2

공개 소스 소프트웨어 프로젝트의 생명 주기와 품질 유지 방안

한성대학교 ▌이 민 석*

1. 서 론

공개 소스 활성화는 수요와 공급 측면, 두 방향이 있다. 수요 측면은 수요에 맞는 품질을 만족하는 공개 소스 소프트웨어가 지원이 가능한 형태로 존재하고, 수요 기관에서 경제적 이득이 있다고 판단하는 경우, 시장 논리에 의거하여 활성화되며, 일부 국가들의 경우처럼 국가적으로 공개 소스 소프트웨어의 수요를 촉진하는 정책을 채택하기도 한다[1]. 반면에 공급 측면에서는 수요가 담보되지 않은 개발 지원 정책들의 효과가 기대에 미치지 못한 경우가 많았다. 이는 공개 소스 소프트웨어 개발, 또는 관련 활동의 동기가 상용 소프트웨어와는 다르기 때문이다.

'Just for Fun'이라는 Linus Tovalds의 저서[2] 제목이 의미하듯이, 그리고 1330명의 소프트웨어 개발자를 대상으로 한 Luthiger의 연구[3]에 따르면 개발자들이 자신의 여유 시간을 이용하여, 여러 의미에서의 '재미'로, 또 141명의 리눅스 커널 공헌자들을 대상으로 한 Hertel 등의 연구[4]에 따르면 주로 '명성'을 얻기 위해, 공개 소스 관련 활동을 하는 것으로 조사되었다. 이는 우리에게 몇 가지 시사점을 제공하는데, 우선 상대적으로 여유 시간이 거의 없는 한국의 소프트웨어 개발자들에게는 공개 소스 활동이 쉽지 않다는 점이며, 다음은 경제적 이득이라는 가장 강력한 동기를 가진 상용 소프트웨어에 비하여 공개 소스 소프트웨어가 수요자, 또는 수요 기관이 요구하는 수준의 품질을 가진 소프트웨어를 원하는 시간에 공급하는 것이 쉽지 않다는 점이다. 따라서 우리나라의 소프트웨어 엔지니어들이 처한 상황에서, 많은 수요자가 원하는 공개 소스 소프트웨어 만들었다는 '명성'을 얻기 위해서는, 기획, 분석, 설계, 구현, 시험, 배포, 유지 등 프로젝트의 전 생명 주기 동안 높은 생산성을 유지할 수 있는 방법을 동원하고, 상용 소프트웨어를 능가하는 품질을 유지해야할 필요가 있다.

이 논문에서는 공개 소스 소프트웨어 프로젝트의 전 생명 주기에 이르는 활동 단계에 소프트웨어 개발자, 사용자들이 어떤 결정을 내려야 하며, 또 어떤 방식으로 참여하고 기여할 수 있는지를 살펴본다. 또 가장 영향력 있는 공개 소스 소프트웨어 프로젝트 개발자 사이트인 SourceForge.net[5]에 등록된 소프트웨어들에 대한 조사를 통하여 성공적인 공개 소스 소프트웨어 프로젝트들이 가진 특징들을 알아본다. 또 공개 소스 소프트웨어 프로젝트를 시작하는 중요한 동기 중에 하나인 '명성'을 얻기 위하여, 즉, 개발된 소프트웨어가 널리 이롭게 이용되도록 하기 위하여, 프로젝트가 가져야 할 품질 관련 요소들에 대하여 기술한다.

2. 공개 소스 소프트웨어 프로젝트의 생명 주기

공개 소스 소프트웨어 프로젝트 역시, 소프트웨어의 개발이라는 관점에서는 기존의 많은 상용 소프트웨어와 유사한 생명 주기를 가지게 된다. 하지만 공개 소스 소프트웨어 프로젝트에서는 '공개'라는 단어가 함축하듯이 여러 명의 개발자가 참여하는 분산 개발, 기존에 공개되어 있는 많은 소프트웨어 자원의 이용, 다양한 부류의 자원자들에 의한 소프트웨어 리뷰 및 시험 과정, 기술 지원 방법, 기능의 확장, 새로운 프로젝트로의 따른 가지치기 과정 등이 상용 소프트웨어와 달리 매우 중요한 의미를 가지게 된다. 그림 1은 공개 소스 소프트웨어 프로젝트의 생명 주기를 포괄하고 있다. 새로운 프로젝트의 경우, 비교적 폐쇄적인 초기 개발 단계를 거쳐, 공개된 뒤에, 커뮤니티와 호흡하는 공개 소스 소프트웨어 순환 구조(그림 1의 OSS LOOP, 즉 그림 2)에 들어간다. 일단 공개 소스 소프트웨어 순환 구조에 들어가면, 프로젝트 관리자들뿐만 아니라 커뮤니티의 모든 참여자들이 공개된 소스에 접근 가능하며, 기능 추가, 버그 리포트 및 수정, 새로운 기능의 요구 등을 함으로써 지속적인 소

* 종신회원

† 본 연구는 2007년도 한성대학교 교내연구비 지원과제임.

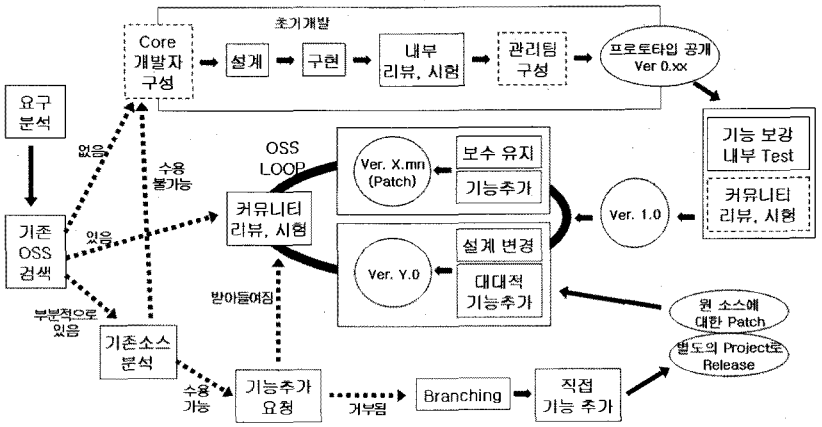

그림 1 공개 소스 소프트웨어 프로젝트의 생명 주기

프트웨어의 개선이 그 안에서 이루어지게 된다.

이 장에서는 공개 소스 소프트웨어 프로젝트의 각 단계를 프로젝트를 시작하는 사람, 개발자, 참여자의 관점에 구체적으로 살펴보고자 한다.

2.1 요구 분석 및 기존 소프트웨어 검색

어떤 소프트웨어를 개발하고자 하는 사람(그룹)은 우선 개발하고자 하는 소프트웨어가 가져야할 기능에 대한 충분한 분석을 한 뒤, 이미 존재하는 공개 소스 소프트웨어 프로젝트들 가운데 주어진 요구 사항을 만족하는 것이 있는지 확인한다. 이 과정에서 요구사항을 모두 충족하는, 또는 충족을 목표로 하는 프로젝트를 성공적으로 발견했다면, 그 사람은 아마도 발견된 프로젝트의 사용자 또는 적극적인 역할을 하는 자원자, 더 나아가 개발자로 공개 소스 소프트웨어 순환 구조에 참여하게 된다. 혹시 부분적으로 요구 사항을 만족하는 프로젝트가 발견된 경우에도 기능 추가를 요구하고, 그 요구가 프로젝트 관리자 그룹에서 받아들여짐으로써 순환 구조에 참여가 가능하다. 실제로 30% 가량의 개발자가 다른 개발자의 결과물을 개선하기 위해 기존 프로젝트 커뮤니티에 참여하는 것으로 조사되고 있다[7].

반면에, 상용 소프트웨어의 경우에는 주어진 요구사항을 만족하는 타 소프트웨어가 이미 시장에 존재하는 경우, 경쟁력(기능, 가격 등) 분석, 자사 관련 제품 라인업, 장기적 제품 로드맵 등을 바탕으로 한 경영적 판단을 거쳐 프로젝트의 진행 여부, 또는 해당 소프트웨어 업체와의 연합, 더 나아가서는 인수, 합병 등을 결정한다.

2.2 프로젝트 가지치기

위 2.1의 검색 단계에서 유사 프로젝트가 발견되었지만, 기존 프로젝트 관리자 그룹에서 새로운 기능 추가 요청을 받아들이지 않은 경우에 개발자는 두 가지의 선택을 할 수 있다. 첫 번째는 검색 단계에서 요구 사항을 만족하는 프로젝트가 없었던 경우와 같이, 전혀 새로운 프로젝트를 시작하는 것이고, 두 번째 선택은 직접 기존 프로젝트에 기능을 추가하는 방법이다. 이를 프로젝트 가지치기(Branching)라 하며, 공개 소스 소프트웨어의 경우, 원 소스를 바탕으로 수정되거나, 추가된 소스를 모두 공개한다면, 소스의 사용과 배포가 자유롭기 때문에 아무런 법률적 문제없이[1] 이런 결정을 내릴 수 있다. 이런 과정으로 개발된 결과는 원 소스에 대한 패치 형태로 원 소스의 개정에 따라가는 형식으로 배포되거나, 원 소스의 특정 버전을 기점으로 하는 새로운 프로젝트로 발전하며, 독자적인 공개 소스 소프트웨어 순환 구조를 성공적으로 구성하기도 한다.

프로젝트 가지치기에는 이전 프로젝트의 결과물을 매우 확고한 프로토타입으로 사용할 수 있다는 점을 제외하고는 새로운 프로젝트의 시작에 버금가는 준비와 여러 가지 선택이 필요하며, 프로젝트 가지치기를 한 개발자에게는 프로젝트를 성공적으로 유지해야 하는 묵시적인 책임도 따른다.

2.3 새로운 프로젝트의 시작

새로운 공개 소스 소프트웨어 프로젝트의 시작은 요구 사항을 만족하는 기존 프로젝트를 발견하지 못한 경우나 기존의 유사 프로젝트에 새로운 기능에 대한 수용 요구가 받아들여지지 않은 경우 등에 내리는 최후의 선택이라고 볼 수 있다. 그 밖에 자주 발생하는 새 프로젝트 시작 요인으로는 개인적 취향에 따른 것인데, 기존 프로젝트에 사용된 개발 언어가 마음에 들

1) 실제 공개 소스 소프트웨어들은 다양한 라이선스 정책을 가지고 있지만, 거의 대부분은 이 가지치기가 문제 되지 않는다.

지 않아 자신이 순환 구조에의 참여가 어렵다고 느끼는 경우, 프로젝트 결과물의 설계 구조에 전혀 동의할 수 없는 경우, 마지막으로는 기존 프로젝트의 핵심 개발자, 관리자 그룹을 개인적으로 선호하지 않는 경우 등이 있다[7].

일단 여러 동기에 의하여, 새로운 프로젝트가 시작되면, 상용 소프트웨어의 개발과 유사한 과정을 거치게 된다. 이 과정은 같은 동기와 목표 의식을 가진 핵심 개발자들로 개발팀을 구성하고, 요구 분석을 더욱 견고하게 한 뒤, 각종 위험 요인 분석, 일정 만들기 등의 절차적인 작업들로 시작된다. 그 가운데 위험 요인 분석에는, 이 새로운 프로젝트가 기존 프로젝트들에 비하여 경쟁력을 가질 수 있는지, 개발과 추후 관리를 위한 충분한 자원자를 확보할 수 있는지, 개발에 필요한 장비가 확보 가능한지 등을 포함한다. 많은 프로젝트의 경우, 표준적인 PC들로 개발 환경을 어렵지 않게 구축할 수 있으며, 소프트웨어 개발 도구 역시 공개 소스 소프트웨어를 사용하기 때문에 큰 비용을 유발하지 않으며, 여러 개발자들의 분산 개발을 지원하기 위한, 버전 관리 시스템(소스 저장소), 메일링 리스트 등도 개인 PC를 이용하여 구축할 수 있다. 최근에는 다수의 공개 소스 프로젝트들의 결과물이 마이크로소프트의 윈도우즈 계열 운영체제를 위해서 개발되고 있다. cygwin과 같은 운영체제 적응 계층을 바탕으로 하는 경우에는 별다른 문제가 없지만, 윈도우즈 운영체제를 직접 지원하는 경우, 도구의 문제가 따른다. 마이크로소프트도 무료로 사용할 수 있는 개발 도구 버전(Express Editions)도 있지만, 일부 기능의 제한이 있기 때문에, 윈도우즈를 지원하는 공개 소스 프로젝트의 경우, 개발 도구 비용을 감당할 수 없는 개발자들의 커뮤니티 참여가 제한되는 경우가 발생할 수 있다.

프로젝트의 시작 단계에서부터 소스의 관리, 버그의 관리, 개발자들 간의 원활한 의사소통을 위하여 SourceForge.net과 같은 공개 소스 소프트웨어 개발자 사이트를 이용할 수 있지만, 많은 초기의 프로젝트의 경우, 프로젝트의 시작 동기, 요구 사항, 설계 등이 기술된 공식적 문서의 부족이나, 다운로드가 가능한 소프트웨어 릴리즈가 없다는 이유로, 개발자 지원 사이트에 등록된다 해도, 커뮤니티 형성 등의 파급 효과를 기대하기는 어렵다.

2.4 프로토타입의 구현

개발된 소프트웨어가 커뮤니티의 관심을 끌기 위한 최소한의 작업은 고품질의 프로토타입을 완성하는 일이다. 프로토타입의 개발은 비교적 소수의 폐쇄적인 핵심 개발자 그룹을 중심으로 이루어진다. 따라서 개발자들 사이의 의견 교환 및 의사 결정을 위한 시스템의 존재 여부는 크게 문제가 되지 않지만, 프로토타입 구현이 진행되면서, 최초의 요구 사항이 일부 수정되고 그 결과가 설계의 변경을 필요로 하는 경우도 많아, 구성원 사이의 의사소통 방법과 최소한의 문서화는 반드시 필요하다.

많은 공개 소스 소프트웨어 프로젝트에서는 상용의 대형 소프트웨어 개발 방식에서 사용되는 소프트웨어 공학적 개발 방법론이 사용되지 않으며, 설계 방식과 참여한 개발자들의 취향에 따른 개발 방식이 사용된다. 하지만, 공개 소스 개발의 가장 중요한 특징인 분산 개발을 효과적으로 수행하고, 소스 코드의 재사용 가능성을 높이기 위하여 모듈화, 계층화된 소프트웨어 설계를 하는 것이 일반적이다. 설계는 이전에 있던 유사 프로젝트의 설계를 바탕으로 이루어지기도 하며, 더 많은 경우는, 개발자들의 혁신 의지에 따라, 새로운 설계를 추구한다.

프로토타입 구현의 완성도와 설계 특징들은, 프로젝트의 동기와 목표에 수긍하는 여러 수준의 참여자들을 커뮤니티에 끌어들이고, 그들의 적극적인 피드백을 유도하는 중요한 원동력이다. 따라서 프로토타입은 기본적인 기능 요구를 만족하며, 안정적인 동작을 해야 하며, 단순 명료한 소프트웨어 구조를 유지하는 것이 바람직하다. 또한 기능적인 부분을 포함하여 개선할 여지도 있어야 자원 개발자들의 참여 동기를 유발할 것이다.

프로토타입의 배포 이전에 반드시 거쳐야하는 단계는 내부 시험이다. 이 단계는 커뮤니티 참여자를 끌어들이기 위하여, 최초로 공개되는 소프트웨어가 그럴듯하게 보여야할 뿐만 아니라 높은 수준의 안정성도 확보되어야 한다는 관점에서 매우 중요하다. 대개의 프로젝트에서 특별한 시험 도구나 정교한 방법론은 잘 사용되지 않으며, 최근에는 가상 머신을 이용하여, 다양한 시스템 환경에서의 시험을 이전 보다는 더 용이하게 할 수 있다. 주로 모듈 단위로 설계, 구현되는 소프트웨어 구조 때문에 공개 소스 프로젝트에서는 기존의 라이브러리들을 적극적으로 활용하게 되며, 많은 경우, 각 모듈 또한 라이브러리 형식으로 개발된다. 개발된 소프트웨어를 배포할 때, 커널, 컴파일러, 라이브러리의 버전, 각 배포판의 미묘한 차이 등에 의해서 이식성 문제가 발생한다. 이식성 문제는 아무리 사소하다 하더라도 공개 소스 개발 환경에 익숙하지 않은 개발자들에게는 넘기 어려운 중대한 진입 장벽으

로 동작한다. 프로토타입 개발자들은, 공개 소스 프로젝트로서 성공적인 정착을 위해 배포 전에 이식성 문제의 해결에 많은 노력을 기울여야 하며, 다행히 GNU의 autoconf, make 등 표준 도구의 사용과 정교한 스크립트의 작성으로 상당히 문제를 완화시킬 수 있다.

2.5 배포

배포는 완성된 프로토타입을 공개함으로써, 프로젝트를 공개 소스 소프트웨어 순환 구조 안에서 발전하도록 만들기 위한 단계이다. 프로젝트의 공개와 소프트웨어의 배포는 SourceForge.net[5], Savannah[8]과 같은 공개 소스 소프트웨어 개발자 사이트를 이용할 수 있으며, Freshmeat[9] 등과 같은 사이트를 통한 홍보와 배포도 가능하다. 하지만 SourceForge.net만 따져도 현재 등록된 프로젝트의 수가 이미 177,000개를 넘었기 때문에, 초기 단계의 프로젝트가 검색 단계에서 발견되어 커뮤니티의 주목을 받기는 쉽지가 않은 상황이다.

배포를 통해 성공적인 커뮤니티 기반 공개 소스 프로젝트가 되기 위해서는 프로젝트 관리 구조에 대한 정비와 여러 가지의 중대한 결정들이 필요하다. 프로젝트 관리 구조는 개발자 그룹과 최종적으로 프로젝트의 방향을 결정하는 의사 결정 그룹, 그리고 의사 결정 과정을 의미하는 것이다. 즉, 배포를 통해서 프로젝트의 소유가 초기의 핵심 개발자 그룹에서 커뮤니티로 바뀐다는 점을 바탕으로, 프로젝트에 더 많은 사람들이 참여할 수 있는 구조와 의사 결정 과정을 만드는 것인데, 이 관리 구조는 프로젝트 결과물의 성격에 따라 다르게 결정되어야 한다. 예를 들어 운영체제의 커널 또는 그의 일부와 같이 기술적 위험이 수반되는 프로젝트의 경우에는 보수적 관점에서의 관리를 추구하여야 한다. 즉, 새로운 기능의 수용, 소스 코드의 수정 등에 매우 신중한 결정을 해야 하며, 시험 및 새로운 릴리즈에 관한 중앙집중형 통제권을 유지하는 것이 바람직하다. 반면에 GUI와 같은 사용자 편의 위주의 프로젝트는 다양한 기능적 요구를 빠르게 수용하기 위한 관리 스타일이 좋다. 초기의 프로젝트에서는 메일링 리스트, 뉴스그룹, 포럼 등을 이용한 의견 교환과 묵시적인 합의에 의하여 프로젝트가 진행될 수 있으나, 프로젝트가 점차 명성을 얻어 커뮤니티가 커지면, 공개적이면서도 좀 더 명확한 의사 결정 구조를 요구한다.

여러 관리 스타일에도 불구하고 커뮤니티를 기반으로 발전하는 공개 소스 프로젝트에서는 개발자, 사용자들의 모든 형태의 기여(기능 추가, 버그 수정, 버그 리포팅, 새로운 기능 요구, 등)가 반드시 권장되어야 하며, 그들의 기여 내용이 빠르게 프로젝트에 반영되어야 한다[2].

배포 전에 결정해야할 또 다른 중요한 사항은 공개될 소스의 라이선스를 결정하는 것이다. OSI(Open Source Initiative)[10]의 공개 소스 정의는 공개 소스에 대한 명확한 가이드라인으로 사용되고 있으며, 라이선스가 이 가이드라인을 만족하면, 공개 소스 소프트웨어라고 할 수 있다. 많은 공개 소스 소프트웨어들이 GPL(Generic Public Licence)[11] 또는 LGPL(Lesser GPL)[12] 라이선스를 가지고 있지만, GPL 버전들 사이의 차이를 비롯하여, OSI에 등록된 다양한 공개 소스 라이선스들의 미묘한 차이점은 공개 소스 개발자들이 프로젝트에의 참여 여부를 결정하는 한 요인이 되기도 한다.

기타 결정 사항에는 커뮤니티 참여자들의 주 통신 방법과 소스 코드에의 접근 방법 등이 있다. 통신 방법에는 커뮤니티 참여자의 성격(개발자, 관리자, 사용자 등)에 따른 메일링 리스트, 포럼과 그들의 아카이브, 버그 리포트, 프로젝트 관련 문서, FAQ 등이 있다. 버그 리포트는 사용자와 개발자의 공식적인 통신 방법으로, 공개 소스 소프트웨어 개발자 사이트들이 공통으로 제공하는 버그 트래킹 시스템을 이용한다. 별도의 홈페이지를 이용하여 프로젝트를 공개한다면, Bugzilla[13]와 같은 버그 트래킹 시스템을 설치하여 이용할 수 있다. 버그 트래킹 시스템은 버그를 등록하고, 누가 그것을 담당하여 수정할 지를 할당하고, 현재 처리 상태는 어떤지, 그리고 그 버그에 대한 의견 교환을 하는 등, 버그의 발생부터 해결까지의 전 과정에 대한 체계적인 관리 방법을 제공한다.

소스 코드의 경우 안정된 배포 버전, 흔히 베타 버전라고 하는 외부용 시험 버전, 그리고 현재 개발이 진행 중인 소스 코드 스냅샷, 세 가지를 모두 공개하는 것이 보통이다. 소스 코드 스냅샷 공개는 개발자들이 소스의 버전 관리를 위하여 사용하고 있는 소스 코드 버전 관리 서버에 로그인하여 소스에 읽을 수 있도록 하는 것이다. 소스 코드의 공개는 공개 소스 소프트웨어의 가장 큰 미덕으로 누구나 쉽게 다운로드, 리뷰, 빌드를 할 수 있도록 하고, 궁극적으로 패치를 만들어 프로젝트 관리자에게 보낼 수 있어야 한다.

2) 많은 공개 소스 프로젝트들은 홈페이지 또는 배포된 소스에 프로젝트 기여자들을 나열함으로써, 그들의 기여에 감사하고, 동시에 그 목록에 없는 개발자, 사용자들에게 동기를 부여한다.

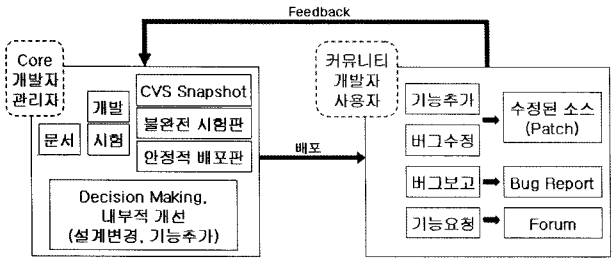

그림 2 공개 소스 소프트웨어 순환 구조

2.6 공개 소스 소프트웨어 순환 구조

일단 프로젝트의 프로토타입, 소스가 공개되어, 점차 알려지고, 사용자, 자원 개발자 등의 관심을 끌어, 커뮤니티가 형성되면 그림 2와 같은 공개 소스 순환 구조에 의해 프로젝트는 진화한다. 공개에 앞서 결정된 의사 결정 구조 등에 의거하여 수정 버전의 릴리즈, 기능이 대폭 보강된 버전업 등이 이루어지며, 사용 중에 발생하는 버그의 리포팅 및 수정, 기능 추가 요구 등이 커뮤니티 측에서도 이루어지며, 그 결과가 프로젝트 관리자 그룹에 의해 반영된다.

이 순환 구조가 얼마나 원활하게 운영되는지가 결국 공개 소스 소프트웨어 프로젝트의 성패를 결정한다. 이 구조의 원활한 순환과 효율성은 모든 참여자 사이의 의사소통 및 의사 결정 과정 등, 배포 전에 내려진 여러 결정에 의해 좌우된다. 공개 소스 소프트웨어가 커뮤니티에 의해 진화한다는 일반적인 인식에도 불구하고, 그 프로젝트를 처음 시작하고, 많은 경우 결국 관리하게 되는 코어 개발자 그룹의 지속적인 관심과 개선 의지는 프로젝트 성공의 가장 중요한 요인이다.

서론에서 많은 공개 소스 소프트웨어 프로젝트 참여의 동기가 '재미' 또는 '명성'이라고 언급한 바가 있다. 하지만, 공개 소스 순환 구조에 안정적으로 들어선 아주 성공적인 프로젝트들은 그 결과가 상업적인 활동에 점차 많이 적용되면서, 기술 지원과 새로운 기능 요구에 대하여 시기적절한 대응이 요구되며, 프로젝트의 핵심 개발자, 관리자 그룹에게는 더 빠른 진화를 할 수 있는 추가적인 동력이 필요하게 된다. 이 과정에서 많은 경우 기술 지원에 대한 대가로서 금전적 보상이 따르게 된다. 최근에는 핵심 개발자 관리자 그룹이 회사와 같은 상업적 조직 형태를 이룰 때, 기업 인수와 같은 방법으로 보상이 이루어지는 경우도 종종 있다[3].

3) 2008년 1월 Sun 마이크로시스템즈의 MySQL AB 인수가 기업의 공개 소스 프로젝트 인수의 전형적인 예이다. 이 과정에서 Sun 마이크로시스템즈는 인수에 미화 8억불을 지불했고, 별도의 스톡 옵션으로 미화 2억불을 제시했다. 인기가 높은 공개 소스 프로젝트를 운영하던 조직이 다른 기업에 인수되면, 프로젝트가 최초로 공개될 때, 또는 인수 직전까지의 공개 소스 라이선스 정책에도 불구하고, 인수 비용을 회수해야 한다는 상업적 명분 때문에, 공개 소스 소프트웨어 정신이 일부 회손될 위험에 처하기도 한다.

3. SourceForge.net의 공개 소스 소프트웨어 프로젝트 분석

이 절에서는 현재 가장 영향력 있는 공개 소스 프로젝트 개발 지원 사이트인 SourceForge.net에 등록된 다양한 프로젝트들에 대한 분석을 통하여, 공개 소스 소프트웨어 프로젝트를 시작하려는 개발자 관점에서 성공적인 프로젝트가 되기 위한, 또는 성공적인 프로젝트로 유지되기 위한 요인들을 살펴본다.

이 분석은 SouceForge.net에 현재 등록되어있는 177,000여개의 프로젝트 가운데 극히 일부만을 대상으로 하였다. 따라서 별도의 개발자 사이트, 배포를 위한 홈페이지 등을 가지고 있는 공개 소스 소프트웨어 프로젝트들뿐만 아니라 SourgeForge.net에 등록되어 있지만, 순위에 들지 않은 훌륭한 많은 프로젝트들이 이 분석 대상에서 제외되었음을 밝혀둔다[4]. 조사는 SourceForge.net에 등록된 프로젝트들에서 표 1의 기준에 따라, 6개 그룹에서 각 20개씩을 선택한 뒤, 각 그룹의 평균 나이, 개발/관리자 수, 현재 상태, 누적 다운로드 수 등을 확인하는 방법으로 이루어졌다. 이 조사는 2008년 5월 14일을 기준으로 하였다.

표 1 조사에 사용된 SourceForge.net의 프로젝트들의 그룹 분류

그룹 A	SourceForge.net을 제외한 Most Active(all-time) 상위 20개
그룹 H	Top Project Hit(past 7 days) 상위 20개 (그룹 A, D1에 속하는 것 제외)
그룹 D1	Most Top Download (all-time) 상위 20개
그룹 D2	Top Download (all-time) 5,000등부터 20개
그룹 D3	Top Download (all-time) 10,000등부터 20개
그룹 D4	Top Download (all-time) 50,000등부터 20개

분류 상 그룹 A, H, D1에 속한 프로젝트들은 SourceForge.net에서 가장 인기 있는 프로젝트들을 의미하는 것으로, 그들의 프로젝트 이름과 장르는 부록에 있다. 그룹 A에서 SourceForge.net이 제외된 이유는 SourceForge.net 홈페이지 자체가 공개 소스로 관리되고 있

4) 따라서, Linux 커널 및 배포판들, KDE, GNOME, Apache, OpenOffice, MySQL 등 규모가 크고, 매우 성공적이지만 SourceForge에서 관리되지 않고 있는 공개 소스 소프트웨어들이 분석에 포함되지 않았다.

표 2 그룹별 평균 특성

평균	그룹 A	그룹 H	그룹 D1	그룹 D2	그룹 D3	그룹 D4
나이 (일)	2,976.1	1,419.8	2,108.7	1,732.2	1,445.8	545.1
개발 상태	5.2	4.9	5.0	4.4	4.8	3.7
개발자 수	26.8	5.8	10.2	3.7	2.1	1.5
관리자 수	3.3	1.8	2.4	1.9	1.5	1.1
다운로드 수(K)	3,536.2	8,115.8	63,747.1	23.4	8.4	0.3
버그 리포트 수	1,004.7	201.1	533.6	13.8	8.9	0.8
지원 요구 수	55.4	7.6	98.3	0.9	1.0	0.0
패치 등록 수	454.9	24.8	26.4	1.4	2.4	0.1
기능 요구 수	175.6	100.9	225.7	7.2	3.9	0.5
포럼 메시지 수	4,413.9	5,826.4	2,603.0	19.5	34.5	2.7
메일링 리스트 수	2.4	0.7	1.2	0.8	0.4	0.2

기 때문에 활성도 통계에서 독보적 1위의 위치를 차지하고 있기 때문이다. 그룹 H는 조사일 기준으로 최근 7일간 가장 많은 페이지뷰를 기록한 프로젝트 가운데 상위 20개를 선택한 것이다. 따라서 그룹 H는 성공적이면서도 최근에 큰 주목을 받고 있는 프로젝트라고 볼 수 있다.

그룹 D2, D3, D4는 그룹 D1과의 비교를 위하여 상대적으로 인기가 낮은 프로젝트들을 선택한 것이다. 이들을 선택한 기준은 프로젝트의 다운로드 회수를 기준으로 각각 5천, 1만, 5만 등 수준의 프로젝트들이며, 1만 등이라 하더라도 실제 다운로드 회수가 8,376회에 이르러 결코 적지 않은 사용자를 확보하고 있다는 것을 고려하여야 한다.

표 2는 각 그룹별 평균 특성을 보여주고 있다. 표에서 개발 상태는 각 프로젝트의 최상위 상태의 평균이다. 프로젝트의 상태는 6:Mature, 5:Production/Stable, 4:Beta, 3:Alpha, 2:Pre-Alpha, 1:Planning, 0:Inactive를 의미한다. 실제 이 상태는 프로젝트 관리자가 스스로 입력하는 것으로 상업적인 프로젝트에 비하여 상대적으로 고평가 되어있는 것이 보통이다.

각 그룹별로 20개씩 선택 조사된 제한적 결과이기는 하지만, 다운로드 수 기준인 그룹 D1, D2, D3, D4를 비교할 때, 여러 가지 명확한 추세를 확인할 수 있다. 우선 프로젝트가 성공적으로 진화되어 감에 따라 더 많은 등록 개발자, 더 많은 관리자가 연관된다는 것으로, 인기가 높은 프로젝트의 경우, 10명 이상의 등록된 핵심 개발자 그룹에 의해 프로젝트가 유지되고 있다는 것을 확인할 수 있다. 또한 일반적인 예상처럼 그룹 A, H, D1과 같이 인기가 높은 프로젝트는 지원 및 기능 요구, 그에 따른 패치를 등록한 수 등 프로젝트 관련한 모든 활동 지수들이 D2, D3, D4 등과 확연이 구분되는 것을 볼 수 있다.

특히, 그룹 H의 경우, 최근에 인기가 높은 프로젝트들이 속한 그룹이므로, 상대적으로 그룹 A나 D1에 비하여 나이, 개발자 수가 적지만 포럼 메시지 수가 가장 높고, 비슷한 평균 나이를 가진 그룹 D3과는 비교할 수 없을 정도의 높은 다운로드 수를 보이는 것을 확인할 수 있다.

상용 소프트웨어 시장에서 상위 한 두 개의 제품이, 시장 점유율의 대부분을 차지하듯이, 공개 소스 영역에서도 게임과 같은 콘텐츠 중심의 소프트웨어 장르를 제외하고는 장르별로 상위 몇 개의 프로젝트와 그 이하의 프로젝트와의 인기 격차가 매우 심한 편이다. 또한 그룹 D2, D3, D4와 같은 하위 그룹의 프로젝트들의 경우, 커뮤니티 활동의 부침이 심한 편이라, 순위 변동이 매우 심한 편이다. 표에서 그룹 D3의 포럼 메시지 수, 패치 등록 수 등 그룹 D2 보다 높은 항목들이 발생한 이유도 그룹 D3에 속한 세 개의 프로젝트에서 이 조사를 실행한 시점 근처에, 급격히 커뮤니티 활동이 증가한 때문이다.

프로젝트 인기도의 집중화는 그림 3의 그래프에서도 확인할 수 있다. 그림 3은 다운로드 수를 기준으로 1위, 2위, 4위, 8위, ..., 65536위를 차지한 프로젝트들의 다운로드 수와, 일수로 따진 나이를 다운로드 순위를 기준으로 표시한 것이다. 그래프에서 x 축은 log2(다운로드 순위)이다. 그래프에서 순위가 조금만 떨어져도 다운로드 수가 급감함을 볼 수 있다. 프로젝트 인기도와 나이와의 관계의 경우, 표 2에서 확인한 바와 같이, 평균적인 특성은 프로젝트의 나이가 많을수록 커뮤니티의 누적 관심도가 쌓여가는 것으로 보이지만, 그림 3은 모든 프로젝트가 오래될수록 인기가 많아지지는 않는다는 점을 보여주고 있다. 실제로 SourceForge.net의 많은 프로젝트들이 중단된 채로, 나이만 먹어가고 있다.

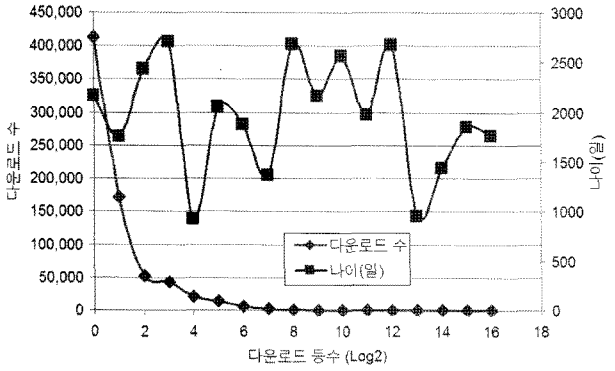

그림 3 다운로드 횟수와 프로젝트 나이의 관계

SourceForge.net에 등록된 프로젝트들에 대한 분석 결과, 우리는 놀랄만한 성과를 거둔 공개 소스 소프트웨어 프로젝트도 있는 반면에, 거의 관심을 끌지 못하는 프로젝트도 많다는 것을 확인할 수 있었다. 공개 소스 소프트웨어 프로젝트가 성공하지 못하는 데에는 다양한 이유가 있을 수 있다. Klincewicz의 연구[14]에서는 가장 중요한 이유로 프로젝트의 혁신성 부족을 들고 있으며, 많은 공개 소스 소프트웨어 프로젝트가 여러 가지 원인에 의해 시작된 'me too' 프로젝트라고 하였다[5]. 다른 중요한 이유로는 프로젝트를 유지하기 위한 동력의 부족이다. 즉, 프로젝트를 시작한 핵심 그룹이 공개 소스 소프트웨어 순환 구조가 정착될 때까지 지속적인 관리를 하지 못한다는 것이다. 이는 많은 프로젝트가 '명성'이나 '재미'를 위해서가 아니라, 대학교의 연구 프로젝트, 또는 단기 과제로 시작되었기 때문이기도 하다. 이 경우, 과제 재출 기한이 지나거나, 핵심적인 개발자가 학교를 떠나거나, 연구비가 지속적으로 지원되지 않는 위기에 쉽게 노출되며, 이러한 종류의 위기는 쉽게 극복되기 어렵다. 특히 연구 과제의 속성 상, 이러한 프로젝트의 경우 특정한 기능에 집중한 개발이 진행되며, 소스 공개 후에도 커뮤니티 기반의 공개 소스 소프트웨어 순환 구조 속에서 진화를 하는 모델을 관리자들이 따르기도 어렵기 때문에 프로젝트의 혁신성에도 불구하고 공개 소스 프로젝트로서 성공하기 어렵다. 또 다른 실패 이유로는 공개된 소프트웨어의 낮은 품질을 들 수 있다. 2절에서 살펴본 바와 같이, 소스 공개를 위한 프로토타입의 완성도는 커뮤니티가 형성되기 위한 중요한 필요조건 중의 하나인데, 이를 만족하지 못하는 경우, 새로운 참여자를 끌어들이지 못하기 때문에, 해당 프로젝트는 커뮤니티 기반의 순환 구조에 이르기 어렵다.

4. 품질: 공개 소스 소프트웨어의 중요한 성공 요인

많은 공개 소스 소프트웨어들은 상용 소프트웨어 못지않거나, 상용 소프트웨어를 능가하는 품질을 갖추고 있다[15]. Raymond는 그의 명저 '성당과 시장(The Cathedral and Bazaar)'[16]에서 공개 소스 소프트웨어의 높은 품질이 부분적으로는 다른 개발자에 의한 소스 리뷰와 개발 과정에서의 사용자 참여 덕분이라고 하였다. 이는 다른 개발자에 의한 소스 리뷰[6]가 소프트웨어의 품질 향상에 매우 중요한 역할을 한다는 소프트웨어 공학에서의 연구 결과와 일치하는 것이다.

새로운 공개 소스 소프트웨어가 지속적인 발전을 이룰 수 있는 순환 구조에 들어서려면, 지속적인 기능 향상과 함께, 상용 소프트웨어 수준의 품질을 유지할 수 있어야 한다. 공개 소스 소프트웨어 프로젝트들은 소스 리뷰라는 강력한 품질 향상 도구를 가지고 있기는 하지만, 지속적인 기술 지원과 빠른 오류 수정을 위해 필요한 인력의 부족, 또는 핵심 개발자 그룹의 기술 부족, 프로젝트 관리 능력의 부족 등은 결과물의 품질에 나쁜 영향을 끼친다.

4.1 공개 소스 소프트웨어의 품질 문제 발생 양상

Michlmayr 등의 연구[17]에서는 공개 소스 소프트웨어 프로젝트에서 발생하는 품질 문제 발생 양상을 다음과 같이 정리하고 있다.

- 기술 지원의 부족 : 가장 중요한 문제는 소스는 제공되지만 유지 보수가 되지 않는 소프트웨어이다. 공개 소스 소프트웨어 프로젝트에서는 어떤 기여자가 잘 알려지지 않은 하드웨어에 소스를 포팅하거나, 특별한 기능을 추가한 소스 코드를 등록을 할 수도 있다. 문제는 다른 개발자들에 의해 원래 소스가 수정되면, 이런 특별한 기능이나 포팅 결과도 지속적인 개정이 있어야만 계속 사용될 수 있다는 것이다. 그러나 불행히도 현재의 소스에 기여한 개발자 중 일부가 더 이상 프로젝트에 참여할 수 없는 상태일 때, 그 소프트웨어는 기술 지원과 유지 보수가 어렵게 질 수 밖에 없다. 핵심 개발자, 관리자들은 이런 상황을 어떻게 다룰 것인지 결정해야 한다.

- 설정 관리 문제 : 많은 공개 소스 소프트웨어 프로젝트들은 모듈화, 계층화를 추구함으로써, 높은 수

5) Klincewicz는 논문에서 SourceForge.net이나 많은 공개 소스 프로젝트에서 프로젝트 설명에 'Yet another'와 같은 구절을 쉽게 볼 수 있다는 점을 이야기하고 있다.

6) 코드 리뷰, 소스 인스펙션, 소스 리딩 등 여러 가지 다른 명칭으로 불리지만, 기본적으로 소스를 읽어 문제를 찾아내는 기법

준으로 커스터마이징이 가능하도록 설계, 개발된다. 이는 사용자에게 많은 유연성을 제공하는 반면, 테스트 문제를 야기한다. 핵심 개발자들은 모든 경우의 수를 테스트하는 것이 매우 어렵거나 불가능하기 때문에 흔히 사용되는 환경에서만 테스트해 보는 경향이 있다. 따라서 새로운 버전의 소프트웨어가 릴리즈 되면, 새 버전이 자신의 환경에서 동작하지 않는다는 버그 리포트가 많은 것이 일반적인 일이다.

- 보안 업데이트 지연 : 공개 소스 소프트웨어의 보안성은 소스 리뷰 과정 덕분에 상용 소프트웨어보다는 더 높은 것으로 알려져 있고, 많은 경우에 보안 업데이트가 적시에 이루어진다. 하지만, 핵심 개발자, 관리자의 본업에 따른 일정 때문에, 또는 다른 여러 가지 이유로, 몇 주 혹은 몇 달 동안 보안 업데이트가 이루어지지 않는 경우도 있다.

- 버그 리포팅 부실 : 전문적인 기술이 부족한 사용자들이 공개된 소프트웨어를 많이 사용함에 따라, 개발자들은 충분한 정보를 포함하지 않거나 중복된 버그 리포트를 더 받게 된다. 그런 버그 리포트는 개발자들의 시간을 빼앗을 따름이다. 일부 프로젝트에서는 버그 리포팅 방법에 대한 더 상세한 설명서를 만들기도 했지만, 많은 사용자들은 버그 리포팅 전에 그 설명서를 읽지 않는다.

- 문서화의 미흡 : 앞의 문제점들은 문서화의 미흡과 관련이 있다고 말할 수 있다. 자원 개발자들은 어떤 한 영역에 기여를 하고 싶어 하는데 어떻게 시작해야 하는지 모를 수 있다. 실제로 많은 프로젝트에서 잠재적 기여자들이 받을 수 있는 도움은 거의 없고, 문서도 거의 존재하지 않는다. 또한 개발자 문서의 부족은 또한 모든 사람이 같은 기술, 같은 개발 과정을 따르고 있는지 확신할 수 없고, 결국 품질의 저하를 가져올 수 있다는 것을 암시한다.

- 의견 조정과 의사소통의 문제 : 일부 프로젝트에는 의견 조정과 의사소통의 문제가 있으며 이는 프로젝트 품질에 나쁜 영향을 줄 수 있다. 때로는 어떤 영역에 대한 책임이 누구에게 있는지 명확하지 않기 때문에 버그에 대해 적절한 의사소통을 할 수 없는 경우도 있다. 또한 작업이 중복될 수도 있고 치명적인 버그를 없애기 위한 조정이 미흡할 수도 있을 것이다.

4.2 공개 소스 소프트웨어의 품질 개선 방안

공개 소스 소프트웨어 프로젝트에서 품질에 나쁜 영향을 끼치는 요인들은 문서화의 부실, 인프라의 부실, 프로젝트 관리 시스템의 부실 등으로 요약될 수 있다. 다시 말하면, 이 세 가지 요인에 대한 적절한 준비, 관리 및 시행이 공개될 소프트웨어의 품질을 담보하는 방법이라고 볼 수 있다.

- 문서화 : 문서화는 아무리 강조해도 지나침이 없는 중요한 요소이다. 성공적인 몇몇 프로젝트를 제외한, 많은 공개 소스 소프트웨어 프로젝트들은 소프트웨어의 기능 분석, 설계, 구현 및 관리에 관한 공식적인 문서가 매우 부족하며, 사용자 설명서도 부실한 경우가 많다. 또, 더 많은 참여자를 유도하기 위한 기초적인 안내 문서를 가지고 있지 않은 프로젝트들도 많다[7]. 참여자 유도를 위한 문서로는, 개발자들이 따라야 하는 소스 코딩 스타일을 설명하는 문서와 소스 등록 절차에 대한 문서가 있다. C언어 코딩 스타일에 관한 한, 공개 소스 소프트웨어의 경우, 리눅스 커널에서 보이는 코딩 스타일[18]이 일반적으로 받아들여지고 있다. 소스 등록 절차에 관한 문서는 누가, 어떤 과정으로, 프로젝트의 버전 관리 시스템에 수정된 소스를 등록할 수 있는지 나타내는 문서로, 프로젝트 관리의 핵심 절차를 기술한 문서이다.

- 프로젝트 인프라 : 공개 소스 소프트웨어 프로젝트는 분산, 협력 개발을 가능케 하는 인프라에 많이 의존한다. 프로젝트 인프라에는 버그 트래킹 시스템, 버전 관리 시스템, 자동 빌드 도구, 메일링 리스트 등이 포함된다. 버그 트래킹 시스템은 주로 사용자의 피드백을 수집하는데 사용되며, 기능에 대한 요구를 저장할 때도 자주 사용된다. 버전 관리 시스템은 여러 사람이 동시에 같은 코드 기반에서 일하는 것과 누가 어떤 것을 변경했는지를 계속 추적할 수 있도록 해준다. tinderbox[19]와 같은 자동 빌드 도구는 버전 관리 시스템 내의 최신 코드가 계속 성공적으로 빌드 되는 것을 보장한다. 시험 빌드는 다른 여러 하드웨어나 소프트웨어 환경에서도 진행될 수 있다. 실제 프로젝트 인프라도 중요하지만 품질은 인프라 자체보다는 인프라를 프로젝트에 이용하는 방법, 즉 프로젝트 관리 방법에 의해 더 좌우된다.

---

7) 많은 연구에서 공개 소스 프로젝트의 문서화 문제를 제기하고 있는 것이 사실이다. 사용법과 같은 최종 사용자를 위한 문서는 대부분 성공적인 프로젝트에서 비교적 정돈된 형태로 준비가 되지만, 공식적인 요구 분석, 설계, 의사 결정 내용과 같은 구현 관련 문서의 부재는 공개 소스 소프트웨어의 장기적 품질 유지에 어려움을 가중시키며, 새로운 참여자에 대한 진입 장벽으로 작용한다. 한편으론, 이러한 내용들이 커뮤니티의 통신 과정에서, 즉 메일링 리스트, 뉴스 그룹 또는 위키의 메시지 형태로 남아있으며, 결정적으로 피어 리뷰를 거친 소스 코드 자체가 훌륭한 문서라고 주장하기도 한다.

- 프로젝트의 절차 : 공개 소스 소프트웨어의 개발에는 많은 단계적 절차들이 개입되지만, 보통 그 절차는 어디에도 문서화되어 있지 않으며, 개발자들이 그 묵시적인 절차들을 따른다. 하지만 프로젝트의 핵심 개발자, 관리자 그룹에는 다음과 같은 합의된 절차와 정책들이 필요하다. 첫째, 참여를 원하는 잠재적 핵심 개발자들 기준과 절차를 가지고 있어야 한다. 대부분의 프로젝트들은 개방적인 정책을 유지하지만, 몇몇 프로젝트는 높은 품질의 결과물을 기대할 수 있는 개발자들에게만 참여를 허용하는 것을 명시적으로 밝히고 있기도 하다. 둘째, 적절한 릴리즈 정책을 가지고 있어야 한다. 릴리즈 정책은 기능 프리즈 단계를 거치도록 만드는 것이 좋다. 기능 프리즈는 새 기능을 코드에 추가하지 않고 버그를 고칠 충분한 시간을 벌어 준다. 이 정책에 따라, 리눅스 커널처럼, 안정적 버전과 개발 버전을 별도로 유지할 수도 있으며, 그 결과 사용자 입장에서의 보는 소프트웨어 안정성에 큰 도움을 준다. 셋째, 소스 리뷰 정책이 필요하다. 전형적으로 버전 관리 시스템을 통한 소스의 수정은 다른 프로젝트 구성원들에 의해 검토되지만, 많은 경우에 이런 형태의 피어 리뷰가 공식화 되어있지는 않다. 즉, 개발자들은 자신이 수정한 것을 다른 프로젝트 구성원들이 검토해주기를 원하지만 실제로 그렇게 하는 것을 보장할 방법이 없다. 일부 프로젝트는 이 절차를 상당히 엄격히 실행하며 소스 등록을 하기 전에 반드시 코드를 검토할 것을 요구한다. 넷째, 확고한 테스트 절차의 확보가 필요하다. 새로운 릴리즈가 프로젝트의 기준을 만족시키는지, 또 이전 릴리즈에서는 없던 문제가 발생하는 지를 확인하기 위하여 프로젝트는 테스트 체크 리스트를 가지고 있어야 한다. 이 체크 리스트는 가장 중요한 기능들에 대한 테스트 절차들 포함하고 있어야 한다. 릴리즈는 테스터들이 다른 플랫폼들에서 체크 리스트의 내용을 하나하나 검사하고, 새 버전이 적어도 주요 기능에서 프로그램의 중단이나 이전에는 잘 동작하던 기능이 동작하지 않는 퇴보 현상이 나타나지 않았음을 확증하고 난 이후에 이루어지도록 한다. 자동 시험 도구를 가지고 있는 프로젝트는 거의 없기 때문에, 테스트 체크 리스트를 가지고 있는 것은 적어도 중요한 구성 요소와 기능의 작동을 보장하기 위한 좋은 해결책이 될 수 있다. 다섯째, 품질 개선 활동이 필요하다. 이 활동의 정형화된 형태는 없으나, 코드 페스트(Code Fest), 버그 데이(Bug Day) 같은 행사를 통하여, 여러 개발자가 모여, 집중적으로 새로운 기능을 추가하거나, 해결되지 못한 버그들을 제거하게 된다.

5. 결 론

많은 공개 소스 소프트웨어는 높은 품질과 안정적인 기술 지원 등을 바탕으로 상용 소프트웨어를 능가하는 성공을 거두고 있다. 또 공개 소스 소프트웨어는 IT 산업과 비 IT 산업 모두에서 소프트웨어에 의한 기술적 진입 장벽을 크게 낮추는 역할을 하기 때문에, 수요와 공급 양쪽 측면 모두 활성화할 이유가 충분하다.

이 조사 논문에서는 공개 소스 소프트웨어 프로젝트의 전 생명 주기의 각 활동 단계에 소프트웨어 개발자, 사용자들이 어떤 방식으로 참여하고 기여할 수 있는지를 기술하였다. 그리고 가장 영향력 있는 공개 소스 소프트웨어 개발자 사이트인 SourceForge.net에 등록된 소프트웨어들을 대상으로 성공적인 공개 소스 소프트웨어 프로젝트들의 특징을 알아보았다. 또 소프트웨어 품질 관점에서 공개 소스 소프트웨어 프로젝트가 가져야 할 요소들에 대하여 기술하였다.

모쪼록 이 논문이 공개 소스 소프트웨어 분야에서 선도적인 역할을 담당하고 있는 국가들의 동급 엔지니어들에 비하여, 상대적으로 여유 시간을 할애하기 어려운 국내의 소프트웨어 개발자들, 또 뭔가 혁신적인 아이디어를 공개 소스 형태로 개발하고자 하는 사람들이 공개 소스 프로젝트를 시작하거나 참여할 때, 좀 더 효율적인 접근을 하는데 도움이 되기를 기대한다.

참고문헌

[1] 이도규, "공개소프트웨어 정책성과 발전방향", 한국정보과학회지, 제24권, 제6호, pp.5-8, 2006.

[2] Tovalds, L., Diamond, D., Just for Fun: The Story of an accidental revolutionary, Harper Business, 2001.

[3] Luthiger, B., "Fun and Software Development", in Proceedings of the First International Conference on Open Source Systems, Genova, pp. 273-278, 2005.

[4] Hertel, G., Niedner, S., Hermann, S., "Motivation of software developers in the open source projects: An internet-based survey of contributors to the linux kernel", Research Policy, Vol. 32, No. 7, pp 1159-1177, 2002.

[5] SourceForge.net, http://sourceforge.net/

[6] Ghosh, R., "Clustering and dependencies in free/open source software development: Methodology and tools", First Monday, Vol. 8, No. 4, 2003.

[7] Senyard, A., Michlmayr, M., "How to Have a Successful Free Software Project", report, Department of Computer Science and Software Engineering.

The University of Melbourne, 2004.
[8] http://savannah.gnu.org/ or http://savannah.nongnu.org/
[9] Freshmeat, http://freshmeat.net/
[10] The Open Source Initialtive, http://www.opensource.org/
[11] The GNU General Public License(GPL), http://www.gnu.org/licenses/gpl.html
[12] The GNU Lesser General Public License(LGPL), http://www.gnu.org/licenses/lgpl.html
[13] Bugzilla, http://www.bugzilla.org/
[14] Klincewicz, K., "Innovativeness of open source software projects", report, School of Innovation Management, Tokyo. Institute of Technology, 2005.
[15] Halloran, T. J., Scherlis, W. L., "High quality and open source software practices", in Proceedings of the 2nd Workshop on Open Source Software Engineering, Orlando, 2002.
[16] Raymond, E. S., The Cathedral and the Bazaar, O'Reilly & Associates, 1999.
[17] Michlmayr, M., Hunt, F., Probert, D., "Quality Practices and Problems in Free Software Projects", Center for Technology Management, University of Cambridge, 2005.
[18] 리눅스 커널 소스의 ./Document/CodingStyle 파일에 설명된 코딩 스타일
[19] tinderbox, http://www.mozilla.org/tinderbox.html

이 민 석

1986 서울대학교 컴퓨터공학과 학사
1988 서울대학교 컴퓨터공학과 석사
1995 서울대학교 컴퓨터공학과 박사
1999~2002 (주)팜팜테크, CTO
1995~현재 한성대학교 컴퓨터공학과 교수
관심분야 : 임베디드 시스템 및 소프트웨어, 멀티미디어 파일 시스템, 공개 소스 소프트웨어
E-mail : minsuk@hansung.ac.kr

부 록

그룹 A 프로젝트들 목록 (Most Active, All-Time)

프로젝트 제목	분야	시작일	다운로드수
Crystal Space 3D SDK	First Person Shooters	99-12-09	381,840
Pidgin	AOL Instant Messenger	99-11-13	22,857,796
Gimp-Print-Top Quality Printer Drivers	Raster-Based Graphic	00-01-17	1,875,929
SquirrelMail	Email Clients (MUA)	99-11-18	2,900,624
Licq	AOL Instant Messenger	99-11-15	1,893,776
jEdit	Software Development	99-12-06	4,567,088
Ghostscript	Graphics Conversion	00-01-26	14,156,767
WebCalendar	Calendar	00-03-22	913,330
NumericalPython	Scientific/Engineering	00-01-12	1,097,319
Python	Interpreters	00-05-08	209,067
Mailman	Mailing List Servers	99-11-07	193,822
Exult	Role-Playing	00-02-07	539,358
Double Choco Latte	Office/Business	00-01-14	76,558
icewm	Gnome	99-11-07	642,271
Linux PCMCIA Card Services	Linux	00-02-08	444,346
Quanta+ Web Development Environment	K Desktop Environment	00-03-29	723,048
Miranda	AOL Instant Messenger	00-02-02	4,401,684
BO2K	Internet	00-04-09	3,059,156
MiKTeX	TeX/LaTeX	00-09-03	5,712,796
wxWidgets	Frameworks	00-08-16	4,077,929

그룹 H 프로젝트들 목록 (Top Project Hit, Past 7 days)

프로젝트 제목	분야	시작일	다운로드수
Simple Directory Listing	Dynamic Content	06-08-14	15,298,890
AutoAP	Wireless	06-12-18	2,667,500
MinGW-Minimalist GNU for Windows	Build Tools	00-02-09	13,617,723
Notepad++	Software Development	03-11-24	8,801,714
XAMPP	Database	02-09-04	13,221,758
ClamWin Free Antivirus	Security	04-03-25	13,483,881
Smart package of Microsoft's core fonts	Desktop Environment	01-08-22	3,302,806
MediaCoder	CD Ripping	05-10-29	8,984,663
TortoiseSVN	Usability	05-05-09	9,168,095
AC3Filter	Sound/Audio	02-10-30	16,207,259
Magical jelly Bean keyfinder	Installation/Setup	08-03-23	421,551
Frets on Fire	Games/Entertainment	06-11-13	4,590,801
Wireshark	Monitoring	99-11-15	7,253,901
ophcrack	Security	05-03-10	3,491,447
DreaMule	File Sharing	04-01-28	4,191,783
emule Xtreme Mod	File Sharing	05-01-23	11,537,950
Dev-C++	Software Development	00-08-31	14,718,834
WampServer	HTTP Servers	04-08-03	4,775,119
ffdshow tryouts	Sound/Audio	06-08-02	3,806,464
StrongDC++	Chat	07-03-23	2,773,429

그룹 A 프로젝트들 목록 (Most Active, All-Time)

프로젝트 제목	분야	시작일	다운로드수
eMule	File Sharing	02-05-13	407,424,851
Azureus	BitTorrent	03-06-24	166,860,494
Ares Galaxy	Chat	04-06-18	153,722,246
BitTorrent	File Sharing	01-08-07	51,850,559
DC++	Chat	01-11-17	51,078,388
GTK+,The GIMP installers for Windows	Raster-Based Graphic	04-10-07	43,209,674
Shareaza	BitTorrent	04-05-28	43,172,851
7-Zip	Backup	00-11-10	42,372,179
Audacity	Analysis	00-05-28	40,519,339
Virtualdub	Conversion	00-08-12	39,541,286
FileZilla	File Sharing	01-02-27	38,447,467
CDex	CD Ripping	99-12-05	35,621,945
eMule Plus	Internet Relay Chat	03-01-18	25,888,249
guliverkli	Capture	03-05-29	23,920,380
PortableApps.com:Portable Software/USB	Chat	05-10-21	19,621,291
aMSN	MSN Messenger	02-05-22	19,129,503
WinSCP	Communications	03-07-13	18,888,324
TightVNC	System Administration	00-11-03	18,391,043
PDFCreator	Office Suites	02-07-13	17,716,373
UltraVNC	Networking	02-10-02	17,566,036