

공개 소프트웨어 가이드

A Guide to Open Source Software

2006 1



정보통신부

요 약 문

1. 제 목

"공개소프트웨어 가이드" 발간

2. 발간의 목적 및 필요성

본 가이드의 목적은 공개소프트웨어 대한 이해를 높이고 공개소프트웨어에 대한 우려와 현실을 객관적으로 분석하여 공개소프트웨어 도입시 필요한 검토사항 및 위험요소 분석, 관리방안 등을 수립하는데 도움을 주기 위함이다.

3. 주요 내용

제1장 "서론"에서는 공개소프트웨어의 이해를 높이기 위한 부분으로써 공개소프트웨어의 정의, 역사, 그리고 비공개소프트웨어와의 비교분석을 다루고 있으며 또한 현재 국내 공개소프트웨어 비즈니스 모델에 대한 소개와 관련업체들에 대해 다루고 있다. 1950년대부터 본격적으로 시작된 소프트웨어는 그 당시의 여러 가지 의미와 개념들이 지금과는 사뭇 다른 모습을 하고 있다는 것을 알게 된다.

공개소프트웨어에 대한 정확하고 객관적인 정의와 소프트웨어 산업적인 측면의 의미를 이끌어 내기 위하여 공개소프트웨어를 역사적인 관점으로 정리하였다. 그리고 공개소프트웨어 또는 자유소프트웨어 그리고 비공개소프트웨어, 독점소프트웨어, 사유소프트웨어 등 혼란스러운 여러가지 용어들에 대하여 설명하고 이를 간략하게 정리하였다.

또한 비공개소프트웨어와의 차이점을 비교분석하고 분석결과를 간략하게 도표로 정리하였으며 공개소프트웨어를 보다 객관적으로 이해하는데 도움이 되도록 하였으며, 공개소프트웨어 기반의 비즈니스모델을 소개하고 국내 공개소프트웨어 관련 산업에 대해 간략하게 분류하고 소개하였다.

제2장에서는 공개소프트웨어에 대한 불필요한 오해들에 대하여 사실과 진실을 객관성 있게 전달하고 있으며 크게는 공개소프트웨어의 비용에 대한 부분, 기술지원 및 가용성 부분, 소프트웨어 신뢰도 부분 그리고 완성도 및 수명의 4가지 부분으로 나누어서 오해에 대한 진실을 정확하게 전달하고 있다.

공개소프트웨어의 라이선스 비용에 대해서는 주로 도입 시 발생하는 라이선스 비용과 총소유비용(TCO)에 대해 잘못 이해할 수 있는 부분을 설명하였다. 특히 "자유소프트웨어"라는 용어의 "자유"라는 단어에서 비롯되는 여러 가지 불필요한 오해들(경제적인 무비용(free)이라는 의미와 자유롭다는 의미의 자유 등에 대한 오해들)에 대하여 보다 정확한 의미전달을 하고 있다. 또한 공개소프트웨어의 사상(자본주의와 공산주의)적인 오해에 대해서도 간략하고 분명하게 진실을 이야기하고 있다.

그리고 공개소프트웨어의 기술지원 및 가용성 부분의 불필요한 오해들에 대해서도 진실을 전달하고 있으며 특히 기술지원부분에서 무조건적으로 신뢰하지 않는다는 것은 그 만큼 공개소프트웨어에 대해 정확히 인지하지 못하였다는 것을 의미하며 인지하지 못한 만큼 또한 오해하고 있었다는 것을 의미한다.

제3장 "공개소프트웨어 도입유형 및 절차"에서는 3가지 도입유형(직접선택, 간접공급, 내부개발)을 설명하였으며 특히 현실적으로 가장 일반적이고 보편적인 도입방법인 간접공급에 대해서 특징, 장점, 단점 등을 깊이 있게 설명하고 있다.

그리고 일반적인 정보화사업 추진절차에 준하는 시스템 및 소프트웨어 도입절차에 따른 표준화, 객관화를 위한 검토사항들을 설명하고 있다. 특히 소프트웨어 도입 시에 제안요청서상의 특정 운영체제 명시, 특정 스펙명시, 특정 조건을 제시하는 등의 비표준, 불공정 소지가 있는 부분에 대하여 객관성과 공정성 유지 방안을 제시하, 공공기관의 정보시스템 구축시 필요한 고려사항과 실제 공개소프트웨어 기반으로 시스템 적용에 필요한 권고사항을 설명하였다.

또한 공개소프트웨어를 도입하기로 결정하였을 경우에 어떠한 부분을 고려해야하는가에 대한 "공개소프트웨어 도입 시 고려사항"들을 현실적인 부분을 고려하여 10여 가지 부분으로 나누어 자세히 설명한다.

제4장 "위험분석 및 관리"편에서는 공개소프트웨어 도입에 따르는 위험요소 분석과 관리방법들을 다루고 있으며 "경쟁 유효성", "보증문제", "소스코드 가용성"이라는 3가지 부분의 위험요소를 분석하고 이를 관리하는 방안을 제시하였다. 경쟁 유효성 부분에서는 공급업체의 신뢰성과 경쟁력 확보여부, 공개소프트웨어 자체의 경쟁력 확보여부, 기능성 및 사용 편의성 등에 대한 경쟁력 확보여부, 비용절감의 경쟁력 확보 여부 등을 주로 다루었다.

그리고 보증문제는 라이선스와 법적인 문제까지 결부될 수 있는 매우 민감한 사안으로써 보증의 결여에 따른 도입기관의 대처문제, 도입하는 공개소프트웨어의 라이선스 문제 등에 대해서도 상세히 설명하였다.

그리고 도입 시 소스코드를 확보해야하는 것은 물론 소스코드의 확보와 함께 수정 및 재사용, 재배포 권한 문제등도 함께 검토되어야 한다. 도입하는 공개소프트웨어의 소스코드를 확보하지 못하였거나 수정권한을 얻지 못하였거나 재사용권한을 확보하지 못하였을 때에는 패치 및 업그레이드를 하지 못하는 상황이 발생 가능하다. 결국 이로 인하여 확장성, 영속성, 안정성 등이 보장되지 않을 수도 있다.

그리고 위험 관리방법과 위험 완화방법을 "도입 유형별 위험평가", "기술로드맵 평가", 위험완화를 위한 검토사항"으로 나누어 설명하였다. 특히 직접선택 방법에서는 직원들의 개발능력이 검토되어야하며 프로젝트 수행경력과 능력이 있는 내부직원이 확보되어야 한다. 반면 간접공급방법에서는 공급업체의 신뢰성과 라이선스 문제를 고려하여야 한다. 정부부처 등 공공기관은 간접공급 방식으로 도입하는 것이 가장 안정적인 공개소프트웨어 도입방식이며 이 때 비용이 발생하는 점을 고려해야 하며 이때, 공급업체의 기술지원 능력과 서비스 능력도 함께 고려되어야 한다.

그리고 위험요소에 따른 위험완화를 위한 검토사항들이 있으며 이들에 대하여 본문에서는 10 여가지의 검토사항들을 제시하고 있으며 공개소프트웨어를 도입하기로 하였다면 최소한 10 여가지의 검토사항들은 기본 점검을 해야 한다.

제5장에서는 공개소프트웨어의 법률문제를 다루었다. 공개소프트웨어와 관련된 법률문제는 주로 라이선스로부터 발생하는 것이기 때문에 GPL, LGPL, BSD License, MPL 등 몇 가지 주요 라이선스에 대하여 알아두어야 한다. 그리고 이들 라이선스가 적용되었을 때에 발생하는 여러가지 법적인 문제들에 대해서도 설명하였으며 도입 후 재배포에 따른 유형별 법적문제들을 설명하고 있다.

공개소프트웨어의 라이선스 종류가 지나치게 많아지고 있다는 점을 사용자들은 매우 혼란스러워하며 또한 이에 대한 위험요소를 가중시킨다고 불평하기도 한다. 하지만 라이선스의 수적 증가는 현실적인 요구에 의한 것으로서 공개소프트웨어가 활성화 될수록 이 요구는 더욱 많아질 것이다. 이에 대한 OSI가 관리하고 있는 라이선스 리스트에 추가되는 것을 현재로서는 막을 방법이 없다.

이보다 더한 위험요소는 법적문제 발생에 따른 GPL의 해석과 적용 범위의 문제이다. 즉, 계약의 성립여부와 그 계약의 유효성, 저작권과 계약의 관계, 준거법과 재판관할, 그리고 저작인격권, 전송권 등에 대하여 해석과 적용범위를 명확히 해야 할 필요가 있다. 그리고 보증문제에 따른 법적문제도 흔히 발생하는 것으로 도입 시에 주의해야 할 사항이다.

그리고 공개소프트웨어를 도입하고 이를 수정하여 확장적용하고 타기관에 재배포하는 각각의 시나리오에 따른 법적문제도 검토되어야 한다.

4. 기대효과 및 활용

공개소프트웨어 도입과 활용은 점점 늘어가고 있다. 반면 도입기관에서는 공개소프트웨어의 정확한 이해가 부족할 뿐 아니라 공개소프트웨어의 도입방법과 그에 따르는 위험요소, 그리고 완화방법, 뿐만 아니라 공개소프트웨어의 라이선스 문제와 그에 따른 법적인 문제들 파악하고 싶어도 이에 대한 전문 안내서가 부족한 실정이다. 따라서 이에 대한 필요성이 제기되어 왔었다. 본 안내서는 이러한 요구환경을 배경으로 제작되었으며 객관적인 시각을 통해 바라본 공개소프트웨어의 진실들을 있는 그대로 전달하고자 노력하였다.

본 안내서는 기관들의 공개소프트웨어 도입에 따른 사전 지식과 도입 후에 필요한 관리지식뿐 아니라 법적인 문제 발생 예방과 문제발생시 대처능력을 확보하고 무엇보다 공개소프트웨어의 도입결정에 따른 유용한 안내서 역할을 충실히 할 것으로 기대한다.

<제목 차례>

1장. 서 론	1
1. 공개소프트웨어의 이해	3
가. 공개소프트웨어의 기원과 역사	3
나. 공개소프트웨어 정의	7
다. 비공개소프트웨어와의 비교	8
2. 공개소프트웨어 산업 개관	13
가. 공개소프트웨어 비즈니스 모델	13
나. 공개소프트웨어 산업 현황	14
2장. 공개소프트웨어에 대한 오해와 진실	21
1. 비용 문제	23
2. 기술지원 및 가용성	25
3. 공개소프트웨어의 신뢰도	30
4. 완성도 및 수명	34
3장. 공개소프트웨어 도입유형 및 절차	37
1. 도입유형	39
가. 직접선택	42
나. 간접공급	45
다. 내부개발	48
라. 종합비교	52
2. 도입절차	53
3. 적용시스템	57
가. 시스템 유형	57

나. 시나리오별 도입 방법	59
다. 적용 가능분야	62
4. 고려사항	64
4장. 위험분석 및 관리	70
1. 위험요소 분석	72
가. 경쟁 유효성	72
나. 보증문제	74
다. 소스코드의 가용성	78
2. 위험관리 및 완화방법	80
가. 도입유형별 위험평가	80
나. 기술로드맵 평가	86
다. 위험완화를 위한 검토사항	89
5장. 법률문제	93
1. 라이선스 유형	95
가. GPL	95
나. LGPL	98
다. MPL	98
라. BSD License	95
2. 법적문제	100
가. 라이선스의 위험성 평가	101
나. GPL의 해석과 적용범위	103
다. 계약의 성립여부 및 유효성	105
라. 저작권과 계약의 관계	105
마. 특허	106
바. 상표	106

사. 보증 책임의 범위	107
아. GPL로의 소스코드 공개방법	107
3. 사용유형별 법적문제 검토	108
참 고 문 헌	113
부록1. 신뢰성이 높은 공개소프트웨어	121
부록2. 비공개 소프트웨어 대 공개소프트웨어 안내	129
부록3. 공개소프트웨어 리소스	140
부록4. GNU 일반 공중 사용 허가서(2판)	142

<표 차례>

<표 1> 비공개소프트웨어와 공개소프트웨어 비교	11
<표 2> 공개소프트웨어와 상용소프트웨어 비교	12
<표 3> 공개소프트웨어 도입 절차	44
<표 4> 직접선택방식 도입 전 체크리스트	45
<표 5> 공개SW도입방식에 따른 비교	52
<표 6> 공개SW도입을 위한 주요 검토 사항	53
<표 7> 비표준, 특정기술 조건 예	54

<그림 차례>

<그림 1> 공개소프트웨어 간접공급 유형	46
<그림 2> 웹기반 정보시스템 구조	59
<그림 3> 웹서버 운영체제 공개SW 도입	60
<그림 4> 웹서버, 웹 어플리케이션 서버 운영체제 공개SW 도입	61
<그림 5> 3계층 서버 운영체제 공개SW 도입	62

1장. 서론

BLANK

1. 공개소프트웨어의 이해

가. 공개소프트웨어 기원과 역사

공개소프트웨어는(Open Source Software) 1984년 자유소프트웨어(Free Software)운동으로부터 분화하여 현재에 이르렀으며¹⁾, 2000년 이후 전 세계의 정부기관과 민간단체에서는 공개소프트웨어의 효율성과 발전성에 대하여 확고한 의지를 가지고 정책추진과 사업화를 시도하고 있다. 국내에서도 2002년부터 공개소프트웨어 정책추진을 위한 본격적인 사업을 착수하여 현재 그 성과를 보이고 있다.

자유소프트웨어 운동(Free Software Movement)은 1984년 리차드 스톨만(Richard Stallman)이 GNU²⁾라는 공개프로젝트를 시작하면서 부터 시작되었으며 리눅스와 함께 자유소프트웨어 운동도 발전하였는데, 이 과정에서 "자유(Free)"라는 용어에 대한 실용주의적 관점에서의 비판이 이루어지고 공개소프트웨어 운동(Open Source Software Movement)이 시작되었다.

자유소프트웨어 운동 진영은 소프트웨어 자체는 항상 윤리적, 도덕적, 사회적으로 타당해야함을 강조하면서 소프트웨어 사용자들에게 이를 바탕으로 한 사용상의 여러 가지 자유³⁾가 주어져 있음을 또한 강조하고 있다⁴⁾.

1) 자유소프트웨어와 공개소프트웨어는 그 역사를 함께 해오고 있으며 같은 개념으로 보는 견해도 있다. (<http://www.debian.org/intro/free> 참조)

2) GNU는 "Gnu is not Unix"의 재귀적 표현임

3) 여기서 자유는 첫째, 어떤 목적으로도 프로그램을 가동시킬 수 있는 자유 둘째, 필요에 맞게 소프트웨어를 수정할 자유, 셋째, 무료 또는 유료로 복사본을 재배포할 수 있는 자유 넷째, 전체 공동체가 혜택을 볼 수 있도록 프로그램의 수정본을 배포할 수 있는 자유를 의미한다.

그리고 공개소프트웨어 운동 진영은 이념적 측면보다는 개발방식에 초점을 맞춘 접근을 취하고 공개소프트웨어 개발자에게도 경제적인 보상을 제공할 수 있어야 한다는 실용적 측면을 강조한다.

자유소프트웨어는 원칙에 입각한 소스코드의 자유로운 사용이라는 측면을 공개소프트웨어는 사회적 실용성과 경제성에 중심을 두고 소스코드의 공개라는 측면을 강조한다고 할 수 있다.

공개소프트웨어가 정부의 정책에 반영되기 시작하고 일반인들에게 알려지기 시작한 것은 불과 몇 년 전 부터이다. 공개소프트웨어가 현재에 이르기까지 어떻게 진화해 왔는가를 살펴보려면 먼저 그 역사를 살펴보아야한다. 공개소프트웨어에 관련된 주요한 사건들을 년대별로 간략하게 정리하였다.

o 1950년대

1950년대는 컴퓨터 프로그램의 초기 사용시기로써 모든 프로그램은 공개적으로 개발되고 또한 소스코드도 공개되었다. 즉, 여기서 중요한 것은 컴퓨터 프로그램의 역사는 공개적으로 개발되기 시작하였고 또한 개발된 소스코드는 공개되었다는 점이다. 이러한 초기 인식이 약 10년 정도 계속되었으나 1950년대 후반부터는 컴퓨터 프로그램이 일부이기는 하지만 판매되기 시작하였다.

o 1960년대

판매되기 시작한 컴퓨터 프로그램이 1960년대 후반에 와서는 소프트웨어라는 상품으로 포장되어 정식으로 판매되는 상품으로 인식되기

4) Richard Stallman, Why "Free Software" is better than "Open Source", <<http://www.gnu.org/philosophy/free-software-for-freedom.html>>

시작하였다. 그리고 이때부터 소프트웨어는 개발한 회사의 일급비밀이 되어 소스코드를 공개하지 않게 되었다. 하지만 이때에도 UNIX를 포함한 공개소프트웨어는 소스코드가 계속 공개되고 있었다. 즉, 1960년대 중 후반기에는 공개소프트웨어와 비공개소프트웨어가 공유한 시기라고 볼 수 있다.

o 1970년대

1970년대도 UNIX 소스코드는 많은 대학에서 자유로이 사용하고 수정하고 할 수 있었다. UNIX에서 사용될 수 있는 많은 공개소프트웨어들도 자유로이 수정되고 사용되었다.

o 1980년대

1980년대에 들어서면서 상용UNIX가 대두되기 시작하여 UNIX에 대한 소유권을 가지고 있었던 AT&T는 수수료를 높이고 그 배포를 제한하게 되었다. 이로써 UNIX 소스코드의 수정본을 공유해왔던 많은 사용자들이 소프트웨어를 수정하는 것을 불가능하게 되었으며, 대학에서도 UNIX에 라이선스를 적용하기 시작하였다. 즉, 1980년도부터 공개소프트웨어의 개념이 희박해지면서 독점소프트웨어가 본격적으로 출현하여 소프트웨어를 완전히 제품화하여 유통하고 판매한 시기라고 할 수 있다.

공개 및 공유되어 왔던 소프트웨어 소스코드가 완전히 상품화, 사유화, 제품화되어 기업의 수익을 위한 상품화된 것에 회의를 느낀 MIT의 리처드 스톨만(Richard Stallman)은 "소프트웨어는 공유되어야 한다."라는 소신을 가지고 그 뜻을 펼치기 위하여 1984년도에 GNU라는 프로젝트를 시작하였다.

자유소프트웨어를 보호하고 자유소프트웨어 운동을 본격적으로 촉

진하기 위하여 리처드 스톨만은 1985년에 자유 소프트웨어재단, 즉, FSF(Free Software Foundation)을 설립하였다. 자유소프트웨어 운동이란 모든 소프트웨어를 누구든지 자유로이 사용할 수 있고, 필요에 따라 변경을 할 수 있으며, 변경된 소프트웨어를 자유로이 배포할 수 있게 하기 위함이 목적이며 이를 위해서는 소프트웨어의 소스코드 공개가 필수적이라는 것이다. 그리고 이를 실현하기 위하여 GPL이라는 자유소프트웨어 라이선스를 만들었다.

o 1990년대

1991년에 핀란드의 대학생인 리누스 토발즈가 80386칩에 최적화된 LINUX라는 시스템 커널을 개발하였다. 그리고 이 소스에 GPL라이선스를 부여하여 GNU프로젝트에 의해 개발될 수 있도록 공개하였다. 이때부터 LINUX 커널은 전 세계의 수많은 개발자들과 함께 GNU프로젝트 하에서 이미 개발되었던 많은 프로그램 및 유틸리티들과 결합되었으며 매우 안정적인 운영체제로 놀라울 정도로 빠른 발전을 하게 되었다.⁵⁾

GNU 프로젝트의 많은 유틸리티들과 합쳐져, GNU/Linux운영체제가 탄생하게 되었으며 드디어 1996년도에 리눅스 커널 2.0이 발표되면서 Linux 운영체제는 완전한 운영체제로 인정을 받게 되었다.

한편 자유소프트웨어 운동 내부에서 이상주의적이고 도덕적인 스톨만의 입장에 대해서 실용주의적 관점에서 비판이 이루어지기 시작하였다. 에릭 레이몬드(Eric Raymond)는 리눅스를 개발하는 방식이 왜 성공적인 결과를 낳았는지를 분석하고 그 결과를 발표하였다.⁶⁾

5) 리누스 토발즈는 운영체제의 일부분인 Linux커널의 개발을 시작한 선도개발자이며, 리누스 토발즈가 개발을 시작한 커널에 수많은 GNU프로젝트와 다른 프로젝트들의 결과물들이 합해져서 결국 완성된 하나의 Linux운영체제가 개발된 것이다.

6) 리누스 토발즈가 리눅스를 만들었다는 사실보다 불완전한 소프트웨어를 자주 빠르게 발표하여 수 많은 사람들이 버그를 수정하는 ‘리눅스 개발 모델’을 만든 것에 큰 의미를 부여한다.(Eric S. Raymond, The Cathedral and The Bazaar 참조)

1998년 넷스케이프사가 레이몬드의 논리를 수용하여 ‘넷스케이프 커뮤니티케이터’의 소스코드를 공개한다고 발표하였다. 레이몬드와 그의 주장에 동의하는 사람들은 공개소프트웨어 운동을 관리하기 위해 Open Source Initiative라는 조직을 결성하고 자유소프트웨어의 이념적 측면보다는 그것이 개발되는 방식에 초점을 맞추는 실용적 접근을 취하는 이들은 자유소프트웨어는 산업계는 물론이고 개발자에게도 경제적인 보상을 제공할 수 있어야 한다고 주장하였다.

나. 공개소프트웨어 정의

공개소프트웨어(Open Source Software)란 소스코드⁷⁾를 공개한 상태로 실행프로그램을 제공하는 소프트웨어로서, 소스코드를 누구나 자유롭게 사용·개작·재배포할 수 있도록 허용한 소프트웨어이다.

공개소프트웨어는 누구라도 소스코드를 읽을 수 있고 사용자가 능력이 있다면 각종 버그의 수정은 물론이고 그것을 개작하여 기능을 추가할 수 있으며, 누구나 소프트웨어 개발에 참여할 수 있다. 따라서 공개소프트웨어는 소스코드에 접근할 수 있는 권리, 프로그램을 복제하여 배포할 수 있는 권리, 프로그램을 개선할 수 있는 권리를 개발자에게 보장한다.

이러한 공개소프트웨어는 소스코드에 대한 접근성이 보장되므로 시스템간의 호환성을 확보할 수 있을 뿐 아니라 사용자의 요구에 부합하는 일관성과 함께 일치성을 보장받을 수 있다.

7) 소스코드는 사람이 읽을 수 있는 형식의 프로그램 코드를 말하며 실행코드 또는 바이너리 코드는 컴퓨터가 인식하고 실행할 수 있는 형식의 코드를 의미한다.

OSI(Open Source Initiative)에서 정의하는 공개소프트웨어 조건은⁸⁾ 자유배포(Free Redistribution), 소스코드 공개(Source Code Open), 2차적 저작물(Derived Works), 소스코드 수정제한(Integrity of The Author's Source Code), 개인이나 단체에 대한 차별 금지(No Discrimination Against Persons or Groups), 사용분야에 대한 제한 금지(No Discrimination Against Fields of Endeavor) 등 10개 항목이 있다.

다. 비공개소프트웨어와 비교

1) 비공개SW와 비교

앞서 공개소프트웨어에 대한 역사와 정의에 대하여 알아보았다. 이제 공개소프트웨어에 대한 현실적인 활용가치와 기능성 등을 확인하기 위하여 비공개소프트웨어와 공개소프트웨어를 차이점을 살펴보면 다음과 같다.⁹⁾

첫째, 가장 대표적인 차이점으로서 비용면에서의 차이점이다.

비공개소프트웨어는 라이선스 비용이 발생하는 반면 공개소프트웨어는 라이선스 비용은 발생하지 않으나, 기술지원이나 서비스에 대한 수수료는 발생한다. 또한 비공개소프트웨어는 시스템에 대한 초기 적용비용이 높지만 공개소프트웨어는 초기 적용비용이 매우 낮다는 점

8) 자세한 내용은 www.opensource.org 및 한국소프트웨어진흥원, 오픈소스소프트웨어연구보고서, 2002 참조

9) 공개소프트웨어는 소스코드의 공개, 제작, 재배포가 자유로운 소프트웨어를 의미하는 것이며, 비공개소프트웨어는 이러한 권리가 제한된 소프트웨어를 의미한다. 공개소프트웨어를 상용소프트웨어와 반대되는 개념의 소프트웨어로 이해하는 경우가 있으나, 이는 공개소프트웨어를 무료 소프트웨어로 잘 못 이해하고 있기 때문에 생긴 오해이다. 공개소프트웨어도 유료 즉 상용으로 배포할 수 있다.

이다.

둘째, 성능면에서의 차이점이다.

비공개소프트웨어의 경우 전문가가 직접 설치 및 최적화 설정을 하기 때문에 규모가 큰 시스템 환경에서 널리 사용되어 왔다. 공개소프트웨어의 경우 최적화가 이루어지지 않은 경우가 많으나, 공개소프트웨어의 경우에도 전문가를 통해 최적화하여 설치한다면 비공개소프트웨어의 경우와 같이 규모가 큰 시스템 환경에서 사용될 수 있을 정도의 성능을 보여준다. 또한 버전이 올라가면서 점차적으로 우수한 성능을 보여주기도 한다. 대표적으로 공개소프트웨어 운영체제인 리눅스의 경우 커널 2.6.X 이상에서는 중대형 시스템에서 안정적 성능을 나타내고 있으며 이러한 추세는 앞으로 더욱 활발해 질 것으로 예상된다.

셋째, 보안면에서의 차이점이다.

비공개소프트웨어는 프로토콜의 호환이 어려워 인증체계가 취약하며 폐쇄적인 운영으로 인해 공개되지 않는 약점이 존재할 수 있다. 반면 공개소프트웨어는 개발 시부터 공개되어 이미 많은 취약점이 개선 및 해결되었으며 안정된 상태에서 운영되고 있다.

넷째, 기술적인 면에서의 차이점이다.

비공개소프트웨어는 프로젝트의 연속성 및 재사용성이 낮은 반면 공개소프트웨어는 소스코드의 공개로 인하여 언제든지 재사용이 가능하기 때문에 프로젝트의 영속성이 보장되며 유지보수 및 업그레이드 비용이 낮다.

다섯째, 확장성에 대한 차이점이다.

비공개소프트웨어는 높은 적용비용과 제한된 시스템 운영환경으로

인하여 확장성이 완벽하게 보장되는 것은 아니다. 하지만 공개소프트웨어는 소스가 공개되어 있기 때문에 시스템의 확장성이 확실히 보장된다는 장점이 있다.

여섯째, 공급권에 대한 차이점이다.

비공개소프트웨어의 생산업체는 하나이므로 독점 및 단일공급 된다. 단, 유통회사는 여러 개 존재할 수 있지만, 원 공급 업체는 하나이다. 반면 공개소프트웨어는 동일한 솔루션에 대하여 다수 업체들로 부터 공급이 가능하기 때문에 사용자의 공급업체 선택권이 넓은 편이다.

일곱째, 저작권에 대한 차이점이다.

비공개소프트웨어는 일반 라이선스를 채택하고 있는 반면 공개소프트웨어는 거의 대부분 GPL 라이선스, BSD 라이선스, MPL 라이선스 등을 원하는 대로 적용할 수 있다.

여덟째, 경쟁력에 대한 차이점이다.

비공개소프트웨어는 핵심 소프트웨어에 대한 개발능력 및 저작권을 이미 보유하고 있는 소프트웨어 선진국에 유리한 소프트웨어이다. 반면 공개소프트웨어는 소프트웨어의 핵심기술 공유할 수 있는 기회가 있기 때문에 신흥 소프트웨어 개발국에 유리하다.

비공개소프트웨어와 공개소프트웨어의 차이점을 요약하면 다음 표와 같다.

<표 1> 비공개소프트웨어와 공개소프트웨어 비교

구 분	비공개소프트웨어	공개소프트웨어
비용분석	<ul style="list-style-type: none"> 적용비용이 높음 유지비용및 시스템 개선비용이 높음 소수의 관리자에 대한 관리비용 높음 라이센스에 대한 비용발생 	<ul style="list-style-type: none"> 적용비용이 낮음 유지비용이 낮고 기능 확장에 대한 추가비용이 낮음 관리비용이 낮음 서비스에 대한 비용발생
성능분석	<ul style="list-style-type: none"> 규모가 큰 시스템 환경에서 비교적 높은 성능 고가의 장비로 인한 고성능 전체적으로 공개SW와 비슷한 성능 	<ul style="list-style-type: none"> 다양한 환경에 최적화된 설정으로 높은 성능 높은 안정성 및 비용효율이 높음
보안성	<ul style="list-style-type: none"> 폐쇄적인 운영으로 인한 공개되지 않은 시스템 취약점 보유 최근에 다수의 취약점 발견으로 보안 위협 노출 프로토콜 호환이 어려워 인증체계 취약 	<ul style="list-style-type: none"> 개발시부터 공개되어 이미 많은 취약점이 해결된 안전화 상태 공개키 기반의 인증 매카니즘 구현을 위한 통합패키지존재 다양한 암호화 알고리즘 및 키 관리에 대한 기능 제공
기술성	<ul style="list-style-type: none"> 재사용성 없음 	<ul style="list-style-type: none"> 재사용성 높음 유지보수,업그레이드용이 독점폐해방지
저작권	<ul style="list-style-type: none"> 일반라이선스 독과점에 의한 가격결정우려 	<ul style="list-style-type: none"> GPL라이선스 BSD라이선스등
확장성	<ul style="list-style-type: none"> 서버의 가용성 측면에서 클러스터링 비효율성 소프트웨어간의 호환성이 보장되거나 높은 적용비용과 제한된 시스템 운영환경 	<ul style="list-style-type: none"> 효율적인 클러스터링 구현가능 소프트웨어간의 적용비용이 거의 들지 않고 낮은 수준에서의 기능 추가 가능
경쟁력	<ul style="list-style-type: none"> 소프트웨어선진국에 유리 	<ul style="list-style-type: none"> 공동개발상식에 따른 기술공유 효과 우수(교육효과 우수) 신흥 소프트웨어 개발국에 적합
공급권	<ul style="list-style-type: none"> 개발업체로 부터 단일공급 	<ul style="list-style-type: none"> 다수 업체로 부터 공급가능

2) 상용SW와 비교

공개소프트웨어는 종종 ‘비상용 소프트웨어’로 간주되나 이는 공개소프트웨어가 자유로운 재배포가 가능하다는 것에 대한 오해에서 비롯된 것이다. 정부 및 업계에서 사용하는 대부분의 공개소프트웨어는

상용이라는 상업적 조건에서 사용이 가능하다.

공개소프트웨어는 영리 목적의 기업으로부터 공급받을 수 있으며, 공개소프트웨어를 제공하는 기업은 라이선스 비용을 지불하지 않으나 서비스에 대한 비용을 청구하는 모델을 통해 공개소프트웨어 제품을 제공하고 있다.

공개소프트웨어와 상용소프트웨어의 차이점은 다음 표와 같이 간단히 정리할 수 있다.

<표 2> 공개소프트웨어와 상용소프트웨어 비교

구분	공개SW	상용SW
소스코드 접근성	가능	불가
도입 비용	유료 또는 무료	유료
개발 및 공급자	다수 개발자(기업)	단일 개발기업
	다수 공급자	다수 공급자
수익모델	서비스 중심	라이선스료 중심

3) 기타 소프트웨어와 비교¹⁰⁾

공개소프트웨어와 혼동하고 있는 소프트웨어 유형으로는 프리웨어(Freeware), 셰어웨어(Shareware), 애드웨어(Adware) 등이 있다. 이들 소프트웨어와 공개소프트웨어의 유사점은 무료로 사용할 수 있다는 점이고 가장 큰 차이점은 이들 소프트웨어는 소스코드가 공개되지 않는다는 점이다. 각 각의 특징을 살펴보면 다음과 같다.

프리웨어는 라이선스 비용없이 무료로 배포되는 소프트웨어로 최종

10) www.naver.com 지식자료 참조(<http://terms.naver.com/item.php?d1id=7&docid=4665>)

사용자가 대금을 지불할 필요는 없으나 영리를 목적으로 배포할 수 없는 소프트웨어로서 국내에서는 흔히 공개판이라 부르기도 한다. 프리웨어는 이용하는 데는 제한이 없지만 배포는 제작자의 의도에 따라야 한다. 대표적인 프리웨어로는 Winzip, RealAudio 등이 있다.

웨어웨어는 대개 판매를 목적으로 제작되지만 일단 사용해 보고 마음에 들면 구입할 수 있도록 하는 프로그램으로 국내에서는 평가판이라 부르기도 한다. 일정 시험기간 동안 무료로 사용한 뒤 해당기간이 지나 계속적 사용을 원할 때는 사용자 등록을 하고 비용을 지불해야 하며 통상 개인에 한정해 자격이 주어지는 것이 보통이다. 가계부 등 주로 가정용에서 사용하는 소프트웨어들이 있다.

애드웨어는 프로그램을 기능이나 날짜상의 제한없이 무료로 사용하는 대신 해당 소프트웨어로 작업하는 동안 광고창을 통해 계속적으로 새로운 광고가 노출되도록 한 새로운 마케팅 기법으로 등장한 소프트웨어이다. 대표적인 소프트웨어로는 제트오디어이다.

2. 공개소프트웨어 산업 개관

가. 공개소프트웨어 비즈니스 모델

공공기관 등 IT 수요자들이 공개소프트웨어를 무료 소프트웨어로 잘못 이해하고 있어 공개소프트웨어 도입 시 비용을 청구하는 것을 문제시 하는 경우가 많다. 그러나 공개소프트웨어 제공업체가 청구하는 비용은 라이선스 비용이 아니라 기술지원 서비스 비용이다. 이 서비스 비용은 제공업체가 공개소프트웨어와 관련된 정보의 바다¹¹⁾에서

필요로 하는 적절한 정보를 선별하여 공급하고, 기능 업그레이드, 버그 개선 등 도입 이후 발생할 수 있는 다양한 기술적 문제에 대한 해결에 필요한 서비스를 제공하기 때문에 청구되는 것이다.

공개소프트웨어는 사업 중점을 어디에 두느냐에 따라 패키지형, 솔루션형, 서비스형으로 구분할 수 있다.

첫째, 패키지형 비즈니스 모델은 패키지의 개발과 판매에 주력하는 것으로 리눅스가 대표적인 성공사례라 할 수 있다.

둘째, 솔루션형 비즈니스 모델은 패키지를 개발, 판매와 함께 AS등에 주력하는 것으로, 소스 코드의 공개로 제품 자체의 차별화 요소가 없는 상황에서 현실적인 모델이다. 솔루션 형의 특징은 마케팅에서 공개소프트웨어 커뮤니티를 활용할 수 있고, 제품이 박스 형태로 제공되기 때문에 판매가 용이하며, 서비스에서 수익이 발생한다. snot가 대표적인 성공사례이며 국내 대부분의 리눅스 기업들의 수익모델이다.

셋째, 서비스형 비즈니스 모델은 패키지 개발을 커뮤니티에 의존하고 우수한 공개소프트웨어 전문인력을 기반으로 공개소프트웨어의 기술검증, 제품 설치, 교육, AS 등을 주력 사업으로 하고 있으며, LinuxCare가 대표적인 사례이다.

나. 공개소프트웨어 산업 동향

공개소프트웨어 플랫폼이 IT산업의 주류로 진입함에 따라 대규모의 소프트웨어 개발기업(ISV;Independent Software Vender)과 하드웨어

11) 공개소프트웨어는 주로 자발적 공헌자 그룹인 커뮤니티에 의해 개발되고 있기 때문에 개발 및 사용과 관련된 정보 교류를 커뮤니티 사이트를 통해서 이루어지고 있다.

개발기업(IHV;Independent Hardware Vender)들이 공개소프트웨어 운영체제에서 실행되는 기업형 어플리케이션이나 시스템 기술 개발을 강화하는 추세에 있다. 또한 주요 아웃소싱 솔루션 제공기업들도 서비스 지원을 확장하여 일반적인 공개소프트웨어 플랫폼을 포괄하도록 하고 있다.

공개소프트웨어 업계는 규모로 볼 때 대규모의 다국적 기업과 중규모 개발업체, 전문유통 및 기술지원 서비스 전문 소규모 기업으로 분류할 수 있고, 사업형태로 볼 때 순수 공개소프트웨어 개발기업, 공개소프트웨어 기반 패키지 개발기업, 공개소프트웨어 기반 SI 및 솔루션 개발기업, 공개소프트웨어 기반 기술지원 및 서비스 기업, 공개소프트웨어 기반 하드웨어 개발 기업 등 5가지 유형으로 분류할 수 있다.

1) 기업 규모에 따른 분류

o 다국적 기업 등 대기업

공개소프트웨어 산업에서 하나의 큰 축을 이루고 있는 그룹은 대규모의 다국적 기업 및 국내 대기업이다. 공개소프트웨어를 기반으로 한 사업을 추진하고 있는 대형 하드웨어 벤더는 IBM, HP, Sun, 삼성 전자 등이 있으며, 소프트웨어 벤더는 Oracle¹²⁾이 대표적이다. 이들 대형 기업들은 공개소프트웨어 개발에 직접 참여, 공개소프트웨어 커뮤니티 발전을 지원, 리눅스 등 공개소프트웨어 운영체제상에서 운용 가능한 다양한 소프트웨어 및 하드웨어 개발 등을 통해서 공개소프트

12) Oracle은 공개소프트웨어 제품을 개발하고 있지는 않으나, 리눅스 등 공개소프트웨어 운영체제에서 운용가능한 소프트웨어를 개발하고, 리눅스 등 공개소프트웨어 발전을 지원한다는 의미에서 공개소프트웨어 업계로 분류한다. Oracle 이외에도 리눅스 등 공개소프트웨어 운영체제에서 운용가능한 소프트웨어 및 하드웨어 개발업체를 공개소프트웨어 업계로 분류한다.

웨어 발전에 참여하고 있다.

o 중소기업 개발 기업

공개소프트웨어 산업의 또 다른 축을 형성하고 있는 그룹은 국내 중규모 개발 기업들이다. 이들은 국내기간 기업들로서 중간 규모의 하드웨어 및 소프트웨어 개발업체들이며 주로 공개소프트웨어기반 패키지 및 솔루션을 공급 하고 있다.¹³⁾ 또한 이들 기업들은 정부기관이나 주요 수요업체들의 요구조건을 수용하고 처리하는데 많은 경험을 보유하고 있으며 때로는 패키지형태의 제품을 개발공급하기도 하기도 한다. 공개소프트웨어 산업에서 이들은 광범위한 수준의 전문지식을 보유하고 있으며 자체적으로 기능성을 추가한 보다 진보한 공개소프트웨어 솔루션 및 패키지 제품을 개발공급하기도 한다. 예를 들면 콘텐츠 관리시스템, 업무지능 소프트웨어, 업무처리 어플리케이션 또는 부문별 그룹웨어 서버, 방화벽, 무선네트워크 환경의 게이트웨이 등을 공급하고 있다.

o 전문유통 및 기술지원 서비스 제공업체

공개소프트웨어 산업의 또 하나의 축을 형성하고 있는 그룹은 소규모 개발 기업들이다. 이들은 주로 공개소프트웨어 산업의 틈새시장이나 특수한 기술시장에서 두각을 나타내고 있다. 공개소프트웨어 산업에서 기술 클러스터를 형성하고 있으며 중규모 개발업체와 긴밀한 관계를 형성하고 있다. 개발이 아닌 분야에서 또 다른 축을 형성하고 있는 그룹은 공개소프트웨어 제품의 전문유통과 기술지원 및 서비스

13) 공개소프트웨어기반 패키지 및 솔루션이라 함은 리눅스 등 공개소프트웨어 운영체제 상에서 운용가능한 패키지 및 솔루션을 의미하며, 공개소프트웨어 기반 SI, 서비스, 하드웨어 개발 등도 같은 의미이다.

를 주 업무로 하는 기술지원 및 서비스 기업들이다. 이들 기업체들은 주로 대규모 개발업체 및 중소기업 업체들의 인기 제품들을 유통하며 이에 따르는 기술 지원 사업을 전개하고 있다. 따라서 이들은 대규모 및 중소기업 업체들과 매우 밀접한 관계를 형성하고 있으며 상호 보완적인 파트너십 관계를 형성하고 있다.

2) 사업 유형에 따른 분류

1990년대 후반부터 리눅스 배포판 기업체들을 선두로 하여 국내에서도 공개소프트웨어를 기반으로 하는 기업들이 나타나게 되었다. 국내에서 공개소프트웨어를 기반으로 사업을 하고 있는 기업체는 현재에는 약 150여개 정도로 파악되고 있다.¹⁴⁾

o 공개소프트웨어 개발 기업

리눅스 등 공개소프트웨어를 개발 및 유통하는 업체로 국내 기업으로는 리눅스윈, 아이젯리눅스, 와우리눅스, 한글과컴퓨터 등이 있고, 외국 기업으로는 레드햇, 수세 등이 있다.

1990년대 후반 벤처 및 닷컴 붐 당시에는 리눅스 배포판 개발 및 유통 기업이 200여개에 달했으나, 현재는 리눅스 전문 개발 기업과 유통 및 기술지원 전문 기업으로 분리 발전하고 있다.

우리나라의 경우 리눅스 이외에 공개소프트웨어 개발 사업에 적극적으로 참여하는 기업은 아직 거의 없다.

14) 국내에서 공개소프트웨어를 기반으로 하는 사업을 하는 150여개 업체들의 상세정보를 확인하려면 『2005 공개SW 기업과 제품정보 가이드(한국소프트웨어진흥원, 2005)』을 참고할 수 있다. 또한, 공개SW 기업체의 기업정보와 제품정보를 DB화하여 help.oss.or.kr에서 제공하고 있다.

o 공개소프트웨어기반 패키지 개발 기업

리눅스 등 공개소프트웨어 운영체제 상에서 운용 가능한 패키지 소프트웨어를 개발하는 기업으로써 대표적인 국내 기업으로는 삼성 SDS, 티맥스소프트, 핸디소프트, 케이컴스 등을 들 수 있고, 외국 기업으로는 오라클, 아도비 등을 들 수 있다.

리눅스 확산의 가장 큰 장애요인으로 작용하고 있는 것이 우수한 어플리케이션이 없다는 인식이나, 대부분의 패키지 개발 기업들이 리눅스에서 운용 가능하도록 개발하고 있으며, 리눅스에 대한 지원을 강화하고 있다.

o 공개소프트웨어기반 SI & 솔루션 개발 기업

삼성SDS, POSDATA, SKC&C, LGCNS 등 대형 SI 기업, 리눅스 코리아, 엔티씨코리아 시스템 등 중소 SI기업 등 대부분의 국내 SI 기업들이 리눅스를 기반으로 한 시스템 구축업체로 분류할 수 있다. 또한, 이니텍, 쓰리알소프트, 사이버다임 등 국내 중견 소프트웨어 개발 기업들도 리눅스 등 공개소프트웨어기반의 솔루션을 개발하고 있다.

리눅스 확산의 가장 큰 장애요인으로 작용하고 있는 것이 훌륭한 어플리케이션이 없다는 인식이나, 대부분의 솔루션 개발 기업들이 리눅스에서 운용 가능하도록 개발하고, 리눅스에 대한 지원을 강화하고 있어 리눅스 도입에 따른 어플리케이션 선택의 제약은 없다고 할 수 있다.

o 공개소프트웨어기반 기술지원 & 서비스 기업

공개소프트웨어 시장 확산에 중요한 사업 분야가 공개소프트웨어 기반의 기술지원 및 서비스 분야로 공개소프트웨어 기반으로 구축된

시스템에 대한 기술지원과 교육을 담당함으로써 사용자의 불안감을 해소해 주는 역할을 하는 사업 분야라 할 수 있다.

국내외적으로 이러한 업종에 속한 기업들은 삼성SDS, 한국IBM, 한국 HP 등의 대기업 및 다국적기업과 리눅스코리아, ITBridge 등의 특화 기술지원, 서비스, 교육을 담당하는 지역단위의 기술지원업체가 있다.

리눅스 교육 분야에서는 수퍼유저코리아, 지역기술지원으로 애니아이티 등이 대표적이다.

o 공개소프트웨어기반 하드웨어 개발 업체

리눅스 등 공개소프트웨어 운용체제가 운용가능한 x86 계열의 하드웨어를 개발 및 생산하는 업체로 국내 기업으로는 삼성전자, 유니와이드 등이 있으며, 외국 기업은 IBM, HP, Dell, Sun 등이 대표적인 업체이다.

BLANK

2장. 공개소프트웨어에 대한 오해와 진실

BLANK

1. 비용 문제

첫째, 공개소프트웨어를 도입하면 도입 이후의 운영비용이 너무 높다는 인식이 있다.

공개소프트웨어를 이용하면 라이선스 비용 절감을 시작으로 하여 전반적인 도입비용, 그리고 더 크게는 TCO(총소유비용)¹⁵⁾를 낮출 수 있다.

IDC가 2002년에 발표한 인텔기반의 리눅스와 RISC기반의 유닉스 서버 간 TCO분석에 따르면¹⁶⁾ 모든 면에서 공개SW의 TCO가 더 낮게 나타났다고 보고하고 있으며, 미국의 시장조사업체인 로버트프랜시스 그룹이 웹서버를 운영하는 전 세계 2천여개 업체들을 대상으로 조사한 결과 3년간 리눅스기반의 웹서버를 운영 했을때 윈도우를 사용 했을 때 보다 39%정도 TCO가 낮은 것으로 나타났다¹⁷⁾. 이 조사에 따르면 하루 10만건의 히트수를 처리하는 것을 기준으로 볼때 3년간 웹서버운영에 솔라리스 기반시스템은 56만2천불, 윈도우기반은 19만1천불, 리눅스는 7만4천불의 TCO가 소요되는 것으로 나타났다.

둘째, 공개소프트웨어는 초기비용보다 시스템 증설 및 유지, 그리고 업그레이드 비용이 더 많이 발생한다는 오해가 있다.

15) TCO(Total Cost Ownership)란 총소유비용을 정보시스템 구축에 필요한 하드웨어, 소프트웨어 등 초기 도입비용 뿐만 아니라 유지관리비용, 교육비용 등 정보시스템 구축과 운영에 필요한 모든 비용을 포괄하는 비용을 의미한다.

16) Al Gillen, Dan Kusnetzky, and Scott McLarnon, The Role of Linux in Reducing the Cost of Enterprise Computing, IDC White Paper, 2002.

17) Total Cost of Ownership for Linux in the Enterprise, Robert Frances Group, 2002

먼저, 독점소프트웨어는 일반적으로 개인당 라이선스(또는 cpu당 라이선스)로 판매 된다. 따라서 현재 시스템을 증설하기 위하여 서버의 숫자가 증가한다면 독점소프트웨어 솔루션 도입비용은 계속 상승하게 된다. 즉, 현재 시스템이 더 많은 클라이언트를 위해 서비스를 해야 한다면 추가되는 클라이언트의 숫자에 비례하여 라이선스 비용을 추가 지불해야 한다는 것을 의미한다.

하지만 공개소프트웨어에서는 시스템이 증설된다 하더라도 하드웨어 구매 비용 외에는 추가 라이선스 비용은 발생하지 아니한다. 공개소프트웨어는 처음부터 라이선스료를 지불하는 것이 아니라 서비스에 대한 비용을 지불하는 것이므로 클라이언트 수가 늘어나는 만큼 얼마든지 복사하여 사용할 수 있다는 것을 의미한다. 따라서 공개소프트웨어에 있어서 시스템 증설에 따른 추가 라이선스 비용은 존재하지 않는다. 단, 시스템증설에 따르는 기술지원료는 발생할 수 있지만 이는 독점소프트웨어도 마찬가지로 일 뿐 아니라 기술지원 업체 또한 얼마든지 선택하여 조건에 맞는 가격에 서비스를 받을 수 있다.

셋째, 공개소프트웨어는 설치시간, 유지비용, 교육비 측면에서 비공개 소프트웨어 구매비용과 별 차이가 없다.

이는 TCO(총소유비용)에 대한 것으로써 공개소프트웨어의 대표적인 장점이라고 할 수 있는 비용 절감에 대한 오해이다. 공개소프트웨어의 대표적인 장점인 라이선스 구매비용이 없다는 것은 공개소프트웨어를 도입하여 적용하려고 하는 기업이나 공공기관에게는 비공개소프트웨어에 비하여 절대적인 비용절감을 가져다주게 된다.

공개소프트웨어는 설치시간, 유지보수비용, 교육비용 등에서 비공개소프트웨어 보다 비용이 많이 들거나 설치가 복잡하다고 일괄적으로 말할 수 없다. 또한 기술지원 서비스 전문업체로부터 다양한 서비스

를 받을 수 있으며 필요한 경우 민간 커뮤니티로부터 무상으로 기술 지원을 받을 수도 있다.

교육비용은 어떤 소프트웨어에서건 처음 접하면 반드시 필요한 것이며 공개소프트웨어라고 해서 더 많이 들거나 작게 드는 비용이 아니다. 단지, 도입한 공개소프트웨어 개발자 커뮤니티¹⁸⁾가 성숙해지기 이전이라면 그 공개소프트웨어는 버그가 존재할 수 있고, 완성도가 떨어질 수 있으므로 이는 도입단계에서 고려해야 할 문제이다. 이 또한 비공개 소프트웨어도 출시 초기에는 수많은 버그를 갖고 있기 때문에 공개소프트웨어만의 문제는 결코 아닐 것이다.

또한, 기존의 시스템을 비공개소프트웨어로 운영하고 있었기 때문에 발생하는 비용이지, 공개소프트웨어를 도입하기 때문에 발생하는 비용은 아니다. 이미 공개소프트웨어를 사용하고 있는 상황에서 비공개소프트웨어를 도입하는 경우라면 더 높은 교육비용을 지불하여야 할 것이다.

이러한 여러 가지 상황을 고려해 볼 때에 비공개 소프트웨어를 도입하는 것에 비하여 공개소프트웨어를 도입할 경우 보다 비용절감이 가능하다.¹⁹⁾

2. 기술지원 및 가용성

리눅스를 중심으로 한 공개소프트웨어 기술지원 전문 기업으로는

18) <부록4> 오픈 소스 소프트웨어 리소스 참고

19) 2005년 7월 기획예산처에서 정보화예산혁신을 위해 추진했던 2006년 정보화사업 중 공개소프트웨어 도입 사업 비용효과분석 결과 초기 도입비용이 평균 약 20% 절감되었다고 발표하였다.

한글과컴퓨터, 리눅스데이터시스템, 리눅스원, 와우리눅스, 리눅스코리아, 아이켓리눅스, 슈퍼유저코리아, 모아시스, 앤티씨큐브, 애니아이티, 케이씨크 등이 있으며 이들 기업은 기업간 파트너십을 통하여 전국적 기술지원 조직을 갖추고 있다.

또한 한국소프트웨어진흥원(KIPA) 내에 설치되어 있는 "공개소프트웨어 기술지원센터"에서 공공기관을 대상으로 공개소프트웨어에 대한 기술지원 서비스가 이루어지고 있다.

기술지원 및 가용성에 대한 편견과 오해를 알아보기에 앞서 수요자들이 공개소프트웨어 사용을 꺼리는 이유 몇 가지 살펴보면 첫째, 익숙함에 길들여져 있기 때문에 익숙하지 않은 것에 대해서는 시도하지 않으려고 하는 성향, 둘째, 타인들이 그냥 불안하다고 평하기 때문에 정확한 정보가 없는 상황에서 막연하게 불안해하는 성향, 셋째, 다른 사람들이 많이 사용하지 않기 때문에 뭔가 문제가 있다고 생각하는 편견 등이다.

공개소프트웨어의 기술지원과 관련된 일반적인 오해와 편견들에 대하여 객관적인 평가를 하면 다음과 같다.

첫째, 공개소프트웨어 기술지원 업체가 없다.

공개소프트웨어에 대한 기술지원을 받고 싶으나 지원업체를 찾기가 어렵다고 호소하나 대표적인 공개소프트웨어인 리눅스의 경우 배포판 업체뿐만이 아닌 다국적 하드웨어 벤더는 물론 국내 주요 SI업체들로부터 기술지원을 직접 받을 수 있다. 또한 공개소프트웨어의 커뮤니티 공동체에 문의하면 무료로 지원을 받을 수도 있을 것이다. 이같은 공동체의 지원 외에도 리눅스의 경우는 유명 하드웨어 업체나 소프트

웨어업체에서도 자사 제품에 대해 공식적으로 리눅스 지원 서비스를 제공하고 있다.

과거 리눅스 기술지원은 주로 중소 벤처기업이 제공했지만 2004년 하반기 이후 이 같은 기술지원의 주체는 SI업체중심 체제로 빠르게 전환되고 있다. 현재 삼성SDS, 포스데이터 등의 주요 SI업체들이 리눅스 전담팀을 강화 하는 등 리눅스 기술지원을 위한 만반의 준비를 하고 있다. 또한 한국소프트웨어진흥원에서는 “공개SW기술지원센터”를 통해 리눅스 또는 공개소프트웨어로 기존 IT인프라를 전환하는 기관을 대상으로 기술지원서비스제공을 하고 있다²⁰⁾.

둘째, 공개소프트웨어에 비해 독점소프트웨어의 기술지원이 우수하다.

공개소프트웨어에 대한 기술지원을 받는 방법은 기술지원 전문기업과 계약체결을 통해 서비스를 받거나, 자발적 민간 커뮤니티로부터 기술지원을 받는 것이다. 사용자는 이들 가운데 선택할 수 있는 권리가 있으며 개인 역량에 따라서 선택할 수 있다. 독점소프트웨어 기술지원에 대해서는 공급업체의 역량에 달려있다고 할 수 있다. 즉, 기술지원에 많은 투자를 하여 체계적인 기술지원시스템을 갖추고 있다면 당연히 공개소프트웨어의 기술지원보다는 좋은 평가를 받을 수 있을 것이나, 그렇지 않다면 좋은 평가를 얻지 못한다. 따라서 이것은 공개소프트웨어의 기술지원이 훨씬 더 나은가? 아니면 독점소프트웨어의 기술지원이 더 나은가? 라는 이분법적인 접근은 바람직하지 않다. 즉, 개별 소프트웨어와 지원업체의 역량문제일 뿐이다.

20) 공개SW기술지원센터는 주요 공개소프트웨어 관련업체의 전문 연구원이 상근적으로 근무하며 공개소프트웨어 기반의 운영체제 배포판과 솔루션에 대한 테스트 및 검증, 하드웨어와 소프트웨어간의 인증, 다양한 기술지원 및 컨설팅 서비스 등을 제공하고 있으며, 매년 6대 광역시를 중심으로 지역기술지원 세미나를 통하여 지방 공공기관에 공개소프트웨어에 대한 소개 및 리눅스 등 공개소프트웨어 사용기관에 대한 기술지원 서비스를 실시하고 있다.

다음으로 공개소프트웨어의 가용성과 관련된 일반적인 오해와 편견들에 대하여 객관적인 평가를 하면 다음과 같다.

첫째, 리눅스 등 공개소프트웨어는 사용이 어렵다.

공개소프트웨어이든 비공개 소프트웨어이든 소프트웨어를 처음 도입하였을 때부터 적응할 때까지는 어렵다는 것은 공통된 것이다. 기존의 시스템 운영자들이 유닉스나 윈도우에 익숙해져 있기 때문에 사용이 어렵다고 인식하는 것에 불과하다. 리눅스는 유닉스와 명령어체계가 대부분 동일하기 때문에 기존의 유닉스 환경에 익숙한 운영자라면 리눅스 환경에 적응하는데 큰 어려움이 없다.

둘째, 리눅스는 주로 웹서버로만 사용된다.

리눅스는 이미 핵심 업무용 서버로도 활발히 도입되고 있다²¹⁾. 또한 서버 외에도 임베디드 분야에서 눈부신 활약을 하고 있다. 리눅스는 단순한 서버용 운영체제가 아니다. 리눅스는 서버용 운영체제는 물론이고 데스크탑, 그리고 소비자 가전 및 휴대폰 등의 임베디드 운영체제로도 널리 활용되고 있다.

셋째, 리눅스는 데스크탑 운영체제로 사용은 시기상조다.

리눅스를 데스크탑 운영체제로 사용이 윈도우에 비하여 편의성이 떨어지고, 인터넷 사용에 현실적으로 제약이 있든 것은 사실이다. 이러한 데스크탑 운영체제로 사용 제약요인은 크게 인터넷 बैं킹 문제, 홈페이지 접근성 제약 문제로 요약할 수 있다.

인터넷뱅킹과 같은 인터넷상에서의 금융거래를 리눅스 기반 웹 브

21) 리눅스 도입 성공사례는 한국소프트웨어진흥원 발간 공개소프트웨어 도입 성공사례집(2004, 2005) 참고

라우저(Mozilla등)에서는 아직 완벽하지 못한 것이 사실이나, 이런 문제는 리눅스용 공인인증서를 개발하고 운영체제에 제약을 받지 않는 128비트, SSL²²⁾ 보안체제가 구축되면 손쉽게 해결될 사안이라고 전문가들은 말한다.²³⁾

리눅스와 모질라 브라우저를 사용하여 인터넷뱅킹이 가능토록 하는 것이 기술적으로 불가능한 것은 아니다. 국내보안업계는 리눅스 기반 인터넷 뱅킹을 위한 기술적 기반은 이미 마련됐다고 한다. 관련업계에 의하면 당장 모든 시스템을 바꾸지 않고도 ActiveX와 유사한 리눅스용 기술을 사용자에게 제공, 리눅스, 맥등 다양한 플랫폼에서 인터넷 뱅킹 서비스를 제공할 수 있으나, 은행업계에서 소수 사용자들을 위해 이를 도입할 의지를 보이지 않고 있다고 한다.²⁴⁾

즉, 리눅스에서 아직까지 해결되지 못한 부분이 인터넷뱅킹문제라고 할 수 있지만 이것은 기술적인 문제가 아니라 산업 환경적인 요인에 기인하는 것으로 보아야 한다. 즉, 리눅스를 사용하는 사용자층이 많아지면 은행업계측에서도 자연스럽게 이를 수용하고 투자를 하게 될 것이기 때문이다.

또 다른 제약요인은 리눅스 데스크탑에서 인터넷 서핑과 검색에 제

22) Secure Sockets Layer는 네트워크 내에서 메시지전송의 안전을 관리하기 위한 프로그램 계층이다.

23) 국내에서 독자적으로 암호화 알고리즘을 개발한 것은 미국지역 외에서 사용되는 익스플로러가 지원할 수 있는 알고리즘 성능이 56비트로 제한됐기 때문으로 문제는 개발된 SEED를 익스플로러에서 구현하기 위해서는 ActiveX라는 MS윈도우의 고유기능을 별도로 사용해야 한다는 것이다. 2000년 이후 미국 외에서도 익스플로러를 비롯한 각종 웹 브라우저에서 128비트암호화 기능사용이 가능해졌지만 국내 은행은 여전히 ActiveX에 기반을 둔 서비스를 제공하고 있다.

24) 정보통신부 우정사업본부에서는 한국소프트웨어진흥원이 추진하는 공개소프트웨어 시범 사업에 참여하여 리눅스 사용자가 인터넷뱅킹이 가능하도록 하는 우체국 인터넷뱅킹 시스템을 2006년 초에 서비스를 제공할 예정이다.

약이 있다는 것이다. 리눅스 및 모질라 등 리눅스용 웹브라우저에서 웹서핑이 제약되는 이유는 리눅스나 모질라의 기술적 제약으로 인한 것이 아니라, 국내 공공기관 홈페이지나 포털사이트 등이 W3C 등 국제 표준을 준수하여 웹을 개발하고 있지 않기 때문에 발생하는 산업환경적 요인이자, 공개소프트웨어가 갖는 기술적 한계가 아니라는 점이다.

그러나, 리눅스의 대표적인 웹브라우저인 Mozilla Firefox 1.1에서는 비표준 문법을 사용하는 수많은 국내 웹사이트들도 문제없이 서핑할 수 있도록 개선하였다.

아직까지 데스크탑용으로는 MS Windows보다 리눅스가 다소 불편한 것은 사실이다. 즉, MS Windows는 사용자의 편리성에 맞추어 오랜 기간 동안 지속적으로 투자되고 개발되어 왔기 때문이기도 하지만 우리가 MS Windows의 GUI환경에 매우 익숙해져 있다는 점이 더욱 큰 이유일 것이다. 하지만, 현재 전 세계적으로 리눅스를 데스크탑용으로 사용하기 위한 많은 노력과 시도들이 이루어지고 있다.

리눅스를 테스트탑 운영체제로 사용하고 있는 대표적인 해외 사례로는 IBM, 노벨, 싱가포르 국방부, 뮌헨시 등이 있으며, 국내는 동북아 시대위원회, 한국소프트웨어진흥원 등이 있다.

3. 공개소프트웨어의 신뢰도

공개소프트웨어 제품에 대하여 자주하는 질문들 가운데 하나는 과연 공개소프트웨어 품질이 우수한가라는 것인데, 이러한 의문이 제기되는 이유는 소스가 공개되어 있고 라이선스 비용 없이 사용할 수 있는 공개소프트웨어 속성 때문이라고 할 수 있다.

독점소프트웨어 업계와 마찬가지로 공개소프트웨어도 매우 다양한 소프트웨어를 사용할 수 있다. 정확한 통계가 나와 있지는 않지만 공개소프트웨어로 등록된 소프트웨어는 대략 6~7만개정도가 되는 것으로 알려져 있다. 이들에 대한 품질과 신뢰도를 측정한다는 것은 매우 어려운 일이나 많은 공개소프트웨어들이 실제 업무에 적용되고 있다는 것은 신뢰도를 입증받고 있는 것으로 해석할 수 있다. 하지만 아직 성숙되지 않은 공개소프트웨어들에 대해서는 그 품질과 안정성을 보장할 수는 없을 것이다. 이번 절에서는 공개소프트웨어의 신뢰도에 대한 여러 가지 오해와 진실을 살펴보면 다음과 같다.

첫째, 공개소프트웨어는 보안에 취약하다.

공개소프트웨어는 소스가 공개되어 있기 때문에 해커에 의한 해킹 공격을 받을 위험이 많은 견해가 있다. 하지만, 현재의 첨단 보안은 소스의 공개여부와 상관관계가 없는 것으로 보고하고 있다. 현재 인터넷에서 사용하는 RSA²⁵⁾보안 프로그램들은 알고리즘부터 소스까지 모두 알려져 있다. 즉, RSA보안 프로그램들의 소스가 공개되어 있다고 해서 RSA알고리즘을 따르는 인터넷 보안체계가 취약한 것은 결코 아니다.

뿐만 아니라 최근 발표되는 자료들에 따르면 공개소프트웨어인 리눅스가 타 플랫폼보다 네트워크 보안이 우수한 것으로 나타나고 있다. 2005년 7월 SANS²⁶⁾가 발표한 보고서에 따르면 20개의 인터넷

25) RSA는 1977년에 Ron Rivest, Adi Shamir와 Leonard Adleman에 의해 개발된 알고리즘을 사용하는 인터넷 암호화 및 인증 시스템이다. RSA 알고리즘은 가장 보편적으로 사용되는 암호화 및 인증 알고리즘으로서, 넷스케이프와 마이크로소프트 웹브라우저 기능의 일부로 포함된다

26) SANS(SysAdmin, Audit, Network, Security)는 1989년에 설립된 보안 전문 연구소로써 정보 보안 트레이닝과 인증 분야에서 큰 영향력을 발휘하고 있다. SANS는 분기별로 리눅스, 유닉스 및 윈도우, 매킨토시와 같은 다양한 플랫폼상의 보안 취약점을 분석하는 보고서를 발표하고 있으며, 보안 전문가들과 네트워크 사용자들에게서 가장 신뢰할 수

보안 취약점들 가운데 단 2개의 가벼운 취약점만이 리눅스 플랫폼 사용에 영향을 주는 것으로 나타났다.²⁷⁾

또한 가트너는 제로데이 어택(Zero-day Attack)²⁸⁾시대라는 점을 기준으로 볼 때 리눅스의 보안성이 높은 것으로 보고하고 있다²⁹⁾. 사실 보안성은 기업 및 기관에서 운영하는 IT인프라의 안정성 및 신뢰성과 직결되는 문제이다. 즉, 사내 인프라의 보안수준이 낮다면 기업이나 기관은 비즈니스 연속성(Business Continuity)을 보장할 수 없을 뿐만 아니라 예기치 못한 장애로 막대한 피해까지 입을 수 있다. 이 같은 관점에서 볼 때 리눅스는 공개소프트웨어만이 제공하는 보안 메커니즘을 통해 안정적인 서비스를 보장한다고 볼 수 있다. 리눅스의 보안성이 얼마나 뛰어난지에 대한 대표적인 사례로는 미국 국가안전보장국(NSA;National Security Agency)에서 학계, 업계의 보안 전문가들의 참여를 통해 추진하고 있는 Security-Enhanced Linux 프로젝트가 있다. Security-Enhanced Linux는 운영체제 관련 가장 진보된 보안 프로젝트 중 하나로 평가받고 있으며 이 같은 프로젝트에 리눅스가 선택된 배경은 공개소프트웨어의 주요 특징 중 하나인 투명성(Transparency)때문이다. 소스를 제공하지 않는 비공개 소프트웨어의 경우 보안 결점이나 버그들이 곳곳에 있어도 이를 공급자가 인지하고 개선안을 마련하기 전까지 알 수 없으나 공개소프트웨어는 소스 코드 분석을 직접 해볼 수 있어 사용자의 능동적인 대처가 가능하며, 공개 소프트웨어 지원업체 및 관련 커뮤니티를 통해 보안결점이나 버그들

있는 보고서라는 평을 받고 있다.

27) ZDNet Korea 2005. 8. 1인용

28) Zero-day란 보안취약점이 발견되고 이를 해결하기까지 걸리는 시간을 의미하며, Zero-day Attack이란 보안 취약점이 발견되었을 때 그 문제의 존재 자체가 널리 공표되기도 전에 해당 취약점을 악용하여 이루어지는 보안 공격. 공격의 신속성을 의미하는 것이다.

29) John Pescatore, Open Source Security-SYMPOSIUM ITXPO 2004, Gartner, 2004.

에 대한 개선이 빠르게 일어나 각종 보안 침해 사고를 최소화 할 수 있다. 즉 소스가 폐쇄된 독점소프트웨어의 경우 보안상의 문제 발생 시 해당 업체의 대응을 기다리는 방법밖에 없지만 공개소프트웨어의 경우 보안 취약성과 같은 결함 발견 시 여러 업체 및 공동체의 공동 대응으로 짧은 시간 내에 패치 또는 업그레이드가 이루어지기 때문에 보안성이 높다고 할 수 있다.

둘째, 공개소프트웨어를 사용하면 바이러스에 감염이 잘 된다.

오히려 소스가 공개되어 있기 때문에 오히려 바이러스 침입에 신속히 대응할 수 있다. 따라서 일반 사용자가 아무리 악의적인 프로그램을 실행해도 시스템에 큰 영향을 주지 못해 바이러스 감염이 매우 적다.

셋째, 공개소프트웨어를 핵심 정보화 프로젝트에 도입하는 것은 매우 위험하다.

리눅스는 단순한 파일, 프린트 서버의 영역을 벗어나 DBMS, 보안 등의 핵심 분야로 도입사례가 확대되고 있다. 2004년 10월 포레스터 리서치가 미국 140개 대기업을 대상으로 조사한 결과에 따르면 조사 대상 업체의 53%가 자사의 주요 핵심 업무용 소프트웨어 운영환경으로 리눅스를 활용하고 있으며, 새로운 업무용 소프트웨어 도입에 있어 리눅스를 선택할 의사 또한 52%에 달하는 것으로 나타났다³⁰⁾. 신뢰하기 어려운 운영체제라면 이렇게 다양한 분야에서 활용되지 못했을 것이다. 최근 초대형 정부 핵심 정보화 프로젝트인 NEIS에 리눅스가 채택되어 구축 중에 있다는 것만 보아도 핵심 정보화 프로젝트에 리눅스가 매우 활발히 이용되고 있음을 잘 알 수 있다.

넷째, 리눅스가 유닉스에 비해 성능이 떨어진다.

30) Brad Day, Laura Koretzle, Linux Crosses Into Mission-Critical Apps, Gartner, 2004

TPC-C(www.ccp.org)나 SPECfp(www.spec.org) 등에서 제공하는 각종 벤치마크 결과를 보면 리눅스는 유닉스와 윈도우에 비해 높은 성능을 나타냄을 쉽게 알 수 있다³¹⁾. 이들 데이터를 종합해 보면 리눅스는 윈도우는 물론이고 유닉스보다도 대체적으로 높은 수준의 성능을 보이고 있음을 알 수 있다. 예를 들어 HP가 발표한 자료에 따르면 아이테니엄2를 장착한 4웨이머신인 HP rx5670의 경우 TPC-C 벤치마크 결과 리눅스를 사용할 때가 가장 높은 결과를 보이고 있으며, 각종 SPEC벤치마크 결과역시 HP-UX에 비해 리눅스가 높은 수치를 보이고 있다.

4. 완성도 및 수명

대부분의 공개소프트웨어 프로젝트에 사용되는 공동제작 모델은 사용이 증가함에 따라서 완성도와 신뢰도가 향상되는 피드백 기반의 개발 모델과 관련이 있다. 제품 수명주기 초기 단계에서, 공개소프트웨어 어플리케이션은 독점소프트웨어의 배포 버전보다 강력하지 못하며 신뢰도가 떨어질 수 있다. 다수의 공개소프트웨어 개발 공동체는 ‘조기 배포, 빈번한 배포(release early, release often)’라는 배포 철학을 따르고 있다. 이러한 제품의 초기 버전들은 일반적으로 제한적인 채택을 의도한 것이므로, 기관들은 중요한 기능을 위해 초기 배포버전을 도입은 주의해야 한다.

하지만 적절하게 설계되고 관리된다면, 공개소프트웨어 어플리케이션은 눈에 띄게 급속도로 발전될 수 있다. 시장 요구를 충족시키는 공개소프트웨어 프로젝트는 추가적인 개발자, 개발업체, 초기 채택 기술자들을 발생시키는 경향이 있는데 이러한 안정적인 프로젝트 관리

31) 이들 사이트에서는 주요 시스템 업체별 제품의 각종 벤치마크 결과를 확인할 수 있다

및 개발 속도가 빠르게 확산되고 있는 소프트웨어가 리눅스이다.

리눅스는 개발자의 확대 및 사용자 증대에 따라 문서화, 웹사이트, 메일링 리스트 지원 및 토론 포럼 등이 활발히 전개되어 소프트웨어 완성도가 성숙단계에 진입한 것으로 인정받고 있는 대표적인 공개소프트웨어이다.

대부분의 공개소프트웨어 개별분야마다 일반적으로 하나 이상의 경쟁업체들이 존재한다. 이는 특히, 다른 생태계 또는 영역으로 구분이 가능한 분야에서 나타날 수 있다. 컴퓨터 운영체제가 가장 적절한 예이다. 이러한 운영 체제의 사용은 다른 공개소프트웨어 프로젝트가 특정용도(이동 장치, 데스크탑 시스템, 서버) 또는 특정 기술 모델(분산 컴퓨팅, 그리드 컴퓨팅, 단일 서버)을 목표로 하여 동시에 성공을 거둘 수 있을 정도로 다양하다.

공개소프트웨어는 비공개 소프트웨어에 비해 안정성이 떨어진다는 견해가 있으나, 포스코, 현대기아자동차 등 리눅스를 도입한 대기업들은 모두 리눅스의 '안정성'을 주요 도입 배경으로 꼽고 있다. 리눅스는 특히 운영체제 수준에서 제공하는 안정성이 뛰어난 것으로 평가받고 있다. 기업이나 공공기관에서 시스템의 안정성은 곧 서비스의 신뢰성과 직결되는 문제이다. 리눅스의 경우 유닉스에 버금갈 뿐만 아니라 윈도우에 비해서는 탁월한 안정성을 자랑한다.³²⁾

그리고 공개소프트웨어의 또 다른 안정성에 대한 의문은 공개소프트웨어를 도입하면 기존의 소프트웨어와 시스템을 모두 바꿔야 한다고 생각하는 경향이 있다.

32) 공개소프트웨어 도입사례는 한국소프트웨어진흥원 발행 공개소프트웨어 도입 성공사례집 참고

주요 소프트웨어 개발업체들은 새로운 소프트웨어 개발 시 가장 먼저 포팅 해야 할 대상으로 유닉스가 아닌 리눅스를 선택하고 있다. 주요 유닉스 벤더들의 최근 유닉스 버전들은 리눅스에서 운용되는 소프트웨어를 운용 하는데 있어 포팅 이슈가 거의 없어졌다. 오히려 리눅스용으로 포팅된 것이 역으로 최신 유닉스 플랫폼으로 이식되기도 한다. 이는 지난 3~4년간 기존 소프트웨어 기업들이 제품 개발 시 목표 플랫폼으로 유닉스를 지원했던 경향이 이제는 리눅스가 최우선 포팅 대상으로 변화하고 있기 때문이다.

3장. 공개소프트웨어 도입 유형 및 절차

BLANK

1. 도입유형

공개소프트웨어를 도입하는 방법에는 크게 3가지 방법이 있다. 첫째 기관이 직접 선택하여 도입하는 직접선택, 둘째 민간기업 등 외부의 추천에 의한 간접공급, 셋째 기관내부에서 공개소프트웨어를 개발 및 수정하는 내부개발 등이다.

이들 3가지 도입방법들에 대한 자세한 설명을 하기에 앞서 공개소프트웨어를 도입함으로써 얻을 수 있는 기대효과와 장점은 어떤 것들이 있는가를 살펴보면 다음과 같다.

첫째, 소프트웨어의 유지관리의 지속성(영속성)을 보장 받는다.

원공급자가 폐업하여 사라지거나 지원을 중단하더라도 공개된 소스를 보유하고 있으므로 기관은 해당코드를 새로운 공급자에게 양도함으로써 정보시스템 유지관리의 지속성(영속성)을 보장받는다.

둘째, 소프트웨어 도입 및 정보시스템 구축비용을 절감할 수 있다.

독점소프트웨어는 라이선스 수수료와 함께 서비스 및 기술지원 비용을 발생시키지만 공개소프트웨어는 라이선스 수수료에 대한 비용 대신에 서비스 및 기술지원비용만을 발생시킨다. 따라서 공개소프트웨어를 도입하면 도입비용을 현저하게 낮춘다. 또한, 유지보수 측면으로 보면 공개소프트웨어는 소스코드에 바로 접근하여 수정할 수 있는 권한이 확보되므로 유지보수 및 업그레이드에 투여되는 예산을 줄일 수 있다.

셋째, 높은 호환성을 보장받는다.

공개소프트웨어는 다양한 형태의 소스코드를 배포하고 있으므로 상대적으로 많은 플랫폼에 이식가능하며 또한 이와 같은 플랫폼 독립성(호환성)은 수요자에게 하드웨어 선택의 폭을 넓혀준다.

넷째, 패치와 업그레이드 주기 및 속도가 빠르다.

공개소프트웨어는 소스코드가 공개되어 있고 많은 개발자들이 함께 공동 작업을 하고 있으므로 보안취약성등과 같은 결함을 발견한 후 몇 시간 또는 몇 일내에 재빨리 패치 및 업그레이드가 이루어지므로 보다 안전하게 시스템을 운영할 수 있다.

다섯째, 독점소프트웨어를 공급하는 업체로 부터의 구속을 탈피할 수 있다.

공개소프트웨어는 동일한 제품에 대해서도 공급자가 다수업체가 될 수 있기 때문에 공급자의 불합리성에 보다 적극적으로 대처할 수 있으며 또한 독점소프트웨어에 대한 구속을 회피할 수도 있다. 즉, 독점소프트웨어의 경우에는 선택의 여지없이 사용해야 만하는 불합리성이 존재할 수 있지만 공개소프트웨어는 공급업체와 제품이 다수이기 때문에 이와 같은 구속은 존재하지 않는다.

여섯째, 소스코드의 수정 및 확장권한을 확보할 수 있다.

독점소프트웨어는 컴파일된 실행파일 즉, 이진파일형태로 제공되기 때문에 소스코드를 수정하거나 확장을 위한 변경작업을 할 수가 없으나, 공개소프트웨어는 소스코드가 공개되어 있기 때문에 현재 시스템에 적용되어 있는 소프트웨어의 수정과 추가 확장을 위한 변경작업이 얼마든지 가능하다.

일곱째, 공공부문의 정보시스템 통합이 용이하다.

정부 및 공공부문의 정보시스템 구축 시에 핵심모듈을 공개소프트웨어로 구축할 경우 공공부문의 개방표준화가 용이하게 되어 정보화 예산을 절감하고 호환성 및 이식성을 확보하여 공공부문 전체의 정보시스템 통합을 촉진시킬 수 있다.

여덟째, 수요자 권리를 확보할 수 있다.

공개소프트웨어 도입 확대는 소프트웨어산업이 공급자 위주의 시장으로부터 소프트웨어 수요자(사용자, 정부) 위주의 시장으로 전환하는데 중대한 역할을 할 수 있다. 독점소프트웨어 업체들이 그들의 소프트웨어를 공급하면서 독점지위를 유지하기 위하여 소스코드가 아닌 실행파일 형태로 제공하고 있다. 하지만 이것은 지극히 공급자 위주의 관점이며 소프트웨어를 구매하는 수요자의 입장에서는 소프트웨어 구매와 함께 소스코드를 요구할 수도 있다고 볼 수 있다. 단순히 소스코드만을 요구하는 것이 아니라 공급업체가 없어지거나 사라지더라도 도입한 시스템의 유지보수와 패치 및 업그레이드 등을 보장받으려면 소스코드가 공개되어 있지 아니하고서는 불가능한 일이기 때문이다. 따라서 소비자들이 이러한 유지보수, 패치, 업그레이드 등을 보장받으려면 독점소프트웨어로는 그 한계가 있기 때문에 공개소프트웨어를 사용하여 그 권한을 보장받을 수 있는 것이다.

공개소프트웨어는 수요자 측면에서의 이러한 장점과 함께 국가산업적 측면에서는 소프트웨어 개발 원천기술을 확보함으로써 지식기반경제 핵심 산업인 소프트웨어산업의 경쟁력을 강화시키고, 최신기술의 공개, 개발자 커뮤니티 등을 통한 기술교육에 의하여 국내 소프트웨어제품의 선진제품과의 기술격차를 줄일 수 있으며, 빠른 신제품 개발 등 시장에 즉시 대응이 용이하며 개발의욕 고취를 통한 양질의 핵심소프트웨어 인력양성이 가능하다는 장점이 있다.

직접선택, 간접공급 내부개발 도입방법 이외에도 다양한 방법들이 존재할 수 있지만 공개소프트웨어를 도입하는 거의 모든 경우는 대부분 이 세 가지 범주 내에 속하기 때문에 이번 절에서는 이들 3가지 방법에 대해서 설명한다.

가. 직접선택

직접선택("내부조달"이라고도 함)이란 공개소프트웨어 리소스 사이트에서 사용자가 특정 공개소프트웨어를 직접 다운로드하여 활용하는 것을 의미한다. 물론 다운로드 행위자체가 기관 내부의 직원에 의해 수행되며 기관 내부에서 공개소프트웨어 관련 정보가 어느 정도 확보되어 있어야 가능한 방법이다. 이 방법은 비용발생이 거의 없다는 장점이 있는 반면 위험부담이 크다는 단점이 있다.

특정 공개소프트웨어를 직접 다운로드 한다는 것은 독점소프트웨어를 공급업체로부터 구매하는 것에 비한다면 기술검증에 대한 책임을 스스로 부담한다는 단점이 있다. 즉, 다운로드하는 소프트웨어 내에 트로이목마와 같은 바이러스 코드가 들어있다거나 키로거(key-logger)와 같은 악성 소프트웨어에 의한 감염을 포함한 소프트웨어가 될 수 있기 때문이다. 그러나 세계적으로 사용빈도가 높은 공개소프트웨어를 공식 사이트로부터 다운로드받는다면 이러한 부담을 제거할 수 있으며, MD5³³⁾ 등의 무결성 체크를 선행하는 경우 대부분 해결이 가능하다.

33) 암호화 알고리즘인 해쉬 함수의 일종으로 미국표준은 SHA-1(Secure Hash Algorithm)이며, 산업계에서는 MD2, MD5 등을 주로 사용한다.

그리고 또 다른 문제점은 다운로드한 공개소프트웨어의 보상과 보증이 보장되지 않는다는 것이다. 즉, 공개소프트웨어 솔루션을 기관내부에서 직접 선택하여 도입하려고하는 경우에는 소프트웨어의 보증과 보상을 보장받지 못하므로 이에 대한 위험감소 대책을 고려해야만 한다. 기관에서 공개소프트웨어 도입시 각 분야에서 신뢰도가 높은 소프트웨어는 < 부록1 >을 참고할 수 있다.³⁴⁾

이러한 위험에 대한 대책으로는 지역 내에 기술지원이 가능한 개발업체가 하나이상 존재하는 공개소프트웨어를 선택하는 것이다. 이와 같은 대책을 마련해 둔다면 공개소프트웨어를 도입한 기관이 직접 해결하지 못하는 기술적 문제에 대해 안정성을 보장 받을 수 있다.

즉, 도입기관의 시스템 운영자는 대부분 매일의 주기적이고 반복적인 업무는 수행할 수 있다 하더라도 기술지원업체를 미리 확인하고 점검해 두는 것은 2중 3중의 안전장치를 가지는 것이며 시스템 가용성을 더욱 높이게 되어 위험요소를 감소시킬 수 있다. 직접선택 방법으로 공개소프트웨어를 도입할 경우 다음의 절차를 따를 것을 권고한다.

34) <부록 1> 신뢰성이 높은 공개소프트웨어와 <부록 2> 공개소프트웨어와 비공개소프트웨어 안내 참고

<표 3> 공개소프트웨어 도입 절차

직접선택 워크플로우	
1 단계	공개소프트웨어 솔루션 조사 ³⁵⁾
2 단계	유리한 공개소프트웨어 솔루션 선택
3 단계	솔루션의 목적에 맞는 적합성과 비용적 가치 검토
4 단계	솔루션의 품질과 보안성 분석
5 단계	예비 프로젝트 수행
6 단계	예비 프로젝트 결과를 통하여 기본적 수준의 예측사항 검토
7 단계	프로젝트 수행 계획 수립
8 단계	프로젝트 수행

특정 공개소프트웨어 제품이 기관의 요구조건을 충족시킬 경우 도입에 앞서 해당 기관에서 보다 상세한 검토를 수행하기로 결정하기로 하였다면 다음과 같은 체크리스트를 작성하고 평가과정을 거치도록 해야 한다.

<표 4> 직접선택방식 도입 전 체크리스트

① 기관에서 사용되는 운영체제 플랫폼에서 해당 공개소프트웨어가 실행되는가?
② 해당 공개소프트웨어가 기존의 운영체제 플랫폼에 대해 획득, 시험, 배치해야 하는 추가적인 시스템 구성요소, 라이브러리 또는 모듈을 필요로 하는가?
③ 해당 공개소프트웨어의 설치 절차가 이해하기 쉽고 명확하게 정의되어 있는가?
④ 도입기관이 목적에 대한 적합성 여부를 결정할 수 있도록 공개소프트웨어를 설치, 배치, 시험할 수 있는 내부 전문지식을 갖추고 있는가?
⑤ 해당 공개소프트웨어의 설치 및 제거 절차가 명확하게 정의되어 있는가?

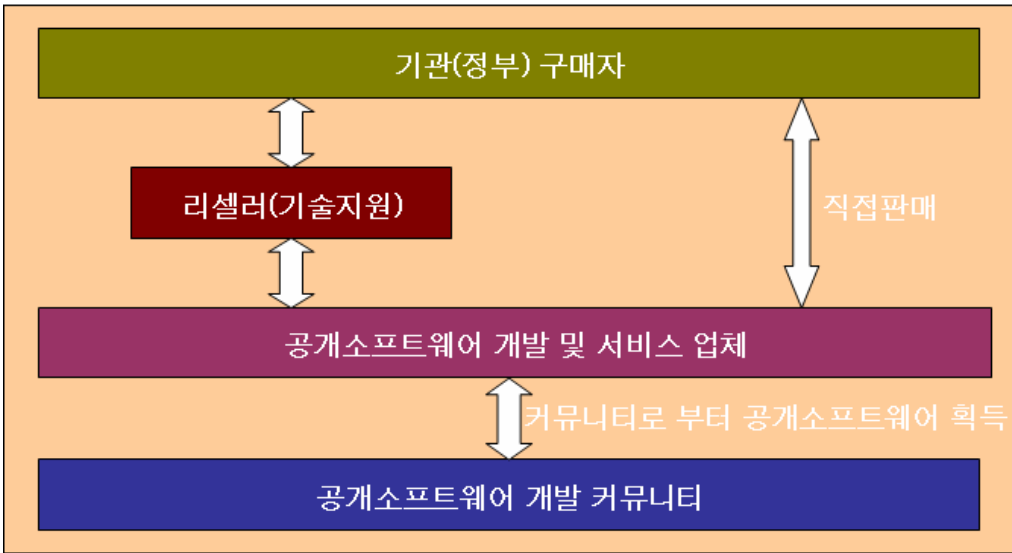
35) 다양한 공개소프트웨어 프로젝트 정보는 <부록 3> 공개소프트웨어 리소스 참조

나. 간접공급

간접공급("외부조달"이라고도 함)이란 공개소프트웨어를 업체로부터 공급받는 것을 의미한다. 즉, 공개소프트웨어 공급 전문기업으로부터 특정 공개소프트웨어를 공급받는 것으로 기술지원과 서비스 비용이 발생한다. 비용이 발생하는 대신 직접선택 방법에 비해 위험부담을 줄일 수 있는 안전한 방법이라고 할 수 있다.

간접공급 방식으로 공개소프트웨어를 도입하였다면 라이선스에 관련된 법적인 위험요소에 대한 책임은 원칙적으로 공급업체에 있다. 하지만 이 경우에도 공급업체와의 계약서 내용에 따라서 법적인 책임 소재가 달라질 수 있으므로 이 부분에 대한 확인을 반드시 해야 한다. 즉, 기관은 기술적인 위험요소와 변경관리 위험요소를 감소시키기 위한 적절한 실사조사를 수행해야 한다. 즉, 기관이 공개소프트웨어 솔루션을 위해 간접공급 방식을 선택한 경우, 해당 업체 즉, 공급업체가 모든 기술 지원에 있어 책임을 져야하며 기관은 공급업체가 해당 공개소프트웨어에 대한 적절한 위험경감 절차를 수행함을 확인해야 한다는 의미이다.

다음은 간접공급 방식의 몇 가지 공급유형을 나타낸 것이다.



<그림 1> 공개소프트웨어 간접공급 유형

이 외의 사항은 비공개소프트웨어를 도입할 때와 고려사항 및 절차가 유사하다. 예를 들어 기관이 네트워크 및 시스템에 도입된 소프트웨어 보안표준에 대한 정책을 이미 보유하고 있을 경우에 개발업체는 위험분석의 일환으로 그러한 정책을 반드시 준수해야 한다. 또한 모든 절차가 끝난 후 검수작업 시에도 보안 표준과의 일치성 여부를 확인하도록 해야 하며 계약서 내에 이러한 모든 절차들이 포함되어 있어야 한다.

또한, 기관은 개발업체에게 모든 소프트웨어 구성요소에 대한 공급을 확인해야 하며 소프트웨어 출처와 라이선스 계약, 그리고 위험평가 문서를 확인해야 한다. 그리고 비공개소프트웨어를 도입할 때와 마찬가지로 도입된 공개소프트웨어의 전체적인 통합문제를 분석하고 이를 파악하여 반드시 문서화해야 하며 개발업체의 수행능력을 확인할 수 있도록 자체평가를 수행할 수도 있다.

비공개소프트웨어를 업체로부터 도입할 때와 마찬가지로 간접공급 방법은 개발업체의 적격성, 확장성, 소프트웨어의 성숙도 등에 관련된 분석이 선행되어야 하며 이러한 문제는 소규모 개발업체와 거래할 때 특히 신중히 검토되어야 한다. 따라서 앞서 설명한 바와 같이 잠재적인 위험요소를 줄이기 위하여 개발업체에 대한 적절한 실사조사 및 평가는 반드시 이루어져야하며 또한 평가 결과에 따른 조치가 이루어져야 한다. 다음은 이러한 평가에 반드시 포함되어야 할 내용들이다.

- 재정적 보장 및 안정성
- 위험관리의 적격성
- 내부 관리의 평가
- 업무 지속성 계획의 검토
- 일반적인 기능 개요
- 서비스 전달 및 관리
- 기타 기관의 특성에 맞는 평가요소

실제로 기관들은 공개소프트웨어 제품을 직접선택(내부조달)을 하기 위한 기술적인 능력을 갖추지 못한 경우가 대부분이므로 외부의 서비스 제공업체를 통해 공개소프트웨어 솔루션을 도입 받는 간접공급(외부조달)방법을 선택하는 것이 바람직하다.

그리고 잘 알려져 있고 인기 있는 대부분의 공개소프트웨어 제품들은 상용 솔루션 제공업체들을 통해 공급이 가능하다. 하지만, 외부의 어떤 업체를 통해 공급받더라도 공개소프트웨어와 비공개 소프트웨어(독점소프트웨어)의 공급에는 차이점이 있다. 즉, 기관이 외부 서비스 업체를 통해 공개소프트웨어 솔루션 또는 제품을 공급받는 경우 일반적으로 관련 공개소프트웨어는 무료로 제공받고 이에 대한 서비스를 구입한다. 이것은 라이선스를 구입하고 부가가치 서비스로 지원을 제

공받는 비공개 소프트웨어(사유소프트웨어)의 경우와 차이가 있다.

그리고 기관이 외부 공급업체를 선택하기에 앞서 후보 업체들에 대한 실사로서 재정 상황, 안정성, 기술 능력 등을 평가해야 한다. 이러한 평가 작업들은 커뮤니티의 지원이 사라질 수 있는 위험한 제품선택에 대하여 위험을 줄여준다. 그리고 도입 후 어느 시점에 외부 공급업체가 사라지더라도 다른 공급업체에게서 지원을 계속 받을 수 있는 안정성과 유연성을 제공해 준다.

다. 내부개발

내부개발이란 기관내부에서 직접 개발하는 방법을 의미하며 공개되어 있는 수많은 공개소프트웨어들 가운데 도입하고자 하는 용도에 가장 적합한 공개소프트웨어를 선택하고 다운로드하여 기관내부의 개발자들에 의해 수정 개발하는 것을 의미한다. 아무것도 없는 상태에서 처음부터 개발하는 것이 아니라 공개되어 있는 수많은 공개소프트웨어들 가운데 도입하고자 하는 용도에 가장 적합한 공개소프트웨어를 선택하고 다운로드하여 기관내부의 개발자들에 의해 수정 개발하는 것을 의미하는 것이다.

기관내부 개발 방법으로 공개소프트웨어를 도입할 때에 고려해야 할 사항들을 간단히 정리해 보면 다음과 같다.

- ① 가장 적합한 공개소프트웨어 선택기준 마련
- ② 다운로드한 공개소프트웨어의 안정성 검토
- ③ 적용되는 라이선스 정책마련
- ④ 가장 적합한 개발방법론 마련
- ⑤ 개발 직원의 개발능력 검토

- ⑥ 사용직원의 사용법 교육 안 마련
- ⑦ 개발소프트웨어의 문서화 작업
- ⑧ 개발된 공개소프트웨어의 안정성과 영속성 대책 마련
- ⑨ 기타 특정업무의 종속되는 대책 마련 등

다음은 실제 내부 개발을 추진한 경우 필요한시에 단계별 개발절차이다.

o 서비스의 정의 및 적용업무 분석

가장 먼저 해야 할 작업이 어떤 업무에 적용하기 위하여 공개소프트웨어를 도입할 것인가를 정의해야하는 작업이다. 즉, 이러한 작업을 "서비스 정의"작업이라고 할 수 있으며 웹기반으로 개발할 것인지 아니면 기관내부 솔루션으로 사용하기위한 개발인지 아니면 유관기관과의 정보공유 및 협동 네트워크를 위한 개발인지를 정확하게 구분하고 이를 정의해야 한다.

o 기반 기술 정의

다음 단계는 개발기술에 대한 분석과 정의이며, 공개소프트웨어의 개발에 필요한 기반기술들을 간단히 정리하면 다음과 같다.

- ① 운영체제, 웹서버, DBMS, 사용 언어 등에 대한 기반기술 분석 및 정의
- ② 공용 framework 개발 및 업무 서비스 요구 분석 및 설계 기술
- ③ 분산 DB 구축 설계 및 관리 기술
- ④ 그룹웨어의 확장 및 공유 수준 지정에 의한 access 범위 설정 기술

⑤ 유관 기관 간 정보 DB 표준화 (XML 기반)

이와 같이 관련 기반기술들이 정의가 되어 있어야 하며 개발 직원에 대한 교육과도 연계되어야 할 항목이다.

o 적용 업무에 대한 분석

개발하려는 공개소프트웨어가 적용될 업무에 대한 기본적인 분석이 이루어 져야한다. 즉, 공개소프트웨어를 직접 개발하기 이전에 다음과 같은 사항들이 이미 검토되어야 한다.

- ① 활용성 : 개발 이후의 당기관 또는 유관기관에서의 활용성
- ② 수요의 규모 : 개발한 공개소프트웨어의 수요정도에 대한 검토
- ③ 재활용성 : 개발한 공개소프트웨어가 타 업무에 재활용 가능 여부
- ④ 표준성 및 표준의 준수여부

o 개발기간 및 투입인력 분석

공개소프트웨어를 직접 개발하기 위해서는 개발기간과 투입인력에 대한 분석과 정의가 이루어져야 한다. 개발기간에 대한 정의는 다음을 고려하여 작성하도록 한다.

- ① 5단계 개발기간 분석 : 요구분석기간, 분석기간, 개발기간, 수정 및 검토기간, 적용 및 안정화기간에 대한 분석 및 각 기간별 위험관리 대책 마련(개발기간의 준수여부에 따라 기간초과에 따르는 위험관리)

② 개발인력에 대한 대책

- 기반 기술(주로 Language)에 가장 적합한 기술인력 활용
- 개발경험이 있는 내부 개발 직원 활용
- 개발책임자(PM)을 선정하여 프로젝트로 진행
- 개발 분야에 따라 외부 개발 직원을 투입대책 마련
- 개발자 교육 대책 마련

○ 개발한 공개소프트웨어의 사후 관리문제

내부개발에 의해 개발 완료된 공개소프트웨어는 일회성 프로젝트로 끝나지 않도록 사후관리가 이루어져야 한다. 사후관리에는 개발 직원에 대한 사후관리를 포함한다.

사후관리는 개발한 공개소프트웨어를 관리대장에 등록하여 지속적인 관리가 이루어지도록 하고, 개발 시에 사용한 기술을 종합 정리하여 이를 문서화하고³⁶⁾, 개발한 공개소프트웨어의 설치문서, 사용법문서, 업그레이드 문서 등을 표준화하여 작성한다.

지금까지 설명한 내부개발 방법을 설명하였다. 기관 내부에 공개소프트웨어 개발자를 확보하기 어려울 뿐만 아니라 프로젝트 진행을 위한 관리자 확보가 어렵기 때문에 현실적으로 내부개발 방법으로 개발하는 경우는 거의 없다. 개발자나 프로젝트 실무자를 확보하고 있다 하더라도 실제 진행을 위해서는 보직변경 및 장기간 동일업무 종사여건등과 같은 조직내부적인 환경이 갖추어져야 할 것이다. 결론적으로 내부개발의 방법으로 프로젝트를 진행하기란 현실적인 어려움이 존재하는 것이 사실이고 이러한 현실적인 어려움을 극복하고 프로젝트

36) 문서화를 할 경우에 업그레이드 및 패치 방법 및 적용 방법 등에 대한 문서를 포함시켜야 한다.

를 진행한다 하더라도 개발 및 프로젝트 노하우의 축적이 지속되기란 더욱 어려운 실정이다.

라. 종합비교

지금까지 설명한 직접선택, 간접공급, 내부개발 3가지 방법에 대한 장단점을 종합비교하면 다음 표와 같다. 공공기관에서 3가지 방법 중 하나를 선택하여 공개소프트웨어를 도입할 수는 있지만 국내 공개소프트웨어 업계의 현실과 기관의 현실을 고려할 때에 가장 현실성 있고, 일반적으로 사용할 수 있는 방법은 간접공급 방식이다.

<표 5> 공개SW 도입방식에 따른 비교

	직접선택	간접공급	내부개발
내용	소프트웨어를 직접 다운로드 하여 사용목적에 맞게 활용하는 것	소프트웨어를 외부업체로부터 공급받는 것을 의미	내부의 개발자들에 의해 수정 개발
장점	비용발생이 거의 없다	직접선택 방법보다는 위험부담을 줄일 수 있는 안전한 방법	개발 및 프로젝트 노하우의 축적
단점	소프트웨어의 보상과 보증이 보장되지 않으므로 위험부담이 크다	기술지원과 서비스 비용을 지불해야하기 때문에 비용적인 부담발생	개발자 확보 어려움 프로젝트 실무자 확보 어려움 보직변경 및 장기간 동일업무 종사여건 등 조직내부적인 기반환경필요
주의 사항	지역 내에 기술지원이 가능한 개발업체가 하나 이상 존재하는 공개소프트웨어를 선택함으로써 도입기관의 안정성 확보	개발업체에게 모든 소프트웨어 구성요소에 대한 완전한 감사를 수행하도록 하여 소프트웨어의 출처와 라이선스계약, 그리고 위험평가 문서를 확인 개발업체의 수행을 확인할 수 있도록 자체평가를 수행	개발기간과 투입인력에 대한 세심한 분석과 정의가 필요 기간별 위험관리대책 필요 개발인력 준비방안 필요 개발한 소프트웨어의 사후관리대책 필요

2. 도입절차

정부기관 및 공공부문에서 시스템을 도입하거나 소프트웨어를 도입하는 일반적인 절차는 다음 표와 같다. 각 단계에서 공개소프트웨어 도입을 위한 주요 검토사항은 다음과 같다.

<표 6> 공개SW 도입을 위한 주요 검토 사항

단 계 (절 차)	내 용
사업계획서 작성	<ul style="list-style-type: none"> ○ 과제 발굴(대부분 사업 전년도), 예산편성 ○ 사업 타당성, 적정성 검토 ○ 사업 추진계획서 작성(최종 사업계획서 작성)
제안요청서 작성	<ul style="list-style-type: none"> ○ 제안요청서 작성 (객관성 및 타당성 검토가 절실히 요구되는 단계)
사업공고 및 제안서 접수	<ul style="list-style-type: none"> ○ 입찰공고 ○ 제안설명회 ○ 사업제안서 접수
제안서평가 및 선정	<ul style="list-style-type: none"> ○ 제안서평가(평가위원회) ○ 원가분석 ○ 기술협상 및 가격협상
계약체결	<ul style="list-style-type: none"> ○ 최종계약체결
사업수행	<ul style="list-style-type: none"> ○ 사업수행(계약체결일로 부터 사업 진행) ○ 진도보고(착수, 주간, 중간보고 등) ○ 완료보고
결과보고	<ul style="list-style-type: none"> ○ 최종 결과보고서 평가(검수)

○ 사업계획서 작성 단계

추진 과제를 발굴하여 사업 타당성과 적정성을 검토한 후 사업계획서를 작성한다. 주관기관은 사업계획 수립 시 비공개소프트웨어와 함께 공개소프트웨어 도입을 고려하되, 시스템 개발 시 개방 표준(Open Standard)³⁷⁾을 지원하고 개방형 플랫폼(Open Platform)³⁸⁾과 상호 호환성을 가지는 제품을 우선 고려해야 한다.

37) 개방 표준이란 ISO, W3C 등 국제기구나 단체에서 수용한 기술표준을 의미한다.

38) 개방형 플랫폼이란 개방표준에 부합하는 소프트웨어, 하드웨어를 의미한다.

o 제안요청서 작성 단계

사업자 선정을 위한 제안요청서(RFP, Request For Proposal)를 작성한다. 제안요청서는 사업 프로젝트 수행에 있어서 매우 중요한 역할을 하는 만큼 제안요청서 작성 시 비표준 조건과 특정기술 등의 제약을 두지 않도록 매우 주의해야하는 단계이다. 즉, 고의, 부주의 또는 업무관습 등으로 인하여 주관기관에서 제안요청서 작성 시 운영체제와 하드웨어 플랫폼 등과 같이 매우 중요한 부분의 요구사항에서 특정 기업 제품 또는 기술표준만 제안 가능한 기술요건을 명시할 경우 불공정경쟁을 초래할 수 있으므로, 다음 표를 참고하여 비 표준적이거나 특정 기술조건을 명시하지 않도록 주의하여야 한다.

<표 7> 비표준 · 특정기술 조건 예

구 분	요구조건	특정기술조건 예
공통	특정업체 특정제품만 가능한 사양 한정	. PCI Slot 48개 . 특정 locking 명시
하드웨어 (Server)	특정 운영체제 아키텍처기반의 CPU를 명시	. RISC, SPARC
	특정 제품만 가능한 성능, 품질등 의 인증요구	. 특정업체 인증 tpmC요청
주변기기	특정 운영체제만 지원하는 주변기 기 채택	. Windows XP 전용 스캐너
운영체제(OS)	특정 운영체제만 지원하는 소프트 웨어 개발언어 및 기술사용	. ActiveX, ASP, WMV로 개발
	특정 운영체제를 명시	. UNIX 플랫폼지원 . Windows 플랫폼지원
WAS	특정 개발 툴에 종속된 개발환경 요구	. 특정 세션관리 기법제시
웹서버 (WebServer)	특정 운영체제상에서만 운용이 가 능한 제품 채택	. IIS 도입
	특정 웹어플리케이션 개발 언어의 사용요구	. ASP 사용 개발
DBMS	특정 운영체제상에서만 운용이 가 능한 제품 채택	. 특정 Locking 기법 제시
어플리케이션	특정업체에 기술 종속전이며 사용 자의 선택권을 제한하는 소프트웨 어 지정	. Windows Media Player

그리고 제안요청서 작성 시에 기관은 시스템 계층별 독립성을 확보하기 위하여 제안요청서상의 ‘도입대상 장비 내역 및 구성요건’ 작성 시 다음 사항을 요구할 필요가 있다.

첫째, 특정 하드웨어에 종속된 특정 운영체제를 선택할 수밖에 없는 상황을 초래하지 않도록 하기 위하여 하드웨어와 운영체제를 별도의 항목으로 명시하고, 별도의 비용으로 계상한다.

둘째, 특정 운영체제에 종속된 특정 응용 소프트웨어를 선택해야만 하는 상황을 초래하지 않도록 하기 위하여 응용 소프트웨어와 운영체제를 별도의 항목으로 명시하고, 별도의비용으로 계상한다.

그리고 포털사이트 구축 사업 또는 웹 기반의 서비스 구축 사업을 진행할 경우 주관 기관은 제안요청서 작성 시 국민의 "정보 접근권 보장"을 위하여 "다양한 컴퓨팅 환경에서 접근 가능하도록 국제표준을 준수하는 제품도입 및 개발"에 대한 항목을 명시하도록 한다³⁹⁾.

그리고 BPR/ISP⁴⁰⁾사업의 제안요청서 작성 시에는 공개소프트웨어 적용 가능성 분석을 포함시킴으로써 사업 초기단계부터 공개소프트웨어 기반의 시스템 구축 가능성을 검토함으로써 시스템의 상호운용성을 확보할 필요가 있다.

마지막으로 공개소프트웨어를 제안하는 사업자는 사업완료 이후 '

39) 사용자되는 운영체제(Windows, Linux등), 웹 브라우저(Internet Explorer, Mozilla, FireFox, Safari등)등에 관계없이 누구나 시스템에 접근 가능하도록 해야 함을 의미한다.

40) 업무프로세스 재설계-Business Process Reengineering / 정보 전략계획-Information Strategy Planning

공개소프트웨어 유지보수 방안 제시'에 대한 항목을 반드시 명시하도록 함으로써 공개소프트웨어에 대한 기술지원 및 유지보수 서비스를 보장 받아야 한다.

o 제안서(제안업체) 평가 및 선정 단계

일반적으로 사용하는 제안서 평가항목은 다음과 같다.

- ① 유사분야에서의 사업수행 경험
- ② 개발대상 업무의 이해도
- ③ 개발전략
- ④ 기능 및 성능
- ⑤ 개발방법론
- ⑥ 기타 주관기관의 평가항목

각 항목별 세부 평가요소들은 주관기관의 사업추진 방향에 따라서 다소 달라질 수 있다. 특히, 소프트웨어 평가 시 동등 성능일 경우 공개소프트웨어를 제안한 업체를 우선 고려할 경우 정보화예산을 절감할 수 있는 장점이 있다.

o 사업 수행 및 결과보고 단계

공개소프트웨어를 적용하여 시스템을 구축하였을 경우 제공하는 소프트웨어의 소스코드를 확보하여야 한다. 특히, 제안업체에서 소스코드를 일부 수정하였을 경우에는 수정부분에 대한 문서화와 필요한 경우 해당 소프트웨어 개발 커뮤니티에 정보를 제공함으로써 공개소프트웨어 발전에 기여할 필요가 있다.

3. 적용시스템

전자정부 등 공공 정보시스템 구현시 목표로 하는 시스템을 서비스 측면으로 보면 크게 정책결정 시스템, 거래 처리 시스템, 지식정보 시스템 등 세 가지 유형으로 구분할 수 있다. 다음에서는 정보시스템 구축 유형별 고려사항과 공개소프트웨어 기반의 정보시스템 구축시 필요한 시나리오별 구현방법을 소개한다.

가. 시스템 유형⁴¹⁾

o 정책결정 시스템

정책 결정 시스템은 어떤 하나의 안건에 대해 다수의 의견을 모아 이를 정책에 반영하고 스로인해 원만하고 활발한 정책 적용 및 이에 영향을 받는 구성원들의 참여를 이끌어 내는 것을 목표로 한다. 따라서 정책 결정 시스템에서 제공하는 서비스는 구성원들의 참여를 이끌어내기 위한 것과 공정하게 의견을 반영하는 것, 그리고 결정된 안건이 구성원들의 의견을 받아들여 투명하게 수행되고 있다는 것을 보여주는 데에 목적을 둔다. 따라서 아래와 같은 세부 서비스를 정의할 수 있다.

첫째, 구성원들의 참여도 향상 서비스로써 구성원들의 참여도를 증가시키기 위해서는 언제 어디서나 구성원들이 자신의 의사를 전자정부 시스템에 반영할 수 있는 인터페이스가 갖추어져야 한다. 전자정

41) 정태영 외, 효율적 전자정부구현을 위한 기반기술 도입 정책연구, 한국소프트웨어진흥원, 2003 참조

부에서는 웹 기반의 의견 수렴 도구를 구축하여 사용자들의 접근이 쉽고 수많은 접속자들로부터의 의견을 반영할 수 있도록 공개된 웹서버를 통한 서비스를 제공해야 한다. 세부적인 서비스로는 선거나 토론을 위한 온라인 선거 서비스, 온라인 토론방 외에도 각종 건의사항을 빠르게 반영할 수 있도록 온라인 건의사항 수렴서비스 등이 존재할 수 있다.

둘째, 의견 반영 서비스로써 구성원들의 참여도가 향상되어 인터넷을 통한 구성원들의 의견을 수렴하면 이를 실제 정책에 적용하고 평가하는 과정에 대해 구성원들에게 신뢰감을 심어줄 수 있는 서비스가 제공되어야 한다. 이러한 서비스의 세부 내역으로는 정책을 적용하고 이를 수행하는 관리자나 혹은 관련 단체에서 보다 빠르게 의견을 모니터링 할 수 있는 서비스와 필요 없는 정보를 줄일 수 있는 필터링 서비스, 의견 분석을 위한 자료수집과 수집된 자료를 일련의 지표로 삼을 수 있는 자동 데이터 분석 서비스가 있다.

셋째, 정책 집행 서비스로써 사용자들의 요구에 부합하고자 정책이 집행되는 모든 과정을 투명하게 사용자에게 제공할 수 있고 신속한 처리과정을 위해 관련 부서간의 유기적인 협력 체제 및 인터페이스를 갖추어야 한다.

o 거래처리 시스템

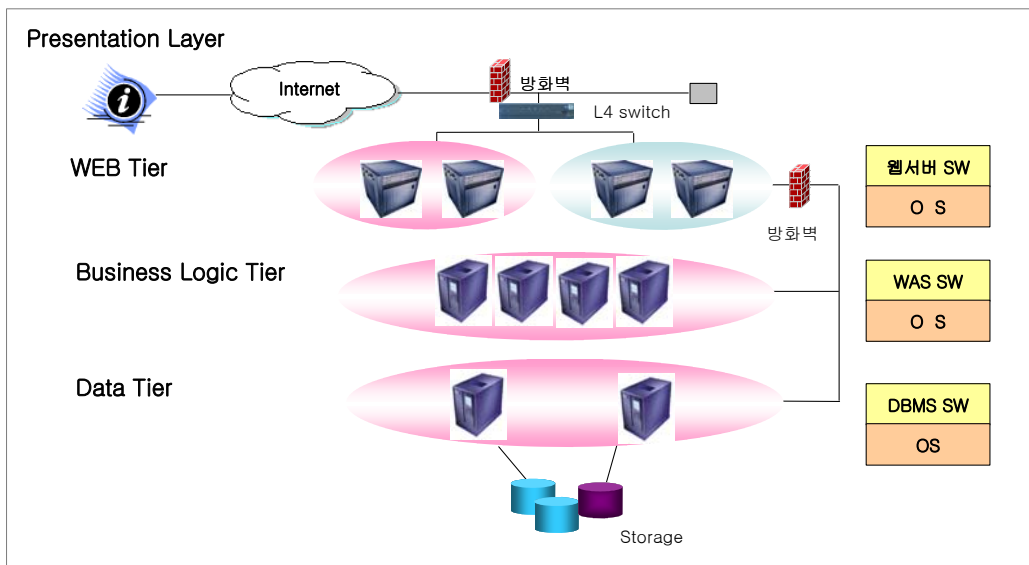
거래처리의 관점에서 보면 전자정부는 네트워크를 통하여 G2G, G2B, G2C, C2G, B2B, B2C 등의 거래를 제공할 수 있어야 하며 이를 위해서 범용적이고 향상된 보안대책이 필요하다.

o 지식 정보 시스템

공공부문과 민간부문에 대한 서비스를 위해 다양한 데이터가 사용된다. 이 데이터들은 종류도 다양할뿐더러 그 양이 너무 방대해 파일관리 위주의 일반적인 데이터 관리로써는 원활한 정보의 공유가 힘들다. 더구나 효율성 면에 대한 고려도 필요하기 때문에 이 데이터를 이용한 지식 정보 베이스를 구축해야 할 필요성이 있다.

나. 시나리오별 도입 방법

웹 기반의 업무환경은 Web Tier, Business Logic Tier, Data Tier의 3계층으로 구성되며 각 각의 계층은 웹브라우저/웹서버, 웹 어플리케이션 서버, 데이터베이스 서버의 순서로 처리된다.



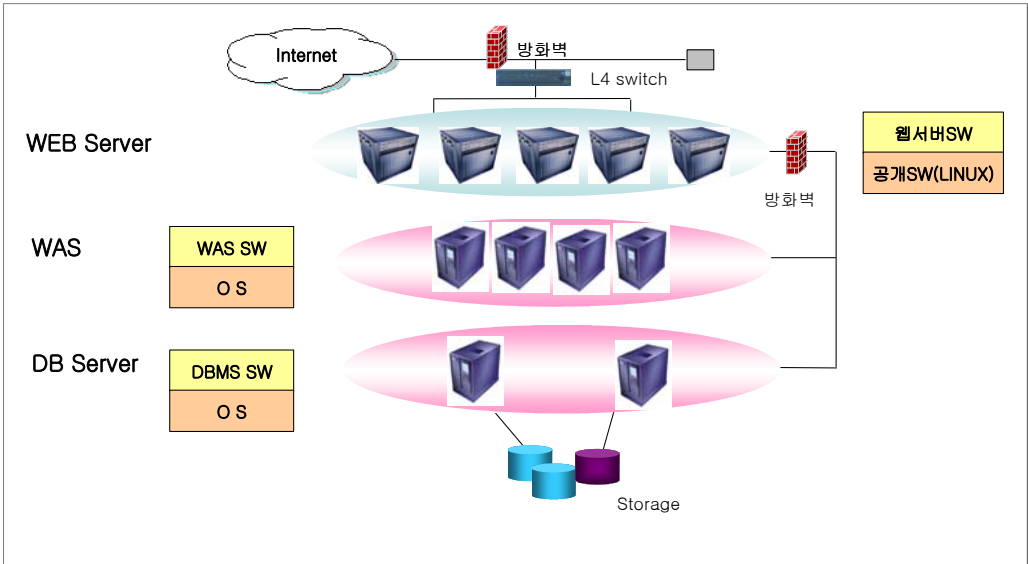
<그림 2> 웹기반 정보시스템 구조

웹 기반의 전자정부 등 공공 정보시스템을 공개소프트웨어기반으로

도입시 다양한 시나리오를 적용할 수 있는데, 그 가운데 대표적인 세 가지 시나리오별 예시하면 다음과 같다.

o 시나리오 1 : 웹 서버 운영체제 공개소프트웨어 도입

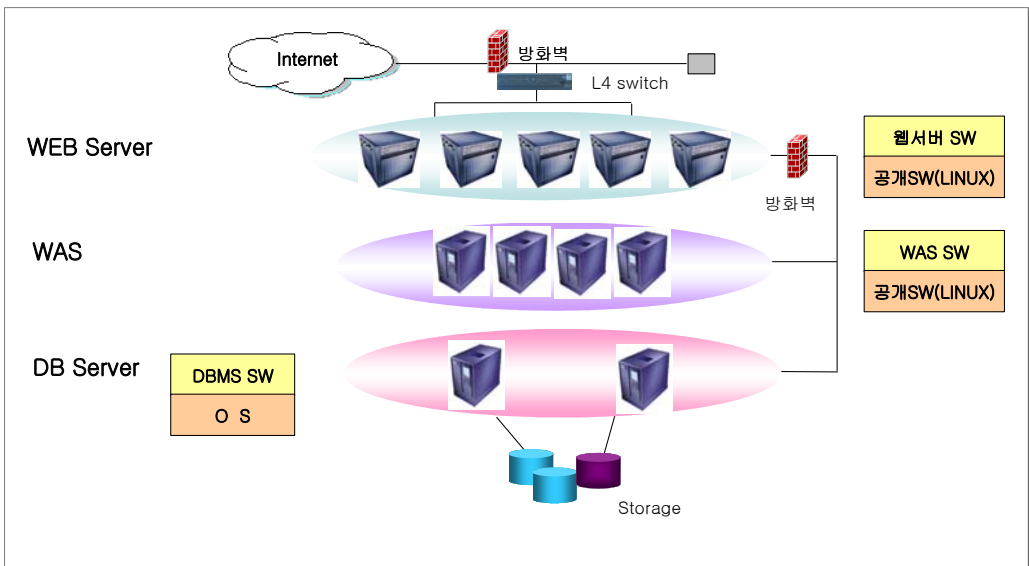
기존 3계층으로 구성된 환경에서 웹 서버 운영체제를 공개SW로 대체하는 경우로써, 공개SW의 안정성 및 성능에 대해 PoC(Proof of Concept)를 실시하고자 하는 경우 적합하다. 웹 서버에 공개소프트웨어를 도입하는 경우는 단순히 서버만을 교체하는 수준의 작업이 요구되므로 전환이 용이하다는 장점이 있다. 웹서버 소프트웨어는 공개소프트웨어(Apache, Tomcat 등)와 비공개소프트웨어(WebtoB, SunOne 등)와 모두 적용할 수 있다.



<그림 3> 웹서버 운영체제 공개SW 도입

o 시나리오 2 : 웹서버 및 웹 어플리케이션 서버 운영체제 공개 소프트웨어 도입

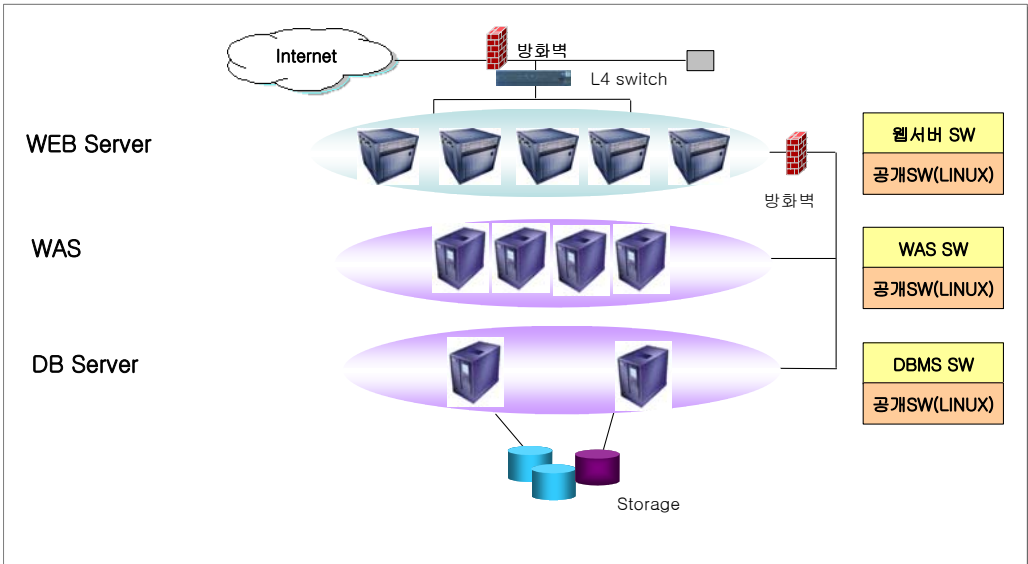
기존의 3계층으로 구성된 환경에서 웹서버, 웹 어플리케이션 서버 운영 체제를 공개SW로 도입하는 경우로써, 비용절감 효과를 원하며, 비교적 안전하게 공개SW를 도입하고자 하는 경우에 적합하다. 어플리케이션 서버의 경우 J2EE 표준준수로 비교적 추가 작업이 적고, 비용절감효과 크다. WAS는 공개소프트웨어(JBoss, Tomcat, JOnAS 등) 또는 비공개 소프트웨어(JEUS, Weblogic, WebShpere 등) 모두 적용 가능하다.



<그림 4> 웹서버, 웹 어플리케이션 서버 운영체제 공개SW 도입

o 시나리오3 : 적극적 공개SW 도입

기존의 3계층으로 구성된 환경 전체 운영체제에 공개SW를 도입하는 경우로써, 신규로 구축되는 사이트에 가장 적합하고, 비용절감 효과가 가장 크다. DBMS는 공개소프트웨어(MySQL, PostgreSQL 등) 또는 비공개소프트웨어(UniSQL, Oracle 등) 모두 적용가능하다.



<그림 5> 3계층 서버 운영체제 공개SW 도입

다. 적용 가능분야

대표적인 공개소프트웨어인 리눅스는 웹서버, 메일서버 등 인프라서버와 1way ~ 4way 엔트리급 서버에서는 이미 기술적 안정성을 입증받고 있으며, 리눅스 커널 2.6 출시 이후 DB서버, AP서버 등 기관 핵심 업무와 8 way 이상의 엔터프라이즈급 서버 운영체제로 폭넓게 활용되고 있는 추세이며, 특히 리눅스가 일반적으로 선택되고 있는 환경은 다음과 같다.

- ① 네트워크 인프라 : 도메인 네임 서버 (DNS), IP 주소 할당 (DHCP), 웹 서비스, 어플리케이션 서비스, 프록시 서버, 디렉토리 패킷 스위핑 및 통신 최적화를 위한 소프트웨어 포함
- ② 데이터베이스 서버 : 탁월한 오픈 소스 데이터베이스 서버로는 MySQL, PostgreSQL, MaxDB Firebird SQL(이전에는 Interbase),

Ingres(이전에는 Adabas)가 있으며, 다수의 비공개 데이터베이스 서버도 리눅스에서 이용 가능

- ③ 보안 시스템: 방화벽, 침입 탐지 및 분석, 허니포트(honeypots), IPSEC 및 기타 가상 사설망(VPN) 시스템, 패킷스니핑(packet-sniffing) 및 분석, 바이러스 백신 소프트웨어 및 스팸 방지 필터링 포함
- ④ 인터넷 및 인트라넷 출판: 웹 서버, 콘텐츠 관리 시스템(CMS) 플랫폼 및 워크플로우 관리 도구 포함
- ⑤ 문서 관리: 자동 전자 문서 캡처 시스템, 개정 관리 시스템, 데이터 캡처 기술, 문서보관 시스템 포함
- ⑥ 이메일 및 통신: 이메일, 일반 그룹웨어(그룹 캘린더 관리, 주소록 공유, 알리미 서비스, 공유 폴더) 및 인스턴트 메시지 교환 서버를 위한 다수의 솔루션 포함
- ⑦ 어플리케이션 서버: Java, PHP, Perl, Python 및 ZOPE 스크립트 툴, (Java 및) Java 2 Enterprise Edition(J2EE) 서버(예: JBoss), dotGNU .NET 공개소프트웨어 어플리케이션 서버에 기반한 광범위하게 사용되고 있는 웹 어플리케이션 서버 포함. 또한, 현재 다수의 비공개 어플리케이션 서버가 리눅스에서 이용 가능
- ⑧ 파일 및 프린트 서버: Unix NFS, Microsoft SMB/CIFS 및 Novell Netware NCP와 같은 대부분의 주요 파일 공유 프로토콜을 포함하는 툴

- ⑨ 저장: 몇몇 네트워크에 부착된 저장 장치들은 주로 공개소프트웨어 플랫폼에 구축되어 있음
- ⑩ 제한된 기능의 워크스테이션: 기본 웹, 이메일, 터미널 액세스 그리고 통화 센터, 키오스크 및 이와 유사한 용도를 위한 사무실 생산성 관련 기능을 제공하는 고정 용도의 워크스테이션
- ⑪ 고성능 컴퓨팅: 여기에는 다수의 마이크로프로세서 (수직 평가), 다수의 저비용 시스템(수평 평가) 및 기타 유형의 슈퍼컴퓨터에 기반한 클러스터를 장착한 단일 영상 시스템 포함
- ⑫ 고성능 기술 워크스테이션: 과학적 분석, 계량, 모델링, 3D 컴퓨터로 생성된 이미지 (CGI) 및 비디오 처리 기능과 같은 컴퓨터의 처리 기능을 강화한 어플리케이션용 멀티 프로세서, 64-비트 및 대용량 메모리 시스템

4. 고려사항

공개소프트웨어를 도입 방법, 절차 그리고 적용 가능한 시스템에 대하여 살펴보았으며, 공개소프트웨어를 도입할 경우 다양한 요구사항들이 도출되는 것을 확인했다. 따라서 이번 절에서는 공개소프트웨어를 도입함에 있어서 고려해야 할 사항들을 검토하면 다음과 같다.

첫째, 도입하고자 하는 공개소프트웨어에 대한 라이선스를 정확하게 확인하고 이해해야 한다.

공개소프트웨어로서 완전한 권리를 획득하기로 하였다면 해당 공개소프트웨어는 소스코드를 모두 공개하고 해당권리를 사용자에게 양도해야 한다. 하지만 대형 프로젝트 등에서는 공급자가 모든 소스에 대한 권리가 없는 경우가 많으므로 소스의 일부는 비공개소프트웨어거나 타인의 저작권에 속하는 기술로 개발된 부분일 수 있으므로, 이러한 부분의 검토도 간과하지 말아야 한다.

둘째, 시스템 호환성 확보를 위해 개방표준을 지원하는 제품을 우선적으로 고려한다.

정보시스템을 설계할 때에는 유연한 상호호환성 확보를 위해 개방표준을 지원하는 제품을 우선적으로 고려해야 한다. 모든 사람들이 읽어야 하는 자료를 웹을 통하여 제공할 경우에는 가능한 모든 정보통신 환경 사용자가 접근 가능하도록 제공하여야 한다.

셋째, 도입하는 공개소프트웨어에 대한 기술지원 및 유지보수에 대하여 고려해야 한다.

만약 간접공급 방식으로 공개소프트웨어를 도입하기로 결정하였다면 사용자지침서, 온라인 튜토리얼, 교육훈련, 헬프 데스크 및 유지보수가 원활해야 하며 또한 여러 플랫폼이나 사용 환경에 맞추어 맞추어 버전관리가 잘 이루어질 수 있는지 확인해야 한다. 직접선택 방식으로 공개소프트웨어를 도입한다면 위 사항에 대한 구체적인 대안이 마련되어야 한다.

넷째, 사업의 취지에 가장 적합한 솔루션인가를 깊이 있게 고려해보아야 한다.

제도적 또는 절차적 공정성을 확보하여 독점소프트웨어와 공개소프트웨어의 장단점을 고려하여 가장 적합한 솔루션을 선택해야 한다.

독점소프트웨어라고 하여 소프트웨어의 완성도가 반드시 높은 것은 아니며 공개소프트웨어가 완성도가 높은 경우도 흔히 있다. 따라서 사업의 취지에 맞는 소프트웨어의 장단점을 검토하여 가장 적합한 솔루션을 선택해야 한다.

다섯째, 도입되는 솔루션에 대한 합리성을 확보해야 한다.

소프트웨어 구매 시 같은 성능일 경우에는 가격이 낮은 제품을 선택하고, 같은 가격일 경우에는 소스코드의 확보 및 확보된 소스코드의 자유로운 변경이 가능한 제품을 우선적으로 고려해야 한다.

성능과 가격이 모두 같은 경우라면 소스코드의 확보가 가능한 것이 향후 프로그램의 유지보수에 대한 보장성과 사업의 영속성을 위하여 보다 합리적이므로 소스코드의 확보가 가능한 것을 선택하도록 한다. 단, 소스코드의 확보는 독점소프트웨어이든 공개소프트웨어이든 구분하지 않고 확보가능성 여부를 고려해야 한다. 여기서 주의할 것은 소스코드의 확보가 가능하다고 하더라도 이것의 자유로운 변경이 허락되어 있지 않은 경우에 추가비용이 필요할 수 있으므로 이러한 경우에도 소스코드의 자유로운 변경이 가능한 제품을 선택해야 한다.

여기서 주의할 것은 소스코드의 확보가 가능하다고 하더라도 한 번 수정이 가해지면 원래의 공개소프트웨어 공급처(또는 커뮤니티)의 소프트웨어와 호환성이 떨어져 향후 업그레이드가 어려워지므로 소스코드의 수정작업은 신중히 이루어져야 한다.

여섯째, 도입되는 공개소프트웨어의 소스코드에 대한 완전한 권리를 획득하였는가를 확인해야 한다.

도입하는 공개소프트웨어에 대한 소스코드의 확보는 매우 민감한

문제이다. 소스코드를 확보하지 못한다면 향후 수정, 패치, 업데이트 등과 같은 작업뿐만 아니라 시스템 증설작업이 필요할 경우에 사업자의 폐업과 같은 극한 경우에도 사업의 영속성을 보장받을 수 있기 때문이다. 따라서 공개소프트웨어 도입 시에는 소스코드의 확보와 함께 확보한 소스코드의 변경 가능여부와 변경한 소스코드를 적용가능항가를 반드시 확인해야 한다.

일곱째, 비공개소프트웨어를 선택하였을 경우와 마찬가지로 선택한 소프트웨어에 대한 기술적인 분석이 이루어져야 한다.

공개소프트웨어를 선택하든 독점소프트웨어를 선택하든 도입하기로 선택한 소프트웨어에 대해서는 기능성, 성능, 효율성, 그리고 확장가능성 및 시스템의 운용에 따르는 부가적인 문제로 구분하여 분석해야 한다.

여덟째, 도입하는 공개소프트웨어에 대한 법적인 분석이 이루어져야 한다.

어떤 무형의 권리를 “지적재산권”으로 총칭하여 언급하는 것이 편리하기는 하지만, 세부적인 내용을 살펴보면 이것은 많은 관점에서 서로 다른 법률들을 포함하는 집합적인 개념(collective term)이다. 따라서 실제로 무엇이 쓰여 지거나, 창조되거나, 개발되었는지, 그리고 어떤 종류의 지적재산권법이 관련되는지를 개별적으로 파악할 필요가 있다.

마지막으로, 웹 클라이언트⁴²⁾ 측면에서의 접근성에 대한 준수여부를 고려해야한다. 현재의 웹 기반 개발 환경이 MS Internet Explorer를 기준으로 개발되고 있어 리눅스, 맥킨토시 등 비 윈도우 운영체제

42) 웹 클라이언트란 3계층 구조 중 Presentation을 담당하고 있는 계층인 웹 브라우저를 의미한다. 현재(2004년 4월 기준) MS의 인터넷 익스플로러(이하 IE)가 80%이상의 점유율을 보이고 있다.

및 모질라, 사파리 등 비 IE 환경 사용자들의 인터넷 사용에 제약을 가하고 있다.

따라서, 웹을 통한 정보공개, 민원서비스 등의 시스템을 구축할 경우 다음의 기준을 준수하는 것이 필요하다

- ① HTML 문서 작성 시 표준 태그만을 이용하여 작성
- ② 자바스크립트 작성 시 표준을 따르는 코드만 이용
- ③ 사용자 인증 등을 위하여 Plug-in 기술을 사용해야 하는 경우 다양한 운영체제 및 웹브라우저에서 접근 가능하도록 설계⁴³⁾
- ④ 기타 특정기업에 의존성 있는 코드 사용 배제
- ⑤ 위에 언급된 대표적인 브라우저(MS IE, 모질라, 오페라, 넷스케이프)에서 동일하게 접근가능한가 테스트를 실시(W3C의 Validation Test(validator.w3c.org) 참조)

다양한 플랫폼 사용자의 웹콘텐츠 접근성 보장을 위한 기술지침은 정보통신접근성향상표준화포럼의 웹콘텐츠 접근성 지침 1.0⁴⁴⁾과 한국 소프트웨어진흥원의 크로스브라우징 가이드를⁴⁵⁾ 통해 세부적인 내용을 참고할 수 있다.

43) IE용 Plug-in 기술인 ActiveX 이외에 XPCOM 등 비 IE 사용자를 위한 대체기술 접근성 보장 필요

44) www.iabf.or.kr

45) user.oss.or.kr

BLANK

4장. 위험분석 및 관리

BLANK

1. 위험요소 분석

공개소프트웨어 도입 시에는 여러 가지 위험요소들이 존재한다. 이번 장에서는 이런 공개소프트웨어의 위험요소들에 대하여 몇 가지 분류로 나누어서 살펴보고자 한다.

가. 경쟁 유효성

공개소프트웨어를 도입함에 있어 존재하는 위험요소들과 공개소프트웨어를 도입하기를 망설이게 되는 여러 가지 도입 저해요소들은 상관관계가 존재한다. 먼저 본 가이드의 앞부분에서 다루었던 도입 저해요인들 가운데 경쟁 유효성에 관한 위험요소에 관련된 항목들을 간단히 정리해 보면 다음과 같다.

- ① 공급업체의 신뢰성과 경쟁력 확보 여부
- ② 도입하는 공개소프트웨어 자체의 경쟁력 확보 여부
- ③ 기능성 및 사용 편의성 등에 대한 경쟁력 확보여부
- ④ 비용절감의 경쟁력 확보 여부 등

첫째, 공급업체가 자체적인 경쟁력뿐 아니라 대외적인 경쟁력을 확보하였는가의 여부에 따라서 위험요소가 존재한다.

이 부분은 공급업체의 신뢰성이 우선 바탕이 되어야하며 신뢰성을 기본으로 하여 두 가지의 경쟁 유효성 즉, 공급업체 스스로가 내부와 외부의 경쟁력을 확보하고 있는가에 대한 것과 공급되는 특정 공개소프트웨어를 기술 지원할 수 있는 업체들이 다수 존재하여 이들 업체들이 상호 경쟁력 확보를 하고 있으며 도입기관에서 기술지원 요청을 할 수 있는 다수 기업체가 존재하는가에 대한 것이다. 이중 후자는

공급받는 공개소프트웨어에 대한 영속성 확보를 위한 위험요소 감소가 그 목적이라고 할 수 있다.

둘째, 도입하는 공개소프트웨어 자체 경쟁력에 대한 부분이다.

공개소프트웨어뿐만 아니라 어떤 소프트웨어든 도입되기 위해서는 동일 분야의 타 제품보다 경쟁력이 확보되어야 한다. 만약 도입한 이후에 보다 나은 소프트웨어를 찾게 된다면 도입자체를 재검토해야 하는 심각한 문제가 발생하게 된다. 따라서 이러한 위험요소를 줄이기 위하여 공개소프트웨어 자체에 대한 경쟁력 확보 여부를 반드시 확인해야 한다.

셋째, 도입하는 공개소프트웨어의 사용자측면의 UI(User Interface)와 기능적인 경쟁력이 있는가에 대한 것이다. 즉, 이것은 도입되는 공개소프트웨어에 사용상의 편리성과 기능상의 우수성에 대한 경쟁력을 확보하고 있는가에 대한 것으로써 훌륭한 소프트웨어라 하더라도 사용이 불편하다면 초기 도입을 위한 교육비용이 높아질 수 있다.

넷째, 공개소프트웨어 도입으로 얻으려고 하는 목적에 가장 높은 경쟁력을 확보한 공개소프트웨어의 선정이다. 공개소프트웨어의 장점은 비용절감 뿐만 아니라 기술 표준성 확보, 호환성 보장, 확장성, 영속성 등 여러가지가 존재한다. 수행하려는 사업의 특성을 분석하여 가장 우선시 되어야 하는 경쟁력을 선정하고 이에 적합한 공개소프트웨어를 도입하여야 한다.

그리고 기존 시스템에 공개소프트웨어를 도입할 때에는 업체의 인적 기술력과 기술지원 경쟁력을 확보했는지 고려해야 한다. 즉, 기존 정보시스템, 기존 소프트웨어와의 호환성 및 기능의 제약 등 기술적

계약 요건을 분석하고 기존 시스템에 공개소프트웨어를 도입함으로써 발생할 수 있는 문제를(예를 들어 기존 비공개소프트웨어와의 호환성 문제, 추가 개발이 필요한 부문이 없는지 등) 기술지원을 제공하는 기업체에서 성실하고 적극적인 기술지원 및 교육을 제공하는지 여부를 확인하여야 한다.

나. 보증문제

공개소프트웨어와 독점 공급 되는 비공개 소프트웨어는 보증문제에 있어서도 큰 차이점이 있다. 독점 공급 되는 비공개 소프트웨어는 구매 시에 보증문제가 이미 언급되어 있고 계약서에도 대부분 명시되어 있으나 공개소프트웨어는 보증에 대해서 다소 분명하지 못한 점이 있다.

GPL 제12항 및 제13항을 보면 GPL 라이선스로 제공되는 모든 공개소프트웨어의 문제발생에 대한 책임은 원칙적으로는 수요자에게 있다는 것을 알 수 있다. 물론 공개소프트웨어를 도입하는 기업과 수요자 사이에 보증에 대한 별도의 계약관계가 있다면 보증에 대한 문제를 해결할 수 있을 것이다. 하지만 GPL에서는 공급자와 수요자 사이에 보증에 대한 별도의 계약을 하였다하더라도 근원적인 보증의 결여를 제한할 수는 없다고 명시하고 있다.

그렇다면 공개소프트웨어를 도입하려는 수요자의 입장에서는 다음과 같은 의문점을 가지게 된다. 많은 장점이 있다고는 하지만 공급자가 확실히 보증하지 않는 공개소프트웨어를 어떻게 믿고 도입해야 하는가?

공개소프트웨어의 보증에 대한 문제의 출발점은 여기서 부터 출발한다. 즉, 이번 절에서는 공개소프트웨어를 도입하면서 어떻게 하면 보증의 결여에 대한 대책을 세우고 안정되고 영속성 있는 공개소프트웨어를 도입할 것인가에 대하여 살펴볼 것이다.

이러한 공개소프트웨어의 보증의 결여에 대한 근본적인 원인은 공개소프트웨어가 공급자의 소유가 아닌 프로젝트의 산출물이기 때문이다. 물론 많은 이유가 존재하겠지만 공동의 프로젝트에 의하여 공동개발된 소프트웨어이기 때문이며 이는 결코 보증의 결여에 대한 문제가 단점으로만 작용하고 있는 것은 아니다. 즉, 많은 개발자들에 의해 자발적인 공동개발 산물이라는 것은 문제 발생 시에 빠른 보완능력을 그 자체로서 가지고 있기 때문이다.

따라서 공개소프트웨어의 보증의 결여에 대한 문제해결의 실마리 또한 여기서 출발한다. 예를 들어 공동프로젝트의 산출물(A), 공개소프트웨어 공급업체(B), 공개소프트웨어 도입 기관(C)이 있는 경우를 가정했을 때, 공공기관C는 공개소프트웨어를 도입하기로 최종 결론을 내리고 공개소프트웨어 공급업체 B와 계약하여 A라는 공개소프트웨어를 GPL 라이선스를 적용하여 공급받기로 하였다.

여기서 B와 C의 계약서 작성 시에 보증문제에 대하여 공공기관C가 주의해야 할 것이 있다. 공개소프트웨어 A의 근원적인 문제에 대해서는 보증을 받을 수 없지만 공개소프트웨어 A를 공동 개발하는 프로젝트 내에서 근원적인 문제가 해결되어 패치버전이 나온다면 B는 즉시 이를 적용하여 도입한 공개소프트웨어를 개선해야한다.

즉, 이는 근원적인 문제를 원천적으로 해결해야한다는 조건이 아니

라 근원적인 문제에 대한 해결이 이루어 졌다면 공급업체에서 도입한 공개소프트웨어에 이를 즉시 적용하여 개선해야 한다는 의미이다. 바로 이것이 공개소프트웨어를 도입할 때에 공공기관C가 가장 우선적으로 검토해야 할 보증문제의 기본적인 해결책이다.

앞서 살펴본 특정 공급업체로부터 공급받았을 경우에 보증의 문제가 발생하지만 업체로부터 공급받지 아니하고 공개소프트웨어를 도입하기 위하여 무료다운로드를 제공하는 사이트에서 직접 다운로드를 하였을 경우에도 다운로드한 소프트웨어의 보상과 보증은 전혀 이루어 지지 않는다. 따라서 이에 대한 위험요소의 대책도 세워야 한다.

즉, 특정한 사이트에서 다운로드를 받은 공개소프트웨어에 대한 가장 기본적인 대책은 배포처 사이트의 주소뿐 아니라 다운로드한 공개소프트웨어에 대한 상세정보를 확보해야 한다. 즉, 공개소프트웨어의 프로젝트 진행자 또는 개발자, 또는 개발 배포회사에 대한 가장 기본적인 정보를 의미한다. 이렇게 확인해 두어야만 배포사이트의 주소가 바뀌거나 프로젝트 진행자가 바뀌거나 또는 개발자, 배포회사가 사라지더라도 최소한의 문제해결 실마리는 얻을 수 있기 때문이다. 이것은 무료 다운로드한 공개소프트웨어의 위험요소를 줄이기 위한 최소한의 조치인 것이다.

그리고 보증의 문제에 대한 도입기관의 또 다른 대책으로는 지역 내에 가능한 많은 공개소프트웨어 지원업체를 알아두어야 한다는 것이다. 공개소프트웨어를 공급한 업체가 사라지거나 또는 기술지원을 중단하더라도 지속적인 보증과 지원을 받을 수 있는 가능성이 높아지기 때문이다. 이것은 도입기관에서 자체적으로 개발하여 도입하는 경우에도 해당된다. 즉, 기관내부에서 공개소프트웨어를 자체 커스터마

이정하여 도입하는 경우에도 외부의 지원업체를 많이 알아 두는 것이 안정성과 영속성을 높이기 때문이다.

이와 같이 보증에 대한 여러 가지 안정장치들은 공개소프트웨어가 핵심적인 업무에 도입되는 경우에 가장 큰 효과를 얻을 수 있다.

지금까지 설명한 공개소프트웨어의 보증의 결여에 대한 위험요소를 감소시키려면 공개소프트웨어의 도입 시 기본적인 조건이 충족되어야 한다. 이는 보증의 결여에 대한 대책이 아니라 지금까지 설명한 보증 결여에 대한 대책 수립을 위한 전제조건이다.

먼저, 도입하는 공개소프트웨어의 라이선스가 명확해야 한다. 물론 거의 대부분 GPL로 도입이 되겠지만 공개소프트웨어에 적용되는 라이선스의 종류가 많기 때문이며 이들은 필요에 의해 수정되거나 새로 만들어진 것이므로 GPL과는 다른 부분들이 존재한다. 따라서 도입하는 공개소프트웨어에 적용되어 있는 라이선스와 그 내용을 꼼꼼히 확인해 보아야 한다.

하지만, 어떤 라이선스를 적용하든 도입하는 공개소프트웨어에 대한 완전한 권리를 획득하기 위해서는 소스코드의 확보와 수정권한 그리고 수정 후 이를 재사용하고 확장할 수 있는 권한을 모두 획득해야 한다. 이는 공개소프트웨어의 보증문제를 해결하기 위한 가장 기본적인 조건이다. 물론 당연한 것이기는 하지만 만약 이 조건이 만족되지 않는다면 공개소프트웨어의 도입자체부터 재검토해야하며 도입된다 하더라도 보증문제는 전혀 보장되지 않는다고 보아야 한다. 따라서 소스코드에 대한 권리는 반드시 확인해 보아야 한다.

그리고 보증문제에 대하여 한 가지 더 확인해 두어야 할 것은 간접

도입 방식으로 전문 기술지원 업체로부터 공개소프트웨어를 도입하는 경우, 도입하는 공개소프트웨어에 대한 기술문서, 사용자지침서, 온라인 튜토리얼, 교육 훈련 등에 대한 확보이다. 이절의 앞부분에서 말씀드렸듯이 공개소프트웨어의 보증에 대한 보장은 1차적으로 공급업체에서 이루어져야하겠지만 불가피하게 보증문제가 발생하였을 경우에는 기술문서등과 같은 문서들이 확보되어 있어야만 해결이 가능하기 때문이다.

다. 소스코드의 가용성

공개소프트웨어를 도입함에 있어 발생할 수 있는 위험요소들 가운데 경쟁우�효성과 보증문제에 대한 위험요소들에 대하여 살펴보았다. 이번에는 공개소프트웨어의 소스코드에 대한 위험요소들이 어떤 것들이 있는가에 대하여 살펴볼 것이다. 즉, 이번 절에서는 소스코드의 가장 핵심적인 요구조건인 가용성에 대하여 살펴볼 것이다. 첫째 소스코드 자체의 확보여부, 소스코드의 수정 및 수정후의 재사용, 확장성 보장등과 함께 소스코드 자체로서 경쟁력을 가지고 있는지 그리고 향후 확장성을 보장할 수 있도록 코딩되어 있는지, 소스코드의 수정 시에 어떤 개발자라도 수정작업이 용이하도록 보장되는지, 그리고 여러 루틴들이 상호 유기적으로 표준화되어 코딩되었는가를 검토해 보아야 하며 이러한 부분들이 보장이 되어야만 소스코드 자체의 경쟁력과 가용성을 보장받을 수 있다.

- ① 소스코드의 확보
- ② 소스코드의 수정 가능성 확보
- ③ 수정한 소스코드의 재사용, 확장성, 재배포 가능성 확보
- ④ 확장 / 수정 용이성

⑤ 소스코드의 표준화

공개소프트웨어로 도입하는 소프트웨어의 소스코드는 많은 측면에서 보장되어야 한다. 만약 독점 공급되는 비공개 소프트웨어라면 소스코드 자체를 확보하기가 거의 어렵기 때문에 이러한 점들을 고려할 필요가 없겠지만 공개소프트웨어는 소프트웨어를 공급받을 때에 소스코드를 함께 공급받기 때문에 이에 대한 가용성 문제를 검토함으로써 공개소프트웨어의 장점을 극대화할 수 있다. 특히 직접선택 방식으로 공개소프트웨어를 사용하는 경우, 소스코드의 재사용에 관련된 가용성 확보는 필수적인 것이다.

다행스럽게도 GPL을 포함한 대부분의 공개소프트웨어 라이선스에는 소스코드를 공개해야 한다는 조건이 있으므로 간단한 확인만으로도 소스코드 확보는 어렵지 않다.

둘째, 소스코드에 대한 재사용, 확장성, 재배포 권한여부를 확인해야 한다. 모든 공개소프트웨어는 소스코드의 제공 및 수정에 대한 권리를 제공하지만 수정한 소스코드의 배포에 대한 권한은 공개 소프트웨어 라이선스의 종류에 따라 틀리다. 예를 들어 수정한 공개소프트웨어를 사업용으로 재배포 가능 또는 불가능한 경우, 재배포할 때에 수정한 소스코드를 함께 공개해야 하는 의무등을 부가하는 라이선스도 있다. 그러므로 보유한 공개소프트웨어에 대한 라이선스를 확인하여 재배포에 부여되는 조건을 확인해야 한다.

앞에서 언급한 두 가지 조건이 소스코드 가용성의 전제조건이다. 즉, 소스코드의 확보, 수정권한, 재사용권한이 소스코드의 가용성에 대한 기본조건이 된다는 의미이다.

결론적으로 소스코드의 가용성에 대한 위험요소를 줄이기 위해서는 도입 시 이 두 가지가 전제되어야 한다는 것을 알 수 있다.

직접선택 방식으로 공개소프트웨어를 도입하는 경우 소스코드의 수정 및 확장성을 높이기 위하여 소스코드의 표준화를 추가하여 검토할 수 있다. 이 또한 소스코드의 가용성을 높이는 것으로써 표준화되어 있는 소스코드는 가독율이 좋으며 개발자가 달라지더라도 수정이 용이해지며 소스코드를 확장하기에 매우 유용하기 때문이다. 따라서 소스코드의 표준화는 가용성을 높이는데 매우 큰 역할을 한다.

2. 위험관리 및 완화방법

가. 도입유형별 위험평가

이미 설명하였듯이 공개소프트웨어를 도입하는 방법에는 크게 3가지가 있다. 도입하려고 하는 공개소프트웨어를 기관에서 직접 선택하는 직접선택 방법이 있고, 외부의 공개소프트웨어 개발업체로부터 도입하는 간접공급 방법이 있으며 나머지 하나는 기관내부에서 공개소프트웨어를 직접 개발하여 도입하는 내부개발 방법이 그것이다.

이번 절에서는 이 3가지 도입유형에 따른 도입 시의 위험요소에 대하여 알아보도록 할 것이다.

○ 직접선택 방법에 따른 위험 평가

공개소프트웨어를 기관에서 직접 다운로드하여 도입하고자 하는 경우 비교적 많은 위험요소가 내재되어 있다.

가장 먼저 주의해야 할 사항은 다운로드한 공개소프트웨어가 정말 깨끗하고 믿을 수 있는 좋은 소프트웨어인가를 확인하기 전까지는 매우 불안한 요소를 내포하고 있을 가능성이 있기 때문이다. 이 말의 의미는 다운로드한 공개소프트웨어 내에 트로이목마와 같은 바이러스 코드나 또는 키로거(key-logger)와 같은 악성 소프트웨어에 의한 감염된 소스등과 같은 악의적인 목적을 가진 소스코드가 내포되어 있을 가능성이 존재하기 때문이다.

그리고 앞 절에서도 살펴보았던 문제이지만 직접 다운로드한 소프트웨어에 대해서는 보상과 보증이 명확하지 않고 보장되지 않는다는 점이다. 즉, 공개소프트웨어 솔루션을 기관내부에서 직접 선택하여 도입하려고하는 경우에는 소프트웨어의 보증과 보상을 보장받지 못하는 위험요소가 존재한다.

또 다른 위험요소는 공개소프트웨어의 배포정보에 대한 부분이다. 즉, 공개소프트웨어를 지속적으로 업그레이드해서 배포해야한다는 의무적인 사항은 그 어디에도 존재하지 않는다. 따라서 오늘 다운로드한 공개소프트웨어가 내일은 배포되지 않을 수도 있고 또한 어떠한 공지사항 없이 배포위치가 바뀔 수 있기 때문이다. 따라서 공개소프트웨어를 직접 다운로드하여 도입하는 기관에서는 배포처나 개발프로젝트에 대한 위험요소 감소대책을 확보해야한다.

그리고 어떤 공개소프트웨어에도 존재할 수 있는 공통적인 위험요소로는 기술지원에 대한 위험요소이다. 특히 공개소프트웨어를 직접 다운로드하여 도입한 기관에서는 자체 기술지원시스템을 확보하고 있

거나 유사시에 도움을 요청할 수 있는 공개소프트웨어 커뮤니티 또는 전문 기술지원업체에 대한 다양한 정보를 확보하고 있어야 한다. 즉 직접 다운로드하여 도입하는 공개소프트웨어에 대해서는 더욱 적극적이고 명확한 기술지원에 대한 위험요소 감소대책을 확보해야 한다.

○ 간접공급 방법에 따른 위험 평가

간접공급에 의한 공개소프트웨어 도입이란 수요자가 공개소프트웨어를 외부업체를 통해 공급받는 것을 의미하는 것으로서 공공기관에서 직접 선택하는 앞의 방법보다는 위험요소를 감소시킬 수 있는 방법이다. 즉, 외부업체를 선택하여 선택한 업체에서 공개소프트웨어를 공급하는 것으로 기술지원과 서비스 비용을 지불해야하기 때문에 비용적인 부담이 있기는 하지만 위험요소를 줄일 수 있다는 장점이 있다. 하지만 이 경우에도 공개소프트웨어의 도입에 따른 위험요소들이 존재한다.

따라서 업체에서 공급하는 공개소프트웨어에 적용되어 있는 라이선스가 어떤 것인가를 명확하게 확인해야 한다. 계속해서 언급하였듯이 공개소프트웨어에 적용되어 있는 라이선스는 여러 가지 종류가 있다. 이들은 필요에 의하여 그 환경과 요구사항에 맞도록 만들어진 것들이며 많은 부분에서 그 내용을 달리하는 경우가 있다. 따라서 간접방식으로 공급되는 공개소프트웨어 공급 계약서 내에 라이선스에 대한 명확한 언급이 있어야만 한다. 물론 도입기관의 담당자는 그 라이선스의 내용에 대하여 명확하게 숙지해야 한다.

그리고 외부 업체에서 공급하는 간접공급방식에서의 또 다른 위험요소로는 하자보증에 대한 부분이다.

1차적으로 공급업체는 하자보수에 대한 지원을 해야 하며 또한 책임을 져야하지만 공개소프트웨어의 GPL 라이선스가 적용된다면 근원적인 문제점에 대해서는 공급업체에서 보증에 대한 책임을 회피할 수 있다는 위험요소가 발생한다. 따라서 공개소프트웨어를 도입하는 기관 담당자는 보증에 대한 위험요소에 대한 대비책을 세워두어야 한다.

그리고 간접방식으로 공급되는 공개소프트웨어에 대한 위험요소는 일반적인 공급계약방식에서 발생할 수 있는 거의 모든 위험요소가 고려되어야 한다.

즉, 공급업체의 적격성에 대한 위험성 평가, 시스템의 확장성 가능 여부에 대한 위험요소 평가, 소스코드의 성숙도 및 신뢰도에 대한 위험요소 평가 등이 모두 고려되어야 한다.

마지막으로 간접방식에서는 외부 공급업체의 기술지원 능력에 대한 위험요소가 존재한다. 즉, 공급업체의 기술지원 능력에 대한 객관적인 검증 없이 공개소프트웨어를 도입하였다면 도입 이후의 기술지원 문제가 발생할 수 있다는 점을 명심해야 한다. 아울러 비공개소프트웨어의 경우와 마찬가지로 공급업체의 파산 등으로 기술지원을 받지 못하는 사태가 발생할 수 있으므로 동일 솔루션의 기술지원업체에 대한 정보를 확보해 두는 것도 공개소프트웨어의 장점을 극대화하여 위험요소를 줄일 수 있는 좋은 방법이다.

○ 내부개발 방법에 따른 위험평가

마지막으로 내부개발 방법에 의한 공개소프트웨어 도입에 따른 위험 요소는 다음과 같다.

- ① 다운로드한 공개소프트웨어의 안정성 검토
- ② 개발된 공개소프트웨어의 라이선스 문제
- ③ 개발된 공개소프트웨어의 안정성과 영속성 문제
- ④ 내부 개발자의 개발능력 문제
- ⑤ 개발 후 문서화 작업 문제 등

앞 절에서 이미 설명하였듯이 기관 내부에서 공개소프트웨어를 개발하기 위해서는 이미 배포되어 있는 공개소프트웨어들 가운데 가장 적합한 공개소프트웨어를 우선 선택하는 작업을 해야 한다. 이때 발생할 수 있는 위험요소는 선택하는 공개소프트웨어의 안정성과 적합성을 검토해야 한다. 즉, 다운로드하는 공개소프트웨어 소스내부에 악의적인 코드가 들어있지 않는가를 확인해야하며 또한 개발하고자 하는 소프트웨어의 목적에 가장 적합한 소프트웨어인가를 검토하는 작업을 해야 한다. 특히 공공기관의 업무에 적용하기 위한 경우 소프트웨어의 기술적 안정성 및 신뢰성 검토에 신중을 기하여야 한다.

다음은 다운로드하는 공개소프트웨어에 적용되어 있는 라이선스, 그리고 개발완료 후에 적용할 라이선스 문제를 검토해야 한다. 개발 전에 공개된 공개소프트웨어를 다운로드할 때에 소스확보가 가능한지, 소스수정이 가능한지, 그리고 소스 수정 후에 업무에 적용가능한지, 유관기관에 이를 재배포할 수 있는 재배포 권한이 있는가에 대한 면밀한 검토가 도입 전에 이루어져야만 한다. 만약 이에 대한 정확한 검토 작업 없이 추진하였다면 개발 완료되었을 때에 업무적용 및 활용, 배포에 심각한 문제가 발생할 수 있다는 위험요소를 가지고 있으

므로 주의해야 한다.

다음은 개발된 공개소프트웨어의 안정성과 영속성에 대한 문제이다. 바로 앞에서 언급한 라이선스 문제와 일부분 결부되는 문제이지만 소프트웨어의 안정성문제는 소스코드의 성숙도측면에서 고려되어야 한다. 그리고 소프트웨어의 영속성 문제는 시스템증설이나 향후 업그레이드시에 재활용 및 재사용이 가능한가라는 확장측면에서 고려되어야 할 문제이다. 따라서 소프트웨어의 안정성부분에 따르는 위험요소와 향후 확장 및 증설측면에서의 영속성에 따르는 위험요소도 고려되어야 한다.

다음은 공개소프트웨어의 소스코드를 도입사업에 적합하도록 수정한 후, 해당 소프트웨어의 버전관리에 대한 방안이다. 공개소프트웨어는 사업에 적합하도록 수정한 이후에는 해당 공개소프트웨어의 업체에서 새로 출시된 버전과 호환성에 문제가 발생할 수 있으며 특히 수정한 부분에서 기술적인 문제가 발생하였을 때 외부의 공개소프트웨어 커뮤니티에서 해결책을 구하기가 어려울 수도 있다.

그리고 개발 완료된 이후에 개발된 공개소프트웨어에 대한 문서화 작업을 해야 한다. 문서화 작업은 크게 두가지로 나누어서 이루어져야 하는데 첫번째는 소스코드에 대한 문서화 작업, 두번째는 소프트웨어 사용에 대한 문서화 작업이 그것이다. 첫번째 소스코드에 대한 문서화 작업은 소스코드의 재사용, 업그레이드 및 패치, 향후증설작업등을 위하여 소스코드에 대한 자세한 설명과 함께 수정 방법 등에 대한 문서를 의미한다. 이 문서화 작업은 처음 개발했던 개발자를 위해서이기도 하지만 다른 개발자가 보았을 때에 도움이 되도록 작성되어야 한다. 두번째 사용법에 대한 문서화 작업으로써 이는 개발완료 이후

에 업무에 적용하여 사용자들이 이를 원활하게 사용할 수 있도록 사용법문서등을 의미하는 것이다. 즉, 설치문서, 사용법문서등을 의미한다. 이와 같은 문서화 작업이 제대로 이루어져 한다. 왜냐하면 개발된 소프트웨어의 확장성과 활용성에 따르는 위험요소가 발생할 수도 있기 때문이다.

나. 기술로드맵 평가

기관에서 소프트웨어를 도입할 때에는 비공개 소프트웨어 뿐만 아니라 공개소프트웨어를 도입할 때에도 기술적인 로드맵을 확보해야 한다. 기술 로드맵이란 도입하는 소프트웨어에 향후 추가할 기능 개선점과 전달 시간계획에 대한 개요를 제공해 주는 문서이다. 이 문서의 목적은 소프트웨어를 도입하는 고객이 해당 소프트웨어의 발전방향을 미리 파악하여 향후 시스템 구축 계획에 반영할 수 있도록 하기 위한 것이다.

물론 기술 로드맵의 일정과 기능이 반드시 지켜진다고 보장이 되는 것은 아니나 이것은 독점 공급되는 비공개 소프트웨어에서도 마찬가지이다. 그러나 이러한 객관적인 기술 로드맵을 통하여 기관은 해당 소프트웨어의 발전 방향을 예측할 수 있으며 특정 기술의 단종에 따른 위험요소를 평가하고 대책을 수립하는데 큰 도움이 된다.

또한 이에 못지않게 중요한 것은 미래 비전에 대한 공급업체의 수행 능력이다. 만약 공급업체가 제품 로드맵, 또는 일반 기술 플랫폼을 수정하거나 변경한다면 이는 위험요소가 매우 높은 업체로 인식해야 한다. 따라서 도입기관은 개발업체에서 공급하는 제품이 원래 의도했던 대로 미래지향적이지 못하거나 훌륭하지 못하다고 판단하게 될 것

이다. 또한 공급받은 제품이 몇 년 후에 발생할지 모르는 기술지원에 대한 보증도 할 수 없을 것이다.

또한 기관은 미래의 위험요소를 보다 정확하고 적절하게 평가할 수 있도록 제품 로드맵을 검토해야 한다. 이러한 로드맵에 대한 정기적인 검토는 기관이 공급업체의 신뢰성과 능력을 파악하는데 큰 도움이 된다. 그리고 이러한 작업들은 미래의 계획 수립 시에 위험요소를 더욱 줄이는 훌륭한 방법이 다.

로드맵 평가 및 검토 과정은 다음사항을 고려한 절차대로 이루어지는 것이 좋다.

- ① 공급받은 제품의 개발업체가 제공한 로드맵을 배치하고 기록한다.
- ② 개발업체(또는 커뮤니티)가 발표한 최신 개발정보 및 제품 배포 일 및 버전에 대한 정보를 비롯하여 사용자 커뮤니티에 대한 정보를 배치하고 기록한다.
- ③ 현재의 타임라인(timeline)을 점검한다.
- ④ 가능한 경우 개발업체 또는 대표 개발자에게 직접 연락하여 (개발업체의 대표자에게 연락한다.) 로드맵 및 배포 일정에 대한 업데이트의 주기 등에 대한 정보를 확인한다.
- ⑤ 제품 로드맵과 배포 일정을 모두 포함하는 웹 페이지의 개별 사본을 배치하고 기록한다.
- ⑥ 변경내용을 확인할 수 있는 이전의 로드맵과 최신 로드맵을 비교한다. 특히 추가 예정되었던 사항이 제거된 항목이 있다면 이를 확인한다.
- ⑦ 추가된 항목을 확인할 수 있는 이전버전과 소프트웨어 개발자의 최신 뉴스 항목을 비교 한다.

- ⑧ 최초로 제안되었던 이정표의 내용들이 실제 실행되었는지 확인할 수 있도록 현재 뉴스 항목 웹 페이지에서 확인된 타임라인과 날짜를 비교 한다.
- ⑨ 분석 기간이 요구된다면 이 주기를 비교한다.

이러한 정보들은 공급되는 공개소프트웨어 제품 뿐 아니라 공급업체(개발업체)에 대한 위험요소 평가의 일부분으로 구성되어야 한다. 이러한 과정은 소프트웨어를 생산하고 유지 관리하는 주요 부분에서 발생할 수 있는 문제점들을 도입기관에게 알려주는 매우 유용한 역할을 하게 된다.

따라서 문제 발생에 대한 조기 경고를 함으로써 도입되는 공개소프트웨어의 위험요소를 줄일 수 있고 또한 효율적인 관리를 할 수 있다.

기술로드맵의 변경에 따른 위험요소 관리 방법은 다음과 같다.

공개소프트웨어를 개발하는 어떤 개발업체들은 품질 및 세부사항에 대한 기술 로드맵을 충분히 제공하지 못할 수도 있다. 더욱이 이미 마련되어 있는 자체 로드맵 또한 일관성 있게 지키지 못하는 경우도 있다. 이러한 개발업체들로 인하여 도입기관은 장기적인 미래전략 수립을 하는 것이 매우 힘들게 된다. 결론적으로 이러한 로드맵에도 위험요소가 존재하고 있다.

이러한 로드맵에 대한 위험요소를 줄이는 방법은 아주 간단한 원칙에서 찾을 수 있다. 기관이 타 기관 또는 사용자들이 광범위하게 사용하고 있는 공개소프트웨어 제품을 미리 선택하고 공급업체에게 이미 선택한 제품을 공급하도록 한다면 위험요소의 상당부분을 줄일 수

있다. 일반적으로 많은 사람들로 부터 광범위하게 사용되어지고 있는 소프트웨어에 대한 위험요소는 매우 낮을 뿐 아니라 기관 및 사용자들이 요구하는 거의 모든 기능과 특징들을 이미 갖추고 있기 때문이다.

이상과 같이 공개소프트웨어의 도입에 따른 로드맵 확인과 로드맵의 평가, 그리고 로드맵의 관리방법과 위험요소들에 대하여 알아보았다. 결론적으로 공개소프트웨어를 선택하고 도입할 때에는 기술로드맵이 있는가를 확인하고 이를 평가하는 과정을 거쳐야 한다. 이러한 작업들으로써 우리는 미래에 있을 위험요소 즉, 미래 위험요소를 줄일 수 있다.

다. 위험완화를 위한 검토사항

지금까지 공개소프트웨어의 도입에 따르는 위험요소들에 대한 분석과 관리방법에 대하여 알아보았다. 이제 이러한 위험요소를 줄이기 위한 방법들을 살펴보겠다.

공개소프트웨어를 도입하기로 결정하기 전에, 도입하고자하는 제품의 기술, 이전 기록, 사용 패턴, 인기도, 기존 사용자의 피드백을 신중하게 검토할 필요가 있다. 다음은 공개소프트웨어를 도입할 때에 검토할 항목들이다.

- ① 도입하는 공개소프트웨어가 배포사이트에서의 알림사항과 제공 문서에 기술되어 있는바와 같이 운영되는가를 확인해야 한다.
- ② 도입기관의 필수 요구항목과 요구 성능이 일치하는지 또는 그 이상의 성능을 보유하고 있는가를 확인한다.

- ③ 도입하는 공개소프트웨어 프로젝트가 지속적으로 진행되고 있는가를 확인한다. 즉, 도입하는 공개소프트웨어의 프로젝트 사이트에 대하여 다음 사항을 검토해야 한다.
- 배포되어 있는 소프트웨어의 보안패치 및 업데이트
 - 해당 프로젝트에 대한 공지사항 및 발표 사항 등의 유무
 - 프로젝트 내에서의 포럼, 메일링, 문서 등의 활용유무
 - 기술방향을 세부적으로 기술해 둔 프로젝트 로드맵
 - 프로젝트 참여 개발자와 사용자의 지속적인 확대 여부
- ④ 제품의 기술지원을 위한 지역 기술지원업체의 가용성을 확인한다.
- ⑤ 최신 발표에 열거된 공개된 웹사이트에서 해당 소프트웨어가 지원되는가?
- ⑥ 해당 소프트웨어의 웹사이트가 이전 버전에 대한 기록을 보유하고 있는가? 그리고 보유기록에 의해 지금까지의 배포횟수와 빈도를 점검할 수 있는가?
- ⑦ 배포 사이트가 보안 수정 및 특징 업데이트에 대한 정보를 제공하는가?
- ⑧ 해당 공개소프트웨어를 위해 공개적으로 사용할 수 있고 독립적인 사용자 포럼이나 메일링 리스트를 보유하고 있는가?
- ⑨ 포럼이 사이트에서 또는 공공의 검색 엔진을 통해 검색될 수 있는 문서보관소를 보유하고 있는가?

-
- ⑩ 소프트웨어가 프로젝트 웹사이트에서 이용 가능한 로드맵을 발행하여 보유하고 있는가? 그리고 이 로드맵이 기관의 요구조건을 충족시키는가?
 - ⑪ 프로젝트가 기관의 요구조건을 충족시키기 위해 추가 기능을 필요로 하는 경우, 개발팀에 새로운 기능이나 개선품을 요청할 수 있는 프로세스를 보유하고 있는가?
 - ⑫ 프로젝트 웹사이트에 지역 기술지원 제공업체가 열거되어 있는가? 그리고 이들 지역 기술지원업체들에 대한 위험요소 평가를 수행하였는가?
 - ⑬ 해당 공개소프트웨어를 적절히 운용하기 위해 필요한 보조 구성요소, 모듈, 라이브러리 또는 시스템 요구조건을 분석하였는가?
 - ⑭ 이러한 구성요소들 각각에 대한 위험요소를 줄이기 위한 체크리스트를 작성하였는가?

또한 공개소프트웨어를 도입하는 기관은 대상 소프트웨어에 대한 사용자의 평가를 직접 확인할 수 있다. 즉, 해당 프로젝트 메일링리스트 또는 포럼은 해당 공개소프트웨어가 보유하고 있는 기술의 적합성, 안전성, 품질의 우수성을 보다 객관적으로 검증할 수 있는 유용한 방법이다. 따라서 사용자들이 참가하고 있는 포럼과 메일링을 통하여 사용자들의 만족도에 대하여 평가할 수 있고, 큰 문제없이 오랜 시간 동안 해당 공개소프트웨어를 사용해오고 있는가를 평가할 수 있다.

마지막으로 도입하고자하는 공개소프트웨어의 위험요소를 완화하기 위하여 도입제품의 장기적인 생존가능성을 평가해야 한다.

공개소프트웨어를 개발하는 공개소프트웨어 프로젝트는 사용자 커뮤니티가 활성화되어 있을수록 안정적인 기능과 지속적인 업그레이드를 보장할 수 있다. 반대로 해당 공개소프트웨어 프로젝트가 소수의 사용자에게 의해 소극적으로 운용되고 있다면 외부 커뮤니티를 통한 기술지원이나 해당 공개소프트웨어의 장기적인 업그레이드는 원활하게 이루어지지 않을 가능성이 많다. 따라서 공개소프트웨어의 위험요소를 평가할 때에는 이러한 사항도 고려하여야 한다.

BLANK

5장. 법률문제

BLANK

1. 라이선스 유형

일반적으로 "라이선스(license)"란 "면허"또는 "허락"을 의미한다. 어떤 프로그래머가 특정 소프트웨어를 개발하게 되면, 저작권등 지적재산권이 프로그래머 또는 그가 속한 회사에 부여되는데, 이러한 지적재산권에 의해 소프트웨어에 대한 독점배타적인 권리를 가지게 된다. 그 결과 원칙적으로 권리자만이 소프트웨어를 사용, 복제, 배포, 수정할 수 있는데, 이들 권리자가 다른 사람에게 일정한 내용의 조건으로 하여 특정행위를 할 수 있는 권한을 부여하는 행위가 "라이선스"이다.

공개소프트웨어의 라이선스는 GPL을 시작으로 하여 많은 종류들이 있다. GPL, BSD 라이선스, MPL 라이선스등 환경과 필요에 의해서 만들어진 라이선스들이 있으며 이들 가운데 현재 주로 사용되고 있는 라이선스들과 국내의 실정에 맞는 라이선스들 위주로 살펴보면 다음과 같다.

가. GPL

GPL 라이선스는 GNU프로젝트에 가장 먼저 적용된 라이선스이며 리눅스에 적용되어 있고 또한 가장 널리 적용되고 가장 대표적인 공개소프트웨어 라이선스이다. GPL은 리차드스톨만(Richard Stallman)에 의해 만들어졌고 자유소프트웨어 재단(FSF : Free Software Foundation)의 철학을 반영하고 있다. GPL이 적용되어 있는 공개소프트웨어의 복제와 유통에는 제약이 없다. 하지만 GPL 라이선스가 적용되어 있는 소프트웨어는 다음과 같은 조건을 따라야 한다. 즉, 자유소프트웨어는 다음과 같은 조건하에서 소프트웨어의 복제와 개작, 배포가 자유롭게 허용되며, 프로그램의 사용(프로그램을 실행시키는 행위)에 대해서는 아무런 제한 없이 자유롭게 사용할 수 있다.

- ① 사용자가 소스코드를 쉽게 사용할 수 있어야 한다.
- ② 배포되는 소프트웨어에는 GNU GPL이 포함되어 있어야 한다. 배포된 소프트웨어를 사용하는 사람은 GPL상의 사용허가를 그대로 유지하는 조건하에 소스코드를 자유롭게 복제, 배포할 수 있다.
- ③ 쌍방향(interactive)프로그램의 경우, 프로그램이 시작될 때 이를 게시하여야 한다.
- ④ 프로그램을 수정할 경우에는 언제, 누구에 의해 수정되는지를 명시해야한다.
- ⑤ GPL 소프트웨어를 수정한 2차적 프로그램을 만들 수 있으며, 이렇게 수정한 프로그램에는 GPL이 적용되어야 한다. 즉, 소프트웨어를 양도받은 자는 소프트웨어를 자유롭게 개작할 수 있고, 개작된 소프트웨어는 GPL을 그대로 유지하는 조건에서 배포할 수 있다.
- ⑥ GPL소프트웨어를 이용하여 만든 소프트웨어에는 반드시 GPL이 적용되어야 한다.
- ⑦ 소프트웨어가 오브젝트 파일(object code)이나 실행파일 형태로 배포될 경우 반드시 소스코드를 함께 제공하거나, 요청을 할 경우 소스코드를 제공하겠다는 약속을 해야 한다.
- ⑧ GPL하에서 배포되는 소프트웨어는 상품의 적합성 등 어떠한 보증도 제공되지 않는다.

위의 ⑥번째 조건에 의해, 기업에서 GPL이 적용된 소프트웨어를 이용하여 개량된 소프트웨어를 개발하였을 경우에 기업은 그들이 개발한 소프트웨어의 소스코드를 공개해야만 한다. 그 결과 상용소프트웨어를 제작 판매하는 기업에서는 GPL에 대해 막연한 두려움을 가지게 되었으며, GPL이 적용된 소프트웨어를 기피하기도 하였다.

1984년 FSF를 설립한 이후 1989년에 FSF에 의해 GPL 1.0이 만들어 졌으며 1991년 FSF에 의해 GPL 2.0이 만들어지고 현재까지 사용하고 있다. GPL은 현실을 반영하면서 계속 진화하고 있으며, 조만간 특허권에 관한 규정, ASP 형태의 서비스 이용에 관한 내용 등을 담은 GPL 3.0 버전이 나올 예정이다.

GPL 라이선스에 있어 가장 핵심이 되는 부분은 소프트웨어를 소스코드의 형태로 복제, 수정, 배포가 가능한 자유가 보장되며 또한 수정된 소프트웨어에 대해서도 동일한 자유와 조건이 계속해서 순차적으로 보장되도록 하는 것이다.

GPL은 저작권을 전제로 하고 있지만 저작권의 본래의 취지를 반대로 이용하여 소프트웨어를 사적인 재산권의 대상으로 삼는 대신에 자유롭게 이용, 복제, 배포, 수정될 수 있는 수단으로 삼은 것이다.

즉, 일반적으로 프로그램의 개발자들이 ‘저작권’을 이용하여 재산적 권리를 취득하는 것과 마찬가지로 자유소프트웨어의 개발자들은 ‘저작권’을 이용하여 프로그램의 공유화를 가능하게 한 것이다. 그래서 ‘저작권(copyright)’을 기반으로 하면서도 이를 역이용하여 프로그램의 공유를 보장하려는 이러한 움직임을 ‘카피레프트(copyleft)’라고 부르게 된 것이다.

따라서 카피레프트의 조건에 따라 배포된 프로그램에 어떠한 수정이 이루어지거나, 여기에 다른 프로그램이 결합되더라도 그 결과물로서의 소프트웨어에는 카피레프트가 적용되는 것이다.

나. LGPL

초기 FSF가 만든 라이브러리들은 상용 라이브러리들보다 뛰어나지 못했었는데, 여기에 GPL이라는 엄격한 라이선스를 적용하게 되면 상용 기업들로 부터 호응을 얻지 못하기 때문에 자유 소프트웨어 재단(FSF)에서는 GPL의 엄격한 조항을 LGPL(Lesser General Public License)이라는 라이선스를 새롭게 만들었다. 결론적으로 FSF에서 LGPL을 만든 궁극적인 목적은 GNU 프로젝트에 의해 개발된 라이브러리와 사적 소프트웨어를 포함한 다른 소프트웨어와의 통합을 허용하기 위함이다. LGPL 라이선스를 만들게 된 궁극적인 목적은 자유 소프트웨어 하에서 개발되는 소프트웨어 제품들이 널리 많이 사용되어 표준이 되도록 하는 것과 독점소프트웨어 제품들과 경쟁을 할 수 있도록 하기 위함이었다. 이렇게 하여 LGPL이 적용된 최초의 소프트웨어가 GNU C라이브러리였다.

다. MPL

MPL은 넷스케이프(Netscape)사가 개발한 모질라(Mozilla) 브라우저 소스코드를 공개하는데 사용한 라이선스로서 "Mozilla Public License"의 약어이다. MPL 라이선스는 소스코드와 실행파일을 분리하여 이 둘을 보완하여 만든 것이다. 먼저 소스코드 측면에서는 소스코드는 반드시 공개되어야 하며 소스코드를 수정하였을 경우에는 최초의 저작자에게 수정한 내용을 통지해야 한다.

실행파일 측면에서는 실행파일 자체를 독점라이선스로 배포할 수 있다. 즉, 저작자의 이익을 보호할 뿐 아니라, 수정, 보완된 소프트웨어의 배포를 통한 상업적인 이익을 보호할 수 있으며 또한 적절한 가격을 요구할 수 있고, 불법복제에 대해 제재를 가할 수도 있다. 결국 이 소프트웨어를 더욱 보완, 발전시키려는 개발자들의 이익을 보호할 수 있게 된다. 즉, 기술적으로 개선을 할 경우, 코드를 보고 수정한 후, 컴파일 하여 새로운 독창적인 버전으로 재배포할 수 있다.

라. BSD License(MIT X License)

BSD(Berkeley software distribution) 라이선스는 소프트웨어 산업과 관련하여 가장 다양하게 사용될 수 있는 라이선스이다. BSD 라이선스가 적용되는 소프트웨어를 수정, 보완한 소프트웨어는 독점 소프트웨어가 될 수도 있고, BSD 라이선스로 배포될 수도 있다. 또한 GPL로 배포될 수도 있다. 즉, BSD 라이선스는 사용자들에게 거의 제한을 가하지 않는 것이 특징이며, BSD 라이선스와 비슷한 라이선스로 아파치(APACHE)웹서버에 적용되어 있는 아파치 라이선스가 있다.

또 BSD 라이선스에는 copyleft 조항도 없기 때문에 사적 소프트웨어 벤더들도 BSD 라이선스로 배포되는 OSS 컴포넌트를 그들의 제품에 무제한으로 사용할 수 있다. 예컨대 X 라이선스는 소프트웨어를 사용, 복제, 변경, 통합, 발행, 배포 및 판매할 권리를 부여한다. 다만 때때로 저작권 표기를 요구하거나, 코드 변경의 날짜 저자 및 변경 목적을 요구하기도 한다.

이상과 같이 공개소프트웨어에 관련된 주요 라이선스들에 대해서 살펴

보았다. 각 라이선스의 특징을 요약하여 비교하면 다음의 표와 같다.

구 분	GPL	LGPL	BSD	MPL
코드의 무료이용	○	○	○	○
코드의 자유배포	○	○	○	○
소스코드의 공개	○	○	○	○
소스코드의 수정	○	○	○	○
수정코드의 소스공개	○	○	X	○
상용소프트웨어와의 링크	X	○	○	○

2. 법적문제

공개소프트웨어의 인기가 높아짐에 따라서 소프트웨어 라이선스부여에 대한 관심이 높아지고 있다. 이러한 관심은 공개소프트웨어 라이선스가 전통적인 독점소프트웨어 라이선스와 매우 다르다는 사실로 인하여 부각되고 있는 것이다.

대부분의 사적소프트웨어 라이선스에는 공통적이고 비슷한 조건들이 포함되어 있으며, 또한 사적소프트웨어를 사용하는 사용자들은 그 라이선스의 조건들을 잘 이해하고 또한 따르고 있다. 즉, 사적소프트웨어 라이선스는 사용자가 해당 소프트웨어를 사용함에 있어 가능한 것과 가능하지 않은 일들에 대한 범위에 초점을 맞추고 있다. 예를 들어 소프트웨어를 설치할 수 있는 시스템의 개수, 동시에 작업 가능한 사용자의 수 등에 대한 조건들을 의미한다.

이와는 반대로 공개소프트웨어 라이선스는 사용자들이 소프트웨어를 어떻게 사용하는가는 중요하게 다루지 않는다. 즉, 공개소프트웨어의 재공급 및 소스코드에 대한 지속적인 접근성에 초점을 두고 있다.

따라서 사적소프트웨어는 공개소프트웨어에 대한 다른 위험요소를 가지고 있다. 결론적으로 사적소프트웨어나 공개소프트웨어를 도입하려는 기관들은 이러한 위험요소에 대한 이해와 관리가 마련되어 있어야 한다.

가. 라이선스의 위험성 평가

대부분의 공개소프트웨어는 앞에서 설명한 몇 가지 공개소프트웨어 관련 라이선스 부여 계획에 따라서 배포된다. 공개소프트웨어 관련 라이선스의 종류가 많다는 것은 기존의 라이선스들 중에 원하는 조건의 라이선스가 존재하지 않을 경우에는 개발자들 혹은 개발업체에서는 새로운 라이선스를 만들게 되는 것이다.

혹자들은 지나치게 많아져 가고 있는 공개소프트웨어 라이선스로 인하여 사용자들에게 많은 혼란과 위험요소를 가중시키고 있다고 불평한다. 공개소프트웨어 업계의 관계자들은 라이선스의 수적증가를 제한하려고는 하지만 요구조건에 부응하는 새로운 라이선스들이 OSI(Open Source Initiative)가 관리하고 있는 리스트에 추가되는 것을 막을 방법은 없다.

따라서 공개소프트웨어를 도입하고자 하는 기관들은 도입을 원하는 해당 공개소프트웨어의 라이선스의 종류가 그 내용을 명확하게 인지하고 있어야 한다.

가장 일반적인 공개소프트웨어 라이선스는 GNU GPL(GNU General Public License)이다. 이 라이선스는 최초의 개발자에 부여한 라이선스 요구조건을 다음 개발자들이 지속적으로 준수하도록 하고 있다. 다른 공개소프트웨어와 마찬가지로 GPL 라이선스가 부여된 소프트웨어는 GPL에 명시되어 있는 저작권의 보호를 받게 된다.

GPL은 공개된 소스코드의 지속적인 가용성을 보장받기 위한 방법으로 저작권 보호를 활용하고 있다. 따라서 GPL 라이선스를 부여받은 제품을 다운로드하여 수정하고 수정한 제품을 재공급하려는 계획을 가진 기관이라면 GPL의 이러한 법적관계에 대하여 파악하고 있어야 한다.

최근에는 이러한 법적분쟁이 다수 있었으며 GPL 라이선스가 준수될 경우에만 소프트웨어의 무료 재배포를 허용하기 위한 법적인 권한을 강화하고 있는 추세이다. 만약 라이선스의 조건을 준수하지 않는 경우에는 GPL 라이선스를 부여받은 코드의 사용에 대한 권리는 모두 철회할 수 있다. 따라서 공개소프트웨어를 도입하는 기관의 이러한 부주의로 인하여 몇년간 지속적으로 도입하여 사용하고 있던 공개소프트웨어를 모두 철회해야하는 불행한 사태를 초래해서는 안될 것이다.

공개소프트웨어를 도입할 때 뿐 만아니라 GPL 라이선스조건에 배포된 소스코드를 수정한 소프트웨어를 제품으로써 공급하고자 할 때에도 GPL의 재배포 조건을 완전히 준수해야만 한다.

이와 같이 공개소프트웨어를 도입하여 적용함에 있어 법적인 타당

성 조사와 라이선스에 대한 정확한 법적 허용 등에 대하여 인식하는 것은 매우 중요한 일이다.

따라서 이번 장에서는 GPL의 해석과 적용범위, 계약의 성립여부 및 유효성, 저작권과 계약의 관계, GPL에 위반되었을 경우의 법적인 재판문제, 저작인격권과 저작권문제, 특허와 상표, 그리고 보증의 책임과 그 범위, 그리고 소스코드의 공개 방법 등에 대해서 자세히 알아볼 것이다.

나. GPL의 해석과 적용범위

그렇다면 GPL하에서 배포되는 공개소프트웨어를 도입하려는 기관들은 실제로 어떤 부분들을 확인하고 주의해야 하는가? GPL에는 다른 라이선스와는 달리, GPL 대상 코드를 다른 코드와 조합시켜 1개의 프로그램으로 한 경우, 프로그램 전체가 하나의 GPL의 대상이 되며, 복제·개작·배포를 허락하며, 소스코드를 공개하여야 한다. 따라서 이러한 조건 때문에 Linux를 이용하는 기업은 다음과 같은 것을 주의해야 한다.

① GPL 대상 코드를 포함하고 있는, 1개의 프로그램(work)으로 볼 수 있는 범위가 GPL의 적용범위가 된다.

② 1개의 프로그램으로 볼 수 있는지의 여부는 모듈간의 링크형태 등의 외형적 기준에 의하는 것이 아니라, 모듈간의 호출관계, 데이터구조의 참조 정도를 고려한 모듈간의 결합 정밀도에 의하여 판단한다. 즉, 모듈내부에서 다른 모듈을 호출하고, 다른 모듈의 구조에 대한 참조가 있는 경우에 이러한 것은 동일 프로그램의 일부로 간주되고, 이

러한 것이 아닌 경우는 별도의 프로그램으로 간주되는 경우가 많다.

다만, FSF(Free Software Foundation)는 GPL 대상 프로그램의 악용에 대비하여 명확한 기준을 제시하지 아니하고, 최종적으로는 법관이 판단하도록 하고 있으며 법원으로 가게 되는 대부분의 경우는 해당 소프트웨어가 GPL의 대상이 되느냐 되지 않느냐에 대한 것이 명확하지 못하기 때문이다. 따라서 GPL의 법적인 문제의 가장 근본적인 핵심은 실제로 해당 소프트웨어가 GPL의 대상이 되느냐 되지 않느냐를 판단하는 것으로 실제 분쟁사례를 분석하여 GPL의 대상 여부를 정리해 보면 다음과 같다.

- ① GPL 대상 프로그램과 정적 링크된 프로그램은 GPL의 대상이 된다.
- ② GPL 대상 프로그램과 동적 링크된 프로그램은 일반적으로 GPL의 대상이 된다. 다만, “main”함수를 호출할 뿐이고, 상호 데이터 구조를 참조하지 않는 경우에는 별개의 프로그램이고, GPL의 대상이 되지 않는다.
- ③ pipe 및 socket, command line 인수만으로 통신하여 결합된 프로그램은 별개의 프로그램이다. 다만, 복잡한 데이터 구조를 참조하는 등 밀접한 관계가 있는 경우를 제외한다. 표준 장치인터페이스(device interface)를 이용하여, 커널의 데이터 구조를 참조하지 않은 장치드라이버 등은 GPL의 대상이 아니다.
- ④ Linux 커널(Linux kernel)의 dynamic loadable module은 일반적으로 커널의 일부로 취급되며 GPL 대상이 된다.

⑤ Linux어플리케이션은 Linux 커널과는 별개의 프로그램이기 때문에 GPL이 될 수도 있고 안 될 수도 있다. 즉, 해당 어플리케이션의 라이선스 조건에 따른다고 하는 것이 정확하다.

다. 계약의 성립여부 및 유효성

GPL 뿐만 아니라, 다른 모든 공개소프트웨어의 라이선스 계약에서는 해당 라이선스 자체가 언제 어떻게 계약에 동의되는가에 대한 명확한 표기가 없기 때문에 계약이 언제 성립되었는가에 대한 분쟁이 발생할 수 있다.

이와 같은 경우 소프트웨어에 계약서가 첨부되어 있는 것만으로는 계약의 유효성에 대한 의문이 발생할 수 있지만, 흔히 말하는 쉬링크랩 계약(포장을 뜻하는 행위가 계약의 동의 의사로 보는 계약)의 경우는 계약동의를 의사가 인정되고 계약으로서 유효한 것으로 보고 있다는 점을 명확히 알아야 한다.

어쨌든, GPL 대상 소프트웨어 이용자는 계약이 유효하게 성립한 것을 전제로 하고 행동하여야 한다.

라. 저작권과 계약의 관계

GPL 계약이 성립하는 것을 전제로 한 경우, GPL 프로그램의 저작권자에게는 저작권에 기초한 청구와 계약에 기초한 청구의 2개의 청구권이 부여되기 때문에, 그 관계가 문제된다. GPL에 대한 계약위반의 효과로서는 GPL 프로그램의 사용허락이 거부된다고 하는 규정은

없다. 따라서 저작권자가 사용자에게 대하여 GPL 위반을 묻는 경우에는 계약위반에 기초한 청구가 아니라, 저작권 침해로 이유로 청구한 것이 된다.

마. 특허

1) GPL의 제삼자의 특허권에 대한 침해 문제

GPL의 프로그램이 제삼자의 특허를 침해하여, 사용금지 및 손해배상을 청구당하는 것은 아닌가라는 것은 공개소프트웨어에 관한 최대문제의 하나이다. GPL은 특허권과 저촉하는 배포를 할 수 없다고 경고할 뿐이고, 현실적으로는 특허권자에게 대항할 수단은 없다고 보아야 한다.

2) GPL 프로그램의 특허권

특허를 취득한 자가 그 특허에 관한 코드를 GPL 프로그램에 편입하여 재배포하는 경우, GPL은 반드시 그 특허의 재라이선스를 포함하여 재배포를 의무화 하고 있다고 해석할 수 있는지는 의문이고, 이것에 기초하여 특허분쟁이 후일 발생할 가능성이 남아 있다.

바. 상표

GPL 프로그램의 명칭이 제삼자에 의하여 상표 등록된 사건들이 발생하고 있다. 프로그램명이 상표 등록된 경우, 그 프로그램의 사용이 방해받지는 않는지가 문제된다.

검토 결과, 상표권의 존재에 의하여 GPL 프로그램의 사용을 할 수 없는 것은 아닌 것으로 판단되지만 무용한 불이익을 피하기 위하여 공개소프트웨어의 개발프로젝트의 중심적인 개발자가 그 프로그램을 상표 등록할 필요가 있다.

사. 보증(Warranty) 책임의 범위

GPL은 면책규정을 가지고 있는데 제조물책임법, 민법의 하자담보 책임규정과의 관계가 문제된다. 일반 민법원칙에 따르면 유상계약의 경우 면책조항을 무효로 하고, 민법원칙에 따라 하자담보책임 및 손해배상책임을 부담하게 된다. GPL 프로그램을 편입시킨 제품 또는 서비스에 대하여 대가를 요구하는 경우가 대상이 된다.

한편, 제조물책임법은 GPL 프로그램 편입 제품 등이 문제가 된다. GPL 프로그램의 부적합에 의하여 제3자에게 손해가 발생한 경우에는 개발자의 책임유무가 문제가 된다. 제3자는 GPL 프로그램을 편입한 제품을 구입만 한 경우에는 GPL 면책규정의 제약을 받는 것이 아니라, GPL 프로그램의 개발자에 대한 불법책임에 기초한 손해배상청구는 가능한 것으로 판단된다.

배포자가 개발자에 대하여 구상을 청구하는 것은 GPL 면책규정에 의하여 방해되는 것은 아닌 것으로 해석된다. 또한 GPL 프로그램을 이용하여 제품 등을 판매하거나 유상서비스를 제공한 경우에는 프로그램의 결함, 하자 등을 충분히 조사, 연구하여 자기의 책임으로 판매·제공할 것이 요청된다.

아. GPL로의 소스코드의 공개방법

그렇다면 소스코드의 공개에 대하여 어떠한 방법으로 공개를 하면, GPL에 합치하는지가 문제가 될 것이다. 그리고 배포비용은 누가 부담하는 지도 문제가 된다. 이 점에 대하여 GPL은 제3조에서

- a) 완전한 소스코드를 제공할 것,
- b) 완전한 원시 코드를 배포한다는 취지의 약정서를 함께 제공할 것,
- c) 소스코드의 배포에 관하여 입수한 정보를 제공할 것등

여러 방법에 의한 공개를 요구하고 있다. 따라서 b)를 선택하는 한, 제3자의 요구에 대하여, 그 우송료 등의 실비부담으로 배포하면 문제가 되지 않을 것이다.

3. 사용유형별 법적문제 검토

이번 장에서 우리는 공개소프트웨어에 적용되는 라이선스들의 종류들을 살펴보고 라이선스 종류에 따른 법률문제들도 살펴보았다. 공개소프트웨어를 도입하고자하는 기관들에서는 혹 발생할지 모르는 법적인 문제들에 있어서 매우 민감할 수밖에 없을 것이다. 힘들게 도입한 공개소프트웨어를 자칫하면 모두 거둬야하는 상황이 발생할 수도 있으며 더욱이 법적인 책임을 져야하는 상황까지 초래할 수도 있다는 점을 인지한다면 공개소프트웨어의 법률문제는 결코 가볍게 다루어져서는 안 될 것임을 쉽게 알 수 있다.

하지만 법률문제라고 해서 어렵게 생각할 문제는 결코 아니다. 거의 대부분 공개소프트웨어의 법률문제는 공개소프트웨어의 소스를 수정하여 재배포하는 경우에 발생하는 것이므로 소스수정 후에 재배포

를 하지 않는다면 크게 고려할 바는 아니다. 하지만 기관에서 도입한 공개소프트웨어를 다른 업무나 타기관에도 도입하기 위해서는 수정과 수정된 소스를 다시 재배포하는 것이 요구되므로 이런 경우에는 재배포에 따르는 법률문제를 반드시 검토해 보아야 한다.

결론적으로 이번 절에서는 기관에서 도입하는 공개소프트웨어의 법률문제들을 공개소프트웨어를 수정하여 재배포하는 경우의 수를 각각 배포 시나리오별로 나누어서 하나씩 살펴보도록 하자.

공개소프트웨어든 독점 공급되는 사적소프트웨어든 모든 소프트웨어의 라이선스 부여는 매우 복잡한 법적인 범주를 고려해야 한다. 그러므로 라이선스 계약을 체결하기 이전에 적절한 법적인 자문을 구할 필요가 있다.

① 공개소프트웨어를 수정하지 않고 기관 내부에서만 사용하는 경우

이 경우에는 법적문제가 발생할 소지가 거의 없는 경우로 볼 수 있다. 따라서 거의 모든 공개소프트웨어 라이선스가 적용가능하다. 예를 들어 GPL, LGPL, BSD, Mozilla Public License, MIT License등을 적용하면 된다.

② 공개소프트웨어를 도입 후 이를 수정하여 기관내부에서만 사용하는 경우

도입한 공개소프트웨어를 목적과 요구조건에 부합하도록 수정 및 확장하여 내부에서만 사용한다. 즉, 수정 후 내부에서만 사용하는 경

우를 의미한다.

이 경우에도 거의 모든 공개소프트웨어 라이선스를 적용할 수 있다. 도입한 기관은 원래 근거하여 배포된 공개소프트웨어 라이선스의 종류에 관계없이 진행할 수 있다. 즉, GPL, LGPL, BSD, Mozilla Public License, MIT License 등의 라이선스를 적용하면 된다.

③ 공개소프트웨어를 수정하여 도입하고, 다른 기관에 재공급하는 경우

한 기관에서 도입한 공개소프트웨어를 특정 목적과 요구조건에 맞게 수정 및 확장한다. 그리고 이렇게 수정된 공개소프트웨어를 다른 기관에 재공급하려고 한다.

이 경우에는 도입한 공개소프트웨어의 라이선스가 허용하는 범위 내에서 거의 대부분의 공개소프트웨어 라이선스를 적용할 수 있다. 예를 들어 한 기관에서 소스코드를 수정하여 이를 중앙정부에 공급하는 경우에 적용 가능한 라이선스는 GPL, LGPL, BSD, Mozilla Public License, MIT License 등이다. 단, 이 경우에는 원래 도입 초기에 원래의 라이선스에 따라 수정된 코드를 공개해야 한다는 조건이 있다. 공개의 범위는 GPL, LGPL, MPL 등 라이선스에 따라 조금씩 차이가 있다.

④ 공개소프트웨어를 도입하여 수정한 후, 소스코드를 공개하지 않고 실행파일 형태로만 배포하려는 경우

한 기관이 공개소프트웨어를 획득하여 특정목적과 요구조건에 맞도록 수정 및 확장한다. 그리고 이렇게 수정된 공개소프트웨어를 클로즈드 소스(Closed Source) 버전으로 공급하려고 한다.

이 경우에 적용 가능한 라이선스는 지금까지의 경우와 사뭇 다르다. 왜냐하면 GPL에서는 소스코드를 공개하지 않는 것을 허용하지 않기 때문이다. 만약 클로즈드 소스형식의 수정된 공개소프트웨어 제품의 재공급을 가능하게 하려면 클로즈 해제(Closing Off)과정을 허용하는 라이선스와 함께 공급하도록 하는 것이 좋다. 따라서 이 경우에 적용 가능한 라이선스는 BSD(Berkeley Software Distribution)가 클로즈 해제를 허용하는 라이선스로서 가장 알맞으며 MIT License도 적용이 가능하다.

⑤ 기관이 개발한 클로즈드 소스 소프트웨어를 공개소프트웨어와 패키징하여 제공하려는 경우

이 기관은 오픈소스 모듈, 컴포넌트, 라이브러리 또는 어플리케이션과 함께 소프트웨어를 패키지로 제공하려고 계획하고 있다.

이 경우 적용 가능한 라이선스는 클로즈드 소스제품이 이러한 다른 공개소프트웨어에 직접 연계되어 있지 않는 경우라면 이 기관은 이러한 번들 제품을 제약 없이 재공급할 수 있다. 그러나, 법적인 위험요소를 완화하기 위하여 번들로 구성된 각 기술의 라이선스 조건을 숙독하여 신중하게 적용해야 한다.

⑥ 기관이 개발한 소프트웨어에 오픈소스 모듈, 컴포넌트 또는 라이브러리가 링크되어 있는 경우

한 기관이 개발한 소프트웨어 제품을 재공급하려고 한다. 이때 재공급하는 소프트웨어가 오픈소스든 클로즈드 소스든 상관없이 재공급하려는 소프트웨어는 오픈소스 모듈, 컴포넌트 또는 라이브러리에 링크되어 있다.

이 경우 법적인 문제가 가장 예민하게 발생할 수 있는 예로서 기관은 소프트웨어가 링크되는 각 제품의 라이선스를 파악하고 이를 반드시 준수해야만 한다. 만약 링크된 제품이 GPL 라이선스를 부여하는 경우에는 소스코드는 반드시 공개되어 사용자에게 제공되어야만 한다. 그리고 링크된 제품이 LGPL 또는 BSD 라이선스라면 소스코드는 제공될 필요가 없게 된다. 즉, 사용자에게 공개소프트웨어의 소스코드를 제공하지 않아도 되는 클로즈드 소스 버전 제품을 작성하는 것을 허용하게 된다.

참 고 문 헌

- 국내 문헌

김성근 외, 공개SW 불공정경쟁 실태조사 연구, 한국소프트웨어진흥원, 2003

김영문 외, 공개소프트웨어 도입가이드라인 연구, 한국소프트웨어진흥원, 2003

김영문 외, 공개소프트웨어 라이선스 연구, 한국소프트웨어진흥원, 2002

김영문 외, 공개소프트웨어활성화를 위한 법제도 개선방안 연구, 한국 소프트웨어진흥원, 2003

김일환 외, 공개소프트웨어 비용구조에 관한연구, 한국소프트웨어진흥원, 2003

김정국 외, 공개SW 중장기 기반기술 기획 연구, 한국소프트웨어진흥원, 2003

정보통신기술협회, 웹콘텐츠 접근성 지침, 2003

정부혁신지방분권위원회, 전자정부사업 공개소프트웨어 도입 권고안, 2004

정태명 외, 효율적 전자정부 구현을 위한 기반기술 도입 정책연구, 한국소프트웨어진흥원, 2003

최성모 외, 공공기관을 위한 리눅스 도입 방안 연구, 한국전산원, 2000
한국소프트웨어진흥원, 공개소프트웨어 도입 성공사례집, 2004, 2005

한국소프트웨어진흥원, 크로스브라우징 가이드, 2003

행정자치부, 정보통신부, 공개소프트웨어 기반 정보시스템 구축 사용자 가이드, 2005

- 외국 문헌

Al Gillen, Dan Kusnetzky and Scott McLarnon, The Role of Linux in Reducing the Cost of Enterprise Computing, IDC White Paper, 2002

-
- Brad Day, Laura Koretzle, Linux Crosses Into Mission-Critical Apps., Gartner, 2004
- CIPR, Integrating Intellectual Property Rights, Commission on Intellectual Property Rights, 2002. 9.
- European Commission's initiative eEurope, An Information Society for all Action Plan, 2000.
- Free Software Foundation, Categories of Free and Non-Free-Software, 1999
- Free Software Foundation, What is Free Software?, <http://www.gnu.org/free-sw/html>
- Gomulkiewicz, R.W. , How Copyleft Uses License Rights to Succeed in the Open Source Software Revolution and Implications for Article 2B, 36 Houston Law Review 179.
- Hannemann, The Patentability of Computer Software, Kluwer Law and Taxation Publishers, Deventer(NL), 1985.
- Jane M. Rolling, Commercial Law Journal Vol. 104 p. 213-221 , 1999.
- Jean Braucher, Why UCITA, like UCC Art. 2B is premature and unsound, UCC Bulletin July 1999,
- John Pescatore, "Nimda Worm shows you can't always patch fast enough", Gartner, 2001.
- John Pescatore, Open Source Security:SYMPOSIUM ITXPO 2004, Gartner, 2004
- Jürgen Taeger, Softwareschutz durch Geheimnisschutz, Arbeitsbericht Nr. 102, Lüneberg, September 1991.
- Kennedy, D.M., A Primer on Open source Licensing Legal Issues: Copyright, Copyleft, Copyfuture, <http://www.denniskennedy.com>
- Koordinierungs- und Beratungsstelle der Bundesregierung für Informationstechnik (KBSt) <www.kbst.bund.de>.

- Larry Smart, Co-Chairs, "Recommendations of the Panel on Open Source Software for High End Computing, 2000.
- Mark A. Lemley, 569 S.C.L. Rev. 1239, 1995.
- Mark A. Lemley, Intellectual Property and Shrinkwrap-License, 68 S.C.L.Rev. 1239, 1995.
- Metzger, A/Jaeger, T., Open Source Software und deutsches Urheberrecht, GRUR Int. 1999.
- Mikro- und makroökonomische Implikationen der Patentierbarkeit von Softwareinnovationen: Geistige Eigentumsrechte in der Informationstechnologie im Spannungsfeld von Wettbewerb und Innovation, Karlsruhe September 2001.
- Neukom & Gomulkiewicz, Licensing Rights to Computer Software, in TECHNOLOGY LICENSING AND LITIGATION 1993.
- NHS Information Authority, Open Source software and The NHS: WHITE PAPER, , 2002. 1,
- Nic Peeling, Julian Satchell, Analysys of the Impact of Open Source Software, QinetiQ, 2001.
- Niva Elkin-Koren, 12 BERKLEY TECH. L. J. 93, 1997.
- Niva Elkin-Koren, Copyright Policy and Limits of Freedom of Contract, 12 BERKLEY TECH. L. J. 93, 1997.
- Office of Government Commerce, Guidance on implementing Open Source Software, 2002.9.
- Office of e-Envoy(OeE), Open Source Software : use within UK government, Version 1, 2002.
- Open-Source Fight Flares At Pentagon, Washington Post 2002. 5. 23.
- P. Samuelson, R. Davis, K.d. Kaor & J. H. Reichman, A Manifesto concerning the Legal Protection of Computer Program, 94 Colombia Law Review 2308.
- Philipp Koehler, Der Erschöpfungsgrundsatz des Urheberrechts im Online-Bereich, Verlag C.H. Beck, 2000.

-
- Pooling Open Source Software (POSS) Feasibility Study, 2002.
- Richard Stallman, Why "Free Software" is better than "Open Source",
- Richard Stallman, Why "Free Software" is better than "Open Source", <<http://www.gnu.org/philosophy/free-software-for-freedom.html>>
- Robert W. Gomulkiewicz, "How Copyleft Uses License Rights to Succeed in the Open Source Software Revolution and the Implications for Article 2B", 36 Hous. L. Rev 179 (Spring 1999)
- Shawn W. Potter, "Opening Up to Open Source", 6 Rich. J.L. & Tech. 24 (Spring 2000)
- Software Licensing Committee of the American Bar Association's Intellectual Property Section, An Overview of "Open Source" Software Licenses,
- Study into the use of open source software in the public sector, 2001.
- Total Cost of Ownership for Linux in the Enterprise, Robert Francis Group, 2002
- Trompenaars, W.M.B., Legal Support for Online Contracts, in Hugenholtz, B., Copyright and Electronic Commerce. Legal Aspects of Electronic Copyright Management, Kluwer Law International 2000.
- Australian Government, A Guide to Open Source Software, 2005
- David Wheeler, Why OSS Look At The Numbers, 2003

- 기타 웹 페이지

<<http://naver.com/>>

<<http://user.oss.or.kr/>>

<<http://www.abanet.org/intelprop/opensource.html>>
<http://www.consumerlaw.org/ucita/twelve_problems.html.>
<<http://www.debian.org/intro/free/>>
<<http://www.iabf.or.kr/>>
<<http://www.oreilly.com/catalog/cathbaz/>>
<<http://cip.umd.edu/osagenda.htm>>
<<http://eu.conecta.it>>
<<http://europa.eu.int>>
<<http://fsub.schule.de/>.>
<<http://help.oss.or.kr/>>
<<http://linux.kbst.bund.de/linuxtag2002/kuester.html>>
<<http://opensource.mit.edu>>
<<http://opensource.ucc.ie>>
<<http://oss.mri.co.jp>>
<http://www.bmi.bund.de/dokumente/Pressemitteilung/ix_82618.htm>
<<http://www.bundestag.de>>
<<http://www.bundestux.de/>>
<http://www.cordis.lu/ist/ka4/tesss/impl_free.htm>
<<http://www.cybersource.com/home.html>>
<<http://www.freestandards.org/>>
<<http://www.gnu.org>>
<<http://www.gnupg.org/>.>
<<http://www.gnupg.org/aegypten/index.de.html>>
<<http://www.ibiblio.org/>.>
<<http://www.idei.asso.fr/ossconf.html>>
<<http://www.infonomics.nl/FLOSS/report/index.htm>>
<<http://www.inria.fr/inria/enbref.en.html>>
<<http://www.itrd.gov/pubs/pitac/pitac-tl-9feb01/>>
<<http://www.kbst.bund.de>>
<<http://www.lernnetz-sh.de/>.>

<<http://www.lernnetz-sh.de/kmLinux/>>

<[http://www.linux-magazin.de/Artikel/ausgabe/2002/09/behoerde/beh
oerde.html](http://www.linux-magazin.de/Artikel/ausgabe/2002/09/behoerde/beh
oerde.html)>

<http://www.nhsia.nhs.uk/def/pages/features/i_250202.asp>

<<http://www.opensource.org>>

<<http://www.orh.bayern.de/Jahresbericht2001.pdf>>

<[http://www.sicherheit-im-internet.de/download/Kurzgutachten-Soft
ware-patente.pdf](http://www.sicherheit-im-internet.de/download/Kurzgutachten-Soft
ware-patente.pdf)>

<<http://www.softwarechoice.org/>>

<<http://zdnet.co.kr/>>

부록1 : 신뢰성이 높은 공개소프트웨어

분야	공개 소프트웨어	추천이유
Web Server	Apache	Apache는 다른 모든 상용 Web Server를 제치고 최고의 시장 점유율을 확보하고 있으며, 성능이나 안정성 등도 상용 Web Server보다 우월하다는 평가를 받고 있어서 자유 소프트웨어 최대의 걸작 몇 개 가운데 하나로 꼽힌다. 현재로서는 큰 적수가 없는 상황으로, 사용자 및 개발자가 많기 때문에 문제에 대한 대처 속도가 빠르고, 도움을 받기 쉽다는 장점이 있다. 유명한 프로그램인 만큼 한국에도 사용자 그룹이 있어서 한글로 도움을 얻기 쉽다.
FTP Server	Proftpd	전통적으로 Unix 계통의 운영체제에서 가장 많이 사용하던 Ftp Server는 Washington University에서 만든 wu-FTPd였지만 wu-FTPd는 역사적으로 많은 보안 문제를 갖고 있으며, 또한 제공하는 기능이 상당히 제약적이며, 설정에 어려움을 겪는 관리자들도 많이 있었다. 이런 모든 문제점을 해결하기 위해 시작된 것이 Proftpd 프로젝트였고, 그 결과 Proftpd는 보안적인 문제점도 거의 없고 (또는 발견된 것이 별로 없고) 상당히 많은 기능을 제공할 뿐 아니라 설정도 Apache와 거의 동일한 방식을 따르기 때문에 매우 쉽고, 기본으로 제공되는 설정을 수정 없이 그대로 사용하더라도 정상적으로 작동한다. 또한 한국에 사용자 그룹이 있어서 한글로 된 도움말을 볼 수가 있으며 많은 도움을 쉽게 얻을 수 있다.
Web Scripting Language	PHP	HTML만으로 Home Page를 만들기에는 기능이 너무 없다. 그래서 만들어진 것이 CGI (Common Gateway Interface) 인데, C로 만들어지는 대부분의 CGI는 (프로그래머의 사소한 실수로) 보안에 많은 문제점을 갖고 있는 경우가 많으며, Perl과 C 모두 유지 보수가 어렵다는 문제점을 갖고 있다. 그래서 나온 것이 Web Scripting 언어인 PHP인데, C와 비슷한 문법으로 배우기가 매우 쉽고 (기존의 C, C++, Java 프로그래머라면 배우지 않고 함수 reference만으로도 사용할 수가 있다) 매우 쉽게 되어 있어서 작성자의 실수를 최대한 줄여주며, Apache의 모듈로 작동하기 때문에 사용자가 폭주하더라도 host에 부하를 최대한 적게 주는 장점이 있다. 비슷하게 ASP와 JSP가 있지만, PHP는 간단한 Module 설치만으로 윈도우즈를 비롯한 어느 곳에서나 돌아가며, 수많은 종류의 DB와 연동을 쉽게 하기 위해 자체 함수를 제공하는 등, 작성자의 편의를 최대한으로 도와주고 있다.

분야	공개 소프트웨어	추천이유
Database Engine	MySQL	<p>많은 종류의 Open Source DB가 있지만, 그 가운데서도 가장 많이 사용되는 것이 MySQL이다. 모든 DB 프로젝트 가운데서 가장 활발하게 움직이고 있기 때문에 가장 빠른 속도로 발전하고 있으며, 성능 또한 그에 맞춰 지속적으로 올라가고 있다. 대부분의 사용자에게서 DB를 쓰는 일은 Home Page를 제작할 때인데, 이를 위해서 Apache + PHP + MySQL (APM)은 환상의 Teamwork을 이루며, 거의 대부분이 이런 방식을 사용하고 있고, 한글 지원과 도움을 쉽게 받을 수 있다. APM에 대한 도움을 무료로 제공하는 곳은 도처에 깔려 있는 상황이다.</p>
Mail Server	Sendmail	<p>전자우편은 Web Service와 더불어 Internet에서 제공하는 가장 중요한 Service로 손꼽히는데, Sendmail은 이미 전세계 표준으로 자리를 잡은 상태다. Sendmail은 엄청나게 많은 기능을 제공하는데, 그 많은 기능 때문에 보안 허점이 발견되기도 많이 했지만, 엄청나게 많은 개발자와 셀 수조차 없는 사용자들 덕분에 거의 대부분의 문제점들은 발견되기가 무섭게 보완되어 나온다. 현재 QMail이 보안을 기치로 Sendmail을 따라오고 있지만, 아직까지는 Mail Server에 Mailing List 관리 프로그램과 같은 Add-on되는 모든 프로그램들이 Sendmail을 기준으로 만들어지고 있다.</p>
Proxy Server (HTTP)	Tinyproxy	<p>Open Source Proxy Server로 가장 많이 쓰이는 것은 SQUID지만, SQUID는 가정에서 개인이 사용하기에는 약간은 덩치가 큰 느낌을 받고, 너무 많은 기능을 제공한다. ADSL을 사용하는 많은 이들이 집에서 Linux로 인터넷을 공유하기도 하는데, 이럴 때에 매우 작지만, 필요한 기능은 다 지원하는 Proxy Server로 Tinyproxy가 제격이다. Tinyproxy는 설정조차 필요 없어서 매우 쉽게 사용할 수가 있으며, 현재 나와있는 모든 Proxy Server 가운데 가장 적은 System Resource를 사용한다고 한다.</p>
Web Client (Browser)	Mozilla	<p>Marc Andreessen이 Mosaic을 만들어 Internet 혁명을 일으킨 뒤, 이는 Netscape으로 발전하였다. 그 뒤 MS의 불공정 행위로 시장의 선두 자리를 MSIE에 빼앗겼지만 Netscape이 Open Source화 되면서 Mozilla 프로젝트가 시작되었다. 또한 Browser의 핵심인 Rendering Engine을 완전 새롭게 만들었는데, 그 것이 바로 유명한 Gecko Engine이다. 이 Gecko Engine 덕분에 Mozilla에서도 MSIE tag를 사용한 page가 제대로 보이며, 속도 역시 매우 빨라져 MSIE보다 더 빠른 속도를 보인다. Mozilla 역시 매우 빠르게 성장하는 프로젝트 가운데 하나이며, 별 지원이 필요 없을 정도로 잘 만들어져 있으며, 아주 많은 Platform을 지원하고 있고 Unix 계열에서 작동하는 대표적인 Web Browser다.</p>

분야	공개 소프트웨어	추천이유
FTP Client	gFTP	FTP는 File Transfer Protocol로 파일 전송을 위해 만들어진 소프트웨어이다. Web만을 Internet으로 알고 있는 경우가 많은데, FTP 역시 매우 중요하고, 상당히 편리한 것이다. gFTP는 Console과 GUI 환경을 모두 지원하며, FTP 뿐만 아니라 HTTP와 SSH 프로토콜을 지원한다. 마우스 끌어서 놓기로 작동시킬 수 있는 매우 직관적인 UI (User Interface)를 지원하며, Multithread로 다중 파일 동시에 받기와 FTP와 HTTP proxy에서도 정상적으로 작동한다. 또한 메뉴와 도움말이 한글로도 제공된다.
Mail Client	Ximian Evolution	Ximian Evolution은 GNOME에서 사용되도록 만들어진 것이지만, GTK Library만 설치되어 있으면 GNOME이든 KDE든 어디에서나 사용할 수 있으며, Mozilla에서 제공되는 전자우편 Client와 다르게 아주 많은 기능들이 통합되어 있다. MS Outlook과 같이 일정 관리 기능에서부터 전자우편 기능까지 다 들어 있어서 Mail Client라기 보다는 PIMS라고 하는 것이 알맞을 것이다.
Downloading Client	Downloading Client	Internet에서 download를 해야 할 일이 많으나 Web Browser에서 click해서 몇 시간이나 걸리는 download를 한다면, 화면에 덩그러니 놓여있는 Web Browser가 눈에 거슬릴 뿐만 아니라 Web Browser가 엄청난 System Resource를 차지하고 있어서 computer 속도도 느려진다. wget은 HTTP와 FTP를 지원하며, 하위 디렉토리까지 모두 받아오기 및 웹사이트 통째로 받아오는 기능까지 갖고 있다. 또한 link를 따라가면서 받아오는 기능까지 갖고 있어서 인터넷에서 다량의 다운로드를 할 경우 최적의 프로그램이다. console용 프로그램이어서 System Resource도 적게 차지한다.
Archiving Software	Gnozip	DOS/Windows에서는 Archiving과 압축이 늘 동시에 이루어 지지만, Unix 계열에서는 Archiving은 tar로, 압축은 Compress나 gzip을 비롯한 여러 가지로 이루어진다. 이 때문에 이를 매우 불편하게 여기는 사람들도 많은데 Gnozip은 여러 Archiving과 압축 Utility들에 대한 Front-End로 Graphical하고 직관적인 Interface를 제공하며, 두 작업을 동시에 해 준다. 또한 매우 많은 종류의 압축 파일을 다룰 수 있다.
CD Burning Software	X CD Roast	CD 제작 도구로 Unix 계열에서는 mkisofs와 cdrecord라는 매우 훌륭한 명령형 도구가 제공되지만, 많은 사람들이 Console이라면 겁부터 먹는 경향이 있다. X CD Roast는 말그대로 X에서 움직이며(Graphic한 사용자 Interface 제공) 또한 많은 CD 제작 프로그램이 Root의 권한을 요구하는데, 이 프로그램은 어느 사용자라 하더라도 CD를 제작할 수 있도록 해 준다. 마우스로 끌어서 놓기 방식의 작동 방법을 지원하는 매우 편리한 프로그램이다.

분야	공개 소프트웨어	추천이유
PDF Viewer	Xpdf	PD (Portable Document Format)는 PS (Post Script)를 Adobe사가 개조해서 만든 것으로 별도의 프로그램이 있어야 볼 수 있다. Xpdf는 PDF를 X Window에서 볼 수 있도록 해주는 Viewer 프로그램이다.
Office Suite	Open Office	Open Office는 원래 독일의 Star Division사에서 만든 Multi-platform Office Suite로 Sun에서 이 회사를 인수하여 Open Source화 하였다. Open Office에 몇 가지 글꼴과 유틸리티를 포함하여 Sun에서 Star Office라는 이름으로 상용 판매하고 있을 정도로 기능과 안정성 등이 입증되었으며, 또한 MS Office와 파일이 거의 완벽하게 호환됨
.IRC Client	Xchat	RC는 Internet Relay Chatting으로 이를 이용하면 전 세계의 누구와도 대화할 수가 있게 된다. Xchat은 X Window에서 작동하는 프로그램으로 Console용인 bitchX와 더불어 가장 유명하며, 가장 많은 사람들이 사용한다.
Remote Login	OpenSSH	Unix 계열에서는 Computer의 원격 조작을 위해서 rlogin, rsh, telnet 등의 프로그램을 전통적으로 제공해 왔다. 하지만, 이는 암호화되지 않아서 보안에 취약하며, 왔다 갔다 하는 패킷을 간단한 도구만으로 훔쳐볼 수가 있다. OpenSSH는 보안을 최우선으로 생각하는 운영체제인 OpenBSD에 속해있는 프로젝트로 암호화와 인증키 방식을 통해서 보안을 강화했으며, 패킷을 훔쳐볼 수가 없게 되었다.
P2P Software	GNUTella	Peer-to-Peer는 Napster를 통해서 세상에 유명해졌는데, Napster가 불법으로 판정난 뒤에 여러 P2P Software들이 만들어지게 되었다. 그 가운데 가장 크고 유명하게 만들어진 것이 GNUTella인데, GNUTella는 중계 서버가 필요하지 않으며, 각각의 client들이 자체적으로 중계 서버 역할을 하기 때문에 전세계의 모든 GNUTella 작동 컴퓨터들을 검색할 수도 있다. 또한 방화벽 뒤에서도 문제 없이 작동하며, 거의 모든 Platform으로 이식되었기 때문에 platform이나 architecture에 구애됨 없이 쓸 수가 있다.
Desktop Management	Nautilus	Nautilus는 Windows의 Explorer와 같은 기능을 수행하는 GNOME용 File Manager로 File 관리를 위한 완전한 기능을 제공하는 것은 물론이고 자체적인 file 내용 확인 기능을 비롯해서 매우 편리한 기능들을 제공한다. 또한 Theme을 제공해서 기분에 따라 모양을 바꿀 수도 있다.
Mail Security	GNUpg	전자우편은 매우 편리하지만, 그 내용이 공개되어 있어서 누구나 볼 수 있을 뿐 아니라 전달되는 과정에서 내용이 변질되어도 알 길이 없다. 그래서 만들어진 것이 PGP인데, GNUpg (GPG)는 이 보다 더욱 강화되었다고 한다. 작동 방식은 PGP와 동일하며 공개키/인증키/비밀키 등을 사용해서 암호화하고 해독한다. 또한 대부분의 전자우편 Client들과 연동되어 사용되기 때문에 쉽게 사용할 수 있다.

분야	공개 소프트웨어	추천이유
Instant Messenger	Gaim	<p>ICQ를 시발점 하는 Instant Messaging은 이제 Web이나 전자우편 만큼 필수적인 것이 되었다. 그러나, 문제점으로 있는 것은 같은 Messenger를 사용하는 사람들끼리만 연락이 된다는 것이고, Messenger의 종류가 너무나 많다는 것이다. Gaim은 AIM (AOL IM), ICQ, Yahoo! IM, MSN, Jabber, IRC, Napster, Gadu-gadu, Zephyr등의 Messenger들을 동시에 사용할 수 있도록 하고 있어서 여러 Messenger를 한꺼번에 사용할 필요 없이 하나로 해결이 된다.</p>
Music Player	XMMS	<p>음악 File의 표준이 되어 버린 mp3 재생기는 종류도 많지만, Windows에서는 WinAmp로 통일이 되어 버린 듯하다. XMMS는 WinAmp와 매우 닮은 X Window용 mp3 재생기로 모양과 기능만 비슷한게 아니고 Skin까지 같이 쓸 수 있도록 만들었다. XMMS는 mp3만이 아니라 Internet Streaming 방송은 물론이고 Plug-in에 따라 동영상까지 재생할 수 있도록 되어 있다. 매우 쉬운 plug-in 연동을 지원한다.</p>
Multimedia Player	MPlayer	<p>Linux용 영화 재생기 (Movie Player)인 MPlayer는 뛰어난 성능뿐 아니라 매우 다양한 종류의 동영상 파일과 코덱을 지원하는 아주 훌륭한 동영상 재생기다. 이 프로그램 하나로 많은 사람들이 'Linux는 Multimedia에 약하다'는 말을 할 수 없어졌다. 오히려 Windows Mediaplayer에서 정상적으로 재생이 되지 않고 화면이 일그러지고 끊기던 동영상이 Linux MPlayer에서 문제없이 재생되는 것을 여러번 겪었다. Desktop Linux를 원한다면 절대 뺄 수 없는 프로그램이다.</p>
Graphic Viewer	GQView	<p>GQView는 다양한 종류의 Graphic File Format을 지원할 뿐 아니라 왼쪽에 Directory를 Tree 구조로 보여주어 쉽고 빠른 Browsing을 가능케 해준다. 그리고 파일 Thumb Nail 기능으로 그림 파일을 열지 않고도 내용을 쉽게 확인할 수 있도록 해 주며, 기초적인 File 관리 기능들을 제공해서 Shell이나 다른 File Manager를 실행하지 않아도 웬만한 File 처리는 할 수 있다. 또한 Slide 기능도 제공하고 있어서 많은 그림을 편안히 감상할 수 있도록 해 준다.</p>
Network Security	firewall-easy	<p>여러 방화벽 관리 도구들이 있지만, firewall-easy는 매우 쉬운 방화벽 관리 도구다. 왜냐하면 전혀 설정이 필요 없기 때문이다. firewall-easy는 IP Table을 기반으로 Masquerading 등을 사용하고 있으며, IP와 Net/Mask, DNS 등을 자동으로 감지한다. 강력한 방화벽 기능제공과 함께 사용이 편리함</p>

분야	공개 소프트웨어	추천이유
Graphic Processor	GIMP	GIMP는 더 이상 말이 필요 없을 정도로 유명한 Open Source Project 가운데 하나로 GNOME에서 사용하는 Gtk Library 자체가 GIMP를 만들다가 나온 부산물일 정도로 방대하고 훌륭한 프로그램이다. Linux의 PhotoShop으로 불리는 GIMP는 Graphic을 처리하고 작업하는데 있어서 더 없이 훌륭한 프로그램이며, Script-Fu라는 기능을 이용하면 여러번 반복해야 하는 작업을 한 번에 해 줄 수 있도록 해 주기 때문에 어느 면에서는 Adobe PhotoShop보다 더욱 편리하다.
Vector Type Graphic Processor	Sodipodi	Sodipodi는 Vector 기반의 그리기 프로그램으로, 같은 기능을 제공하는 상용 프로그램으로는 CorelDraw와 Adobe사의 (가장 유명하고 널리 쓰이는) Illustrator가 있다. 많은 특화된 기능들을 갖고 있는 Sodipodi는 Web Designer들에게는 매우 유용한 도구가 될 것이다. Vector 기반의 그리기 도구가 PhotoShop과 같은 종류보다 더 나은 점은 아무리 그림을 확대해도 계단현상이 발생하지 않는다는 것이다. Vector를 사용하기 때문에 늘 매끄러운 맵시를 유지한다.
3D Image Processor	Blender	Linux는 이제, 영화 산업에서 터줏대감이 되어 가고 있다. Shrek, Blade II, Titanic, Lord of the Rings, Toy story, Star Wars Episode II 등을 비롯한 영화와 Dreamworks사에서 만들어지는 모든 영화(최근에는 Sinbad - the Legend of Seven Seas가 있다)에서 Linux가 아주 중요하게 사용되고 있는 것이다. Computer를 이용한 Modeling/Animating 가운데 상용으로는 SGI의 자회사가 만든 Maya가 가장 유명하데 일본의 유명한 게임인 Final Fantasy가 이를 사용했다. Free Software에서는 Blender가 있는데, 압축된 Binary가 겨우 2MB밖에 되지 않으면서도 Modeling/Animation/Post-production/Real-time interactive 3D/Game 제작까지 수많은 기능을 갖고 있는 Cross-platform 3D Graphic Program이다. Blender의 성능을 알고 싶다면, “유명한 영화”를 감상하면 된다.
Video Conference	GNOME Meeting	해외에 있는 지사 혹은 협력사와의 회의. 전자우편으로만 하기에는 찝찝하고, 전화로만 하기에는 뭔가 어색하고 부족한 듯하다. 그러나 만나러 해외에 가기에는 턱도 없는 소리다. 이럴 때에 필요한 것이 화상회의인데, GNOME Meeting이 바로 답이다. 다양한 음성 및 영상 코덱을 지원하고 있으며, 국제 표준인 H.323을 완벽히 지원하기 때문에 MS Windows에서 지원하는 NetMeeting과도 100% 호환되며, 같이 사용할 수 있다.

분야	공개 소프트웨어	추천이유
File/Printer Sharing	Samba	File 및 Printer를 공유할 수 있는 소프트웨어로서 Windows NT Server인 것처럼 작동하면서 사내 Network 공유를 가능하게 해 준다. 너무나 필수적이어서, 오히려 없다면 이상한 Service가 바로 Samba Service다.
Network Monitoring	Nagios	Nagios는 Notices Any Glitch In Our System의 약자로 전반적인 Network 상황에 대해 Monitoring할 수 있을 뿐만 아니라 Network에 있는 각 Host들에 대한 상태도 점검할 수 있는 매우 유용하고 뛰어난 Network Monitoring Tool로 Web을 기반으로 조작하고 작동되기 때문에 상당히 쉽고 설정에 따라 어디서나 확인하고 작동시킬 수 있다.
ERP/CRM Software	Compiere	ERP(Enterprise Resource Planning)와 CRM(Customer Relationship Management)는 아마도 기업에서 가장 중요한 것 가운데 하나가 아닐까 한다. 중요하지만, 어려운 것이 바로 ERP와 CRM이 아닐까 한다. 이를 쉽게 해 주는 프로그램들이 있는데, 상용 프로그램들은 일반인이 상상하기 힘들 정도로 비싼 것들이 대부분이다. 그러나 많은 기업들이 ERP와 CRM을 엄청나게 비싼 돈을 들여 도입하고 난 뒤에 제대로 사용하지도 못하고 사장시키는 경우가 많다. Compiere는 ERP와 CRM이 통합된 프로그램으로 기본 라이선스가 MPL이며(상업적 지원을 받고 싶다면 일정 금액을 내고 받을 수 있다) 물론 무료로 구할 수 있다. 그렇다고 Compiere가 사용할 수 없을 만큼 품질이 형편없거나 나쁘지 않다.
Integrated Development Environment	KDevelop	KDevelop은 KDE에서 개발을 돕기 위해서 만든 통합 개발 환경으로 KDE와 GNOME을 모두 지원하며, 언어적으로는 C와 C++을 지원하고 있다. Library를 QT와 GTK를 모두 사용할 수 있는 것은 더 말할 필요도 없다. 이미 수 없이 많은 훌륭한 프로그램들이 KDevelop으로 만들어졌기 때문에 성능에 대해서도 말할 필요조차 없다.
Delphi on Linux	Kylix	Delphi는 Pascal을 기본으로 Borland에서 만든 개발 언어로 MS Windows에서 매우 많은 개발자들의 사랑을 받고 있으며, 훌륭한 프로그램들이 Delphi로 만들어졌다. 이러한 Delphi가 이제 Linux에서도 사용할 수 있게 되어서 많은 개발자들을 더욱 Linux로 끌어들이 수 있게 되었다. 또한 이미 Delphi로 만들어진 Windows용 응용 프로그램들을 쉽게 Linux로 이식할 수 있게 되었다. Kylix는 여러 버전이 있는데 그 가운데 Open Kylix는 GPL로 배포된다.

분야	공개 소프트웨어	추천이유
Easy Programming on Linux	wxBasic	<p>BASIC의 목적은 초보자들도 쉽게 프로그램을 만들 수 있도록 하는 것이다. 그렇기 때문에 왕년에 엄청난 위력을 떨쳤던 Cobol과 Fortran이 이제는 거의 안 쓰이거나 특수한 분야에 국한되어 쓰이고 있는데도 Basic은 대단한 인기를 누리고 있는 것이다. Linux에서 돌아가는 Basic이 매우 많지만, wxBasic을 고른 이유는, 인터프리터 방식으로 작동해서 쉽게 실행과 디버깅을 할 수가 있으며, 다른 Basic보다 문법이 QBasic과 아주 유사해서 더더욱 사용자들 (특히 초보자들)과 친숙하다는 점이다. 그리고 윈도우즈에서도 움직이기 때문에 한 번만 만들면 윈도우즈와 Linux에서 완벽하게 작동한다. Linux에서는 C만 쓸 것 같은 분위기, Shell Script를 하려고 해도 C같은 느낌 때문에 어렵다. 그래서 더욱 사용자들을 멀리하게 하는 것 같지만, Linux에서도 쉽게 자신만의 프로그램을 만들 수 있다.</p>
Computer Aided Design	QCAD	<p>CAD는 그리 많은 사람들이 쓰는 것은 아니지만, Linux에도 매우 훌륭한 CAD 프로그램이 있다. 상용으로는 AutoCAD가 꽉잡고 있는 것 같은데, QCAD도 그에 못지 않는 성능을 갖고 있다. QCAD는 License가 GPL로 바뀐 다음부터 매우 역동적으로 개발되고 있으며, 매뉴얼 및 도움이 한글로도 제공된다.</p>
Game	Mame	<p>Linux가 Game에 약한 것은 사실이지만, 그렇다고 전무한 것은 아니다. 이미 Linux용으로 많은 훌륭한 Game들이 나와 있는데, Mame는 Game은 아니고 Emulator다. 바로 오락실에 놓여 있는 오락기들을 Emulation해 주는 프로그램으로 Game Rom만 구한다면 어떤 오락실 Game이라도 즐길 수 있다.</p>

부록2. 비공개 소프트웨어 대 공개소프트웨어 안내

가. Networking

소분류	상용 제품명	공개 소프트웨어	제품 설명 및 특징	라이선스
Web Browser	Internet Explorer	Mozilla	1) 넷스케이프의 공개소프트웨어 버전 2) 웹브라우저, 이메일 리더, 챗 클라이언트 등등 핵심 어플리케이션의 통합	MPL NPL GPL
		Konqueror	1) KDE기반의 기본 웹브라우저 2) 윈도우의 탐색기와 유사한 기능제공(파일 관리, 웹 브라우저, 다양한 뷰어 기능 등)	GPL2
		Galeon	1) GNOME기반의 웹 브라우저 2) 다양하고 편리한 기능 제공	GPL
Email client	Outlook Express	Ximian Evolution	1) 그놈 기반의 개인정보관리 프로그램 2) 이메일, 캘린더, 통합정보관리, 일정관리, 메시지 보드 지원	GPL2
		Mozilla Mail	1) 모질라에 포함되어 있는 전자우편 2) 웹과 통합된 전자우편 기능 지원	MPL
		Kmail	1) KDE를 기반으로 한 이메일 클라이언트 2) 아웃룩 익스프레스와 같이 이메일 관련 기능만 있음	GPL2
FTP-clients	Cute-FTP WS-FTP	Gftp	GTK+ 기반의 FTP 클라이언트 프로그램	GPL2
		KBear	KDE 기반의 FTP 클라이언트 프로그램	GPL
		ncftp	콘솔용 ftp 클라이언트. 메타문자 지원, 리커시브 디렉토리 지정 가능, 탭 완성 기능	Artistic
IRC-clients	Mirc	Xchat	1) 유닉스 기반의 IRC 클라이언트 프로그램 (GTK +) 2) 거의 대부분의 플랫폼 지원	GPL
		Kvirc	유닉스 기반의 IRC 클라이언트 프로그램 (QT)	GPL
		bitchX	콘솔 기반의 IRC. 쉬운 사용법. 많은 기능 지원	GPL2 / Others
Instant Messaging clients	ICQ MSN Yahoo Miranda	Gaim	1) 인스턴스 메세지 클라이언트 2) 대부분의 플랫폼과 메시징 프로토콜을 지원	GPL
		Kmess(MSN)	리눅스에서 MSN 사용시 이용하는 KDE기반의 메시지 클라이언트	GPL
		Jabber	1) 여러 플랫폼에서 사용가능 2) MSN을 비롯한 여러 메신저와 메시징 가능	GPL / JOSL
IDS	BlackICE	Snort	1) 널리 쓰이는 네트워크 침입 탐지 시스템 2) 다양한 플랫폼 지원 및 다양한 플러그인과 롤	GPL2
Peer-to-pe er clients	WinMX Napster	GNUtella	1) 강력한 기능을 갖는 파일 공유 프로그램 2) 방화벽 뒤에서도 사용가능 3) 서로가 서로를 계속 중계	GPL

소분류	상용 제품명	공개 소프트웨어	제품 설명 및 특징	라이선스
Remote management	pcAnywhere VNC	VNC	현재 운영 중인 컴퓨터뿐만 아니라 인터넷으로 연결된 다양한 아키텍처의 컴퓨터 데스크탑 환경을 보면서 제어할 수 있도록 해주는 원격 디스플레이 시스템	GPL
		Open SSH	1) 콘솔 기반의 원격 셸 (telnet, rsh 대응) 2) 매우 강력한 암호화 및 호스트 인증으로 보안에 최적	BSDL
Network monitoring tool	Dumeter, Netmedic	Gkrellm	X 윈도우 상에서 CPU 부하량, 프로세스, 디스크 액세스, 네트워크를 통한 서버 접속 상황 등등을 한눈에 모니터링 가능	GPL
		Etherape	네트워크 상태를 그래픽할 수 있도록 함	GPL
		MRTG	1) SNMP를 지원하는 네트워크장비가 발생하는 트래픽을 모니터링 2) 지정한 시간마다 모니터링 결과값을 GIF(or PNG)이미지 생성하여 HTML 페이지로 뿌려주기 때문에 쉽게 트래픽을 알 수 있음	GPL
		Nagios	1) 웹 기반의 쉬운 작동 2) 각 인터넷 서비스 감시 3) 각 호스트의 상태 점검	GPL2
Video conference	NetMeeting	GnomeMeeting	1) H.323 호환되는 비디오 컨퍼런스 2) MS NetMeeting과 연동 가능 3) 비디오 컨퍼런스에 필요한 모든 기능 제공	GPL
Firewall tool	ZoneAlarm	Kmyfirewall	KDE용 방화벽 구성 툴	GPL
		firewall-easy	1) 강력한 여러 기능 2) 매우 쉬운 설정 (거의 설정 필요 없음)	GPL
Network maintance tool	HP OpenView, MS SMS, Tivoli	Big Brother	웹 기반으로 쉬운 사용 웹 기반으로 어디서나 사용 실시간 네트워크 모니터링	BTFL / 상용 라이선스
Protocols analysing	MS Network Monitor	Ethereal	1) 네트워크 프로토콜 분석기 2) 다양한 프로토콜 지원	GPL
		Tcpdump	네트워크 프로토콜 분석에 가장 기본이 되는 분석기	BSD License
Security scanner	ISS	Nessus	1) 원격 보안 스캐너 2) 분류된 결과 보고서와 다양한 포맷 변형 가능	GPL2
		Nmap	포트 스캐너의 가장 기본이 되는 분석기	GPL2
Sharing data/files	Novel Netware	Samba	윈도우 시스템과 리눅스의 다양한 퍼미션을 통한 파일서버 및 프린터 서버 관리가능	GPL

나. Work with files

소분류	상용 제품명	공개 소프트웨어	제품 설명 및 특징	라이선스
File manager	Windows Explorer	Konqueror	1) KDE용 파일 관리자 2) 웹 브라우저와 겸용	GPL2
		Nautilus	1) GNOME용 파일 관리자 2) 멀티미디어 파일을 비롯한 파일 쉽게 보기 지원 3) 테마지원으로 모양 변경 가능	GPL2
compressed files	WinZip	Gnzip	1) 워프과 비슷함. 2) 프론트 엔드 형식으로 다양한 압축파일 지원	GPL2
Console archivers	arj, rar, zip	tar, gzip, bzip2	압축과 아카이빙이 분리되어 있어서 더욱 강력한 작업이 가능	GPL

다. System Software

소분류	상용 제품명	공개 소프트웨어	제품 설명 및 특징	라이선스
Text editor	Notepad, WordPad	Kedit	KDE용 기본 문서 편집기	GPL2
		Gedit	GNOME용 기본 문서 편집기	GPL2
Console text editor	FAR Editor	Vi	1) 유닉스 계열의 필수/기본 편집기 2) 정규식을 이용한 막강한 검색/치환기능 3) 키보드에서 손이 거의 움직이지 않는 편리한 명령키 배열 4) 개발용 문법 표시 기능 및 자동 들여쓰기	GPL
		Emacs	1) GNU 프로젝트의 시초가 된 편집기 2) LISP 언어 사용으로 학습기능 3) Add-on 기능으로 메일, 유즈넷, 게임등 추가	GPL
Multi-purpose text and source code editor	UltraEdit, Editplus	Kate (KDE)	1) 파워풀한 하이라이팅 기능 2) 킥커러 HTML 뷰어	GPL2
		Glimmer	1) 강력한 문법 표시 기능 (HTML을 비롯해 20개 이상의 언어 기본 지원) 및 괄호 검사 기능 2) 여러 파일 동시 검색/치환 기능 3) 다단계 실행 취소 기능	GPL
Viewing PostScript	RoPS	GhostScript/ GhostView	1) PS와 PDF를 동시 지원 2) 거의 모든 플랫폼에서 사용 가능 3) 인쇄를 비롯한 다양한 기능 지원 및 다른 프로그램에서 붙여 쓸 수 있음 4) GhostScript는 해석기, GhostView는 프론트엔드	GPL
Viewing PDF	Adobe Acrobat Reader	Xpdf	1) 쉽게 사용 가능 2) 다양한 플랫폼 지원 3) PDF 전용 뷰어	GPL2

라. Multimedia(audio/CD)

소분류	상용 제품명	공개 소프트웨어	제품 설명 및 특징	라이선스
Programs for CD burning with GUI	Nero, Roxio Easy CD Creator	K3b. (KDE)	1) 일반적인 CD 제작을 기본으로 비디오CD, CD 립, DVD 립 등 다양한 CD 제작 가능 2) 직관적인 그래픽 인터페이스 3) KDE 기반으로 만들어짐	GPL
		XCDRoast	1) 사용자나 그룹을 가리지 않고 사용 가능 2) 끌어 놓기를 지원 3) 여러 플랫폼을 지원	GPL2
		Gnome Toaster	1) 끌어 놓기를 지원 2) GNOME 기반으로 작성됨 3) On-the-Fly 방식의 CD 복사 지원 4) Disk-at-Once 모드로 On-the-Fly를 지원하는 유일한 프로그램	GPL
Music mp3 players	Winamp	XMMS (X multimedia system)	1) 윈앰프와 비슷한 모양 및 비슷한 사용법 2) 윈앰프와 스킨을 공유할 수 있다. 3) 다양한 플러그인으로 쉽고 막강한 기능 확장	GPL

마. Multimedia(graphics)

소분류	상용 제품명	공개 소프트웨어	제품 설명 및 특징	라이선스
Graphic files viewer	ACDSee	Xnview	1) 간단한 사용법 2) 이미지 파일 조작 및 변환 가능 3) 썸네일 생성 4) 400개 포맷 보여주기 가능 5) 40개 포맷으로 내보내기 가능 6) 명령행 변환기와 개발자용 라이브러리 제공	Freeware
		Gqview	1) 쉬운 사용법 (끌어 놓기) 2) 기본적인 파일 관리 기능 제공 3) 디렉토리 브라우징 기능	GPL
graphic editor	Adobe Photoshop	Gimp	1) Gnome 기반의 그래픽 프로그램 2) 사진 리터칭과 이미지 작업에 최적화 됨	GPL
vector graphics	Adobe Illustrator	Sodipodi	벡터 그래픽 제작 기능	GPL
Flash creation	Macromedia Flash	DrawSWF	JAVA2로 만들어져서 플랫폼에 구애받지 않고 실행 가능하며, 쉽고 간단한 플래시 제작 가능	GPL
		Ming	1) 플래시를 만들기 위한 라이브러리 2) C를 기본으로 C++, PHP, Python, Ruby에서 사용 가능	LGPL
3D-graphics	3D Studio MAX, Maya	Blender	2002년 7월 톤 루센탈에 의해 10원 라이선스를 GPL로 바꾸고, 2003년 2월 블랜더 2.6이 발표되어 블랜더의 부활을 다시 한번 꿈꾸고 있다.	GPL
		KPovModeler	KDE기반의 프로그램 POV-Ray 파일을 만들어줌 계층적 객체 구조	GPL2
making screenshots	Snag it	Ksnapshot	KDE 기반의 화면 캡처 프로그램	GPL

바. Multimedia(video and others)

소분류	상용 제품명	공개 소프트웨어	제품 설명 및 특징	라이선스
Video / mpeg4 players	Windows Media Player	Mplayer	1) 리눅스에서 가장 많이 이용하는 동영상 재생기 2) DivX/AVI/ASF/MPEG1/MPEG2 지원 3) DVD, VCD 지원	GPL2
DVD players	PowerDVD, WinDVD	Ogle	1) 첫번째 Opensource DVD 플레이어 2) JPG로 저장, 전체모드 등 다양한 기능 제공	GPL
Professional video creation and editing	Adobe Premiere, Media Studio Pro	Cinelerra	형상 편집을 위해 필요한 무수히 많은 기능들	GPL2
Cutting video	Virtual Dub	Avidemux	영상 편집, 추가, 절단, 필터링, 인코딩 지원	GPL2
		Kino	음성, 영상 편집 기능 IEEE 1394 캡션 지원 99회까지 실행 취소 가능	GPL2
Converting video	Virtual Dub	Transcode	영상 편집 기능 제공 영상 크기 변경 가능 DVD를 비롯한 다양한 포맷 지원 포맷 변환 가능	GPL2
		Ffmpeg	영상 포맷을 간단히 변환 가능 영상 스트리밍 서비스도 지원	GPL
Animation	Animation Shop	CinePaint	35mm 필름에 최적화된 리터칭 프로그램 해리포터, 스튜어트 리틀을 비롯한 여러 영화에 사용됨 (안정성과 기능이 입증됨)	GPL

사. Emulators

소분류	상용 제품명	공개 소프트웨어	제품 설명 및 특징	라이선스
Virtual machine emulator	VMWare for Windows	VMWare for Linux.	1) 복수의 가상 시스템을 운영할 수 있음 2) 다양한 운영체제 지원 3) 사운드 입출력, SCSI, CD-ROM 드라이버 지원	상용
Windows emulator		Wine	1) 다양한 윈도우 어플리케이션 사용 가능 2) Win32 (9x/NT/XP), Windows 3.x and DOS binaries 지원 3) Win32 and Win16 함수 콜 지원	GPL2
		Mame	여러 플랫폼 지원 게임기들을 에뮬레이트	Mame License

아. Office and Business

소분류	상용 제품명	공개 소프트웨어	제품 설명 및 특징	라이선스
Office suite	MS Office	OpenOffice	1) StarOffice에 기반한 오픈 소스 오피스 2) MS Office와 완벽한 호환성 3) 높은 신뢰도 4) 유럽시장에서 약 40%의 점유율 보임 5) 다양한 플랫폼 지원	GPL/ LGPL/ SISSL
		StarSuite	1) OpenOffice에 몇 가지 유틸리티를 붙여 판매 2) Sun에서 제품 지원 및 보증 3) 약 11개 언어로 판매되고 있음	상용
		Koffice	1) KDE기반의 오피스 2) 다양한 기능을 갖는 프로그램이 융합되어 있음	GPL
		HancomOffice	1) 한글을 리눅스로 포팅한 제품 2) 프리젠테이션과 스프레드 시트 등을 추가하여 판매 3) 한컴 리눅스에서 판매	상용
		Gnome Office	GNOME기반의 오피스 역시 다양한 프로그램이 섞여 있음	GPL2
Word processor	MS-Word	Abiword	GNOME Office에 포함되어 있는 워드프로세서	GPL
		Ted	많은 기능을 갖고 있는 워드프로세서	GPL
		Kword	KDE Office에 포함되어 있는 워드프로세서	GPL
		OpenOffice Write	OpenOffice에 포함되어 있는 워드프로세서	GPL
Spreadsh eets	Excel	Gnumeric	GNOME Office에 포함되어 있는 스프레드시트	GPL2
		Kspread	KDE Office에 포함되어 있는 스프레드시트	GPL
		OpenOffice Calc	OpenOffice에 포함되어 있는 스프레드시트	GPL
Graphing / charting data	Excel	Kivio	Visio 스타일의 흐름도 프로그램 KDE Office에 포함되어 있다.	GPL
		Dia	Visio와 비슷한 다이어그램 프로그램 GNOME Office에 포함되어 있다.	GPL2
		OpenOffice Draw	OpenOffice에 포함되어 있는 그리기 프로그램	GPL

Creating presentations	MS PowerPoint	OpenOffice Impress	OpenOffice에 포함되어 있는 멀티미디어 프리젠테이션 프로그램	GPL
		Kpresenter	KDE에 포함되어 있는 프리젠테이션 프로그램	GPL
Local database	Access	Kexi	KDE Office에 포함되어 있는 통합 데이터베이스 환경 액세스	GPL
		Gnome DB	GNOME Office에 포함되어 있는 데이터베이스 연결 프로그램	GPL2
Software for e-commerce and web business	Weblogic, IBM WebSphere Application Server	JBoss	1) J2EE 기반의 WAS(Web Application Server) 중 EJB 컨테이너 2) J2EE 상품으로 기업용 시장에서 두각	LGPL
Personal finances manager	MS Money	GNUcash	GNOME Office에 포함되어 있는 개인 재정 관리 프로그램	GPL
		GnoFin	1) GNOME에 기반한 개인 재정 관리 프로그램 2) 작고 사용하기 쉽다.	GPL
Project management	MS Project	Mr Project	1) GNOME Office에 포함되어 있는 프로젝트 관리 프로그램 2) 인터페이스가 Ms Project와 유사하다	GPL2
		Toutdoux	GNOME Office에 포함되어 있는 프로젝트 관리 프로그램 데이터베이스 백엔드를 갖고 있다.	GPL
ERP, CRM	BOSS-Corporation	Compiere	CRM과 통합된 ERP 프로그램 제품 지원을 받을 수 있다.	MPL 1.1

자. Programming and Development

소분류	상용 제품명	공개 소프트웨어	제품 설명 및 특징	라이선스
IDE	Microsoft VisualStudio.net	CodeForge	디버거가 통합된 환경	상용
		Kdevelop	1) KDE(QT), GNOME, C, C++ 지원 2) 프로젝트 관리 기능 3) 대화상자 편집기 4) 클래스 관리 도구 5) 통합된 디버거 6) 어플리케이션 마법사	GPL2
		Eclipse	IBM, 레드햇, 수세를 비롯한 유수업체 참여 ETRI도 참여하고 있음	CPL
		Glade	1) GTK (GNOME)와 C를 지원 2) XML 설치만으로 C++, Ada95, Python, Perl 지원 3) 쉽고 빠르게 비주얼한 사용자 인터페이스를 만들 수 있도록 해줌	GPL
Visual C++ IDE	Borland C++ Builder, MS Visual C	Anjuta + Glade + Devhelp	MS Visual Studio와 같이 통합된 환경에서 컴파일, 디버그, 도움말까지 참조할 수 있도록 됨	GPL2
		Kylix Open Edition	1) 오랜 경험의 볼랜드에서 만든 리눅스용 통합환경 2) 리눅스에서 델파이를 사용할 수 있는 유일한 도구	GPL
		Kdevelop	위 설명 참조	GPL

C++ IDE	Borland Turbo C++ 3.0 for DOS	GCC + vi/Emacs...	강력하고 편리한 콘솔 개발환경 쓸 수록 빠져드는 매력적인 환경	GPL2
		Eclipse	위 설명 참조	CPL
Pascal	Turbo Pascal	Freepascal	1) Turbo Pascal 7.0과 호환되는 문법 2) 인텔 계열과 모토롤라 계열의 CPU 지원 3) Linux, FreeBSD, DOS, OS/2, BeOS, SunOS, QNX, 아미가 등의 여러 OS 지원 4) 32bit 컴파일러	GPL
		GNU Pascal	1) 32/64bit 컴파일러 2) gcc가 작동하는 모든 시스템에서 사용 가능 3) 모든 GNU 프로그래밍 유틸리티와 호환 (gdb 포함) 4) Turbo Pascal 7.0과 호환되는 문법 5) Borland Delphi의 일부로 사용됨 6) 표준 및 확장 파스칼 문법을 완벽히 지원	GPL
Basic	GW-Basic, Quick Basic, Visual Basic	wxBasic	1) Linux와 윈도우즈에서 사용 가능 2) QBasic과 비슷한 문법 3) 인터프리터 언어	GPL
		Gambus	1) 비주얼한 개발 환경 (Visual Basic과 비슷) 2) 컴파일러와 인터프리터로 모두 사용 가능 3) 통합된 개발 환경	GPL
		ScriptBasic	1) 확장과 임베디드가 수월함 2) 컴파일러와 인터프리터로 모두 사용 가능 3) 멀티 쓰레드를 지원	GPL
		X11-Basic	1) 아타리용 베이직과 비슷한 문법 2) X11 기능을 완전히 사용 가능 3) 인터프리터 언어로 작동 4) 셸이나 CGI로 사용 가능	GPL
		KBasic	1) KDE 용으로 개발된 베이직 2) 비주얼한 쉬운 개발 환경 3) 인터프리터 언어로 작동	GPL
		Blassic	1) 행번호가 반드시 있어야 하는 전형적 베이직 2) 고전적인 PEEK, POKE 명령 사용 가능	GPL
		YaBasic	윈도우즈와 유닉스에서 사용 가능	GPL
		XBasic	1) 윈도우즈와 리눅스에서 사용 가능 2) 통합된 그래픽한 개발 환경 제공 3) 컴파일러 언어로 작동	GPL/ LGPL
Prolog	VisualProlog	GNU Prolog	1) 표준 프롤로그 지원 2) 저수준 디버거 제공 3) Native Code는 WAM-CC와 비슷한 속도 4) Byte Code는 WAM-CC보다 5배 정도 빠름	GPL
		Mercury	1) Logic 프로그래밍 언어이면서 2) 동시에 함수형 프로그래밍 언어	GPL or LGPL

Assembler	TASM, MASM, NASM	NASM	1) 인텔 계열의 어셈블러 2) a.out 파일 형식 지원 3) 리눅스/BSD/COFF/MS16bit/MSwin32 파일 형식 지원	LGPL
		FLAT Assembler	1) 인텔 계열의 어셈블러 2) DOS/Linux/Windows 지원 3) MMX, SSE, SSE2, 3DNow 지원	BSDL
Debugger		gdb	1) 여러 언어에서 사용할 수 있는 디버거 2) 윈도우즈와 대부분의 유닉스에서 작동 가능 3) 로컬뿐 아니라 네트워크를 통해서도 실행 가능	GPL2
WYSIWY G html editor	Dreamweaver , Frontpage	Netscape / Mozilla Composer.	1) 모질라 웹 브라우저에 기본으로 붙어 있는 편집기 2) 문서 편집하듯 보면서 할 수 있다.	MPL
		Openoffice HTML editor.	상동	GPL
HTML editor	HomeSite	Quanta Plus	1) KDE용 웹 개발 환경 2) 디렉토리 트리를 이용해 대규모 웹을 쉽게 제작 가능	GPL
		Bluefish	1) 숙련된 웹 디자이너와 개발자를 위한 에디터 2) 통합 환경을 제공 3) WYSIWYG 환경 제공 4) 다양한 언어를 지원	GPL
Java IDE	Jbuilder	NetBeans	1) 자바용 통합 개발 환경 2) 자바 외에 C, C++, XML, HTML을 지원 3) CVS지원 4) 비주얼한 디자인 환경 제공 5) 마법사와 코드 생성 관리 기능 제공 6) JSP, XML, RMI, CORBA, JINI, JDBC, Servlet 지원	SPL
		Eclipse	위의 설명 참조	CPL
Graphical libraries	WinAPI, MFC	Tk (Tcl, C)	스크립트 언어인 TCL와 거의 함께 사용	BSD License
		Qt (C++)	C++을 기반으로 하며 윈도우즈용과 비슷한 컴포넌트제공	GPL
		GTK+ (C, C++)	C를 기본으로 하고, C++용 래퍼 있음	GPL2

차. Scientific and Special Programs

소분류	상용 제품명	공개 소프트웨어	제품 설명 및 특징	라이선스
CAD/CAM	Autocad, ArchiCAD	Qcad	1) 강력한 2차원 캐드 프로그램 2) 많은 참여로 활발히 개발되고 있음	GPL2
Diagram and chart designer	Microsoft Visio	Kivio (Koffice)	1) KDE 기반의 다양한 흐름도(Flow Chat) 프로그램 2) 기본 표준규격 지원 (letter, A4, etc...)	GPL
		Dia	GTK+ 기반의 다이어그램 생성 프로그램	GPL
		GNUplot	1) 콘솔 기반의 그리기 프로그램 2) 다양한 공식으로 그래프 생성 가능 3) 강력한 그리기 기능	GPL
GIS (Geographical information system)	ArcView	Grass	1) 명령행으로 사용하거나 Tcl/Tk 그래픽 인터페이스 사용 가능 2) 윈도우에서도 실행 가능 3) Intel x86, Motorola PPC, SGI MIPS, Sun SPARC, Alpha AXP, HP PA-RISC, CRAY를 비롯한 여러 플랫폼에서 사용 가능 4) Linux/Intel, Linux/PowerPC, Solaris/SPARC, Solaris/i86, SGI IRIX, HP UX, Mac OS X (Darwin), IBM AIX, BSD-Uinx variants, FreeBSD, CRAY Unicos, iPAQ/Linux handhelds and other UNIX compliant platforms (32/64bit) 등의 운영 체제에서 사용 가능 5) 방대하고 부족함 없는 기능	GPL2
		Quantum GIS.	Unix/Linux에서 작동, PostgreSQL 지원, 지리 파일 화면 출력, 동적인 플러그인 지원, 지도 출력, 스크립트 엔진, 메타 데이터 지원	GPL
HDD testing / benchmarking	SiSoft SANDRA	Bonnie++.	1) 2G 이상의 스토리지 지원 2) ReiserFS를 비롯한 여러 프로그램 지원 3) 동일 디스크의 다른 Zone의 성능 측정 가능	GPL2

카. Server Software

소분류	상용 제품명	공개 소프트웨어	제품 설명 및 특징	라이선스
Web-server	Apache, IIS	Apache	1) 웹서버 시장 점유율 65% (2003년 3월기준) 2) 아파치 관련된 다양한 모듈 3) 멀티 플랫폼	Apache Software License 1.1
FTP-server	Internet Informatio n Server	wu-ftpd	레드햇 6.x 에서 가장 널리 쓰임	Wu-Ftpd Software License
		proftpd	1) 보안이 강화된 ftp 서버 2) 아파치 스타일 설정 파일로 관리가 쉽다.	GPL
Language for Web-devel opment	ASP	PHP	Apache 모듈로 포함되면서 Mysql과 함께 널리 쓰이는 서버 스크립트 언어	PHP License
		Perl	과거 시스템 관리 및 CGI 프로그래밍 시에 널리 이용되었던 서버 스크립트 언어	Artistic license
Database engine	MS SQL	PostgreSQL	가장 발전된 공개소프트웨어 데이터 베이스 엔진	BSD License
		MySQL	1) 가장 많이 쓰이는 공개소프트웨어 데이터 베이스 엔진 2) 멀티유저, 멀티쓰레드 지원	GPL2
Email server	Mdaemon	Sendmail	1) 가장 널리 쓰이는 MTA(Mail Transfer Agent)	Sendmail License
		Qmail	1) 안정성에 가장 큰 기반을 두고 설계 2) 성능, 신뢰성 및 간편성 우수 3) 가장 널리 쓰이는 Sendmail의 아성에 도전	Qmail License
Email / PIM / Groupware server	Microsoft Exchange	Evolution	1) 그놈 기반의 개인정보관리 프로그램 2) 이메일, 캘린더, 통합정보관리, 일정관리, 메시지 보드 지원	GPL2
		Kroupware	독일 정부 지원으로 만들어진 KDE기반의 개인정보관리 시스템	GPL2
Proxy server	MS Proxy Server	Squid	1) 가장 널리 쓰이는 프락시 캐시 서버	GPL
		TinyProxy	설정이 필요없는 간단한 프록시 서버	GPL
Server for supporting Java Servlets	Tomcat	Tomcat	1) 자카르타 프로젝트의 하위 프로젝트로 J2EE 기반의 WAS(Web Application Server) 중 서블릿 컨테이너 2) 최신 버전 4.0, 개발버전 5.0.1	Apache Software License 1.1

부록3. 공개소프트웨어 리소스

사이트 이름	웹 주소
Apache Software Foundation	www.apache.org
European Commission Free and Open Source Software project	europa.eu.int/information_society/activities/opensource/index_en.htm
Freshmeat	www.freshmeat.net
GNU/Free Software Foundation	www.gnu.org
GNUWin II	gnuwin.epfl.ch/en/index.html
Government Open Code Collaborative (GOCC)	www.gocc.gov
Java-Source.net	www.java-source.net
Linux Australia	www.linux.org.au
Linux HQ	www.linuxhq.com
Linux Software Equivalents	linuxshop.ru/linuxbegin/win-lin-soft-en/table.shtml
LinuxWorld	www.linuxworld.com
Mac OS X Open Source Directory	www.apple.com/downloads/macosx/unix_open_source/
Open Source Industry Australia	www.osia.net.au
Open Software Initiative	www.opensource.org
Open Source Software Educational Society	www.softpanorama.org
Open Source Software Institute (OSSI)	www.oss-institute.org
OSdir.com	www.osdir.com
Samba	www.samba.org
Scientific Applications on Linux	ftp.llp.fu-berlin.de/lsoft/index.shtml
Slashdot	www.slashdot.org
SourceForge	www.sourceforge.net
Tigris.org	www.tigris.org
World Wide Web Consortium	www.w3.org/Status
자바스터디	www.javastudy.co.kr
자바서비스넷	www.javaservice.net
제이스툼	www.jstom.pe.kr
프로자바	www.pro-java.com
자바카페	www.javacafe.or.kr

자바모델링	www.javamodeling.com
J2EE스터디	www.j2eestudy.co.kr
JSP School	www.jspschool.com
OKJSP	www.okjsp.pe.kr
플렉스개발자커뮤니티	www.flex.or.kr
자바크래프트넷	www.javacraft.net
자바유저스넷	www.java-users.net
데이터베이스사랑넷	database.sarang.net
PostgreSQL 국내 공식 커뮤니티 사이트	www.postgresql.or.kr
리눅스포탈	www.superuser.co.kr
Linux.co.kr	www.linux.co.kr
KLDP	www.kldp.org
OOPS	www.oops.org

부록4. GNU 일반 공중 사용 허가서(2판)

[전 문]

소프트웨어에 적용되는 대부분의 사용 허가서(license)들은 소프트웨어에 대한 수정과 공유의 자유를 제한하려는 것을 그 목적으로 합니다. 그러나 GNU 일반 공중 사용 허가서(이하, 'GPL'이라고 칭합니다.)는 자유 소프트웨어에 대한 수정과 공유의 자유를 모든 사용자에게 보장하기 위해서 성립된 것입니다. 자유 소프트웨어 재단이 제공하는 대부분의 소프트웨어들은 GPL에 의해서 관리되고 있으며, 몇몇 소프트웨어에는 별도의 사용 허가서인 GNU 라이브러리 일반 공중 사용 허가서(GNU Library General Public License)를 대신 적용하기도 합니다. 자유 소프트웨어란, 이를 사용하려고 하는 모든 사람에 대해서 동일한 자유와 권리가 함께 양도되는 소프트웨어를 말하며 프로그램 저작자의 의지에 따라 어떠한 종류의 프로그램에도 GPL을 적용할 수 있습니다. 따라서 여러분이 만든 프로그램에도 GPL을 적용할 수 있습니다.

자유 소프트웨어를 언급할 때 사용되는 '자유'라는 단어는 무료(無料)를 의미하는 금전적인 측면의 자유가 아니라 구속되지 않는다는 관점에서의 자유를 의미하며, GPL은 자유 소프트웨어를 이용한 복제와 개작, 배포와 수익 사업 등의 가능한 모든 형태의 자유를 실질적으로 보장하고 있습니다. 여기에는 원시 코드(source code)의 전부 또는 일부를 원용해서 개선된 프로그램을 만들거나 새로운 프로그램을 창작할 수 있는 자유가 포함되며, 자신에게 양도된 이러한 자유와 권리를 보다 명확하게 인식할 수 있도록 하기 위한 규정도 포함되어 있습니다.

GPL은 GPL 안에 소프트웨어를 양도받을 사용자의 권리를 제한하는 조항과 단서를 별항으로 추가시키지 못하게 함으로써 사용자들의 자유와 권리를 실제로 보장하고 있습니다. 자유 소프트웨어의 개작과 배포에 관계하고 있는 사람들은 이러한 무조건적인 권리 양도 규정을 준수해야만 합니다.

예를 들어 GPL 프로그램을 배포할 경우에는 프로그램의 유료 판매나 무료 배포에 관계없이 자신이 해당 프로그램에 대해서 가질 수 있었던 모든 권리를, 프로그램을 받게 될 사람에게 그대로 양도해 주어야 합니다. 이 경우, 프로그램의 원시 코드를 함께 제공하거나 원시 코드를 구할 수 있는 방법을 확실히 알려주어야 하고 이러한 모든 사항들을 사용자들이 분명히 알 수 있도록 명시해야 합니다.

자유 소프트웨어 재단은 다음과 같은 두 가지 단계를 통해서 사용자들의 권리를

보호합니다. (1) 소프트웨어에 저작권을 설정합니다. (2) 저작권의 양도에 관한 설정법에 의해서 유효한 법률적 효력을 갖는 GPL을 통해 소프트웨어를 복제하거나 개작 및 배포할 수 있는 권리를 사용자들에게 부여합니다.

자유 소프트웨어를 사용하는 사람들은 반복적인 재배포 과정을 통해 소프트웨어 자체에 수정과 변형이 일어날 수도 있으며, 이는 최초의 저작자가 만든 소프트웨어가 갖고 있는 문제가 아닐 수 있다는 개연성을 인식하고 있어야 합니다. 우리는 개작과 재배포 과정에서 다른 사람에 의해 발생된 문제로 인해 프로그램 원저작자들의 신망이 훼손되는 것을 원하지 않습니다. GPL에 자유 소프트웨어에 대한 어떠한 형태의 보증도 규정하지 않는 이유는 이러한 점들이 고려되었기 때문이며, 이는 프로그램 원저작자와 자유 소프트웨어 재단의 자유로운 활동을 보장하는 현실적인 수단이기도 합니다.

특허 제도는 자유 소프트웨어의 발전을 위협하는 요소일 수밖에 없습니다. 자유 프로그램을 재배포하는 사람들이 개별적으로 특허를 취득하게 되면, 결과적으로 그 프로그램이 독점 소프트웨어가 될 가능성이 있습니다. 자유 소프트웨어 재단은 이러한 문제에 대처하기 위해서 어떠한 특허에 대해서도 그 사용 권리를 모든 사람들(이하, '공중(公衆)'이라고 칭합니다.)에게 자유롭게 허용하는 경우에 한해서만 자유 소프트웨어와 함께 사용할 수 있다는 것을 명확히 밝히고 있습니다.

복제(copying)와 개작(modification) 및 배포(distribution)에 관련된 구체적인 조건과 규정은 다음과 같습니다.

[복제와 개작 및 배포에 관한 조건과 규정]

제0조

본 허가서는 GNU 일반 공중 사용 허가서의 규정에 따라 배포될 수 있다는 사항이 저작권자에 의해서 명시된 모든 컴퓨터 프로그램 저작물에 대해서 동일하게 적용됩니다. 컴퓨터 프로그램 저작물(이하, '프로그램'이라고 칭합니다.)이란 특정한 결과를 얻기 위해서 컴퓨터 등의 정보 처리 능력을 가진 장치(이하, '컴퓨터'라고 칭합니다.) 내에서 직접 또는 간접으로 사용되는 일련의 지시 및 명령으로 표현된 창작물을 의미하고, '2차적 프로그램'이란 전술한 프로그램 자신 또는 저작권법의 규정에 따라 프로그램의 전부 또는 상당 부분을 원용하거나 다른 언어로의 번역을 포함할 수 있는 개작 과정을 통해서 창작된 새로운 프로그램과 이와 관련된 저작물을 의미합니다(이후로 다른 언어로의 번역은 별다른 제한 없이 개작의 범위에 포함되는 것으로 간주합니다). '피양도자'란 GPL의 규정에 따라 프로그램을 양도

받은 사람을 의미하고, ‘원(原)프로그램’이란 프로그램을 개작하거나 2차적 프로그램을 만들기 위해서 사용된 최초의 프로그램을 의미합니다.

본 허가서는 프로그램에 대한 복제와 개작 그리고 배포 행위에 대해서만 적용됩니다. 따라서 프로그램을 실행시키는 행위에 대한 제한은 없습니다. 프로그램의 결과물(output)에는, 그것이 프로그램을 실행시켜서 생성된 것인지 아닌지의 여부에 상관없이 결과물의 내용이 원프로그램으로부터 파생된 2차적 프로그램을 구성했을 때에 한해서 본 허가서의 규정들이 적용됩니다. 2차적 프로그램의 구성 여부는 2차적 프로그램 안에서의 원프로그램의 역할을 토대로 판단합니다.

제1조

적절한 저작권 표시와 프로그램에 대한 보증이 제공되지 않는다는 사실을 각각의 복제물에 명시하는 한, 피양도자는 프로그램의 원시 코드를 자신이 양도받은 상태 그대로 어떠한 매체를 통해서도 복제하고 배포할 수 있습니다. 복제와 배포가 이루어 질 때는 본 허가서와 프로그램에 대한 보증이 제공되지 않는다는 사실에 대해서 언급되었던 모든 내용을 그대로 유지시켜야 하며, 영문판 GPL을 함께 제공해야 합니다.

배포자는 복제물을 물리적으로 인도하는데 소요된 비용을 청구할 수 있으며, 선택 사항으로 독자적인 유료 보증을 설정할 수 있습니다.

제2조

피양도자는 자신이 양도받은 프로그램의 전부나 일부를 개작할 수 있으며, 이를 통해서 2차적 프로그램을 창작할 수 있습니다. 개작된 프로그램이나 창작된 2차적 프로그램은 다음의 사항들을 모두 만족시키는 조건에 한해서, 제1조의 규정에 따라 또다시 복제되고 배포될 수 있습니다.

제1항 파일을 개작할 때는 파일을 개작한 사실과 그 날짜를 파일 안에 명시해야 합니다.

제2항 배포하거나 공표하려는 저작물의 전부 또는 일부가 양도받은 프로그램으로부터 파생된 것이라면, 저작물 전체에 대한 사용 권리를 본 허가서의 규정에 따라 공중에게 무상으로 허용해야 합니다.

제3항 개작된 프로그램의 일반적인 실행 형태가 대화형 구조로 명령어를 읽어 들이는 방식을 취하고 있을 경우에는, 적절한 저작권 표시와 프로그램에 대한 보증이 제공되지 않는다는 사실(별도의 보증을 설정한 경우라면 해당 내용), 그리고 양도받은 프로그램을 본 규정에 따라 재배포할 수 있다는 사실과 GPL 사본을 참고할 수 있는 방법이 함께 포함된 문구가 프로그램이 대화형 구조로 평이하게 실행

된 직후에 화면 또는 지면으로 출력되도록 작성되어야 합니다. (예외 규정 : 양도 받은 프로그램이 대화형 구조를 갖추고 있다 하더라도 통상적인 실행 환경에서 전술한 사항들이 출력되지 않는 형태였을 경우에는 이를 개작한 프로그램 또한 관련 사항들을 출력시키지 않아도 무방합니다.)

위의 조항들은 개작된 프로그램 전체에 적용됩니다. 만약, 개작된 프로그램에 포함된 특정 부분이 원프로그램으로부터 파생된 것이 아닌 별도의 독립 저작물로 인정될 만한 상당한 이유가 있을 경우에는 해당 저작물의 개별적인 배포에는 본 허가서의 규정들이 적용되지 않습니다. 그러나 이러한 저작물이 2차적 프로그램의 일부로서 함께 배포된다면 개별적인 저작권과 배포 기준에 상관없이 저작물 모두에 본 허가서가 적용되어야 하며, 전체 저작물에 대한 사용 권리는 공중에게 무상으로 양도됩니다.

이러한 규정은 개별적인 저작물에 대한 저작자의 권리를 침해하거나 인정하지 않으려는 것이 아니라, 원프로그램으로부터 파생된 2차적 프로그램이나 수집 저작물의 배포를 일관적으로 규제할 수 있는 권리를 행사하기 위한 것입니다.

원프로그램이나 원프로그램으로부터 파생된 2차적 프로그램을 이들로부터 파생되지 않은 다른 저작물과 함께 단순히 저장하거나 배포할 목적으로 동일한 매체에 모아 놓은 집합물의 경우에는, 원프로그램으로부터 파생되지 않은 다른 저작물에는 본 허가서의 규정들이 적용되지 않습니다.

제3조

피양도자는 다음 중 하나의 항목을 만족시키는 조건에 한해서 제1조와 제2조의 규정에 따라 프로그램(또는 제2조에서 언급된 2차적 프로그램)을 목적 코드(object code)나 실행물(executable form)의 형태로 복제하고 배포할 수 있습니다.

제1항 목적 코드나 실행물에 상응하는 컴퓨터가 인식할 수 있는 완전한 원시 코드를 함께 제공해야 합니다. 원시 코드는 제1조와 제2조의 규정에 따라 배포될 수 있어야 하며, 소프트웨어의 교환을 위해서 일반적으로 사용되는 매체를 통해 제공되어야 합니다.

제2항 배포에 필요한 최소한의 비용만을 받고 목적 코드나 실행물에 상응하는 완전한 원시 코드를 배포하겠다는, 최소한 3년간 유효한 약정서를 함께 제공해야 합니다. 이 약정서는 약정서를 갖고 있는 어떠한 사람에 대해서도 유효해야 합니다. 원시 코드는 컴퓨터가 인식할 수 있는 형태여야 하고 제1조와 제2조의 규정에 따라 배포될 수 있어야 하며, 소프트웨어의 교환을 위해서 일반적으로 사용되는 매

체를 통해 제공되어야 합니다.

제3항 목적 코드나 실행물에 상응하는 원시 코드를 배포하겠다는 약정에 대해서 자신이 양도받은 정보를 함께 제공해야 합니다. (제3항은 위의 제2항에 따라 원시 코드를 배포하겠다는 약정을 프로그램의 목적 코드나 실행물과 함께 제공 받았고, 동시에 비상업적인 배포를 하고자 할 경우에 한해서만 허용됩니다.)

저작물에 대한 원시 코드란 해당 저작물을 개작하기에 적절한 형식을 의미합니다. 실행물에 대한 완전한 원시 코드란 실행물에 포함된 모든 모듈들의 원시 코드와 이와 관련된 인터페이스 정의 파일 모두, 그리고 실행물의 컴파일과 설치를 제어 하는데 사용된 스크립트 전부를 의미합니다. 그러나 특별한 예외의 하나로서, 실행물이 실행될 운영체제의 주요 부분(컴파일러나 커널 등)과 함께 (원시 코드나 바이너리의 형태로) 일반적으로 배포되는 구성 요소들은 이러한 구성 요소 자체가 실행물에 수반되지 않는 한 원시 코드의 배포 대상에서 제외되어도 무방합니다.

목적 코드나 실행물을 지정한 장소로부터 복제해 갈 수 있게 하는 방식으로 배포 할 경우, 동일한 장소로부터 원시 코드를 복제할 수 있는 동등한 접근 방법을 제공한다면 이는 원시 코드를 목적 코드와 함께 복제되도록 설정하지 않았다고 하더라도 원시 코드를 배포하는 것으로 간주됩니다.

제4조

본 허가서에 의해 명시적으로 이루어지지 않는 한 프로그램에 대한 복제와 개작 및 하위 허가권 설정과 배포가 성립될 수 없습니다. 이와 관련된 어떠한 행위도 무효이며 본 허가서가 보장한 권리는 자동으로 소멸됩니다. 그러나 본 허가서의 규정에 따라 프로그램의 복제물이나 권리를 양도받았던 제3자는 본 허가서의 규정들을 준수하는 한, 배포자의 권리 소멸에 관계없이 사용상의 권리를 계속해서 유지할 수 있습니다.

제5조

본 허가서는 서명이나 날인이 수반되는 형식을 갖고 있지 않기 때문에 피양도자가 본 허가서의 내용을 반드시 받아들여야 할 필요는 없습니다. 그러나 프로그램이나 프로그램에 기반한 2차적 프로그램에 대한 개작 및 배포를 허용하는 것은 본 허가서에 의해서만 가능합니다. 만약 본 허가서에 동의하지 않을 경우에는 이러한 행위들이 법률적으로 금지됩니다. 따라서 프로그램(또는 프로그램에 기반한 2차적 프로그램)을 개작하거나 배포하는 행위는 이에 따른 본 허가서의 내용에 동의한다는 것을 의미하며, 복제와 개작 및 배포에 관한 본 허가서의 조건과 규정들을 모두 받아들일겠다는 의미로 간주됩니다.

제6조

피양도자에 의해서 프로그램(또는 프로그램에 기반한 2차적 프로그램)이 반복적으로 재배포될 경우, 각 단계에서의 피양도자는 본 허가서의 규정에 따른 프로그램의 복제와 개작 및 배포에 대한 권리를 최초의 양도자로부터 양도받은 것으로 자동적으로 간주됩니다. 프로그램(또는 프로그램에 기반한 2차적 프로그램)을 배포할 때는 피양도자의 권리의 행사를 제한할 수 있는 어떠한 사항도 추가할 수 없습니다. 그러나 피양도자에게, 재배포가 일어날 시점에서의 제3의 피양도자에게 본 허가서를 준수하도록 강제할 책임은 부과되지 않습니다.

제7조

법원의 판결이나 특허권 침해에 대한 주장 또는 특허 문제에 국한되지 않은 그 밖의 이유들로 인해서 본 허가서의 규정에 배치되는 사항이 발생한다 하더라도 그러한 사항이 선행하거나 본 허가서의 조건과 규정들이 면제되는 것은 아닙니다. 따라서 법원의 명령이나 합의 등에 의해서 본 허가서에 위배되는 사항들이 발생한 상황이라도 양측 모두를 만족시킬 수 없다면 프로그램은 배포될 수 없습니다. 예를 들면, 특정한 특허 관련 허가가 프로그램의 복제물을 직접 또는 간접적인 방법으로 양도받은 임의의 제3자에게 해당 프로그램을 무상으로 재배포할 수 있게 허용하지 않는다면, 그러한 허가와 본 사용 허가를 동시에 만족시키면서 프로그램을 배포할 수 있는 방법은 없습니다.

본 조항은 특정한 상황에서 본 조항의 일부가 유효하지 않거나 적용될 수 없을 경우에도 본 조항의 나머지 부분들을 적용하기 위한 의도로 만들어 졌습니다. 따라서 그 이외의 상황에서는 본 조항을 전체적으로 적용하면 됩니다.

본 조항의 목적은 특허나 저작권 침해 등의 행위를 조장하거나 해당 권리를 인정하지 않으려는 것이 아니라, GPL을 통해서 구현되어 있는 자유 소프트웨어의 배포 체계를 통합적으로 보호하기 위한 것입니다. 많은 사람들이 배포 체계에 대한 신뢰 있는 지원을 계속해 줌으로써 소프트웨어의 다양한 분야에 많은 공헌을 해주었습니다. 소프트웨어를 어떠한 배포 체계로 배포할 것인가를 결정하는 것은 전적으로 저작자와 기증자들의 의지에 달려있는 것이지, 일반 사용자들이 강요할 수 있는 문제는 아닙니다.

본 조항은 본 허가서의 다른 조항들에서 무엇이 중요하게 고려되어야 하는지를 명확하게 설명하기 위한 목적으로 만들어진 것입니다.

제8조

특허나 저작권이 설정된 인터페이스로 인해서 특정 국가에서 프로그램의 배포와 사용이 함께 또는 개별적으로 제한되어 있는 경우, 본 사용 허가서를 프로그램에 적용한 최초의 저작권자는 문제가 발생하지 않는 국가에 한해서 프로그램을 배포한다는 배포상의 지역적 제한 조건을 명시적으로 설정할 수 있으며, 이러한 사항은 본 허가서의 일부로 간주됩니다.

제9조

자유 소프트웨어 재단은 때때로 본 사용 허가서의 개정판이나 신판을 공표할 수 있습니다. 새롭게 공표될 판은 당면한 문제나 현안을 처리하기 위해서 세부적인 내용에 차이가 발생할 수 있지만, 그 근본 정신에는 변함이 없을 것입니다. 각각의 판들은 판번호를 사용해서 구별됩니다. 특정한 판번호와 그 이후 판을 따른다는 사항이 명시된 프로그램에는 해당 판이나 그 이후에 발행된 어떠한 판을 선택해서 적용해도 무방하고, 판번호를 명시하고 있지 않은 경우에는 자유 소프트웨어 재단이 공표한 어떠한 판번호의 판을 적용해도 무방합니다.

제10조

프로그램의 일부를 본 허가서와 배포 기준이 다른 자유 프로그램과 함께 결합하고자 할 경우에는 해당 프로그램의 저작자로부터 서면 승인을 받아야 합니다. 자유 소프트웨어 재단이 저작권을 갖고 있는 소프트웨어의 경우에는 자유 소프트웨어 재단의 승인을 얻어야 합니다. 우리는 이러한 요청을 수락하기 위해서 때때로 예외 기준을 만들기도 합니다. 자유 소프트웨어 재단은 일반적으로 자유 소프트웨어의 2차적 저작물들을 모두 자유로운 상태로 유지시키려는 목적과 소프트웨어의 공유와 재활용을 증진시키려는 두 가지 목적을 기준으로 승인 여부를 결정할 것입니다.

제11조

본 허가서를 따르는 프로그램은 무상으로 양도되기 때문에 관련 법률이 허용하는 한도 내에서 어떠한 형태의 보증도 제공되지 않습니다. 프로그램의 저작권자와 배포자가 공동 또는 개별적으로 별도의 보증을 서면으로 제공할 때를 제외하면, 특정한 목적에 대한 프로그램의 적합성이나 상업성 여부에 대한 보증을 포함한 어떠한 형태의 보증도 명시적이나 묵시적으로 설정되지 않은 '있는 그대로의' 상태로서 프로그램을 배포합니다. 프로그램과 프로그램의 실행에 따라 발생할 수 있는 모든 위험은 피양도자에게 인수되며 이에 따른 보수 및 복구를 위한 제반 경비 또한 피양도자가 모두 부담해야 합니다.

제12조

저작권자나 배포자가 프로그램의 손상 가능성을 사전에 알고 있었다 하더라도 발생한 손실이 관련 법규에 의해 보호되고 있거나 이에 대한 별도의 서면 보증이 설정된 경우가 아니라면, 저작권자나 프로그램을 원래의 상태 또는 개작한 상태로 제공한 배포자는 프로그램의 사용이나 비작동으로 인해 발생한 손실이나 프로그램 자체의 손실에 대해 책임지지 않습니다. 이러한 면책 조건은 사용자나 제3자가 프로그램을 조작함으로써 발생한 손실이나 다른 소프트웨어와 프로그램을 함께 동작시키는 것으로 인해서 발생한 데이터의 상실 및 부정확한 산출 결과에만 국한되는 것이 아닙니다. 발생한 손실의 일반성이나 특수성 뿐 아니라 원인의 우발성 및 필연성도 전혀 고려되지 않습니다.

[새로운 프로그램에 GPL을 적용하는 방법]

생략