기술혁신학회지 제5권 제2호
2002년 7월 pp. 212-227

오픈소스 소프트웨어의 기술혁신 특성: 리뷰

The Innovation Characteristics of Open Source Software: A Review

송 위 진*

〈目 次〉

<Abstract>

This study reviews the institutional frameworks of Open Source software and develops the stylized facts of Open Source software innovation. Open Source software have to solve two difficult problems for encouraging innovation. First, the source code of Open Source software program should be open and freely distributed and it is very difficult for developers to appropriate the results of their investments. Second, as Open Source software development process is characterized by the participation of communities of developers, it is not easy to coordinate and manipulate the development process. These difficulties of developing Open Source Software have been solved by the particular incentive schemes and coordinating mechanisms. This study reviews the study on the motivation of Open Source software development and the mechanisms which coordinate innovation process of Open Source software with peer review and meritocracy, and how these characteristics promote innovation in Open Source software communities.

Key words : 오픈소스 소프트웨어, 리눅스, 기술혁신, 모듈라 시스템, 기술개발 동기

* 과학기술정책연구원 연구위원, songwc@stepi.re.kr

1. 머 리 말

오픈소스 소프트웨어는 소스코드가 개방된 기술이다. 오픈소스 소프트웨어는 제품에 대한 소유권에서 시작하여 기술개발방식까지 기존의 사적독점 소프트웨어(proprietary software)와 커다란 차이가 있다. Windows 98, XP 등과 같은 소프트웨어는 소유권이 마이크로소프트와 같은 특정 기업에 있고 그 소프트웨어를 사용하기 위해서는 라이센스 요금을 지불해야 하며, 소스코드는 기업의 비밀이기 때문에 공개되지 않은 사적으로 독점된 시스템이다. 그리고 소프트웨어의 개발 과정은 전적으로 그 소프트웨어를 소유한 회사에 의해 통제된다. 이에 반해 리눅스와 같은 오픈소스 소프트웨어는 공공의 소유이며 필요한 사람들은 누구나 '자유롭게' 개방된 소스코드를 사용하여 필요한 소프트웨어를 개발할 수 있을 뿐만 아니라 배포할 수도 있다.

이와 같은 특성으로 인해 오픈소스 소프트웨어는 기술혁신과 관련해서 독특한 모습을 보여주고 있다. 우선 오픈소스 소프트웨어의 기술혁신은 오픈소스 소프트웨어를 좋아하고 지지할 뿐만 아니라 그것을 자체적으로 개발하고 사용하는 '개발자 · 사용자 공동체'를 통해 이루어진다. 공급자가 기술개발 과정 전체를 통제하고 혁신을 주도하는 사적독점 소프트웨어와는 매우 다른 양상을 보이고 있는 것이다.

둘째, 오픈소스 소프트웨어는 소스코드를 공개해야 되기 때문에 그것을 개발하는 과정에 시간과 자금과 같은 사적 자원을 투입했다고 해서 그 결과물에 대한 專有(appropriation)를 통해 경제적인 이득을 얻기가 어렵다. 그럼에도 불구하고 수많은 개발자들과 IBM이나 HP와 같은 민간 기업들이 오픈소스 소프트웨어 개발 프로젝트에 적극적으로 참여하고 있다. 다시 말하면 오픈소스 소프트웨어 개발에는 전유체제가 제대로 작동하지 않음에도 불구하고 개발자들의 참여를 이끌어내는 독특한 인센티브 제도가 작동하고 있는 것이다.

셋째 오픈소스 소프트웨어 개발과정에는 프로젝트 전체를 기획 · 통제하는 조직 위계는 존재하지 않는다. 많은 개발자들이 자발적이고 자율적인 규율에 따라 소프트웨어 개발에 참여한다.

본 글의 목적은 일견 모순적으로 보이는 오픈소스 소프트웨어의 기술혁신 특성이 무엇이며 왜 그런 현상이 나타나는가에 대한 논의들을 기술혁신론(Innovation Studies)[1]의 관점에서 검토하는 데 있다. 이는 제도주의적 관점에서 기술혁신 주체들이 혁신활동을 수행하는 방식, 상호작용 하는 방식 등 기술혁신이 이루어지는 조직과 제도의 구조적 특성에 초점을 맞추어 기술혁신을 파악하는 것이다. 따라서 본 글에서는 오픈소스 소프트웨어가 개발되는 조직적.제도적 조건에 초점을 맞추어 논의를 전개할 것이다.[2]

1) 기술혁신론은 1980년대 후반에 등장한 논의로서 기존의 신고전파적 접근과는 다른 시각에서 기술혁신과 과학기술정책을 파악하고 있다(Coombs et al, 1997; Freeman and Soete, 1997; Tidd et al, 1997). 기술혁신론은 영국의 SPRU(Science Policy Research Unit)를 모태로 해서 이루어진 연구와 Nelson and Winter(1982)로부터 시작된 진화론적 경제학을 이론적 자산으로 하여, 조절이론 등과 같은 제도주의 경제학, 그리고 자원기반 전략경영론 등과 상호작용을 통해 발전해왔다. 기술혁신론에서는 기술혁신 성과는 기술혁신관련 제도 변수에 의해 결정된다고 보고 있다.

2) 이는 오픈소스 소프트웨어(또는 자유소프트웨어)가 가지고 있는 이념적 측면에 대한 논의보다는 오픈소스 소프트웨어가 개발되고 사용되는 실제적인 측면에 초점을 맞춘다는 것을 의미한다. 오픈소스 소프트웨어(자유소프트웨어)와 그것의 개발과정이 갖고 있는 이념적 의미를 다룬 논의들은 홍성태(2002), 홍성태.오병일외(2000)을 참조하라. 이러한 측면에서 본 글은 자유소프트웨어와 오픈소스 소프트웨어를 구분해서 접근할 때, 오픈소스 소프트웨어에 무게 중심을 둔 연구라고 할 수 있다. 자유소프트웨어에 중점을 두는 연구는 홍성태(2002)를 참조하라.

그 동안 오픈소스 소프트웨어에 대한 다양한 분석들이 이루어졌지만 오픈소스 소프트웨어를 지지하는 개발자들이 다소 낭만적인 방식으로 정리한 글들(Raymond, 1997; DiBona et al, 2000; 토발즈와 다이아몬드, 2001)이 주로 소개되었으며, 사회과학적 분석 특히 기술혁신론에서의 접근은 최근에서야 시작되고 있다(McKelvey, 2001; von Hippel 2002; 2001; von Hippel and von Krogh, 2002; Kogut and Metiu, 2001; Feller and Fitzerald, 2000; 2002).[3)]

이 글에서는 최근에 이루어지는 분석적인 연구들을 중심으로 오픈소스 소프트웨어의 특성과 기술혁신 동기와 과정, 정부 지원정책들을 둘러싼 쟁점과 논의들을 살펴볼 것이다. 이는 오픈소스 소프트웨어의 기술혁신을 본격적으로 분석하기 위한 정형화된 사실(stylized facts)들을 정리[4)]하는 작업이 될 것이다.

논의의 순서는 다음과 같다. 제2절에서는 오픈소스 소프트웨어의 정의와 특성을 살펴보고 제3절에서는 오픈소스 소프트웨어 개발에 참여하는 동기와 오픈소스 소프트웨어 개발의 조정에 관한 논의들을 살펴본다. 제4절에서는 오픈소스 소프트웨어에 대한 정책적 지원을 둘러싼 논의들을 정리한다.

2. 오픈소스 소프트웨어의 정의

1. 오픈소스 소프트웨어의 정의

오픈소스 소프트웨어는 라이센스 요금이 무료이면서 소스코드[5)]를 개방한 소프트웨어로서 누구나 자유롭게 사용·활용·개선할 수 있는 소프트웨어이다. 사적독점 소프트웨어는 소프트웨어 사용권한을 라이센스하면서 요금을 징수한다. 이 때 소스코드는 공개되지 않는다. 따라서 버그가 발견되거나 문제가 생기면 개발자나 회사에 연락해서 대책을 요청할 수는 있지만 사용자의 희망을 들어줄 것인가의 여부는 전적으로 개발자의 판단에 따른다(클리프 밀러, 2000: 71). 반면 오픈소스 소프트웨어는 누구라도 소스코드에 접근할 수 있고 사용자가 능력만 있다면 각종 버그의 수정은 물론이고 그것을 개선하여 기능을 추가할 수 있다. 누구나 그 소프트웨어의 개발에 참여할 수 있는 것이다. 따라서 오픈소스 소프트웨어는 프로그램을 복제하여 배포할 수 있는 권리, 소프트웨어의 소스코드에 접근할 수 있는 권리, 프로그램을 개선할 수 있는 권리들을 개발자 또는 사용자에게 보장한다. 이렇게 소스코드를 수천, 수만 명의 프로그래머, 사용자와 공유함으로써 오픈소스 소프트웨어는 자유롭게 버그를 해결하고, 성능을 강화하고, 자신의 목적에 코드를 수정·사용할 수 있도록 하여 공동체의 창조력과 협력을 이끌어낼 수 있다고 이야기되고 있다.

---

3) 오픈소스 소프트웨어의 기술개발과정들을 분석한 사회과학적 연구에 대해서는 http://opensource.mit.edu, http://opensource.ucc.ie가 매우 유용한 정보를 제공해주고 있다.

4) 정형화된 사실들을 정리하는 것은 진화론적 경제학 또는 기술혁신연구에서 연구의 초기 단계에 주로 사용하는 방법이다. 현실적합성을 갖는 이론들을 도출하기 위해 경험적으로 파악되는 규칙성을 우선 발견하고 그로부터 이론화 작업을 수행하는 것이다(Dosi, 1988).

5) 컴퓨터가 읽을 수 있는 언어와 인간이 읽을 수 있는 언어는 차이가 있다. 프로그래머는 사람이 읽을 수 있는 형식의 프로그래밍 언어로 소스코드를 작성한 다음, 그것을 컴퓨터가 읽을 수 있는 형식으로 변환하여 프로그램을 만든다. 소스코드는 사람이 읽을 수 있는 형식의 프로그램 코드를 말하며 실행코드 또는 바이너리 코드는 컴퓨터가 인식하고 실행할 수 있는 형식의 코드를 지칭한다. 소스코드를 컴퓨터가 읽을 수 있도록 바이너리 코드로 변화하는 것을 컴파일(compile)이라고 하는데 컴파일러(compiler)는 그것을 위한 프로그래밍 툴이다.

오픈소스 소프트웨어는 상업용 소프트웨어와 비상업용 소프트웨어가 있다.[6] 비상업용 오픈소스 소프트웨어는 소프트웨어를 사용하는 데 비용이 들지 않는 오픈소스 소프트웨어이다. 상업용 오픈소스 소프트웨어는 사용자가 그 소프트웨어를 사용하는 데 돈을 지불해야하는 소프트웨어이다. 래드햇의 리눅스, 한컴 리눅스의 리눅스용 한글 등이 상업용 오픈소스 소프트웨어라고 할 수 있는데, 이 때 사용자가 지불하는 비용은 사용권한에 대한 라이센스 요금이 아니라 관련 소프트웨어를 묶어서 쉽게 사용할 수 있도록 한 서비스에 대한 비용이라고 할 수 있다.[7]

이렇게 다른 유형의 소프트웨어와 대비시켜 오픈소스 소프트웨어를 정의할 수는 있지만 여전히 모호한 점이 있다. 그리하여 좀 더 엄밀한 의미에서 오픈소스 소프트웨어를 정의하면 "오픈소스 소프트웨어는 '오픈소스 소프트웨어 정의(Open Source Software Definition)'[8]에 순응하는 조건으로 배포되는 소프트웨어를 지칭한다(Feller and Fitzgerald, 2002; 12)". 이 오픈소스 소프트웨어 정의는 단순히 라이센싱 관련 사항만을 정의해 놓은 것이 아니라 소프트웨어의 배포기준도 명시하고 있다.

한편 오픈소스 소프트웨어의 정의에 부응하는 라

〈표 1〉 소프트웨어의 유형

		소스코드가 공개	
		예	아니오
사용자가 지불하는 소프트웨어 관련 비용	무료	비상업용 오픈소스 소프트웨어	프리웨어 쉐어웨어
	유료	상업용 오픈소스 소프트웨어	사적독점 상업용 소프트웨어

자료: International Institute of Infonomics and Berlecon Research(2002), Part III, p.11
주 : Shareware는 도입 초기에는 무료이나 일정기간이 지나면 유료로 라이센스 받아야한다. 소스코드는 활용할 수 없다. Freeware의 경우 라이센스료가 없다. 그러나 이것은 제대로 활용하기 위한 보완제품의 경우에는 라이센스료가 추가되는 경우가 많다. 소스코드는 활용할 수 없다.

6) 자유/오픈소스 소프트웨어 운동이 분화되면서 오픈소스 소프트웨어를 상업용과 비상업용으로 구분해서 보는 관점이 등장했다. 스톨만(R. Stallman)의 주도해왔던 자유소프트웨어 운동과는 입장을 달리하는 Open Source Initiative라는 조직이 2001년 레이몬드(E. Raymond)를 중심으로 구축되면서, 공개/오픈소스 소프트웨어의 이념적 측면보다는 그것이 개발되는 방식과 실용성에 초점을 맞추는 움직임들이 구체화되기 시작했다. 이들은 오픈소스 소프트웨어는 산업계는 물론이고 개발자에게도 경제적인 보상을 제공할 수 있어야 한다고 주장하고 있다. 이들의 관점에서 본다면 오픈소스 소프트웨어는 상업용으로 활용될 수 있기 때문에 상업용 소프트웨어와 오픈소스 소프트웨어를 대립시키는 방식은 지나치게 이데올로기적이다. 오픈소스 소프트웨어는 상업용 일수도 있고 비상업용 일 수도 있다.

7) 이는 리눅스와 관련하여 제일 먼저 등장한 비즈니스 모델이다. 사용자들이 리눅스를 사용하기 위해서는 리눅스 커널이외에 리눅스에서 동작하는 다양한 프로그램을 필요로 한다. 전문 지식을 가진 사용자는 필요한 응용프로그램을 모아서 목적에 맞게 시스템을 구성할 수 있지만 일반 사용자들에게는 이것은 어려운 작업이다. 이 때문에 몇몇 기업들이 리눅스 커널과 함께 필수적인 툴이나 유틸리티, 애플리케이션을 하나의 패키지로 포장해서 일반 사용자들에게 배포하기 시작했는데 이것이 바로 '배포판(distribution)'이라고 불리는 것이다. 물론 이들 제품들은 인터넷을 통하여 무료로 다운로드받을 수도 있으며 유료로 구입한 경우에는 서비스 지원을 받을 수 있도록 되어 있다. 오픈소스 소프트웨어 진영은 이러한 접근을 확장시켜 이제 소프트웨어 산업은 패키지 소프트웨어와 같은 일상용품(commodity)을 파는 산업이 아니라 서비스를 파는 산업으로 구조전환되고 있다고 주장하고 있다.

8) http://www.opensource.org/osd.html에 오픈소스 정의가 게재되어 있다. 여기에는 배포의 조건, 2차적 저작물에 대한 기준, 소스코드의 수정과 관련된 제한, 차별금지 조항, 라이센스의 적용 등에 대한 규정들이 다루어지고 있다.

이센스들은 여러 가지가 존재한다. GPL(GNU Public Licence), LGPL(Library GNU Public Licence), BSD 라이센스(Berkely Software Development Licence), X 컨소시움 라이센스, Artistic 라이센스, MPL(Mozilla Public Licence) 등이 그것들이다. 이들은 모두 오픈소스 소프트웨어의 정의를 만족시키고 있지만 상용제품에의 사용여부나 재배포 조건에서 다른 입장을 취하고 있기도 하다.

2. 오픈소스 소프트웨어의 활용

오픈소스 소프트웨어 제품들은 주로 개발도구, 하부구조형 소프트웨어, 네트워킹 유틸리티들이다(<표 2> 참조). 이들 제품들은 주로 성능과 신뢰성이 중요시되는 제품들이다. 또한 오픈소스 소프트웨어는 주로 인터넷, 웹과 관련이 있는 소프트웨어들이다. 인터넷을 가능하게 했던 TCP/IP 프로토콜이나 DNS, 전자우편 유틸리티들은 전부 오픈소스 소프트웨어라고 할 수 있다.

이들 분야에서 오픈소스 소프트웨어는 사적독점 소프트웨어에 대해 우위를 확보할 것으로 전망되고 있다. 영국정부의 오픈소스 소프트웨어 지원정책의 기본적인 관점을 정리한 Peeling and Satchell(2001)의 보고서에 따르면 오픈소스 소프트웨어는 하부구조형 소프트웨어[9] 시장에서 근본적인 변화를 추동하는 역

〈표 2〉 오픈소스 소프트웨어 제품 사례

제품	특성
Linux	운영체제
Apache	웹 서버
Sendmail	인터넷 메일 유틸리티
Bind	웹에서 Domain Name Server를 운영하는 데 사용되는 소프트웨어, Berkeley Internet Name Daemon
Perl, Python	프로그래밍 언어
GNU software	다양한 컴파일러, 유틸리티, EMACS editor
GNOME, KDE	Linux GUI 인터페이스
Mozilla	Netscape Navigator의 오픈소스 소프트웨어 버전
Jikes	IBM이 개발한 Java 컴파일러
GIMP	Image Workshop
Darwin	Mac OSX 서버 시스템
SAMBA	마이크로소프트의 SMB 프로토콜과 통합을 허용
Ghostscript	Aladdin Enterprises PDF 유틸리티
Doom	게임

자료: Feller and Fitzgerald(2000)

9) 이 보고서에서는 소프트웨어를 하부구조 소프트웨어(Software Infrastructure)와 응용소프트웨어(Software Application)로 구분하고 있다. 하부구조 소프트웨어는 운영체제, 데이터베이스, 웹서버, 그리고 응용소프트웨어를 구동하는 데 필요한 주요 소프트웨어를 지칭한다. 응용소프트웨어는 워드 프로세서나 스프레드 쉬트, 프레젠테이션 툴, 프로젝트 관리 툴 등 조직을 운영하는 데 필요한 여러 소프트웨어를 지칭한다.

할을 하고 있으며 5년 이내에 하부구조형 소프트웨어 시장의 50% 정도를 차지할 것으로 전망하고 있다.

3. 오픈소스 소프트웨어의 기술혁신 특성

1. 기술개발의 동기

오픈소스 소프트웨어 개발의 동기 문제는 전통적인 관점에서 보았을 때 모순을 담고 있는 현상으로서 사회과학자들의 관심의 대상이 되고 있다. 오픈소스 소프트웨어 개발은 사적 자원의 투자(개인적 시간과 노력, 기업의 투자)를 통해 공공재(오픈소스 소프트웨어)를 생산해내는 매우 독특한 특성을 지니고 있기 때문에, 기존의 사적 투자에 기초한 기술개발모델이나 시장실패에 따른 공공적 투자에 기초한 기술개발 모델로는 그 동학을 제대로 설명하기 어려운 새로운 현상이다(von Hippel and von Krogh, 2002). 일반적으로 투자 결과물을 전유할 수 없는 경우 사적투자는 이루어지기 어렵다. 이런 경우 시장실패가 나타난다고 하여 공공자원의 투자가 이루어지는 것이 일반적인 현상인데, 오픈소스 소프트웨어의 경우에는 많은 경우 개인과 기업들의 사적투자가 나타나고 있는 것이다.[10)]

그렇다면 개인적으로 보상이 크지 않은 공공재적 성격을 지니고 있는 오픈소스 소프트웨어를 왜 능력있는 개발자들이 많은 시간과 비용을 들여 개발하는가 더 나아가 기업들은 왜 직접적인 수익에 도움이 되지 않음에도 불구하고 막대한 비용을 투입하여 오픈소스 소프트웨어를 개발하는가?

Feller and Fitzerald(2002)는 그 동안 오픈소스 소프트웨어의 개발동기에 관한 연구들을 체계적으로 정리하여 동기를 기술적 동기, 경제적 동기, 사회.정치적 동기로 구분하고 분석의 수준을 미시수준(개인수준)과 거시수준(조직.커뮤니티 수준)으로 분류하여 개발동기들을 살펴보고 있다.[11)]

<표 3>에서도 알 수 있듯이 개인 개발자나 민간 기업, 공동체 구성원들이 오픈소스 소프트웨어 개발에 참여하는 이유는 매우 다양하고 복합적이다. 개인 차원에서는 개인의 기술적 문제를 해결하기 위해서부터, 기술학습, 명성의 획득[12)], 이타주의까지 여러 가지 동기요인들이 있으며, 기업 차원에서 소프트웨어 개발의 생산성 제고, 안전성의 확보, 브랜드 가치의 제고까지 여러 가지 요인들이 지적되고 있다. 공동체 차원에서는 개발도상국의 소프트웨어 개발지원이나

10) IBM의 경우 다양한 형태의 오픈소스 소프트웨어 개발에 적극적으로 참여하고 있다. IBM은 리눅스기술센터를 설립하여 70여개 정도의 리눅스 관련 프로젝트에 관여하고 있으며, 리눅스 관련 소프트웨어 개발에 10억불 정도를 투자했다고 이야기되고 있다(International Institute of Infonomics and Berlecon Research, 2002; Part II, 9-12). HP나 SUN Microsystems도 매우 적극적으로 다양한 오픈소스 소프트웨어 개발활동에 참여하고 있다.

11) 여기서 Feller and Fitzerald(2002)의 논의를 중심으로 기술개발 동기를 살펴보는 이유는 이들이 기존에 여러 연구들에서 언급되었던 동기요인들을 거의 망라적으로 정리했기 때문이다.

12) 명성의 획득은 자존심을 충족시켜줄 뿐만 아니라 개발자에게 경제적인 이득을 가져다 준다. 합리적 주체를 논의의 출발점으로 삼고 있는 경제학자들은 명성획득의 경제적 측면을 강조하고 있다. 그들에 따르면 유사한 활동을 수행하고 있는 공동체로부터 훌륭한 개발자라는 평판을 얻는 것은 그 후 개발자가 기업체에 자리를 잡거나 다른 작업을 수행할 때 그에게 더 많은 급여와 프로젝트 예산을 획득할 수 있게 하는 자산이 된다. 소프트웨어 개발자에게는 암묵적인 인센티브(implicit incentive)가 되는 것이다. 또 오픈소스 소프트웨어를 개발하는 공동체에서 훌륭한 소프트웨어를 개발하고 배포하는 것이 그 사람이 가지고 있는 지적 자산을 외부에 알리는 방안이 될 수 있다는 것이다. 소프트웨어를 개발한 사람의 능력과 활동에 대한 정보를 확실하게 알려주는(signaling) 것이 되는 것이다(Learner and Tirole, 2000).

〈표 3〉 오픈소스 소프트웨어의 개발동기

	미시수준(개인적 동기)	거시수준(조직/커뮤니티 수준 동기)
기술적 동기	· 개인적 기술적 필요성을 충족시키기 위해 · 동료평가(Peer Review)시스템을 활용하여 소프트웨어 개발의 효율성을 높이기 위해 · 최첨단의 기술을 가지고 작업하기 위해 · 사용자들 및 고급 소프트웨어 개발자와의 상호작용을 통한 기술학습 기회 확보(사용자 니즈의 파악 및 사사(mentorship)를 통한 학습)	· 개발비용은 높아지고 질은 떨어지는 소프트웨어의 위기에 대응하기 위해(오픈소스 소프트웨어 개발 방식의 도입을 통해 적은 비용으로 신속하게 높은 질의 소프트웨어 개발) · 단순하고 지루한 개발작업(테스트나 문서화)들을 사용자들과 나눠서 수행하기 위해 · 오픈소스 소프트웨어 커뮤니티를 통한 기업 연구개발활동의 보완 및 강화(예: Netscape의 Mozilla) · 기업의 혁신을 촉진하기 위해(IBM의 오픈소스 소프트웨어 개발) · 소스코드의 공개를 통해 소프트웨어 개발 및 활용의 투명성을 확인하기 위해(NASA나 국가안보기구 등에서 오픈소스 소프트웨어를 선호하는 이유)
경제적 동기	· 경력관리에서 이익을 얻기 위해(오픈소스 소프트웨어 개발 경험을 경력관리에 적절히 활용) · 코딩 스킬을 향상시키기 위해 · 스톡옵션 등을 통해 부를 얻기 위해 (오픈소스 소프트웨어 개발 및 배포회사에서 스톡옵션의 획득) · 낮은 기회비용(잃을 것이 별로 없음)	· 오픈소스 소프트웨어를 가지고 투자자의 열정을 이끌어내기 위해 · 일상용품(commodity) 중심의 소프트웨어 산업을 소비자 지향적 서비스 산업으로 패러다임 전환을 촉진하기 위해 · 기업의 브랜드 가치를 높이기 위해 · 관련 제품과 서비스, 악세사리 등을 판매해 간접적인 수익을 얻기 위해 · 개발도상국에 소프트웨어를 공급하기 위해 · 사적독점 소프트웨어에 비해 저렴한 가격을 통해 비용절감
사회 · 정치적 동기	· 자기 성취욕구(소프트웨어 기획 · 개발에서 기업 경영진이 아니라 개발자 자신의 직접적인 통제를 통해 성취욕구의 달성) · 자신의 능력을 알리기 위해(개발사례를 제시함으로써 인적자원가치에 대한 정보제공) · 코딩자체에 대한 동기 · 커뮤니티에의 소속감을 얻기 위해 · 이타주의	· 사회운동(특히 마이크로소프트에 대항하는) · 디지털 격차의 극복 (소프트웨어는 누구에게나 자유로워야 한다) · 미래의 작업방식 모델 개발 (소프트웨어의 영역을 넘어서는 새로운 작업방식으로서 오픈소스 소프트웨어 개발방식)

자료: Feller and Fitzerald(2002), 138-139 수정 · 보완

반독점 사회운동 등과 같은 요인들이 지적되고 있다.

한편 이들 요인들 중에서 어떤 요인들이 주요인이고 어떤 요인들이 부요인인지, 그리고 각 요인들은 어떻게 연관되어 있는지, 분석의 수준이나 개발자의 여러 조건에 따라 어떤 요인들이 중요한지는 아직 명확하게 해명되지 않은 상태이다. 그리고 각 국가의

조건에 따라 어떤 요인들이 더 중요한지에 대한 연구들도 이제 시작단계라고 할 수 있다. 각 요인들의 중요성을 검증하기 위한 실증연구도 최근에 와서야 이루어지고 있다.

2002년 상반기에 이루어진 조사결과(International Institute of Infonomics and Berlecon Research(2002: Part IV, 45))에 따르면 개발자들이 오픈소스 소프트웨어 개발에 참여하고 활동하는 가장 중요한 동기는 새로운 소프트웨어 관련 스킬을 학습하고 발전시키며, 그 스킬들을 공유하기 위한 것이었다. 이것은 오픈소스 소프트웨어 개발활동에 참여하는 것이 기술학습을 수행하는 데 매우 유용하다는 것을 보여주는 것이다. 한편 돈은 벌기 위해 오픈소스 소프트웨어 커뮤니티에 참여하고 활동한다는 항목은 적은 비율을 보이고 있다. 따라서 직접적인 금전적 동기에 따라 오픈소스 기술개발에 참여하는 경우는 그리 높지 않다고 할 수 있다.

주요 소프트웨어 기업들이 오픈소스 기술개발에 참여하는 이유는 표준화의 촉진, 저비용 컴포넌트로서 오픈소스 소프트웨어 사용 등이 지적되고 있다(International Institute of Infonomics and Berlecon Research, 2002: Part II). 많은 기업들이 오픈소스 소프트웨어 특히 리눅스 개발활동에 참여하는 주된 이유중의 하나는 과거 UNIX에서 나타났던 分枝(forking) 문제를 해결하기 위해서라고 이야기하고 있다.[13] 오픈소스 소프트웨어는 특정 업체에 운영체제 소프트웨어에 대한 통제권을 주지 않으면서도 개방된 표준 운영체제를 제공하여 분지문제를 해결할 수 있기 때문에 주요 소프트웨어 관련 기업들이 오픈소스 소프트웨어 개발에 참여하고 있다는 것이다. 개방표준의 채택을 통해 경쟁관계에 있는 기업들이 공통의 운영체제 하부구조를 마련하고 있는 것이다.

또 주요 소프트웨어 기업들은 하드웨어와 소프트웨어를 결합하여 시스템을 구축할 때 발생하는 비용을 줄이기 위해서 오픈소스 소프트웨어 특히 리눅스를 사용하는 것으로 조사되었다. 운영체제 그 자체는 이제 제품을 차별화하는 수단이 되지 못하고 있으며 사용자 요구사항에 맞게 하드웨어와 소프트웨어를 잘 통합하여 적은 비용으로 공급하는 서비스 능력이 경쟁우위의 원천이 되고 있기 때문이다.

2. 기술개발과정의 조정

오픈소스 소프트웨어 개발이 이루어지는 방식은 매우 다양하다. 여러 개발자들과 조직들은 각기 다른 툴과 방법을 사용하고 있다. 그러나 전체적인 측면에서 오픈소스 소프트웨어 개발 및 그것이 조정되는 방식에는 일반적인 특성들이 있다(Feller and Fitzerald, 2002: ch. 6). 여기서는 이 일반적인 특성을 살펴보기로 한다.

전체적으로 오픈소스 소프트웨어는 기술적인 측면에서는 모듈라 시스템적 아키텍처에 의해 조정되고, 조직적인 측면에서는 동료평가와 업적주의에 입각해서 공동체로부터 신뢰를 받고 있는 인물들이나 집단

13) UNIX가 상용 소프트웨어로 본격적으로 사용되면서 하드웨어 공급업체들이 자신들의 제품에 부합되는 UNIX를 개발하여 다양한 유형의 UNIX가 존재하게 되었다. 다양한 UNIX의 존재는 하드웨어 공급업체, 사용자, 소프트웨어 공급업체 모두에게 상당한 부담을 주었다. 하드웨어 공급업체들은 운영체제를 업데이트하기 위해 계속해서 많은 비용을 소모해야 했으며 그러한 활동들 자체가 경쟁우위의 확보에 별로 도움이 되지 못했다. 소프트웨어 공급업체의 경우 계속해서 늘어나는 하드웨어 공급과 여러 운영체제를 통합시키는 작업에 상당한 비용을 지불해야 했다. 이러한 활동들은 결국 사용자들의 비용지출을 높이는 결과를 낳았다. 이로 인해 단일의 운영체제를 사용하는 것이 모두에게 도움이 되는 상황이 만들어졌다.

의 권위에 근거해 조정되고 있다고 볼 수 있다.

1) 모듈라 시스템과 병행적 개발

운영체제와 같은 하부구조형 소프트웨어는 다양한 구성요소들로 복잡하게 구성되어 있다. 따라서 시스템을 개발하여 활용하기 위해서는 복잡성을 관리하는 방안이 필요하다. 소프트웨어 공학에서 이야기하는 '브룩스의 법칙(Brooks' Law)'에 따르면 프로그래머가 n명 늘게되면 할 수 있는 일의 양이 n배 증가하지만 일의 복잡성과 버그 출현 가능성은 n의 제곱에 비례해서 늘어난다. 이 법칙에 따르면 수많은 사람들이 참여하는 프로젝트는 분열되기 쉽고, 불안정한 혼란 상황에 처하는 경우가 많게 된다. 따라서 상당수의 사람이 참여하는 소프트웨어 개발이 원활히 조정되기 위해서는 프로젝트를 규율하는 독특한 방식이 필요하다. 오픈소스 소프트웨어 개발의 복잡성을 관리하기 위한 방안의 하나로서 제시된 것이 시스템의 구성요소들을 모듈로 만들어 전체 시스템을 모듈라 시스템으로 만드는 것이다.

Schilling(2000)은 모듈라 시스템을 컴포넌트들이 느슨하게 결합되어있는 시스템이라고 정의하고 있다. 이 시스템에서는 표준적이고 개방된 인터페이스에 따라 구성요소들이 느슨하게 결합되어 있다. 각 구성요소들은 독립적인 기능을 수행한다. 또 각 구성요소들이 수행하는 기능 및 인터페이스에 대한 기술명세들이 기술개발에 참여하는 행위자들이 준수해야 하는 규격으로 제시된다. 이로 인해 구성요소들간의 관계가 명확해지면서 전체 시스템이 단순화된다.[14] 이런 특성을 지니고 있는 모듈라 시스템의 경우 각 구성요소의 기술개발을 담당하는 개인이나 집단, 조직은 전체 시스템이나 다른 구성요소에 대한 충분한 기술적 지식이 없이도 기술을 개발할 수 있다. 전체 시스템 차원에서 규정되는 기능과 요건을 만족하는 구성요소들을 만들기만 하면 되기 때문이다. 이러한 방식을 통해 매우 복잡한 소프트웨어 개발과정도 좀 더 덜 복잡해질 수 있다.

오픈소스 소프트웨어의 핵심적인 특성으로서 이야기되는 병행적 개발방식(parallel development)은 오픈소스 소프트웨어들이 모듈라 시스템이기에 가능하다. 각 개발자들은 인터페이스의 규정을 지키면서 전체 시스템의 부분들을 동시에 병행적으로 개발할 수 있다. 개발자 A는 '가' 부분을 개발하고 개발자 B는 '나' 부분을 동시에 개발할 수 있는 것이다. 결국 소프트웨어 개발속도는 빨라지게 된다.

또한 병행적 개발방식에서는 '가'부분을 개발자 A와 개발자 B가 병렬적으로 개발할 수 있다. 이는 중복적인 작업이 될 수도 있지만 다수의 개발자들이 존재하고 기술개발에 몰입하는 상황에서는 상대적으로 더 우수한 요소들을 공급하는 토대가 된다. 병행적 개발방식을 통해 다양성이 높아지게 되면 성능이 더 좋은 소프트웨어 모듈이 개발될 수 있다. 그리고 인터넷을 통해 재빠르게 의사소통이 이루어지고 동료평가에 의해 개발된 모듈에 대한 엄격한 평가가 이루어지기 때문에 다양성이 증대해도 선택이 이루어지는 과정에는 그렇게 많은 비용이 필요하지 않다. 따라서 다양성이 낮을 수밖에 없는 기업 내에서의 사적 독점 소프트웨어개발과정 보다 질과 수준이 높은 소프트웨어가 개발될 수 있다.

14) 이 때문에 시스템을 분해하여 기존의 컴포넌트를 새로운 컴포넌트 --- 공개된 인터페이스 기준을 준수하는 ---로 대체해도 시스템의 원활한 작동에 무리가 생기지 않는다. 반면 '강하게 결합된 시스템(integrated system, tightly coupled system)'의 경우 컴포넌트를 다른 컴포넌트로 바꾸어서 시스템을 구성하면 시스템이 제대로 작동하지 않거나, 기존의 컴포넌트들을 새롭게 대체한 컴포넌트에 맞게 재구성해야만 시스템이 기능할 수 있게 된다.

2) 동료평가 방식과 능력주의, 조정

오픈소스 소프트웨어 개발과정에서는 개발된 기술에 대한 독립적인 동료평가(peer review)가 이루어진다. 일반적으로 기업조직에서 동료평가가 이루어질 때, 조직 내에 형성되어 있는 다양한 정치적.사회적 지형에 의해 독립적인 동료평가가 어려운 경우가 많다. 그러나 오픈소스 소프트웨어 커뮤니티의 경우 對面的 관계가 아니라 인터넷을 통해 상호작용을 하기 때문에 복잡한 정치적.사회적 관계에 이끌리는 경우는 상대적으로 적다. 독립적인 평가가 이루어질 수 있는 조건이 형성되어 있는 것이다.

이렇게 독립적인 동료평가가 이루어지기 때문에 오픈소스 소프트웨어 커뮤니티에서는 능력주의가 강력하게 자리잡고 있다. 그 사람의 나이나 성별, 소속이 무엇이든 소프트웨어 개발능력이 뛰어나고 오픈소스 소프트웨어 공동체의 대의에 동의한다면 그 개발자는 높은 평가를 받게 된다. 그리고 높은 평판을 지닌 개발자들의 존재는 오픈소스 소프트웨어 개발을 조정하는 토대가 된다.

오픈소스 소프트웨어 개발에는 프로젝트 전체를 기획 · 통제하고 예산을 배분하며, 마케팅에 신경을 쓰는 회사 경영진이나 특정 의무를 수행하도록 공권력을 행사하는 국가는 존재하지 않는다. 많은 개발자들이 분산된 형태로 자율적인 방식에 따라 소프트웨어를 개발하는 것이다. 그러나 이렇게 분산적.자율적으로 소프트웨어 개발이 이루어질 경우 경쟁자나 경쟁집단이 형성되어 소프트웨어 개발에 분열이 생길 수 있다. 같은 뿌리에 서있지만 코드 기반이 전혀 다르고 호환성이 없는 제품이 개발될 수 있는 것이다. 리눅스의 경우에는 이러한 일이 나타나지 않고 있다. 그것은 리눅스 커널 개발에서 사용되는 방식 때문이다. 리눅스 커널의 경우 리누스 토발즈, 앨런 콕스(Allan Cox) 등과 같이 높은 명성을 지니고 있으며 공동체 구성원들로 권위를 인정받고 있는 인물들이 갈등의 소지가 있는 문제를 해결하는 리더쉽을 발휘해 왔으며, 계속해서 그 리더쉽을 유지하고 있다. 공동체로부터 능력과 도덕성에서 정당성을 획득한 고수들이 전체적인 방향을 제시하고 분쟁 발생시 그것을 해결하는 공공재를 공급함으로써 복잡한 기술개발과정이 조정되는 것이다. 그렇지만 이와 같은 정당성을 획득한 인물이나 그룹이 없는 경우 다른 집합적 의사결정 방식을 취할 수밖에 없다.

한편 동료평가가 이루어지기 위해서는 공통의 지식기반(shared knowledge base)이 오픈소스 소프트웨어 커뮤니티에 존재하고 있어야 한다. 이 공통의 지식기반은 신속한 의사소통과 다양한 모듈의 결합을 가능하게 하는 핵심적 요소이다. 이것이 뒷받침되지 않을 때에는 아무리 인터넷을 통해 개발자들이 연결되어 있어도 밀도 있는 의사소통과 지식과 정보의 이전이 제한된다. 이들은 배경지식(contextual knowledge)으로서 정보.지식의 창출.전달.해석의 기본적인 토대가 되기 때문이다. 그렇다면 이와 같은 공통의 배경지식기반은 어떻게 구축되었을까? 이 배경지식은 암묵적인 성격을 지니고 있기 때문에 대면적인 접촉 없이 쉽게 획득하기 어려운 특성을 지니고 있고 메일링리스트나 웹사이트와 같은 매체를 통해서 학습하기도 어렵다.

Kogut and Meitu(2001)은 리눅스가 널리 받아들여지고 빨리 진화할 수 있었던 요인으로 많은 개발자들이 이미 UNIX 소프트웨어에 익숙했던 사실을 들고 있다. UNIX와 리눅스의 기본 구조가 유사하기 때문에 개발자들간에 공통의 지식기반이 이미 형성되어 있었고 이것이 신속한 정보의 확산과 공유를 가능하

게 했다는 것이다. 과거에 UNIX를 학습하고 활용하면서 가지게 된 암묵적 지식과 경험이 리눅스를 개발하는 데에 배경적 지식으로 활용되었다는 것이다.[15)]

조사결과에 따르면 개발자들간의 차이는 있지만 오픈소스 소프트웨어 공동체에 소속된 사람들은 공개된 소프트웨어의 기능과 내용을 평가할 수 있는 지식을 가지고 있다고 볼 수 있다. International Institute of Infonomics and Berlecon Research(2002: Part IV)에 따르면 오픈소스 소프트웨어 개발에 참여하고 있는 상당수의 개발자들은 전문적인 개발자들이다. 전체 개발자 중에서 소프트웨어 엔지니어가 전체의 33.3%, 프로그래머가 11.2%, IT 컨설턴트가 9.8%이고 IT관련 학생들이 15.8%를 차지하고 있다. 이들은 전문가로서 관련 분야의 지식을 공유하고 있다고 할 수 있다.

4. 오픈소스 소프트웨어 지원정책

1. 정부개입의 근거

오픈소스 소프트웨어에 대한 정부개입의 근거는 크게 산업 발전이라는 측면과 소프트웨어 사용이라는 두 가지 측면에서 살펴볼 수 있다. 정부는 산업발전과 공정경쟁을 위한 틀을 형성하는 역할과 그 자신이 공공업무 수행을 위해 소프트웨어를 사용하는 수요자의 역할을 하고 있기 때문에 이 두 영역에 개입할 수 있다.

우선 소프트웨어 산업 발전의 차원에서 정부개입에 대한 논의를 살펴보기로 하자(<표 4> 참조). 정부가 소프트웨어 산업발전과 관련해서 오픈소스 소프트웨어를 지원해야 하는 이유로서 1) 오픈소스 소프트웨어가 사적독점 소프트웨어(proprietary software)보다 혁신적이고 이 때문에 소프트웨어 시장에 경쟁을 촉진시키며 2) 후발국의 경우에 소프트웨어 산업의 혁신능력을 향상시켜 선진국을 추격할 수 있는 계기를 마련해줄 수 있다는 것이 지적되고 있다(Working Group on Libre Software, 2000).

이들 논의에 따르면 오픈소스 소프트웨어는 사용자가 동시에 개발자가 되는 경우가 많기 때문에 요구사항을 적절히 충족시킬 수 있으며, 다양한 인력들이 참여하고 또 소스코드를 공유함으로써 매우 빠른 시간에 소프트웨어를 개발하고 버그를 제거할 수 있는 잠재력을 가지고 있다. 그리고 이러한 특성 때문에 현재 특정 대기업을 중심으로 독과점적 구조를 형성하고 있는 소프트웨어 시장(특히 운영체제나 하부구조적 성격을 갖는 소프트웨어 시장)에 경쟁을 촉진할 수 있다. 또한 후발국의 경우에는 제품의 설계도라고 할 수 있는 소스코드가 공개되어 있기 때문에 그것을 활용할 수 있으며, 또 선진국 소프트웨어 개발자와 공동작업을 통해 소프트웨어 개발 능력을 향상시킬 수 있기 때문에 선진국과의 소프트웨어 개발 능력 격차를 좁힐 수 있는 기회를 마련할 수 있다는 것이다.

다음에는 수요자의 측면에서 정부의 개입 근거를 살펴보기로 한다. 공공부문은 상당액수의 소프트웨어를 구매한다. 이 때 오픈소스 소프트웨어를 구입하는 것이 공공성을 확보하는 데 큰 도움이 된다. 왜냐하

15) 이러한 측면에서 이미 사용되고 있는 사적독점 소프트웨어와 유사한 소프트웨어를 개발하는 오픈소스 소프트웨어 진영의 전략은 기존에 사적독점 소프트웨어를 사용하면서 축적된 배경적 지식을 효과적으로 활용할 수 있는 전략이라고 할 수 있다. 사적독점 소프트웨어에 의해 지배되어 있는 상황을 역으로 활용하는 전략인 것이다.

면 오픈소스 소프트웨어를 사용하면 1) 호환성의 확보와 특정제품에의 고착(lock-in)방지 2) 비용 절감 3) 보안성 확보 4) 투명성 확보가 가능하기 때문이다 (International Institute of Infonomics and Berlecon Research(2002: Conclusion).

사적독점 소프트웨어는 폐쇄적이기 때문에 그것이 표준이 되면 성능이나 신뢰성이 더 뛰어남에도 불구하고 그것과 호환되지 않은 다른 제품을 사용할 수 없게된다. 이에 반해 오픈소스 소프트웨어는 소스코드가 개방되어 있기 때문에 쉽게 호환성을 확보할 수 있어 다른 제품을 용이하게 활용할 수 있다. 그리고 소스코드가 공개되어 있기 때문에 다수의 공급업체가 경쟁하게 되어 시스템구축 비용을 절감할 수 있다. 또 소스코드가 공개되어 있기 때문에 보안문제가 발생했을 때 그것을 재빨리 확인하고 전파하여 (직접적인 사용자만이 아니라 공동체의 도움을 얻어) 문제에 대한 대안을 조기에 확보할 수 있다. 이에 반해 사적독점 소프트웨어는 제품이 암흑상자이기 때문에 백도어가 있는지, 보안과 관련된 버그가 있는지를 확인하기가 어렵다. 따라서 보안문제가 발생했을 때 오픈소스 소프트웨어의 대응이 훨씬 신속할 수 있다.

그리고 오픈소스 소프트웨어는 투명성이 높기 때문에 공공부문에 적합한 소프트웨어이다. 민주주의 국가에서 시민들은 공공분야 정보에 대한 접근 권리만이 아니라 공공정보가 처리되는 과정(예: 전자투표, 전자과세가 이루어지는 과정)에 접근할 수 있는 권리가 있다. 오픈소스 소프트웨어는 소스코드가 공개되어 있기 때문에 시민들의 이와 같은 정보처리과정과 정보에 대한 접근 권리를 실제로 보장해줄 수 있다.

그러나 이러한 지원 논리에 대해 반대가 없는 것은 아니다. 사적독점 소프트웨어를 지지하는 기업이나 단체들은 정부개입에 대해 오픈소스 소프트웨어에 대한 차별적인 지원이라고 반박하면서, 지원의 근거가 타당하지 않으며 이 때문에 정부가 소프트웨어 산업발전을 왜곡할 수 있다고 주장하고 있다(Evans and Reddy, 2002).

이들의 주장에 따르면 소프트웨어 시장은 독과점적 시장이 아니며 이미 충분한 경쟁이 이루어지고 있다. 따라서 오픈소스 소프트웨어에 대한 지원은 이미 많은 실패를 낳은 '승리자를 선발하는(picking the winner)' 방식의 정책이고 결국에는 시장의 왜곡을 가져올 것이다. 후발국의 오픈소스 소프트웨어에 대한 지원도 마찬가지의 결과를 가져올 것이며 결국 선진국이든 후발국이든 소프트웨어 산업을 자유로운 시장의 작동에 맡기라고 주장하고 있다. 또한 수요측면에서도 그것이 사적독점 소프트웨어이든 오픈소스

〈표 4〉 오픈소스 소프트웨어에 대한 정부개입의 근거

분 야	오픈소스 소프트웨어에 대한 정부개입의 근거
소프트웨어 산업발전	ㅇ 오픈소스 소프트웨어의 확산을 통해 독과점적인 소프트웨어 시장의 경쟁 촉진 ㅇ 소프트웨어 산업의 혁신능력을 향상시켜 선진국 추격 계기 마련.
수요자의 공공구매	ㅇ 호환성의 확보와 특정기업에의 고착(lock-in)방지 ㅇ 경쟁을 통한 비용 절감 ㅇ 소스코드 공개를 통한 우월한 보안성 확보 ㅇ 시민들의 정보처리과정에 대한 접근권 보장

소프트웨어이든 많은 사람들이 특정 소프트웨어를 사용한다면 그것이 실제적인 표준이 되어 호환성을 확보할 수 있으며 충분한 효용을 누릴 수 있다고 주장하고 있다. 그리고 오픈소스 소프트웨어의 경우 사용자들의 교육 · 훈련이나 시스템 관리에 숨겨져 있는 비용들이 있으며, 오픈소스 소프트웨어를 사용하는 경우 발생하는 운영상, 보안상 문제점을 책임질 수 있는 주체가 불명확하기 때문에 사적독점 소프트웨어보다 우월할 수 없다고 이야기하고 있다. 오픈소스 소프트웨어에 대한 정부의 개입을 주장하는 논의들이 오픈소스 소프트웨어와 시장지배적인 사적독점 소프트웨어의 '출발점을 같게 하기(levelling the playing field)'를 주장한다면, 이들 논의들은 현재의 상황은 합리적인 시장진화의 결과이며 자원의 효과적으로 분배되고 있는 상황이기 때문에 정부의 개입이 필요 없다고 주장하고 있는 것이다.

2. 오픈소스 소프트웨어 지원정책의 유형

오픈소스 소프트웨어 지원정책들은 크게 오픈소스 소프트웨어의 공급을 촉진하는 정책(기술공급정책)과 오픈소스 소프트웨어의 수요를 촉진하는 정책(기술채택정책)으로 분류해서 접근할 수 있다(Mowery, 1995). 전자가 소프트웨어 산업 발전이라는 정부개입 논리에 근거하고 있다면 후자는 수요자 측면의 정부개입 논리에 기반하고 있다.

지원정책들을 구체적으로 살펴보면[16] 공급정책의 경우에는 우선 공공부문의 자금을 통해 직접적으로 오픈소스 소프트웨어를 개발하는 프로젝트를 추진하

〈표 5〉 오픈소스 소프트웨어 지원정책의 유형

공개 소프트웨어 공급정책	ㅇ 오픈소스 소프트웨어를 개발하기 위한 기술개발 프로젝트의 추진 - 유럽연합 프레임워크 프로그램의 오픈소스 소프트웨어 기술개발 프로젝트(http://www.cordis.lu/ist/ka4/tesss/impl_free.htm) - 독일의 오픈소스 소프트웨어 기반 보안소프트웨어 개발 프로젝트 ㅇ 정부연구개발사업이나 공공부문 사업을 통해 개발된 소프트웨어의 공개화 촉진 - 유럽연합의 IDA(interchange of Data between Administrations) 프로그램에서의 소프트웨어 공유 타당성 평가 프로젝트(http://europa.eu.int/ISPO/ida) ㅇ 오픈소스 소프트웨어 관련 수요자의 니즈나 관련 정보, 소프트웨어 라이브러리를 제공해주는 정보거래소(Clearinghouse)를 설립 - 독일의 BerliOS(Der Open-Source-Mediator)(http://www.berlios.de/index.php.en) ㅇ 오픈소스 소프트웨어 개발을 지원할 수 있는 하드웨어 및 플랫폼 지원 ㅇ 오픈소스 소프트웨어의 개발과 사용과 관련된 교육.훈련지원
공개 소프트웨어 수요정책	ㅇ 정부 소프트웨어 구매시 사적독점 소프트웨어와 동등한 위치에서 오픈소스 소프트웨어에 대한 고려 - 영국의 Open Source Software: Use within UK Government ㅇ 정부구매하는 소프트웨어에 대해 최대한의 공개화 추진 ㅇ 정보통신 표준 설정시 오픈소스 소프트웨어와 친화성이 있는 개방 표준(open standard)의 우대

16) 각 국의 오픈소스 소프트웨어 지원정책에 대한 정보는 STEPI 사이버연구회의 오픈소스 소프트웨어 연구회에서 살펴볼 수 있다(www.stepi.re.kr).

거나 공공부문이 그 동안의 개발사업을 통해 획득한 소프트웨어들을 공개화 하는 정책들을 지적할 수 있다. 물론 개발자나 기업들의 오픈소스 소프트웨어 개발을 촉진하기 위해 하부구조를 조성해주는 작업들도 중요한 정책의 하나이다. 기술채택정책의 경우에는 그 동안 불이익을 받아왔던 오픈소스 소프트웨어에 대해 경쟁적 환경을 조성하여 정부구매를 촉진하는 정책을 들 수 있다. 그리고 더 나아가 정부가 구매하는 소프트웨어의 경우 소스코드에 대한 권한을 최대한도로 확보하여 특정 소프트웨어에 대한 고착을 탈피하고 정보통신시스템의 유연한 진화를 유지해나갈 수 있도록 하는 정책도 이야기 할 수 있다.

5. 맺 음 말

리누스 토발즈가 리눅스라는 소프트웨어와 함께 리눅스 개발 방법론 또는 오픈소스 개발방법론을 개발했다는 레이몬드의 주장은 핵심을 지적한 것이다 (Raymond, 1997). 리눅스 개발과정에서 오픈소스 소프트웨어만이 아니라 오픈소스 소프트웨어가 (재)생산될 수 있는 제도적 틀 또는 조직과정이 개발되었기 때문이다. 더구나 그 방법론은 상당히 성공적인 결과를 가져왔다. 오픈소스 소프트웨어는 소스코드를 개방하고 자유롭게 사용할 수 있다는 독특한 제품 특성과 함께 사적독점 제품과 경쟁할 수 있는 새로운 기술개발방식을 창조해냈다. 사적독점 소프트웨어 개발과는 구분되는 이와 같은 인센티브 시스템, 기술혁신의 조정 방식 등은 제도적 혁신이라고 할 수 있다.

이 글에서는 이러한 제도적 혁신을 살펴보기 위한 배경 지식과 함께, 새로운 제도에 대한 논의들을 간략하게 살펴보았다. 오픈소스 소프트웨어의 등장과 발전이 비교적 최근의 일이고 사적독점 소프트웨어들이 지배하고 있는 상황에서 전개되는 새로운 현상이기 때문에 오픈소스 소프트웨어의 기술혁신에 대한 연구는 이제 시작하고 있다고 볼 수 있다. 다음에서는 이러한 새로운 제도적 틀이 갖는 의미를 숙고할 수 있는 몇 가지 연구주제들을 지적하면서 글을 마무리하고자 한다.

우선 오픈소스 기술개발과정과 그 제도적 틀을 미시적으로 분석하여 조직관리 일반에 대한 시사점을 도출하는 연구들이다. 이미 이러한 연구들은 여러 주제들을 대상으로 이루어지고 있다. Markus et al (2000)은 오픈소스 소프트웨어 개발과정에서 활용되는 방식을 이용하여 지식노동과 가상조직의 관리에 대한 교훈을 도출하고 있으며 Moon and Sproull (2000)은 컴퓨터를 통한 분산 작업의 관리방식에 대한 시사점을 얻으려고 하고 있다. 오픈소스 기술개발방식과 최근에 나타나고 있는 새로운 조직적 현상들을 연계시키는 논의들이 전개되고 있는 것이다.[17]

또한 오픈소스 소프트웨어 기술개발과정의 제도적 틀은 과학기술과 같은 전문적인 분야에서의 시민참여 방안 탐색과 관련해서 의미 있는 준거점을 제공해 줄 수 있다. 과학기술부문에서의 시민참여는 과학기술지식의 전문성으로 인해 시민이 참여하기도 어렵고 참여할 수도 없는 것으로 파악되어 왔지만 최근

17) 한편 오픈소스 소프트웨어 기술개발 방식은 Merton(1942)에 의해서 제시된 과학자들의 과학활동 방식 ---- 물론 그것이 과연 진정한 과학자들의 과학활동 방식이냐에 대해서는 문제가 제기될 수 있다 ----과 유사한 성격도 지니고 있다. 공동체적인 특성, 지식의 공유, 금전적 이익보다는 명성 획득의 선호 등 과학적 지식의 형성과정에서 나타나는 양상은 오픈소스 소프트웨어의 그것과 상당히 비슷하다. 전통적인 과학발전 모델과 오픈소스 기술개발과정을 비교해보는 것도 이러한 측면에서 여러 가지 의미가 있을 것이다.

전문가와 비전문가의 경계를 낮추면서 시민참여의 의미와 방법을 논구하고 실천하는 활동(합의회의, 시나리오 워크샵, 시민배심원제 등)들이 활성화되고 있다(참여연대시민과학센터, 2002). 이들은 주로 과학기술 지식이나 인공물의 기획단계에서 어떻게 시민사회의 요구를 반영하고 리스크를 줄일 수 있을까에 초점을 맞춘 제도적 틀이라고 할 수 있다. 따라서 집행 및 평가분야인 실제적인 기술개발에서의 참여방식은 상대적으로 소홀히 다루어지고 있다. 오픈소스 소프트웨어 개발방식은 '사용자를 위해, 사용자에 의해 혁신이 이루어지는 수평적인 혁신 네트워크(horizontal innovation network: by and for users)'로서(von Hippel, 2001), 기술의 기획에서 사용.활용.평가까지 지속적으로 사용자들의 참여[18]가 보장되고 촉진되는 특성을 지니고 있다. 이러한 특성으로부터 도출된 제도적 틀들은 과학기술부문에서의 시민참여를 집행 · 평가부문까지 확장하는 데 실마리를 제공할 수 있을 것이다. 또한 인터넷과 웹을 통해 사용자들의 참여가 이루어지는 기제도 과학기술부문에서의 시민참여의 기술적 방식과 관련해서 여러 시사점을 제시해 줄 수 있을 것이다.

참 고 문 헌

리누스 토발즈.데이비드 다이아몬드(2001), 『Just for Fun(국역: 리눅스 그냥 재미로)』, 한겨레신문사.

참여연대시민과학센터(2002), 『과학기술.환경.시민참여』, 한울아카데미.

클리프 밀러(2000), 『리눅스비즈니스 .COM』, 세종서적.

홍성태(2002), "운영체계의 사회화와 정보공유운동", 《경제와 사회》, 제54호(2002년 여름호).

홍성태.오병일 외(2000), 『디지털은 자유다: 인터넷과 지적재산권의 충돌』, 이후.

Coombs, R., Saviotti, P. and Walsh, V.(1992), *Economics and Technological Change*, Rowman & Littlefield.

DiBona, C., Ockman, S. and Stone, M.(2000), 『오픈소스』, 한빛미디어.

Dosi, G. (1988), "Sources, Procedures, and Microeconomic Effects of Innovation" *Journal of Economic Literature*, Vol.26, 1120-1171.

Evans, K. and Reddy, B.(2002), "Government Preferences for Promoting Open Source Software: a Solution in Search of a Problem", National Economic Research Associates, http://opensource.mit.edu/online_papers.php.

Feller, J. and Fitzgerald, B. (2002), *Understanding Open Source Software Development*, Addison Wesley, London.

Feller, J. and Fitzgerald, B. (2000), "A Framework Analysis of the Open Source Software Development Paradigm", Proceedings of the 21st International Conference in Information Systems. http://afis.ucc.ie/jfeller/publications/ICIS2000.pdf.

Freeman C. and Soete, L. (1997), *The Economics of Industrial Innovation*, 3rd edition, The MIT Press.

International Institute of Infonomics and Berlecon Research(2002), *FLOSS(Free/Libre and Open Source Software: Survey and Study)*, FINAL REPORT, www.infonomics.nl/floss.

18) 오픈소스 소프트웨어 개발과정에서 사용자들의 참여가 소프트웨어의 개발만을 의미하는 것은 아니다. 도큐멘테이션이나 사용 경험을 알려주는 것도 중요한 참여활동의 하나이다(클리프 밀러, 2000: 제4장).

Kogut, B. and Metiu, A.(2001), "Distributed Knowledge and the Global Organization of Software Development", Working Paper, http://opensource.mit.edu/online_papers.php.

Lerner, J. and Tirole, J.(2000), "The Simple Economics of Open Source", NBER Working Paper 7600.

Marus, L., Manville, B. and Agres, C.(2000), "What Makes an Virtual Organization Work?", *Sloan Management Review*, Fall, 2000.

Mckelvey, M.(2001), "The Economic Dynamics of Software: Three Competing Business Models Exemplified through MIicrosoft, Netscape and Linux", *Economics of Innovation and New Technologies*, Vol. 10, 196-236.

Merton, R.(1942), "The Normative Structure of Science", in *The Sociology of Science*, Chicago Univ. Press.

Moon, J. and Sproull, L.(2000), "Essence of Distributed Work: The Case of Linux Kernel", *First Monday*, Vol. 5. No.11. http://firstmonday.org/issues/issue5_11/moon/index.html.

Mowery, D. (1995), "The Practice of Technology Policy", in Stoneman, P.(ed.), *Handbook of the Economics of Innovation and Technological Change*, Blackwell, Oxford.

Nelson, R. and Winter, S. (1982), *An Evolutionary Theory of Economic Change*, Harvard University Press, Cambridge, MA.

Raymond, E.(1997), "The Cathedral and Bazaar". http://www.tuxedo.org/~esr/writings/cathedral-bazaar/cathedral-bazaar.html.

Schilling, M.(2000), "Toward a General Modular System Theory and its Application to Inter-firm Product Modularity", *Academy of Management Review*, Vol. 25, 312-334.

Tidd, J., Bessant, J. and Pavitt, K.(1997), *Managing Innovation: Integrating Technological, Market and Organizational Change*, John Wiley & Sons.

von Hippel, E. amd von Krogh, G.(2002), "Exploring the Open Source Software Phenomenon: Issues for Organization Science", http://opensource.mit.edu/online_papers.php.

von Hippel, E.(2001), "Open Source Shows the Way: Innovation By and For Users. No Manufacturer Required", http://opensource.mit.edu/online_papers.php.

von Hippel, E.(2002), "Horizontal innovation networks- by and for users", MIT Working Paper, http://opensource.mit.edu/online_papers.php.

Working Group on Libre Software(2000), "Free Software/Open Source: Information Society Opportunities for Europe", Ver. 1.2. http://eu.conecta.it.