컴퓨터 프로그램의 특허성 원론

Basic Theory on the Patentability of Computer Program

이상무(S. M. Lee) 지적재산개발실 연구원

소프트웨어 산업이 발전하면서 다양한 기능을 수행하는 복잡한 프로그램들이 개발됨에 따라 그 특허성을 인정하는 수용의 폭이 크게 확대되는 추세에 있다. 컴퓨터 프로그램은 컴퓨터(하드웨어)와 결합함으로써 그 실체가 나타나므로 잠재적인 특허성을 가지고 있는 것으로 본다. 단지, 컴퓨터 프로그램은 하드웨어와의 관련성에 따라 계층적인 상호작용을 갖고 있는 특성이 있어 프로그램의 유형에 따라 특허성의 정도 차이가 있다는 것을 알 수 있다. 컴퓨터 프로그램은 반드시 하드웨어와 결합하여 동작할 때 의미가 있으므로, 컴퓨터 프로그램에 대한 최근의 특허심사 기준이 실체성 표명의 벽을 크게 완화하였다고는 하여도 프로그램 자체만으로는 특허성이 인정되지 아니하며 어떠한 형태로든 산업상 유용한 사물과의 연관관계를 이용하여 설명하여야 한다.

I. 서 론

컴퓨터 프로그램은 그 자체의 본질로 볼 때, 수학적 알고리즘 또는 인간의 논리적 사고의 흐름으로 해석되어 특허를 받는데 특정한 제약 조건이 뒤따라 왔다. 특허는 추상적인 것은 대상이 되지 못하며 실제의 효과를 요구하기 때문이다. 컴퓨터 프로그램은 첨단 기술분야로서 무궁무진한 변화와 발전을 해오고 있다. 초기의 단순한 기능에서 부터 점차 복잡한 기능으로 다변화하므로 여기에 특허의 기준을 적용한다는 것은 어려운 문제이며 이에 따라 어떻게 복잡한 컴퓨터프로그램의 성격에 적절한 특허성 해석 기준을 적용하여야 할 것인가가 여러가지 사건들을 통하여 변천되고 있다. 이러한 변화의 과정 속에서 근본적인 성격 차원에서 컴퓨터프로그램과 특허의 성질을 분석하고 그들의 관계에 따른 컴퓨터프로그램의 특허성을 이론적으로 정립하고자 한다.

II. 컴퓨터 프로그램의 특성 분석

컴퓨터프로그램의 특성이 여러가지가 있겠으나 본 고의 주제에 따라 특허성 원론에 밑받침이 되는 영역에서의 주요 특성을 분석하고 전제하도록 한다.

1. 컴퓨터 프로그램의 발생 배경

컴퓨터프로그램의 성격을 이해하기 위해서 컴퓨터프로그램이 어떻게 해서 발생하게 되었는지,

그 역사적 배경을 짚어 보는 것이 기본적인 도움이 될 것이다. 컴퓨터프로그램이 나타나게 된 것은 그 이전의 기원으로부터 비롯되었다. 컴퓨터란 어의가 계산의 뜻을 가지고 있는 것은 이것이 원래 계산기에서부터 시작되었다는 말하는 것이다. 즉, 오늘날의 복잡한 컴퓨터에 이르까지 최초에는 그것이 아주 원시적인 계산기가 발전하여 이루어진 것이다.

계산기의 유래는 고대 메소포타미아 문명 당시, 작은 돌이란 의미를 갖고 있는 'kalkul'이 계산기(calculate)의 어원으로 사용되었다. 이후, 중세에는 오늘날의 코인(coin)에 해당하는 것으로서 도박할 때, 득점 계산에 쓰는 나뭇조각 따위의 작은 원판인 산가지(jetton)라는 것을 사용하기도 하였다. 17C부터는 초보 수준의 전문 계산기가 등장하게 되었다. 그 대표적인 것들에는 네피어(Napier)의 계산기로서 대수(對數)용 승제산(乘除算)을 할 수 있는 것과 파스칼의 계산기로서 치차(齒車)를 조합시킨 가산기와 라이프니찌의 계산기로서 반복 가산에 의한 승산기가 있었다. 19C에 바베지가 개발한 해석 엔진이 나타났는데 이것이 세계 최초의 자동 계산기였다. 여기서 자동 계산을 위하여는 가장 원시적인 프로그램 방식이 적용되었을 것이다.

20C에 접어들면서 1944년에 세계 최초의 전기기계 계산기인 하바드 마크I이 만들어졌으며 1945년에 노이만에 의하여 '프로그램 기억 방식, 2진수 연산방식'이 제창되었다. 1946년에 세계 최초의 전자 계산기인 에니악(ENIAC)이 탄생하고 계속 개발되어져 나감으로써 이 때부터 본격적인 컴퓨터 프로그램의 개념이 적용되기 시작하였다. 1945년에 'Plan Kalkul'이라는 최초의 고급언어가 기계어로 개발되었다. 이어 유명한 당시 대표적 프로그램밍 언어인 Fortran과 COBOL 등이 등장하게 되었으며 Unix 운영체제를 지나 IBM PC가 등장하면서 MS-DOS로 이어지게 되고 이외에 다양한 목적 기능의 응용 프로그램들이 개발되게 되었다. 이러한 변천 과정을 정리하면 <표 1>과 같다[1].

아래의 프로그램 발생 배경에서 알 수 있는 바와 같이 프로그램 발생의 근원은 원시적인 계산에서부터였다. 단순한 편의 목적의 계산 기구로부터 자동화된 계산기로 이어지면서 프로그램의 개념이 부여되기 시작하였다. 그리하여 컴퓨터가 등장하게 되고 이에 따라 점차 고급 프로그램으로 발전하게 되었다.

2. 컴퓨터 프로그램과 하드웨어와의 관계

가. 하드웨어와의 일체성

최초의 계산기 시대에는 사람의 손으로 직접 단순히 계산기구를 다루면 되었으므로 별도의 프로그램이라는 것은 필요치 않았으며 인간의 논리적 사고에 따라 직접 이용되었다. 그러나 점차 복잡한 동작 기능을 요구하게 되면서 이것을 사람의 생각에 따라 처리하기에는 한계가 있으므로 계산기를 복잡하게 동작시켜 좀 더 다향한 결과를 얻고자 하는 논리적 수순을 결정하여 이것을 기계에 자동 적용할 목적으로 창작된 것이 프로그램이었다. 즉, 먼저 하드웨어가 있고 이것을 인간이 원하는 목적에 따라 유용하게 동작시키는 것이 프로그램이다. 이와 같이 컴퓨터와 프로그램은 서로 결합됨으로써 필요한 기능을 수행한다. 각각은 하나의 특성 개체이지만 독립적으로는 존재의 의미가 없으며 양자가 결합되었을 때, 실제의 의미를 갖게 된다. 프로그램은 컴퓨터를 통해서 그 뜻을 나타내며 컴퓨터는 프로그램을 통해서 동작한다.

<표 1> 컴퓨터 프로그램의 역사

연대	구분	하 드 웨 어	프로그램(소프트웨어)
고대	컴퓨터	Kalkul (메소포타미아): 자갈(小石)의 뜻, calculate 의 어원	
		Abacus (희랍): 평판의 뜻. Kalkul 과의 조합이 주판으로 발전. 3 계 단위의 취급	
중세		Jetton (중세 유럽): 위조 코인에 의한 계산. 현재, 공중전화용 코인으로서 이름을 남김.	
		산목(算木), 산반(算盤) (중국, 일본): 정/부(正/負)의 개념을 도입한 계산 용구. 주산으로 발전	
17C	이전의 계산기	네피어(Napier)의 계산기: 대수(對數)에 의한 승제산	
		파스칼의 계산기: 치차를 결합한 가산기	
		라이프니찌의 계산기: 가산의 되풀이에 의한 승산기. 컴퓨터가 등장할 때까지는 널리 사용되었다.	
19C		바베지의 해석엔진: 세계 최초의 자동 계산기. 완성에 도달 못함. 바베지의 협력자 Ada 부인은 세계 최초의 프로그래머라고 불렸다. 1970 년대에 미국 국방성이 개발한 Ada 는 동 부인의 이름과 같이 명명되었다.	
20C		홀라리스의 펀치 카드 시스템(PCS): 펀치 카드에 의한 대량 데이터 처리기. 컴퓨터가 등장하기까지 대기업, 관청 등에서 다수 사용되었다.	
1944	제1세대/진공관	하바드 마크 I: 세계 최초의 전기 기계 계산기. 바베지의 꿈을 실현하였다.	
		1945: 폰 노이만에 의한 '프로그램 기억방식, 2 진수 연산방식'의 제창	
		1946: ENIAC(세계 최초의 전자 계산기)	
		1949: EDSAC	
1950		1950: EDVAC 어느 것이든 '프로그램 기억방식', '2 진수 연산방식'을 채용. 현재의 컴퓨터시스템의 막이 열림 1951: UNIVAC I 1952: IBM701 1954: 바라메트론 계산기(일본)	1945: 콘라드.쯔제에 의한 'Plan Kalkul'의 개발 (최초의 고급 언어, 미완성) <기계어에 의한 개발> 1951: 서브루틴, 라이브러리의 개념 (Wilks, Wheeler, Gill) <어셈블러에 의한 프로그램 개발> 1954: 최초의 컴파일러 가동(MIT) 1956: FORTRAN 개발 (IBM : John Backus) 이후 기술계산용 언어로서 널리 사용된다. 1956: IBM704 용 OS 의 개발 (IBM, GM, North America) 1958: ALGOL 발표, 프로그래밍 언어 이론에 영향이 큼 1959: LISP 개발(MIT : John McCarthy) 1959: COBOL 개발(CODASYL) 사무처리 부문에 공헌이 큼 <가상기억기능: 영 멘체스터 대학>

연대	구분	하드웨어(계산기)	프로그램(소프트웨어)	주된 처리형태
1960	제 2 세대 / 트랜지스터	1959: IBM7070 UNIVAC III	OS 의 개발 - 프로그램 로더 - 모니터 - Job 관리 - 다중 프로그래밍 1963: SABRE 시스템 (IBM, American Airline: 트랜잭션 처리) 1965: PL/I 발표 (IBM/SHARE)	Batch 처리
	제 3 세대 / IC	1964: IBM 시스템/ 360 미니컴의 등장	1965: BASIC 개발 (John keweny, Thomas kurtz) <애플리케이션 소트웨어의 등장> 1968: Pascal 개발 (Niklaus Wirth) 1968: 소프트웨어공학회의 개최(NATO)	온라인 처리 TSS 처리 회화형 처리
1970	제 3.5 세대 / LSI	1970: IBM 시스템/370 애플 II	1970: 릴레이셔널 데이터베이스의 개념 (IBM : E.F. Dodd) 1970: 구조화 프로그래밍의 개념 1970: UNIX 의 개발(Edsger Djkstra) (벨 연구소, Thompson, Ritchie) 1972: "C"언어의 개발(상동) 1975: 미국 국방성 Ada 개발에 착수 <맨-머신 인터페이스의 개선 : XEROX. PARC> <오브젝트 지향언어의 등장>	분산처리 네트워크 LAN
1980	제 4 세대 / VLSI	1981: IBM PC IBM308X 슈퍼컴퓨터 1984: 매킨토시 제 5 세대 컴퓨터	1981: MS-DOS 발표 1987: OS/2 발표 AI 엑스퍼트 시스템	

나. 컴퓨터 프로그램의 계층성

컴퓨터 사용자가 컴퓨터를 사용하기 위해서는 프로그램을 통하여야 한다. 프로그램을 통해서 하드웨어 자원을 활용하는데 그 방법과 목적과 기교를 프로그램이 구사한다. 프로그램은 장치와 사용자를 연결시켜 주는 기능을 갖고 있다. 즉, 사용자와 하드웨어 사이에는 프로그램이 중개자로 놓여 있어서 사용자의 요구를 받아 하드웨어에 작용하며 하드웨어의 상태를 사용자에게 보여준다. 기계의 입장에서 인식할 수 있는 것은 사람이 생각하는 것 같이 복잡한 것이 아니라 오직 신호의 유무만을 감지한다. 사람의 고차원적인 사고와 양자 택일만이 있는 기계를 상호 연결시키기 위해서는 프로그램이 필요한 것이다. 그런데 이렇게 현격히 다른

차원을 연결시켜 주기 위해서는 자연히 프로그램의 구조가 복잡해질 수 밖에 없는데 여기서 프로그램이 이해하는 차원이 기계 쪽에 가까운가 아니면 사용자 기준에 가까운가에 따라 프로그램의 성격이 구분되는 특성이 나타나며 이것을 계층성이라고 한다.

컴퓨터 본체가 있고 사용자가 있으며 이들을 연결시켜 주는 프로그램을 컴퓨터 본체와 합하여 가상 컴퓨터 영역이라고 할 때, 일반적 계층성의 표면적 구조를 도시하면 (그림 1)과 같다[2].

가상 컴퓨터 상의 프로그램의 계층성을 설명하기 위해서 3가지 사용자의 형태를 주지하도록 한다. 먼저 이미 상품화된 프로그램의 사용자이다. 이것을 'level 3 languages'라고 한다. 두번째는 프로그램을 작성하는 소프트웨어 엔지니어들이다. 이것은 'level 2 languages'라고 한다. 세번째는 마이크로프로세서 따위의 전자 하드웨어 혹은 장치의 근접한 레벨에 매우 밀접하여 작업하는 일부 소프트웨어 엔지니어와 아마츄어 프로그래머들이다. 이것을 'level 1 laguages'라고 한다.

프로그래머와 하위 계층의 프로그래머는 언어번역기에 접근하여 기억장치에 변수와 프로그램을 입력하고 프로그램을 편집하며, 프로그램을 연결시키고 로딩시켜 실행하는 특성을 갖을 것이다. 모두 열거된 것은 아니지만 이러한 특징들은 집합적으로 PSE 즉, 관련된 컴퓨터 장치와 보조장치의 Programming Support Environment로 알려져 있다. Ipse(integrated project support environment)은 PSE들이 모여진 통합 그룹이다. 일반적인 경우에 있어서, 사용자와 프로그래머는 전자 하드웨어의 상부에 물리적인 것이 아닌 실효적으로 아마도 수십만 프로그램 문장의 표준 소프트웨어로서 상기 특성과 편의가 존재할 소위 가상 컴퓨터시스템에 접근할 것이다. 하드웨어의 상부에 있는 실제의 혹은 표준 프로그램의 이러한 도식은 유용하지만 엄밀히 구분되는 것은 물론 아니다. 프로그램은 컴퓨터의 회로 내부나 자기 디스크 저장 장치, 카세트 테이프 등과 같은 보조장치상에 2진 비트 형태로 저장된다. 가상 컴퓨터 영역의 프로그램이 하드웨어에 중재되어 있는 관점은 그 절차가 회로와 다른 표준 프로그램과 프로그래머들의 응용 프로그램으로의 접근을 아주 강하게 조절(혹은 완전한 차단)한다는 점에서 실제로 옳은 개념이다. 이러한 통제구조는 특히 전자 기사들로 하여금 마치 강 어귀가 모래로 덮여있는 것처럼 프로그램의 편리 기능들로 막혀 있는 컴퓨터 하드웨어를 보기 위하여 level 1 프로그래밍 언어안에서 작동시키고자 하는 충동을 유발해왔다.

소프트웨어는 그 용도에 따라 큰 범주가 나누어지게 되었다. 공통적으로 유용한 서비스를 제공하는 소프트웨어는 시스템 소프트웨어라고 한다. 오퍼레이팅 시스템, 컴파일러와 어셈블러는 시스템 소프트웨어의 예이다. 프로그래머에 목적된 프로그램에 반하여, 애플리케이션 소프트웨어 혹은 단지 애플리케이션은 스프레드시트나 텍스트 편집기와 같은 컴퓨터 사용자에 목적된 프로그램에 주어진 이름이다[3].

3. 창작과 기술의 양면성

컴퓨터프로그램의 또 다른 특성은 이것이 단지

산업기술 분야에 적용된다는 것 외에는 마치 인간의 일상 생활을 소재로 한 내용을 글로써 표현한 보통의 소설과 같은 저작물의 성격과 동일하다는 것이다. 프로그램에서는 언어를 기술한 글이 이 분야에 특수하게 맞도록 개발된 프로그램 언어에 해당된다. 컴퓨터 기술분야에 쓸 수 있는 다양한 표현이 프로그래밍 언어를 사용하여 이루어진다. 즉, 프로그램 자체는 일종의 표현 수단이라고 볼 수 있다. 따라서 이 표현수단에는 근본적으로 인간의 논리적 사고가 들어 있는 것이다.

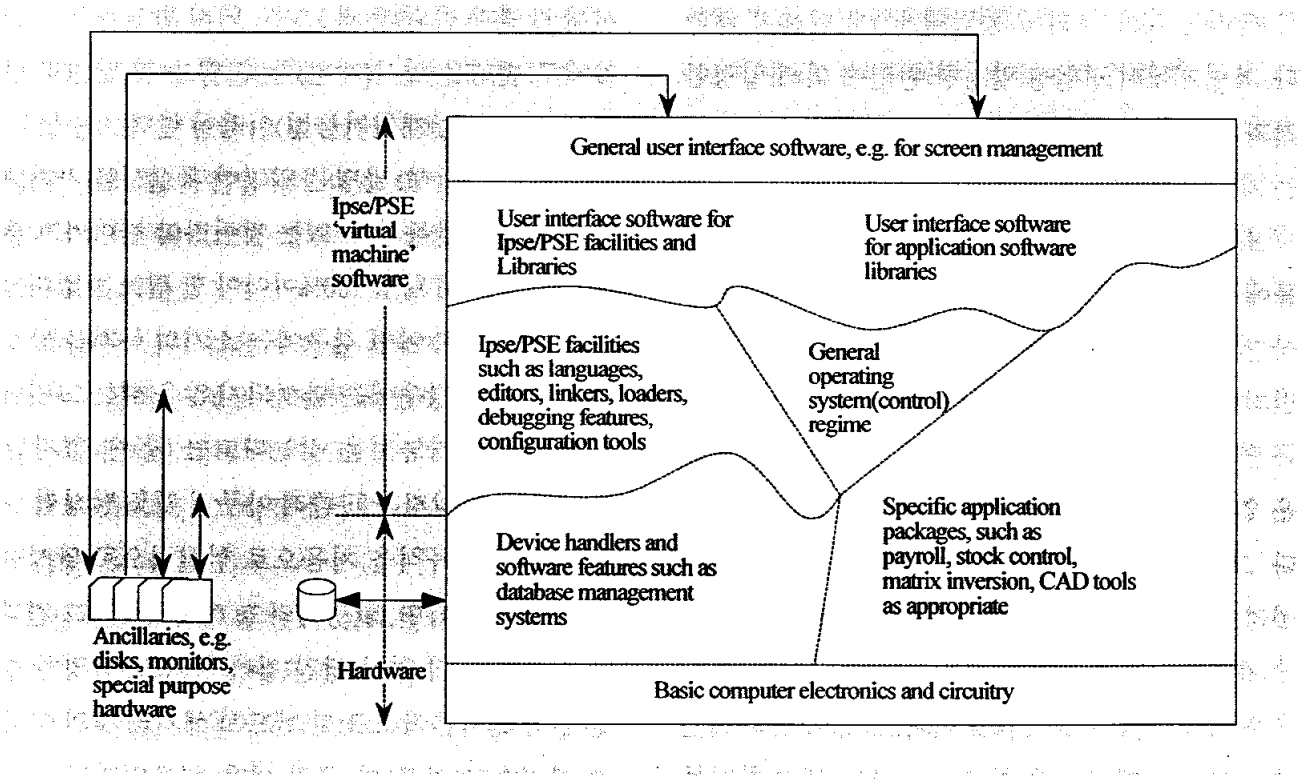
이와 같이 컴퓨터프로그램은 산업기술 분야에 적용할 수 있는 기술(技術)성과 프로그래밍 언어라는 표현 매체를 통하여 기술(記述)할 수 있는 표현성을 동시에 보유하고 있다[4].

III. 특허의 요구 분석

1. 발명의 정의에 따른 해석

국내 특허법 제1장(총칙) 제2조(정의)의 1에 "'발명'이라 함은 자연법칙을 이용한 기술적 사상의 창작으로서 고도한 것을 말한다."라고 되어있다[5]. 이 문귀를 잘 분리 해석하면 기본적인 특허요건들을 <표 2>와 같이 기술할 수 있다.

<표 2>에 나타난 바와같이 특허요건은 4가지이다. 이 중에서 컴퓨터프로그램에 문제의 관건이 되는 요건은 성립성이다. 이것은 미국 특허법에서 소위 process, machine, manufacture, composition of matter 등의 'subject matter'에 해당하는 사항이다[6]. 이것은 다른 요건들보다도 더 기초적인 요건으로 해석할 수 있다.



(그림 1) 가상 컴퓨터상의 프로그램의 표면적 계층성

<표 2> 발명의 정의로부터 나온 특허요건

No.	분리문구	해당특허 요 건	해 설
1	'자연법칙을이용한'	성립성	
2	'기술적사상의'	유용성	산업상이용 가능성 이라고도한다.
3	'창작으로서'	신규성	새로운것을말한다.
4	'고도한것을말한다.'	진보성	

2. 자연법칙의 이용성

전통적인 해석에 의하면 자연법칙이란 자연의 영역(자연계)에서 경험에 의해 발견되는 법칙을 말한다. 예컨대 물은 높은 곳에서 낮은 곳으로 흐른다. 또는 통나무는 물에 뜬다는 것 등은 자연법칙이다.

자연법칙이라고 해도, 자연과학상 XX법칙으로 불리는 것(뉴튼의 운동법칙 등)에 한하지 않는다. 자연계에서 경험상 일정한 원인에 의해 일정한 결과가 생긴다고 하는 것(경험칙)도 여기서 말하는 자연법칙이다.

또한, 인간의 추리력 기타 순지능적 · 정신적 활동에 의해 발견되고 인출된 법칙(수학또는 논리학상 법칙), 인위적인 결정, 경제학상 법칙 등은 자연법칙이 아님은 말할 나위도 없다.

인간의 심리현상에 입각한 경험칙 즉 심리법칙은 일반적으로 자연법칙이 아니라고 해석되고 있다. 그러나 생리학상의 법칙에 대해서는 자연법칙이라고 해석해야 할 것이다.

자연법칙 그 자체로는 발명이 되지 않는다. 발명은 이것을 이용한 것이어야 한다. 예컨대 앞의 예인 물은 높은 곳에서 낮은 곳으로 흐른다, 또는 통나무는 물에 뜬다는 자연법칙을 이용해서 수차(水車)를 만들거나, 통나무를 결속해서 뗏목을 만들면 자연법칙을 이용한 것으로서 발명이 될 수 있다.

자연법칙을 이용한 것이라고 할 수 있게 하기 위해서는 다음의 것이 필요하다.

전체적 이용: 자연법칙의 이용은 전체로서의 이용이어야 한다. 일부라도 자연법칙을 이용하지 않은 부분이 있는 것은 발명이 아니다. 예컨대, 자연현상에 대한 잘못된 인식을 전제로 한 것은 설령 기타의 부분이 이론적으로 정확하다고 해도 발명은 아니다. 이와 같은 것은 결국 실시할 수 없는 것에 불과하기 때문이다.

즉, 발명은 실시가능성이 있는 것이라야 하며 더구나, 자연법칙을 이용한 것인 이상 몇 번 되풀이 해도 실시할 수 있으며, 또 항상 일정한 확실성을 가지고 같은 결과를 반복할 수 있는 것임과 동시에 발명자 이외의 제 3 자도 역시 발명자와 마찬가지로 발명을 실시(재현)할 수 있는 것(발명의 반복 가능성 또는 재현가능성)이라야 한다.

일정한 확실성: 발명은 일정한 확실성을 가지고 같은 결과를 반복할 수 있는 것이라야 한다라고 해도 그 확실성이 항상 100%이어야 할 필요는 없다.

어떤 발명이 어떤 경우에는 소기의 목적을 달성하지만, 어떤 경우에는 달성되지 않을 때는 그것이 100%의 반복가능성을 갖는다고는 할 수 없지만, 이와 같은 것일지라도 종래 아무리 해도 달성할 수 없었던 어떤 목적을 처음으로 현실적으로 달성할 수 있는 것이라면, 발명이라고 해야 한다. 그 결과가 단순히 근거 없이 생긴 것이 아니고, 일정한 수단에 의해 처음이고 또한 반드시 생기는 것인 이상, 그 확실성(성공률)이 극히 낮은 경우일지라도 발명임을 잃지 않는다. 발명이 개척적 내지 기본적인

것일 경우에는 오히려 확실성(성공률)이 낮은 경우가 많다.

자연법칙에 대한 인식: 발명은 자연법칙을 결과로서 이용하는 것이라면 충분하며, 발명자에게 있어 그 법칙에 대한 정확하고도 완전한 인식을 반드시 가져야 할 필요는 없다. 경험상 취득한 것이라면 충분하다. 즉 발명이 어떤 이론에 의해 효과를 가져오는가에 대한 설명이 없어도, 또는 설명이 다소 불충분하거나 또는 잘못이 있어도 상관 없다. 일정한 수단에 의해 일정한 목적을 달성하는 것이 이론이외일지라도 확실히 증명할 수 있는 한, 결과로서 자연법칙을 이용한 것이 되기 때문이다.

또 발명자는 일정한 수단 및 일정한 결과에 대한 인식에서 잘못이 있어서는 안되지만 그 인식은 반드시 완전할 필요는 없다.

실패예와 발명의 성립: 특정한 목적을 달성할 수 없다고 하여 선인(先人)이 단념한 수단(실패예)은 발명으로 인정할 수 있는가. 목적달성수단으로서의 인식 즉, 자연법칙에 대한 인식을 완전히 결한 것이므로 발명으로 성립하지 않는다는 것은 말할나위도 없다.

후인(後人)이 같은 수단을 채택하여 목적을 달성할 수 있다는 것을 분명히 하였을 때에는, 후인은 자연법칙에 대해 처음으로 인식하고 이것을 이용한 자이므로, 그 시점에서 그 수단은 발명으로 성립한다.

단, 여기서 유의할 것은 실패예가 발명으로 성립하고 따라서, 그 발명은 신규성을 가진다고 해서, 그 발명이 진보성을 갖는다고 하여 특허를 받을 수 있는 것은 아니다. 발명의 성립성과 특허성은 별문제이기 때문이다[7].

3. Subject Matter –실체성

미국 특허법상에서는 국내 특허법의 성립성 요건에 상당하는 내용으로 기본적인 특허의 대상을 규정하고 있다. 그 대상은 상기 발명의 정의에 따른 해석에서 언급한 바와 같으며 각 subject matter를 요약하면 다음과 같다.

기계(machines): 이것들은 간단한 플라이어로부터 많은 다른 구조를 갖는 복잡한 강철 주조물에 이르기까지 복잡하게 변하는 기계적 기계들을 포함한다. 이 범주는 또한 구성 요소들, 회로와 전반적인 시스템을 포함할 수 있는 전기 장치들을 포함한다.

공정(process): 공정은 방법과 같은 것이며 무언가를 하나의 상태에서 또 다른 상태로 전환하기 위하여 취해지는 연속된 단계 혹은 조치로 이루어진다. 여기에는 석유 화학 정제와 같은 화학 공정, 금속 부분을 형성하는 기계 공정과 전기 공정들이 포함된다.

조성물(composition of matter): 이 범주는 화학 합성물, 약제물, 금속 합금과 고온 초전도 물질과 같은 새로운 세라믹 형성물을 포함한다.

제조품(articles of manufacture): 이 범주는 새로운 인간공학적 의자로부터 기계적 도면 작성용 연필에 이르기까지 사람에 의해 만들어진 모든 것을 포함한다.

사업을 하는 방법과 같은 특허받을 수 없는 실체물(subject matter)의 어떤 범주들은 특허받을 수 있는 자격 요건으로부터 제외된다. 두번째 특허받을 수 없는 범주는 단순한 연산식이다. 즉, 연산식의 산업적 이용을 위한 적용으로부터 분리되어 있는 수

학 공식 그 자체(예: $E=MC^2$)는 특허받을 수 없다[8].

4. 특허심사기준의 변화에 따른 적용

이와 같이 법적인 특허요구 정의가 있지만 실제로 그 개념을 적용함에 있어서는 워낙 다양한 사건들이 전개되기 때문에 이러한 것의 특허성을 분류하기 위해서 별도의 특허청 심사기준 혹은 지침, 안내(guideline)를 마련해서 이것을 통해서 실제의 해석을 내리고 있다. 그래서 법령 자체는 변하지 않더라도 지금까지 고도의 성장을 이루어 온 컴퓨터프로그램의 변천에 따라 그것의 특허성을 판단하는 심사기준도 변화되어 올 수 밖에 없었다.

초기에는 수용의 폭이 좁았으나 점차 프로그램의 물량이 확산됨에 따라 심사기준의 폭도 대폭 완화되어 컴퓨터프로그램의 특허성에 대한 적극적 부정요건이 아닌 태양 아래 인간이 만든 모든 것은 자연법칙 자체, 자연현상, 단순한 추상적 아이디어가 아닌한 특허가 될 수 있다는 소극적 부정요건[9]으로 전개되어 실제에 있어 컴퓨터용 데이터 기록매체에 수록된 내용에 대해서도 특허를 허용하는 획기적인 사건의 단계에 이르게 되었다[10].

IV. 컴퓨터 프로그램과 특허와의 관계

1. 산업기술분야로서의 컴퓨터 프로그램

컴퓨터프로그램은 "정보기술의 응용형태로서 하드웨어로 하여금 특정기능을 수행하도록 하는 일련의 명령군"으로 정의되는데 모든 정보기술응용분야 중에서도 핵심적인 위치에 있다. 정보기술의 급속한 진보로 컴퓨터의 기능이 더욱 고도화되었고 이에 따라 컴퓨터의 이용이 폭넓게 확산되고 있다. 컴퓨터 이용의 확대는 또한 컴퓨터프로그램의 발전을 촉진시켜 컴퓨터프로그램은 이제 가장 중요한 정보기술산업으로 부각되고 있다.

컴퓨터프로그램은 사회 및 산업구조의 정보화와 더불어 산업구조고도화 측면에서 뿐만 아니라 범국가적 전략산업으로서의 중요성을 갖는다. 정보화사회에서 컴퓨터프로그램산업이 갖는 의미는 공업화사회에서의 중화학공업보다는 더 포괄적이므로 컴퓨터프로그램산업에 대해서는 새로운 시각과 전략이 필요하다. 따라서 기업뿐만 아니라 정부에 대해서도 컴퓨터프로그램산업에 대한 방향감각 제시가 절실하다.

컴퓨터프로그램의 개발과정은 현저하게 다른 기술과 방법을 필요로 하는 몇가지 다른 단계의 연속이라고 할 수 있다. 이 단계는 크게 네 가지 그룹으로 나누어지는데 1) 시스템 정의 및 세분화, 2) 프로그래밍, 3) 테스팅 및 수정, 4) 보수가 그것이다.

다른 산업활동과는 달리 소프트웨어 개발과정에 있어서 보수는 새로운 요구와 상황에 대해 프로그램을 지속적으로 적응시키는 과정이기 때문에 이를 앞의 세 단계인 시스템 정의 및 세분화, 프로그래밍 및 테스팅 단계의 축소판이라고 할 수 있다. 소프트웨어 패키지의 경우 이 과정은 같은 패키지에 대한 새로운 판(version)을 내는 것으로 나타나는데, 이 때에는 전판에 비해 보다 다양하고 고급화된 기능을 부여하게 된다. 이처럼 프로그램 개발과정은 그 수명이 끝날 때까지 계속되기 때문에 라이프사이클은 보수를 포함한 위의 네 단계 모두를 지칭한다[11].

2. 컴퓨터 프로그램의 특허권 요구

상술한 바와 같이 컴퓨터가 초창기에는 단순한 계산에서 비롯되었으나 인간의 유용한 목적 달성을 위한 복잡하고 다양한 형태로 진화하여 오늘날 산업구조의 중요한 지위를 차지하게 됨에 따라 그 지적재산권 보호가 날로 중대한 문제로 부각하게 되었다. 산업기술 분야로서의 컴퓨터프로그램의 기술적 독창적 아이디어에 대한 특허출원이 증대하게 되고 이러한 현상을 수용하기 위하여 특허 법조계의 변화가 이루어지고 있다.

무궁무진한 프로그램 자원의 저작권 보호를 위하여 신지적재산권으로서의 컴퓨터프로그램 보호법 등이 수립되었지만 권리보호 영역에는 한계가 있다. 유럽공동체 이사회가 1991 년 5 월 14 일에 확정 · 공표한 컴퓨터프로그램의 법적 보호에 관한 지침에 의하면 컴퓨터프로그램의 기초가 되는 아이디어나 기본 원리들은 저작권 보호에서 제외되고, 지침 전문(前文)에 의하면 프로그램의 해법과 규칙 및 프로그램 언어가 아이디어나 기본 원리에 해당되는 한도에서 그러한 아이디어와 기본 원리는 저작권 보호에서 제외된다. 최근 미국의 Computer Associates vs. Altai[1]사건에서는, 아이디어/표현의 이분법(二分法)을 컴퓨터 프로그램에 적용하여, 1) 효율성의 필요에 의하여 규정되어진 프로그램요소들, 2) 외부적 요건에 의하여 규정되어진 프로그램요소들, 그리고 3) 공용 · 공지의 프로그램 요소들은 저작권법적 보호를 받을 수 없는 사항들이라고 판시한 바 있다. 따라서, 프로그램내의 일련의 지시(또는 구체적인 프로그램코드)가 당해 프로그램의 기능을 실현하기 위하여 긴요하고 유일한 수단에 해당되면, 아이디어와 표현이 합쳐져서 따로 보호할 프로그램표현이 없다고 할 수 있다. 특히, 프로그램 개발업자는 항상 "가장 효율적인 방법으로" 프로그램 이용자의 수요를 충족시킬 수 있는 프로그램을 개발하려 하고, 효율성을 추구하려면 필연적으로 가장 간결한 계산방법 또는 가장 논리적이고 간결한 절차를 프로그램화하여야 할 것이고, 따라서 그러한 프로그램에 있어서 아이디어와 표현이 분리되기 어려운 경우가 많을 것이다. 즉, 아이디어와 표현이 일체화(merger of idea and expression)된 부분에 한하여서는 저작권 침해가 있을 수 없게 된다. 프로그램 화면에 있어서의 유사성을 근거로 하여 저작권 침해의 소송이 제기된 사건에서, 프로그램 화면에서 프로그램 이용자가 효율적으로 커서를 움직이도록 하는 방법과 절차는 극히 한정되어 있기 때문에 그러한 프로그램 화면 운영상의 유사성은 효율성을 생명으로 하는 프로그램의 보호대상에서 제외된다고 판시된 바 있다[2]

아이디어/표현의 이분법 및 아이디어/표현 일체화의 원리가 프로그램의 보호범위 판단에 적용된 또 다른 결과로서, 프로그램이 컴퓨터 하드웨어 등의 외부적인 요인에 의하여 규정되어지는 한도에서는 저작권법적 보호를 받지 못한다[12].

이렇듯이 컴퓨터 프로그램에 대하여 저작권 보호법에 의하여 보호받는 것과 특허로서 독창적 아이디어를 보호받는 것은 본질적인 성격의 차이가 있어 양자간에 그 의미가 서로 다르며, 특허의 강

1) 1992 U.S. App. LEXIS 14305(2nd Cir., June 22, 1992)

2) Manufacturers Technologies, Inc. v Cans, Inc., 706F. Supp. 984, 995 (D. Conn. 1989)

점이 여기에 있다.

3. 특허의 컴퓨터 프로그램에 대한 제약

특허의 컴퓨터프로그램에 대한 제약은 컴퓨터 산업기술의 발달과 컴퓨터프로그램에 대한 특허성 심사기준의 변천에 따라 변화되어 왔다. 눈에 보이지 않는 기술적 창작을 담고 있는 유동적인 컴퓨터프로그램을 특허라는 법적 틀에 껴맞추자니 많은 논란과 실무 사건들이 전개되어왔다. 그래서 이 제약의 틀이 항상 일정할 수 없으며 결정적인 상황에 따라 개조되어왔다.

프로그램이 특허에 대하여 위배되는 첫번째 문제는 프로그램의 본질상 근본적으로 인간의 논리적 사고에 근거한 것이라는 것이다. 미국 특허법에서 특허는 추상적인 것이 아닌 실체일 것을 요구하며 한국과 일본의 특허요건중 성립성의 기조인 자연법칙의 이용성이 바로 이 실체성과 일맥상통한 것이라 볼 수 있다. 특허의 실체성은 바로 이 자연법칙의 이용성을 통해 나타나기 때문이다. 단지 동·서양의 사고방식에 따른 관점상의 표현이 다르다고 할 수 있다.

프로그램을 특허받기 위해서는 이 제약조건에 맞춰주어야 하므로 그 자체로서는 비정형인 프로그램이 컴퓨터라는 장치를 통하여 나타나는 실체성을 표현해주어야 하는데 이것이 복잡한 컴퓨터 프로그램의 영역과 하드웨어와의 관련성에 따라 다양하기 때문에 적잖은 표현상의 기법이 요구되기도 한다. 이러한 표현은 특허의 권리서이자 기술서인 명세서상에 나타나게 된다. 과거에는 고급 계층의 응용프로그램으로 갈수록 실체성 규명이 용이하지 않아 특허받기가 까다로왔으나 현재는 그 포문이 열려 복잡한 수준의 응용프로그램이라고 하더라도 그 특성에 따라 많은 특허권을 형성하게 되었다. 그렇다고 하더라도 단순히 프로그램 자체의 기술로서는 안되며 어떠한 형태로든 실질적인 유용 가치를 표명하여야 한다.

V. 컴퓨터 프로그램의 특허성

1. 전통적인 해석

근래에 컴퓨터의 이용기술, 소위 컴퓨터프로그램이 급격히 발달함에 따라, 가장 중요한 부분인 프로그램(전자계산기를 기능시켜서 하나의 결과를 얻을 수 있게 이것에 대한 지령을 조합한 것으로 표현한 것)을 특허대상으로 할 것인가의 여부에 대해 많은 국가에서 논의되어 특허성을 부정하는 견해와 이것을 긍정하는 견해 및 중간설 등이 대립해 왔다.

부정설: 부정설의 대표적인 것은 컴퓨터프로그램은 인간이 머리 속에서 하는 정신적 · 지능적인 수단 또는 과정(mental process, step)과 동등하므로 자연법칙을 이용하는 것이 아니라 본질적으로는 하나의 계산방법에 불과하다는 것이다.

긍정설: 한편, 긍정설의 대표적인 것은 웨어하우스설이다. 즉 프로그램되기 전의 컴퓨터는 부품의 웨어하우스(창고)에 불과하지만 프로그램은 컴퓨터에 입력되었을 때에 그 물리적인 구조의 일부가 되어 이것들의 부품을 유기적 · 일체적으로 결합시킴으로써 특정한 목적에 적합한 구체적인 장치를 만들어 내는 배선 또는 접선 수단과 동일시할

수가 있으므로 자연법칙을 이용하는 것이다라고 하는 설이다.

중간설: 프로그램의 종류에 따라 특허성을 긍정할 것과 부정할 것이 있다는 설이다. 즉 프로그램 그 자체에는 특허성이 없다고 해도 공작기계 · 발전기 등의 자동제어를 위한 프로그램처럼 제어 방법에 기술적 특징이 있으면 그와 같은 방법으로서 특허성이 있는 것으로 한다. 그 이유로서 대표적인 것은 이같은 프로그램은 직접적으로는 문제의 수학적이나 논리적 성질을 이용한다고는 해도 그같은 성질이 더욱 근본적으로는 기술적 효과인 자연법칙에 관한 성질에서 시작되는 경우가 있는데, 그 경우에는 궁극적으로는 자연법칙의 이용이라고 할 수 있다는 것이다. 기존의 특허청의 심사기준은 대체로 이 중간설에 따라왔다고 할 수 있다[7].

2. 잠재적 특허성

이상의 부문별 분석에서 살펴본 바와 같이 컴퓨터프로그램의 하드웨어와의 일체성에 따라 컴퓨터프로그램 자체로서 독립된 개체이기는 하나 특허요건상의 실체성은 하드웨어와의 결합을 통해서 나타나는 것은 프로그램의 종류를 막론하고 공통된 사실이다. 프로그램은 인간의 논리적 사고가 담겨있는 무형의 저장소이고 컴퓨터는 자연법칙에 따라 동작하는 장치이므로 실체성을 나타내는 그릇이다. 이 자연법칙을 이용한 실체를 나타내는 그릇에 프로그램이라는 창작 사상이 주입되면서 비로소 인간이 추구하는 유용의 활동을 구사하게 된다. 이것은 마치 사람이 육체와 정신으로 이루어져 있는 것에 비유할 수 있다. 인간의 육체만으로는 아무런 동작을 할 수 없으나 이 육체에 정신이 들어갈 때, 갖가지 고도의 활동들을 펼치게 된다.

따라서 컴퓨터프로그램의 어떠한 종류와는 상관 없이 기본적으로 컴퓨터와 결합되기 이전의 모든 프로그램에는 특허성이 잠재되어 있다고 볼 수 있다. 단지 다양한 프로그램의 종류에 따라서 특허성에 어떤 영향력을 지닌 변수가 있음을 추측할 수 있다. 바꿔 말해 잠재된 특허성이 일률적이라고는 할 수 없으며 프로그램의 유형에 따라서 다소간의 상호의존적인 변화가 있음을 알 수 있다. 그렇기 때문에 어떤 프로그램의 경우에는 특허받기 쉬운 반면, 어떤 프로그램의 경우에는 특허받기가 용이치 않은 성격의 차이가 나타난다. 컴퓨터프로그램의 잠재적 특허성은 하드웨어와의 결합을 통해서 나타나고 그 실질적인 사용 영역과 가치로서 실체성을 느낄 수 있다. 이 때 특허의 대상이 되는 것은 프로그램 자체를 독립해서 놓고 보는 것이 아니라 컴퓨터와의 일체로서 그 안에 담겨있는 독창적인 기술적 사상을 보호하는 것이다.

3. 컴퓨터 프로그램의 계층성에 의한 영향

앞서 언급한 바와 같이 컴퓨터프로그램의 실체성은 하드웨어를 통해서 나타난다. 그런데 컴퓨터프로그램의 특성상 하드웨어와의 연관관계에 있어서 긴밀도에 따른 계층성이 있으므로 컴퓨터프로그램의 특허성인 실체성을 나타내는 데에 난이도의 차이가 있다. 하드웨어 레벨에 가까운 운영체제와 같은 프로그램은 하드웨어와의 상호작용 관계에 있어서 하드웨어를 제어하는 성격이 강하므로 자연법칙을 이용한 실체성을 표명하기가 용이한 만큼 특허성이 강하게 된다. 그러나 운영체제를 통

하여 하드웨어의 동작에 관여하는 응용프로그램들의 경우, 하드웨어와의 직접적인 관련성을 표명하기가 용이치 않은 경우가 있어 특허성이 감쇄하게 된다. 이러한 특성은 컴퓨터프로그램을 포함한 1996년 상반기 미국의 소프트웨어 특허의 종류별 분포 통계를 볼 때에도 그 전체적인 흐름상에 적용된 영향력을 알 수 있다. 그 통계분포는 <표 3>과 같다[13]. <표 3>에서 네트워크 및 통신 관련 소프트웨어는 매우 넓은 범위를 허용하고 있다. ISDN과 인터액티브 텔레비젼, 개인 통신, LANs/WANs/GANs, 그리고 전자적으로 다자간 통신을 수행하는 모든 범주가 이 영역에 해당한다. 더군다나 근래에 접어들면서 네트워크과 통신 기술은 CDMA를 이용한 이동통신, 인터넷 접속, 초고속 정보통신망, 광대역 네트워크, 인트라넷, 셀룰라폰, 위성통신, 케이블 TV, 그룹웨어, PC 통신, 전자우편, 윈도즈 NT 등 다양한 분야에서 각광을 받고 있다. 또한, 특허성의 관점에서 네트워크이나 통신 소프트웨어는 주로 대형 시스템이나 처리 공정과 신호체계 상에서 동작하기 때문에 특허받기에 유리하다.

<표 3> 1996년 상반기 소프트웨어 특허 통계

No.	소프트웨어 종류	특허 건수	점유율	No.	소프트웨어 종류	특허 건수	점유율
1	Networking and communication	919	14.14%	25	Geophysical	66	1.02%
2	Image processing	528	8.12%	26	Distributed processing	64	0.98%
3	Automobiles and vehicles	421	6.48%	27	Compression	63	0.97%
4	Operating systems	404	6.21%	28	Object oriented	63	0.97%
5	Graphics	365	5.61%	29	Character recognition	60	0.92%
6	Process control	291	4.48%	30	Speech recognition	55	0.85%
7	Graphical user interface	289	4.45%	31	Robotics	52	0.80%
8	Signal processing	232	3.57%	32	Music	49	0.75%
9	Medical	225	3.46%	33	Algorithms	48	0.74%
10	Engineering	215	3.31%	34	Artificial intelligence	47	0.72%
11	Databases	193	2.97%	35	Neural networks	44	0.68%
12	Security	190	2.92%	36	Numerical analysis	43	0.66%
13	Office automation	174	2.68%	37	Games	39	0.60%
14	Computer aided engineering	157	2.42%	38	Natural language analysis	35	0.54%
15	Computer aided software engineering	136	2.09%	39	Parallel processing	30	0.46%
16	Navigation/GPS	130	2.00%	40	Education	30	0.46%
17	Finance	117	1.80%	41	Multiprocessing	28	0.43%
18	Entertainment	115	1.77%	42	Fuzzy logic	27	0.42%
19	Computer aided design	94	1.45%	43	Simulation	20	0.31%
20	Physics	94	1.45%	44	Virtual reality	18	0.28%
21	Chemistry	85	1.31%	45	Vision	11	0.17%
22	Biology	81	1.25%	46	Spreadsheet	5	0.08%
23	Word processing	74	1.14%	47	Genetics	3	0.05%
24	Pattern recognition	72	1.11%	계		6,501	100%

세부 소프트웨어 항목들의 정확한 계층성의 구분은 어렵지만 전체적인 흐름상에서 오퍼레이팅 시스템을 비롯한 하드웨어의 제어 성격이 강한 소프트웨어들이 상위 점유률에 분포되어 있으며 시스템 자원의 관련보다는 컴퓨터 자체 레벨에서 동작하는 하드웨어의 단순 사용의 성격이 강한 프로그램들은 하위 점유률에 분포되어 있다.

VI. 결 론

컴퓨터프로그램의 특허성은 컴퓨터프로그램 자체의 상태에서는 잠재해 있으며 하드웨어와의 결합을 통해서 나타난다. 컴퓨터프로그램은 독립적인 무형의 인간의 논리적 사고를 표현한 존재이나 그 자체만으로는 의미가 없으며 컴퓨터와 일체가 됨으로써 인간 생활에 유용한 목적 달성을 이룰 수 있다. 하드웨어 장치는 자연법칙에 의하여 동작하는 것이고 여기에 컴퓨터프로그램이 탑재되어 필요한 기능을 수행하는 것이므로 하드웨어와 일체로서 컴퓨터프로그램의 특허성이 나타난다.

그런데 다양한 컴퓨터프로그램의 종류에 따라 특허성의 결정에 차이가 있는 것은 컴퓨터프로그램의 특성상 하드웨어와의 관련성에 따른 계층적 성격에 기인한 것이다. 계층적 성격이란 컴퓨터프로그램의 하드웨어와의 상호 작용 관계에 있어서 하드웨어에 밀접한 레벨에서 동작하는 프로그램이 있는가 하면 하드웨어 상에서 동작은 하더라도 사용자의 편의성에 편향되어 하드웨어 레벨은 이용자의 입장에서 감추어 보이고 응용프로그램의 레벨에서 동작하여 하드웨어를 직접 제어하는 프로그램의 상위 계층에서 동작하는 프로그램들이 있다.

컴퓨터프로그램의 특허성을 표명하기 위해서는 그 실체성을 나타내어야 하는데 그것이 곧 하드웨어와의 관련성을 어떻게 구사하느냐에 달려있는 문제이며 이 때에 프로그램의 계층성에 따라 그 난이도에 차이가 있게되고 이것이 특허받기 위하여 필요한 특허성에 영향을 미치게 되는 것이다.

참 고 문 헌

[1] 今井賢一, 소프트웨어 진화론, NTT출판주식회사, 1989, pp.16-17.

[2] Allen Macro, *Software Engineering – Concepts and Management*, Prentice Hall, 1990, p.15.

[3] John L. Hennessy and David A. Patterson, *Computer Organization and Design – The Hardware/Software Interface*, Morgan Kaufmann Publishers, 1994, p. 8.

[4] 과학기술처 정보산업기술 담당관, "컴퓨터프로그램의 법적 보호현황 및 대책," 한국전자통신연구소, 1987. 4., p. 1.

[5] 한빛지적소유권센터 편집실, 수험용 특허법전, (주)한빛지적소유권센터, 1997. 8., p. 6.

[6] Roger E. Schechter, *Selected Intellectual Property and Unfair Competition*, West Publishing Co., 1993, p. 366.

[7] 吉藤幸朔, 특허법 개설, 대광서림, 1993, pp.52-54, 134.

[8] Thomas L. Crisman, "The Patentability of Software," *J&G Intellectual Property Law Section*, http://www.jenkens.com/software.htm (current Nov. 1997).

[9] 이상무, 소프트웨어특허의 신종류, 전자통신동향분석, 제12권, 제5호, 한국전자통신연구원, 1997. 10., p. 107.

[10] J. F. Villella, Jr., "컴퓨터 S/W의 특허성에 관한 미국의 최근 동향," '96 컴퓨터 S/W 특허세미나, 특허청, 1996, p. 39,43,

[11] 김용욱, 김휘석, 김태진, 권업, 소프트웨어산업의 구조와 발전방향, 산업연구원, 1987, p. 21, 27.

[12] 정상조, "컴퓨터프로그램 보호의 국제 신동향과 대응전략," 소프트웨어 재산권보호의 신조류와 대응 세미나, 과학기술처·한국정보산업연합회, 1992, pp. 29-36.

[13] Gregory Aharonian, "1996 Half Year Software Patent Statistics," http://www.rw.lv/home/friends/mails.htm (current Nov. 1997).