조는 RUNOFF라 할 수 있다. 특히 PUB의 블록 개념과 비슷한 스위칭 환경(switch environment)개념을 두었는데 약간의 차이가 있다. 스윗칭 환경은 문서내에 설정되는 파라미터의 영역을 정의하기 위한 것으로 새로운 파라미터를 갖는 환경을 정의하기 위해서 push down 방식으로 스윗칭이 가능하며 이전에 적용된 환경변수들은 복구(restore)될 수 있다.

포맷팅 명령은 두 가지 방식을 지원하였는데 그 중 하나는 분리된 라인에 '.'으로 시작하는 명령을 두므로서 다른 텍스트 라인과 구별시키는 방식이고, 다른 하나는 '\'로 시작하는 escape character를 텍스트 속에 두어 시스템에서 특정한 용도로 사용하기 위한 명령을 갖는 방식이다.

1970년대 후반에 개발된 UNIX 문서작성 툴[2], TeX[3], Scribe[4]와 같은 대표적인 시스템들의 두드러진 특징은 문서내에 복잡한 수학식, 테이블, line-drawing등의 객체를 포함하고 장치 독립적인 특성을 갖는 문서의 입출력 표현 및 처리에 노력을 기울였는데, 이는 현재 사용되고 있는 많은 포맷터들에 영향을 주고 있으며 TEX와 UNIX 문서 작성 툴등은 그 후로 기능이 보강되어 현재에도 사용되고 있다.

TEX는 1970년대 후반에 스텐포드대학의 D. Knuth에 의해서 개발된 시스템으로 특히 복잡한 수학식을 처리하기 위한 강력한 포맷터를 가진다.

포맷팅 명령은 복잡한 텍스트, 테이블, 수학식등의 폭넓은 범위의 객체 표현을 위한 융통성에 중점을 두었다. 그러나 사용자는 원하는 문서를 얻기 위하여 낮은 단계의 포맷팅 명령을 사용하여 직접 출력될 페이지 위의 객체를 지정해야 하는 복잡성을 가지는데, 즉 사용자가 출력될 문서의 모습에 책임을 갖는 부담을 준다. 시스템의 사용자 인터페이스를 보강하기 위해서 매크로 패키지, 프리프로세서, 선언적 명령형식과 같은 개발이 기대되어졌는데, 현재 사용되고 있는 LaTex 시스템은 Tex의 이러한 문제를 보강하여 사용자가 문서를 쉽게 작성할 수 있게 하고 있다.

Scribe는 1970년대 후반 카네기 멜론대학의 B. Reid에 의해서 개발된 시스템으로 현재 사용되고 있는 많은 시스템들에 영향을 주고 있다. 이 시스템이 강조하고 있는 것은 문서의 내용을 문서 포맷으로 부터 강하게 분리하는 것으로, 사용자는 문서의 논리적 내용과 관계되는 명령을 사용함으로서 실제적으로 출력될 문서의

모습에 책임을 지는 것으로 부터 떠나게 해준다.

특정 출력 디바이스 및 문서 양식등에 관계되는 낮은 수준의 명령들은 시스템으로 부터 결정되어지며 사용자는 문서의 논리적 구조만을 지정하도록 하므로서, 똑같은 문서 표현이 다른 컴퓨터에서도 사용되어 질 수 있으며 다른 출력 장치에도 적용되어 질 수 있다.

그러나 Scribe 시스템의 이러한 철학으로 말미암아 다룰 수 있는 객체의 형식은 실제적인 페이지 위에 위치시키기 위한 명령없이 객체의 내용에 의해서 완전히 기술되어 질 수 있는 것으로만 한정되어 졌다. 특히 텍스트 객체로만 제한되어 졌고 선도형, 테이블, 복잡한 수학식등을 표현하는 기능을 갖지 않았다.

UNIX 환경에서 문서처리 시스템은 여러개의 독립된 문서 포맷팅 툴들로 구성되어 있으며 서로 연결되어 다양한 문서를 만들 수 있도록 하는 "빌딩 블록" 접근 방식으로 설계되어 있다. 이러한 접근방식은 UNIX 프로그래밍 환경에서 강하게 영향받은 것으로 간단하면서 강력한 기능을 갖는 여러개의 툴들을 지원하므로서 각각은 독립적으로 간단한 기능을 수행하며 'pipe'와 같은 UNIX의 기본 메커니즘을 통해 쉽게 상호 연결될 수 있는데, 이는 각 툴의 개발을 쉽게하고 그 툴에 대한 수정이 요구될 때 쉽게 변경할 수 있으며 새로운 사용자 요구사항이 생기면 원래의 프로그램을 변경하여 기능과 복잡성을 증가시키기 보다는 이미 존재하는 프로그램을 이용하면서 새로운 프로그램을 만들고자 하는 철학이다. 이러한 철학이 포맷팅 시스템의 구조에도 강력하게 반영되어졌는데 기본적으로 NROFF/TROFF 포맷터를 갖으며 여기에 여러가지의 매크로 패키지, 프리프로세서, 포스트 프로세서등이 지원된다. UNIX 문서처리 시스템은 포맷터에 의해서 기본적으로 제공되는 낮은 단계의 포맷팅 명령보다는 고수준으로 명령을 사용할 수 있게 하는 ms 매크로 패키지 및 수학식의 포맷팅을 위한 EQN, 테이블 포맷팅을 위한 TBL, 참고문헌 작성을 위한 REFER, 선도형을 위한 PIC등의 프리프로세서로 구성된다.

그러나 여러개의 작은 모듈로 구성된 이 시스템은 서로 다른 객체들 사이에 또한 이들 객체를 기술하기 위한 문서기술언어 사이에서의 통합이 힘들다는 단점을 가진다.

GML(generalized markup language)은 IBM의 C.F. Goldfard에 의해서 1970년대 중반 초 부터 개발되어 1978년에 일반적으로 사용되기 시작하였으며, RUNOFF와 비슷한 SCRIPT라는 포맷터를 위한 매크로를 이용하여 구현된 포맷터이다. GML은 태그라 불리는 고급 선언 명세(high level declarative specification)와 많은 writer's workbench 특징을 제공하는데 예를들면 장, 절, 목록(list elements), 각주(footnote)등의 자동 번호 매김, 목차(table of contents)와 색인(index)에 포함되는 포맷팅 정보의 관리기능등을 가지고 있다.

1. 대화형 문서처리 시스템

1970년 중반에 런던대학에서 개발된 QUIDS는 에디팅과 포맷팅이 결합된 환경을 제공하며 사용자는 세개의 그룹으로 구분되는 명령들을 사용하여 문서를 작성할 수 있다.

'>' mode에서는 기존의 문서를 편집하거나 출력시키는 등의 명령을 입력한다. '*' mode에서는 문서의 포맷팅을 제어하는 파라미터를 지정하기 위한 명령을 입력시킬 수 있는데, 예를들면 제목이 각 출력 페이지에 프린트되게 할것인가 아닌가를 선택하는 명령을 입력시킨다. ',' mode에서 사용자는 단락 또는 절과 같은 논리적 객체를 선택하는 명령을 입력하고 그 객체와 관련된 텍스트를 입력하며 저수준의 포맷팅 파라미터를 입력한다.

이 시스템에서 사용자는 최종 출력되는 문서표현을 직접 조작(direct manipulation)할 수 없지만 문서의 논리적 표현을 변경할 수 있고 사용자의 요청시 최종문서의 모습을 보여줄 수 있다.

1973년에 개발된 Xerox Alto 워크스테이션은 마우스와 같은 포인팅 디바이스와 8.5 × 11inch 비트맵 디스플레이를 포함하고 있었는데 이러한 환경 위에서 많은 대화형 문서처리 시스템들의 개발이 가능하게 되었다. 그중 하나인 Bravo 시스템은 에디팅, 포맷팅, 이미징 과정이 하나로 결합되어 사용자는 마우스를 사용하여 문서에 표현되는 객체를 직접 조작하고 그 결과는 포맷되어 즉시 디스플레이에 표시된다. 이 시스템에서 다룰 수 있는 객체는 문자, 단어, 라인, 단락과 같은 비교적 간단한 텍스트 정도로 한정되어 졌지만 Markup, Draw와 같은 drawing 패키지에서 작성되는 그림을 문서내에 merge 시키는 것이 가능하였다.

1981년 Xerox연구소에서 발표된 Star 워크스테이션에서의 문서처리 시스템은 상품화를 위한 실험적 토대인 Alto, Bravo, Markup, Draw등의 개발경험과 개선을 통하여 개발된 사무용 문서시스템으로 문서의 편집 및 보관기능과 전자우편, 데이타 처리기능등이 통합된 환경을 제공하였다.

특히 이 시스템은 텍스트는 물론 수학식, 선도형을 직접 조작하여 그 처리된 결과를 즉시 보여주고 여러개

의 overlapped 윈도우를 제공하며 시스템에서 처리되는 모든 객체를 아이콘 또는 pictorial 심볼로 표시하는등 사용자 인터페이스에 많은 노력을 기울였다.

1980년대 중반부터 PC 및 워크스테이션을 토대로한 상용화된 WYSIWYG(what you see is what you get) 시스템들이 본격적으로 발표되기 시작했는데 이들 시스템은 기본적으로 마우스와 같은 포인팅 디바이스, 고해상도 비트맵 디스플레이를 가지며 여기에 메뉴, 아이콘, 윈도우와 같은 그래픽 사용자 인터페이스 환경을 제공하여 실질적으로 대화적 환경에서 문서를 작성할 수 있게 하고, 출력장치로는 300dpi 이상의 고성능 레이저 프린터를 사용하여 고품질의 문서 출력이 가능하게 하고 있다. 상업화에 성공한 대표적인 시스템으로는 FrameMaker[5], Interleaf[6], Ventura[7], PageMaker 등을 예로 들 수 있다.

이들 시스템들은 사용자 위주의 편리한 사용자 환경 위에 강력한 텍스트 포맷팅 기능과 섬세한 그래픽, 이미지 편집 기능등을 하나로 결합하고 있고, 여러가지의 객체를 포함하고 있는 문서의 입력 및 편집을 디스플레이 상에서 직접 조작하여 그 결과를 실제 출력될 모습과 비슷하게 보여준다. 한편 이들 시스템은 여러가지의 객체를 편집하는 에디터들(텍스트, 그래픽, 테이블, 수학식, 스프레드 쉬트등의 에디터)의 결합정도에 따라 크게 두 가지 방식으로 구분된다. 그 하나는 DesignStudio, PageMaker, Ventura와 같은 시스템으로 다른 시스템에서 작성된 여러 형식의 텍스트 화일 및 그래픽, 이미지 화일들을 import하는 강력한 기능을 기본 전략으로 하고 있으며, 자체에 가지고 있는 페이지 레이아웃 에디터를 이용하여 텍스트 블럭을 설정하거나 그래픽 영역을 설정한 다음 import된 객체를 배치시키고 포맷팅을 행한다.

다른 하나는 FrameMaker, Interleaf와 같은 시스템으로 여러 복합문서 에디터들이 한 시스템속에 강력하게 결합되어 있어 기술 보고서와 같이 수학식, 복잡한 추상적인 도형들로 구성되는 많은 분량의 문서를 쉽게 작성할 수 있다.

그러나 이들 WYSIWYG 시스템들은 시스템들 사이에 약간의 차이는 있지만 대부분 고품질의 폰트를 제공하고 강력한 레이아웃 설정 기능 및 포맷팅 처리능력을 가지며 문서 작성자를 위한 편리한 작업환경을 제공하며 네트워크 기능 및 문서 전송기능등을 포함하는 고도의 출판기능을 갖고 있다.

국내에서는 1980년대 후반부터 WYSIWYG 시스템들이 개발되기 시작하였는데 휴먼 컴퓨터의 문방사우, 서울시스템의 페이지매니저, 쌍용컴퓨터의 세종 퍼블리쉬, 신명시스템즈의 신명전자출판시스템, 금성소프트웨어의 하나 퍼블리쉬등이 있다.

2. 문서처리모델

대부분의 문서작성 시스템은 그림 1과 같이 크게 에디팅, 포맷팅, 이미징 단계로 구성된다.

에디팅은 새로운 문서를 만들거나 이미 작성된 문서를 수정하기 위해 텍스트 입력, 삭제등의 동작들로 이루어지며 문서의 표현을 구조화시키는 처리, 예를들면 장, 절, 단락, 각주와 같은 문서의 논리구조 속성을 작성된 텍스트에 설정하고 이들 논리 객체(logical object)간의 관계를 정의할 수 있도록 하는 과정이다. 또한 다음 단계의 포맷팅을 위해 텍스트속에 포맷팅 명령을 지정하는 문서기술(document description) 작업도 에디팅 과정속에 포함된다.

포맷팅은 에디팅과정에서 텍스트와 포맷팅 명령이 혼합되어 작성된 논리적인 문서를 하나 또는 그 이상의 이차원적 페이지 공간에 배치시키는 처리로 정의된다. 이 논리적인 문서에 표현된 텍스트, 그림, 테이블, 수학식등과 같은 논리적인 객체들을 페이지에 배치시키기 위해서 레이아웃 처리를 행하며, 특정 문자 또는 단락에 지정된 특정 폰트 스타일 또는 크기에 대한 처리 및 단락 내에서의 수평/수직 정렬 즉, 인덴트, 탭, 정렬(left/right/center/justify), 라인 간격등의 처리를 행하여 논리적 문서를 라인들, 페이지들로 분해한다. 포맷팅 결과는 하드카피 또는 소프트 카피 장치에 출력될 수 있도록 표현되어야 하는데 특정 출력장치에 한정된 형태의 명령 또는 장치 독립적인 명령들의 형태로 구성되는 포맷된 문서를 출력한다.

이미징 과정은 포맷된 문서를 디스플레이 또는 프린터 장치에 출력하기 위한 변환과정으로 실질적으로 종이나 디스플레이로 완성된 문서를 얻을 수 있게 한다.

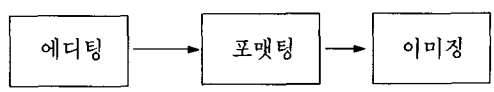


그림 1. 문서 처리 모델

Ⅲ. 전자출판 시스템의 구성요소

1. 그래픽 사용자 인터페이스

마우스와 같은 포인팅 디바이스 및 비트맵 디스플레이 장치와 윈도잉 시스템, 메뉴, 아이콘, 메세지 상자와 같은 그래픽 환경은 편리한 사용자 인터페이스를 제공한다. 또한 이러한 환경은 대화적인 응용프로그램의 개발을 쉽게 하고 표준적인 인터페이스 개발을 가능하게 한다.

그래픽 사용자 인터페이스(graphical user interface) 시스템은 크게 세가지로 구성된다. 윈도윙 시스템, imaging 모델, 응용프로그램 인터페이스(application programming interface)가 이에 해당한다. 윈도윙 시스템은 윈도우, 메뉴, 대화상자등 스크린에 나타나는 객체들을 만들기 위한 프로그래밍 도구와 명령어의 집합체로 구성된다. 윈도우를 만들고, 크기를 조절하며, 스크린상에서 이동하는 방법을 제공한다.이에 해당하는 가장 대표적인 예는 MIT의 X window 시스템이다. 모든 X window GUI들이 같은 윈도윙 시스템을 쓰기 때문에 응용 프로그램을 개발하기 위한 프로그램 도구들을 공유하는 것이 가능하다. Imaging 모델은 어떻게 폰트와 그래픽들이 화면에 만들어지는가를 정의한다. 워드프로세서나 탁상출판시스템의 텍스트의 크기나 모양 및 직선이나 곡선등은 imaging 모델에 의해서 정의된다. 가장 잘알려진 imaging 시스템은 PostScript이며 Macintosh의 QuickDraw, OS/2의 Presentation Manager (PM)는 GPI(graphical programming interface)라는 imaging 모델을 가진다. 또한 몇몇 GUI들은 하나이상의 imaging 모델을 제공하기도 하는데 Sun의 NeWS는 PostScript Imaging 모델을 제공하나 PHIGS나 GKS와 같은 imaging 모델을 제공하기도 한다. API는 어떻게 프로그래머가 윈도우, 메뉴, 스크롤바, 아이콘등을 스크린상에 나타나게 해주는가에 관련된 프로그래밍 언어 function call이다. OpenLook은 Sun의 새로운 API이며 DEC윈도우는 XUI(X user interface)라 불리는 API를 가진다.

Motif, OpenWindow와 같은 그래픽 사용자 인터페이스 시스템은 영어문화권에 맞게 정의된 Style Guide라는 지침서가 정해져 있으므로, 우리 환경에 맞는 guideline의 연구가 요구된다.

2. Font

폰트는 표현방식에 따라 크게 세가지로 분류된다. 글자를 그림 또는 화상으로 간주하여 화소단위로 표현하는 비트맵 폰트(bitmap), 글자의 윤곽선을 기하학적으로 표현하여 수학적인 직선이나 곡선식으로 구성하는 윤곽선 폰트(outline font), 글자의 구조를 획의 위상으로 표현하는 구조적 폰트(structual font)가 있다[12].

비트맵 폰트는 대부분 도트 매트릭스 프린터 및 화면용 폰트로 많이 사용되고 있다. 이 방식은 각 글자를 구성하는 점(pixel)정보를 직접 출력장치에 매핑 시키므로 속도가 빠르지만, 각 글자는 사각형안에서 점행렬로 표현되기 때문에 출력장치의 해상도에 의존성을 가지며 확대, 축소, 회전 변환 처리등에 약하고, 한 서체에 대하여 여러가지의 크기, 굵기등 속성변형을 갖는 모든 글자에 해당하는 정보를 기억해야 하는 제약을 갖는다.

윤곽선 폰트는 글자의 윤곽선을 여러 부분으로 나누어 각각 직선, 원호 또는 자유곡선으로 표현하는 방식으로 자유곡선으로는 Bezier 곡선, 3차 운형(cubic spline)곡선 또는 B운형(B spline)곡선등의 수학적인 표현이 이용된다. 이 방식은 폰트의 제작이 매우 수월하다는 장점을 가지며 글자의 크기, 폭, 기울임등의 변환처리에도 쉽게 적응될 수 있는데 어떤 한 글자에 대해 글자의 윤곽선에 대한 직선과 곡선부분의 좌표값만을 기억하고 변형할때 이들 좌표값에 대한 변형 행렬식만 계산하면 된다.

구조적 폰트의 대표적인 예로는 D.E. Knuth가 개발한 Metafont시스템을 들 수 있다. 이 시스템에서는 원형이나 타원형등 여러가지 모양의 펜을 사용하여 글자의 중심선을 따라 펜의 움직임을 정의함으로써 글자를 설계한다. 이 방식은 폰트의 설계가 아주 용이하며 적은 양의 기억장소를 차지하며 펜의 교체 또는 크기의 변경과 움직임을 나타내는 좌표값을 변경함으로써 크기 변화부터 획의 변화까지 글자꼴의 모든 변화에 적용가능하다. 그러나 출력된 글자를 생성하는데 소요되는 계산량이 윤곽선 폰트보다 훨씬 많이 걸리기 때문에 프린터 출력용 폰트보다는 설계 단계에서 많이 이용된다.

한편 윤곽선 폰트는 디스프레이와 같이 낮은 해상도를 가지는 장치에서 어느 수준이하의 크기를 가질 때는 알아보기 힘들다. 이러한 단점을 보완하기 위해서 현재 많은 폰트제작 업체들은 600dpi 이하의 낮은 해상도를 가지는 장치에서도 좋은 출력 결과를 보장하기 위해서 hinted outline 폰트를 제공한다. 따라서 프린터 뿐만 아니라 스크린상에서도 같은 폰트가 사용되기 때문에 진정한 WYSIWYG 시스템이 가능하게 된다. Adobe의 Type 1 format, Sun의 F3 format등은 이러한 hinting 정보를 포함하고 있다.

3. PDL(Page Description Language)

페이지 기술언어는 포맷팅을 거쳐 출력된 최종 문서

표현을 레이저 프린터와 같은 hardcopy 장치에 출력하기 위해 기술되는 언어이다. 완성된 문서내에는 문자와 그래픽, 이미지와 같은 객체를 포함하고 있으며 PDL은 이들 각 객체들을 실질적인 페이지에 표현하기 위한 위치, 방향, 크기, 이동, 회전, 확대, 축소등의 동작을 다룰 수 있는 operator들로 구성된다. 또한 장치 독립적인 특성을 갖는 사용자 좌표계와 물리적인 장치의 시스템 좌표계로 구성된다.

현재 대표적인 PDL로는 Adobe의 PostScript, Xerox의 Interpress, Imagen 의 DDL(document description language)이 있다. 특히 Adobe의 PostScript[10]는 내부 사양이 공개되어 있어 응용프로그램의 개발이 용이하며, Sun 워크스테이션의 X11/News 윈도우 시스템의 이미징 모델로 사용하고 있고 IBM, Apple, HP사등에서 계속 채택해서 사용하고 있는 상황이다.

한편 하드카피 장치로 사용되고 있는 프린터는 충격형 프린터(impact printer) 및 도트 매트릭스 프린터와 같은 저수준의 프린터로부터 300dpi 이상의 레이저 프린터, CTS(computer typography system) 용등의 고품질의 문서 출력을 가능하게 하는 프린터에 이르기까지 매우 다양한 기능을 갖는 프린터가 존재하고 있다. 이러한 여러 업체의 페이저 기술언어와 레이저프린터등이 등장함에 따라 보다 포괄적인 사무 문서처리 환경을 조성하고 정보교환 분야의 표준화를 위한 노력으로 ISO에서는 표준 프린터 제어 포맷인 SPDL(standard page description language)을 제정하게 되었고, Adobe의 PostScript와 Imagen사의 DDL은 SPDL의 제정을 위한 모델로 검토중이다.

4. 문서구조

문서는 논리적인 구조(logical structure)와 배치적인 구조(layout structure)로 구성된다[8]. 논리구조는 장(chapter)과 절(section), 단락과 같은 기본 논리객체로 정의되고 제목, 목차, 색인 같은 이들 논리 객체간의 관계를 정의한 것으로 계층구조로 표현된다.

배치구조(layout structure)는 문서의 내용을 물리적인 매체 즉, 종이나 화면에 배치시키기 위한 구조이며 페이지 세트, 페이지, 프레임, 블록 및 그 내용으로 구성된다. 페이지 세트는 페이지의 집합을 식별하기 위한 것이고 페이지는 문서 내용부의 위치 및 표현형식을 정의하기 위한 직사각형 영역이다. 프레임은 페이지내의 레이아웃 경계를 결정하고 논리 레이아웃 관련 규칙을 제어하는데 기본이 된다. 블록은 문서 내용부를 포함하는 기본 단위이다.

Ⅳ. WYSIWYG 전자출판시스템의 처리방식

WYSIWYG(what you see is what you get)이라는 용어는 직접 조작(direct manipulation)방식과는 다르게 사용된다. WYSIWYG은 디스플레이상에 표현된 내용과 최종 hardcopy된 내용과의 거의 비슷하다는 개념으로 사용된다. 직접조작 방식은 사용자 인터페이스를 모델화하는 더 일반적인 개념을 갖는다. 직접조작 방식을 갖는 문서처리 시스템은 디스플레이된 내용과 최종 hardcopy된 내용사이에 WYSIWYG 관계가 아닐 수도 있다.

1) 에디팅

일반적으로 텍스트 및 그래픽 에디팅을 포함하며 이 밖에도 테이블, 수학식, 비지니스 그래픽, 음성, 화상, 애니메이션등과 같은 다양한 여러 객체의 에디팅을 포함한다. 이들 객체들은 서로 혼합되어 문서내에 표현될 수 있으며 이러한 문서를 복합문서(compound document)라 부른다. 복합문서를 다루는 이들 에디터를 결합하는가 아니면 분리시키는가는 WYSIWYG 환경에서 중요한 설계 요소이다. 에디터를 결합하는 경우 마우스와 같은 포인팅 디바이스를 사용하여 직접 객체를 지정하고 그 객체에 해당하는 메뉴 또는 에디팅하기 위한 환경을 사용자에게 보여준다. 에디터와 포맷터를 분리하는 경우 특정 객체를 선택했을때 그 객체에 해당하는 에디터가 다른 윈도우에 보여지고 실행된다.

또한 객체의 에디팅시 요구되는 속성 설정들은 메뉴등과 같은 형태로 감춰진 선언적 언어 형태로 내부적으로 지정된다. 즉 사용자에게 시스템에 대한 전문적인 지식을 갖지 않게 하고 쉽게 사용할 수 있도록 하며 사용자의 인지 작용과 실제적인 작업영역 사이에 gap을 없앤다.

2) 포맷팅

대화적 문서 처리 환경에서 포맷팅은 가장 중요한 문제이다. 왜냐하면 그림영역을 위한 페이지 레이아웃을 지정한다든지 특정 문자 또는 단락에 다양한 크기의 폰트 및 서체를 지정했을때 즉시 그 결과가 화면에 피드백되어 사용자에게 보여야 한다. 전통적인 문서처리 방식인 batch mode 문서 처리 시스템에서는 문서내의 일부 또는 새로운 내용을 추가하기 위해서는 분리된 에디터를 사용하여 작성한 다음 포맷터로 넘겨져 첫 페이지부터 문서의 끝 페이지까지 처리되어 포맷터 출력을 만들게 된다. 포맷처리된 결과는 프린터로 하드카피 되거나 사용자가 요청시 해당 페이지는 디스플레이상에서

preview 되어질 수 있다. 그러나 대화형 시스템에서의 처리속도와 시스템으로부터 사용자에게 피드백되어 디스플레이되는 속도는 시스템 성능에 절대적인 영향을 미친다. 중요한 것은 사용자에게 특정 페이지 위의 객체를 다루기 위해서 얼마나 많은 기능을 제공하느냐 하는 것보다는 포맷팅 속도이다. 이 포맷팅 속도를 높이기 위해서 처음부터 끝 페이지까지 포맷팅을 하지 않고 가능한 한 현재 보여지고 있는 페이지에 대한 처리만을 하는 동적 포맷팅 방식이 연구되고 있다[13].

3) 이미징

포맷팅된 결과를 디스플레이 또는 프린터에 출력시키는 과정을 말한다. 대화적 환경에서 디스플레이로의 출력은 사용자에게는 즉시 보여지고 있어 포맷터와 결합되어 있는 것 같지만 이 과정은 엄밀히 말하면 포맷팅과 분리되어 있다. 포맷터에서 처리된 결과는 내부 데이타 구조형태로 갖고 있어 사용자가 디스플레이상의 객체를 지정하고 명령을 설정했을때 그 결과를 디스플레이에 표시하고 포맷터에서 다시 사용될 수 있도록 파라미터 형태로 저장된다.

프린터로의 출력은 formatted file을 PostScript와 같은 PDL로 변환하는 과정을 말한다.

V. 문서 교환과 국제표준

점차 우리 사회가 고도의 정보화 사회로 되어감에 따라 컴퓨터를 이용한 문서처리와 그 문서를 이 기종간의 문서 시스템에서 상호교환하는 정보교환의 중요성이 날로 증대되고 있다. 특히 사무용 문서 시스템간의 전자적인 문서교환의 필요성이 증대됨에 따라 텍스트, 그래픽, 화상등 여러가지 정보들을 포함하는 문서를 시스템 독립적(system-independent)으로 작성, 수정 또는 교환할 수 있는 환경이 요구되고 있는 실정이다.

따라서 이러한 분산 환경하에서 문서교환을 위해 현재 국제표준화 기구(ISO), 전기통신 자문 위원회(CCITT)와 유럽 컴퓨터 제작 협회(ECMA)에서 사무통신에 관련된 OSI protocol 표준과 데이타 객체 양식 표준화 활동을 진행중이다. 그 중 ISO에서는 ISO/IEC JTC1/SC18에서 사무통신 표준화 작업을 추진하고 있다. SC18/WG4는 분산환경에 관련된 protocol로써 MHS(message handling system), DPA(document printing application), RDT(referenced data transfer) 및 DFR(document filing and retrival)을, 데이타 객체 양식 표준은 SC18/WG3에서 ODA(office document architecture)/ODIF (office document interchange format)의 표준을 제정하였고, WG8에서는 SGML(standard generalized markup language)/SDIF(SGML document interchange format) 표준을 그밖에 WG5에서는 multimedia 및 hypermedia 정보에 관련된 HyTime 표준은 WG8에서, 또한 multimedia content architecture를 WG9에서는 user interface에 관련된 표준화 작업을 진행 중이다.

여기서는 ISO/IEC JTC1/SC18에서 표준으로 제정한 데이타 객체 양식 표준인 ODA와 SGML에 대한 간단한 개요를 알아보도록 하겠다.

1. SGML

현재 간단한 레이아웃과 폰트 정보를 갖는 한 문서의 생성을 문자 입력에서부터 레이아웃 지정 그리고 인쇄까지 포함하여 1대의 시스템, 1인의 조작자(operater)에 의해서 실행하고 있다. 이러한 시스템들(troff, nroff, TeX, Scribe 등)은 한 문서를 만들때 그 문서에 대한 폰트, 레이아웃 등을 포함하는 시스템 의존적인 포맷팅 명령들을 가지게 된다. 이렇게 각자의 시스템에서 쓰이는 낮은 수준의 포맷 명령을 소위 절차적 마아크업(procedural markup)이라 한다. 이런 절차적 마아크업 명령으로 포맷된 문서는 이 기종 시스템의 출력장치(종이,화면 등)들을 통하여 볼 수 없으며, 다른 시스템의 포맷터나 프린터, 화면으로 출력하기 위해서는 그 문서의 절차적 마아크업 명령들의 변경이 필요하다.

작금에는 이 기종간의 시스템사이의 문서교환시 시스템들은 매크로기능을 제공하여 이러한 문제를 해결하려 하였으나, 이 매크로가 일관되게 모든 시스템에서 사용되는지 어떤지를 조사할 방도가 없다. 현재 마아크업의 개념이 "마아크업에 대해 수행되어야 할 프로세싱을 특정화하기 보다는 문서의 구조와 다른 속성(attribute)들을 나타내야 한다"는 쪽으로 변화하면서, 개괄적 마아크업(generalized markup)이 나오게 되었다. 이 개괄적 마아크업은 수행처리될 것을 각각 기술하는 것이 아니라, 문서의 논리구조와 속성을 기술함으로써 한 마아크업으로 여러 속성들을 포함 가능하게 한다.

이에 국제표준화 기구(ISO:International Organization for Standardization)에서는 서로 다른 이기종 시스템 간의 효율적인 문서 교환을 목적으로 마아크업의 일관성을 강화한 SGML을 ISO 8879로 제정하게 되었다. SGML은 이미 미국 국방성의 조달사양(CALS)에 채용된데 이어 미국출판협회, 유럽 공동체 출판국, 옥스퍼드 대학 출판부 등에서 사용하고 있고, 가까운 일본

만 하더라도(1989년) 통산성의 주도에 의해 SGML의 본격적인 보급에 대해서 검토를 개시하였다.

1) SGML의 구성 및 처리모델

SGML은 어떤 표준 마아크업의 언어(standard makrup language)나 포맷터가 아니며, 한 개괄적 마아크업 언어 (generalized markup language)의 구문만을 정의한 메타 언어(meta language)이다. SGML은 이기종 간의 시스템에서 한 문서를 전송하기 위해 다음의 3가지 요소의 구문을 제공하고 있다.

(1) SGML 선언부(SGML declaration)

(2) 문서 타입 정의부(document type definition : DTD)

(3) SGML 문서(SGML document)

각 SGML 문서를 다른 시스템으로 전송할 때 반드시 그 문서 앞에 문서 정보(폰트, 사용된 기능 등)을 담은 SGML 선언부가 전달된다. SGML에서 제공하는 구문 중 중요한 한 부분이 문서의 논리 구조를 정의하는 문서 타입 정의부이다. 이 문서타입 정의부에 따라 실제 한 문서가 작성될 수 있는데 이것이 SGML문서이다. SGML을 사용한 문서처리 모델은 그림 2와 같이 구성할 수 있다.

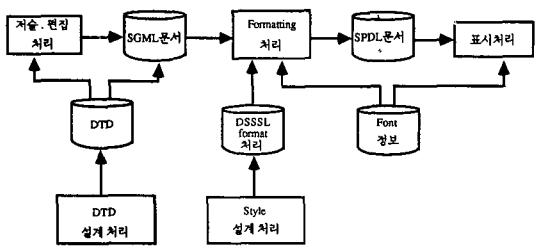


그림 2. SGML의 문서처리 모델

2. ODA

ODA는 문서 표현 메디아와 문서의 논리적 구조와 배치 구조를 체계적으로 규약하여 이 기종간의 효율적인 문서교환을 실현하기 위해 문서 교환 형식을 규정한 표준안이다. ODA의 시작은 1982년에 ECMA TC29의 2번째 모임에서 논의가 되어 ODA 표준인 ECMA-101이 1985년에 나왔다. CCITT SGⅦ은 팩시밀리 텔레텍스를 통합한 혼합 모드(mixed-mode : 권고 T.72)와 통신을 위해 문서표현과 통신방법을 규약하는 문서교환 프로파일(profile : 권고 T.73)을 1984년에 권고하였으며, 비디오텍스(videotex)와 같은 회화형 서비스등 텔레마틱 전반에 적용하기 위해 문서교환 프로토콜을 특정 서비스에 의존하지 않게 구성하고, 수신단에서 재처리를 가능하게 하는 논리구조를 도입하여 도형정보등 문서표현 메디아를 표현 가능하게 하는 것을 SGⅦ 회의에서 합의하였다. 또한 ISO의 ODA 표준과 상호 호환성을 갖고 정합하기 위해 권고 T.73을 권고 T.410 시리즈로 표준화를 진행하였다. 한편 텍스트와 사무용 시스템을 연구하는 ISO TC97/SC18에서는 문서의 논리구조와 문서표현 미디어의 연구가 진행되어 문서 구조와 교환 형식(ISO 8879)이 초안되었다. 이렇게 정보처리 즉 텍스트와 사무용 시스템에 인식을 같이하여 공동 연구한 결과 1988년에 국제표준 ISO 8879가 제정되었다.

1) ODA의 구성과 처리모델

문서는 상대방에게 전달되어 이해되기 위한 목적과 상대방에게 전달된후 편집되기 위한 목적으로 교환된다. ODA는 문서구조, 내용구조 및 교환양식을 기술하는 표준안으로서 이러한 목적에 따라 문서를 수정가능한 형태(processable form)과 문서의 수정이 불가능한 포맷된 형태(formatted form) 또는 이러한 두 형태의 결합 형태인 포맷된 수정 가능한 형태(formatted processable form)로 교환할 수 있게 한다.

ODA에서는 한 문서의 구조적 모델을 논리적인 관점과 배치적인 관점에서 구성하였으며, 그 내용은 다음과 같다.

(1) 공통 논리 구조(generic logical structure)

(2) 공통 배치 구조(generic layout structure)

(3) 특정 논리 구조(specific logical structure)

(4) 특정 배치 구조(specific layout structure)

(5) 내용 정보 (content information)

(6) 배치 스타일 (layout style)

(7) 표현 스타일 (presentation style)

ODA의 문서처리 모델은 주 처리 즉 편집, 배치, 표시 처리로 나누어져 있으며 그림3에 이들간의 관계를

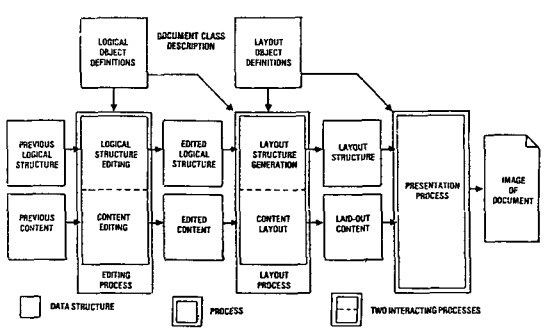


그림 3. ODA의 문서처리 모델

개념적으로 보여주고 있다.

Ⅵ. Multimedia Documentation

국제표준기구에서는 앞으로의 전자출판 환경이 stand-alone에서 벗어나 시스템간의 문서정보의 교환을 필요로 하기 때문에 그 목적에 따르는 국제표준 SGML과 ODA을 제정하였다. 이 표준안은 현재 문자, 도형, 이미지등의 정적 문서표현 미디어만을 포함하고 있으나, 최근의 multimedia 기술의 등장으로 미디어의 종류도 이전의 정적인 미디어에서 음성, 동화상등 시변요소가 적용되는 동적 미디어로 폭이 넓어지고 있다. 이에 ISO JTC1/SC29 및 SC18에서는 Hypermedia 정보처리 및 각각의 미디어들을 표준화 하고 있으며 SC29의 작업은 다음과 같이 진척 중이다.

WG7 : Computer graphic image
WG8 : Coded representation of picture, audio and multimedia information
WG9 : Bi-level image
WG10 : Photographic image
WG11 : Moving picture image and associated audio
WG12 : Multimedia and hypermedia inforamtion

또한 이들 미디어를 수용하기 위해서 JTC1 SC18에서는 SGML의 syntax를 사용하여 이들 미디어를 표현하는 Hytime(hypermedia/time-based document structuring language : DIS 10744)을 현재 국제 표준안으로 확정하기 위한 작업을 진행중이며, ODA에서는 Hyper ODA로 확장시키고 있다 .

Ⅶ. 결 론

지금까지 전자출판에 관하여 여러가지 측면을 살펴보았다. 필자는 지난 15년간 각종 연구, 개발해온 문서처리 또는 전자출판시스템의 경험에 근거함과 동시에 1980년도 중반부터 JTC1/SC18을 중심으로 하는 문서처리 관련 국제표준 활동 과정에서 느껴온 몇가지를 정리하고 이를 결론에 대하고저 한다.

(1) 문서의 논리구조 및 배치구조의 구분이 명확하여야 한다.

국내에서 개발된 몇몇 전자출판 시스템에서 볼때 논리구조 및 배치구조의 구분이 명확하지 않는 경향으로 생각된다. 문서의 논리구조는 일반적으로 포맷팅에 필요하지만 그밖에 논리구조를 이용하여 문서의 저장, 검색등 응용범위는 매우 넓다. 또한 각기 다른 저자에 의해서 작성된 문서정보의 공유도 명확한 문서논리구조의 정의를 통하여 가능할 것이다. 즉, 문서의 논리구조를 정의하고 공유할 수 있는 공통의 방식이 요구된다.

(2) Font의 개발, font 정보의 보급, 관리 필요성

Font 정보는 DTP의 품질을 결정하는 매우 주요한 요소이며 font의 연구, 개발에는 막대한 시간과 경비가 필요하게 된다. 이와같이 font 정보는 font 개발자를 포함하는 모든 사람을 위한 귀중한 자산이다. Font 정보의 공유는 UNIX나 Open System의 공개 정책과 마찬가지로 보편화된 국제적 흐름이다. 국내에서도 font의 연구, 개발은 개별적 또는 공동으로 이루어질 수 있으나 개발자의 저작권등의 확보되는 환경에는 귀중한 font 정보의 공통 보급 및 관리할 수 있는 메카니즘이 절실히 요구된다.

(3) Graphical user interface의 공유

보다 사용자에게 친근함을 줄 수 있는 GUI의 구축이 요구되며 여기에는 표준적인 사용자 지침서도 필요할 것이다. 표준적인 GUI의 제공은 보다 폭넓게 전자출판의 보급을 가능케 할 것이며, 급변하는 전자출판 관련 기술에 독립적인 한글처리 환경의 구축이 필수적인 것으로 판단되며, 궁극적으로 전자출판시스템의 개발을 용이하게 함과 동시에 개발비의 절감을 통한 고품질의 저가격화를 도모할 수 있을 것이다.

(4) 워드프로세서를 포함하는 전자출판시스템간의 상호 정보교환방식의 구축이 시급하다.

국내에서 개발된 DTP시스템간 그리고 워드프로세스간의 문서정보의 상호교환은 매우 어려운 실정이며 문서정보의 상호교환의 중요성은 매우 높아지고 있다. 국내에서도 외국과의 문서정보 교환이 가능한 상호교환 방식을 실현하기 위한 기반구조의 구축이 절실하다. 이를 위하여 업계의 공동적인 노력과 동시에 정부 차원에서의 적극적 자원이 요구된다.

(5) 새로운 형태의 multimedia documentation에 공동 대처하여야 한다.

멀티미디어 기술등 급변하는 기술혁신에 각기 대처하기는 사실상 무리일 것이다. CD-ROM의 포멧, MPEG, HyTime 기술, MPC 사양등을 포함하는 멀티미디어 기술과 이를 이용하는 새로운 형태의 전자출판을 국내 기업이 개별적으로 대응 또는 대처하기는 도저히 무리일 것이다.

지난 4월 미국 샌프란시스코에서 열린 multimedia 및 CD-ROM conference에서 발표된 외국의 유수기업간의 합병 또는 공동 연구개발의 선언이 우리에게 뜻하는 것은 무엇일까?

參考文獻

[1] R. Furuta et al, "Document formatting systems: Survey, concepts, and issues," *ACM Computing Surveys*, vol.14, no.3, pp.417-472, Sep. 1982.

[2] B.W. Kernighan, "The UNIX Document Preparation Tools-A Retrospective," Proc. First Int'l Conf. Text Processing Systems, pp. 12-25 Oct. 1984.

[3] D.E. Knuth, *The TeX Book*, Addison Wesley, Reading, Mass., 1986.

[4] B.K.Reid, Scribe:A Document Specification Language and Its Compiler, PhD thesis, Canegie Mellon Univ., Pittsburgh, Oct. 1980.

[5] FrameMaker Reference Manual, Version 2.0, Frame Technology Corp., San Jose, Calif., Oct. 1989.

[6] Interleaf Technical Publishing Software Reference Manual, vol.1-4, Interleaf Inc., Cambridge, Mass., Mar. 1989.

[7] Ventura Publisher Reference Manual, Version 1.1, Ventura Software, Jul. 1987.

[8] Information Processing-Text and Office Systems -Office Document Architecture and Interchange Format, ISO 8613, May 1989.

[9] Information Processing-Text and Office Systems -Standard Generalized Markup Language (SGML), ISO 8879, 1986.

[10] Adobe Systems, Inc., *PostScript Language Reference Manual*, Addison-Wesley, Reading, Mass., 1985.

[11] O. Jones, *Introduction to the X Window System*, Prentice-Hall, 1989.

[12] 임순범, "글자체 설계 및 자동생성 시스템의 개발," 폰트개발과 표준화 워크샵, 한국정보과학회 SW공학연구회, pp.3-6, 1989. 4

[13] 임광택, 이수연, "Dynamic Document Formatting," 광운대학교 정보공학연구실, 1992.

[14] 이수연, "도표의 자동 배치 기능을 갖는 영문 원고 WP 시스템," IECE 65(5), pp.558-562, 1982.

[15] 이수연, "동적 합성에 의한 2차 문서 화상의 자동 작성 시스템," IPSU(Trans.) 24(4), pp.442-452, 1983.

[16] 이수연, "GPSS를 이용한 real-time 컴퓨터 시스템의 설계 및 분석에 관한 연구," 한국정보과학회 논문집, 1978.

[17] 이수연, "문자 패턴의 크기변환," 한국통신학회 논문집 11(2), 1986.

[18] 김차종, 이수연, "범용 터미널을 이용한 문서화상 작성에 관한 연구," 한국정보과학회 논문집 14 (1), pp.3-10, 1987.

[19] 정회경, 이수연, "4방향 선분 분리법에 의한 도트 패턴의 크기 변환에 관한 연구," 전자공학회 논문집 24(6), pp. 1013-1019, 1987.

[20] 김차종, 강한종, 이수연, "다목적 폰트제작 시스템," 한국정보과학회 논문집 16(5), pp. 422-433, 1988.

[21] 임광택, 김차종, 이수연, "확장 Preview 기능을 갖는 한글 전자출판 시스템," 한국정보과학회 논문집 17(6), pp. 742-759, 1990.

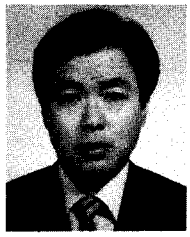
[22] 홍온선, 윤근중, 이수연, "ODIF 데이타스트림의 포스트스크립 변환," 한국통신학회 논문집 16 (11), pp.1027-1036, 1991.

[23] 진익희, 홍온선, 이수연, "X 원도우를 이용한 복합문서 그래픽 편집기," 한국정보과학회 논문집 19(2), pp.178-186, 1992.

[24] 고창환, 정회경, 이수연, "X 원도우를 이용한 CGM 환경 구현," 한국정보과학회 논문집, 7월 게재 예정

[25] 홍온선, 이수연, "SGML 문서 에디터 구현에 관한 연구," 한국통신학회 논문집, 게재 예정 1992.

筆 者 紹 介



李 壽 淵
1946年 10月 24日生
1969年 광운대학교 전자통신학과 (학사)
1977年 연세대학교 대학원 전자공학과 (석사)
1983年 일본 교또대학교 정보공학과 (박사)

1973년 ~ 현재 광운대학교 전자계산기공학과 교수

주관심분야 : 문서화상, ODA, SGML, Multimedia Documentation