소셜 컴퓨팅 요소가 오픈 소스 개발 프로젝트의 성과에 미치는 영향에 대한 연구

: 소셜 코딩 플랫폼 Github 사례를 바탕으로

Get Social and Get Better
: How social computing features help open source software projects

최주희, Joohee Choi*, 최정홍, Junghong Choi**, 문재윤, Jae Yun Moon***

요약 본 연구에서는 오픈 소스와 소셜 컴퓨팅의 결합인 소셜 코딩 플랫폼의 대표적인 사이트 Github(http://github.com)를 대상으로 소셜 컴퓨팅 요소들이 프로젝트 결과물에 어떠한 영향을 미치는지 알아보고자 하였다. 소셜 컴퓨팅 요소의 적용 및 효과에 대한 논의는 갈수록 활발해지는 추세이나, 오픈 소스와 같이 대표적인 온라인 협업 공간에 소셜 컴퓨팅 요소를 적용하였을 때 그것이 협업 성과에 어떤 영향을 미칠 것인지에 대한 연구는 아직 미흡한 실정이다. 보다 세부적인 연구 질문은 다음과 같다: 1) 소셜 코딩 플랫폼의 시스템 요소들은 기능적으로 어떻게 구분할 수 있는가? 2) 실제 프로젝트의 결과물과 시스템 요소 이용 간에는 어떠한 관계가 있는가? 이에 답하기 위해 본 연구에서는 정성 및 정량적 분석을 수행하였다. 정성적 분석으로는 Github 이용자들의 인터뷰 결과에 기반하여 Github에서 제공하는 시스템 요소들을 기본적인 기능에 따라 분류하였고, 정량적 분석으로는 시스템 요소들과 프로젝트 결과물 간의 관계를 파악하기 위해 2,034개 프로젝트 샘플을 대상으로 다중 회귀 분석을 실시하였다. 결론적으로, 외부 연결 시스템 요소들이 프로젝트들의 코드에 큰 영향을 미친다는 사실을 발견하였다. 마지막으로는 본 연구의 결과에 기반한 함의를 제시하였다.

Abstract In this study, we aim to understand how social computing features affect open source project's outcome based on the representative social coding platform, Github (http://github.com). Though there is growing interest regarding the application and effect of employing social computing features, yet empirical evidences related to the subject are still short. To bridge the gap, we conducted our research based on the following research questions: 1) How the system features of social coding platform are classified? 2) How are the use of system features and project performance related to each other? Qualitative and quantitative analysis are performed: The system features of Github are clustered according to their usage in qualitative analysis, and the relation between the feature uses and project outcome is identified by multiple linear regression test. In conclusion, we found that the use of inter-project support features has a positive relation with a project outcome. Implication based on the results is also discussed.

핵심어: Social Coding, Social computing, Open source, Github

본 논문은 2010년 한국연구재단의 우수학자 지원사업에 의하여 연구되었음.
*주저자: 연세대학교 인지과학협동과정 석사과정; e-mail: tigerjh@yonsei.ac.kr
**공동저자: 연세대학교 인지과학협동과정 석사과정; e-mail: junghong@yonsei.ac.kr
***교신저자: 고려대학교 경영학과 교수; e-mail: jymoon@korea.ac.kr
■ 접수일 : 2012년 3월 26일 / 심사일 : 2012년 4월 20일 / 게재확정일 : 2012년 6월 28일

1. 서론

온라인에서 일어나는 사람들의 사회적인 활동을 지원하는 기술을 소셜 컴퓨팅이라고 일컬으며[1], 최근 소셜 매체의 성공과 확장에 힘입어 소셜 컴퓨팅의 적용 사례와 범위가 빠르게 확대되고 있다. 온라인 대규모 협업 또한 소셜 컴퓨팅 기술의 적용을 통해 확산된 새로운 형태의 사회적 활동 중 하나다. 대표적인 예시로는 온라인 백과사전인 Wikipedia와 소셜 네트워킹 사이트(SNS)인 Facebook, LinkedIn 등을 들 수 있으며, 이 밖에도 소셜 컴퓨팅 요소를 적용한 서비스의 범위는 온라인 지식 공유 플랫폼인 Stackoverflow, 공유 및 협업 플랫폼인 openprocessing.org 등을 아우른다.

오픈 소스 개발은 온라인을 통한 대규모 협업의 대표적인 사례 중 하나이나, 기존의 오픈 소스 개발 지원 사이트들은 전통적인 포럼 형태의 게시판과 사이트 외부 소통 채널, 예를 들어 이메일, 메신저 등을 활용하는 방식으로 협업을 진행하는 등, 소셜 컴퓨팅의 요소가 크게 관여하는 방식을 적용하고 있지 않다. 그러나 최근 Github를 필두로 소셜 코딩이라는 용어가 주창되었고, Github의 성공으로 말미암아 소셜 컴퓨팅 요소와 오픈 소스의 결합이 점차 주목 받고 있다. 소셜 코딩은 앞서 언급한 컴퓨팅 요소를 적용한 오픈 소스 개발 플랫폼을 지칭한다[2]. Github[2]는 버전 관리 시스템(Version Control System)인 Git을 차용한 오픈 소스 코드 호스팅 사이트로, 2008년 설립된 이래로 매우 빠른 성장세를 보여 최근에는 가장 인기 있는 오픈 소스 개발 사이트로 선정[3]되었으며, 소셜 코딩의 대표적인 성공 사례라고 볼 수 있다.

오픈 소스 개발 방식은 온라인 협업의 대표적 성공사례로 항상 거론될 정도로 크게 확장되었다[4]. 일례로, 2007년 IDC(International Data Corporation)의 조사에 따르면 전 세계 개발자의 71%가 오픈 소스 방식으로 개발된 소프트웨어를 사용하고 있다고 한다[5]. 도입 초기에는 소스 코드를 공개하는 것에 대해 회의적인 시각이 팽배했음에도 불구하고, 이 개발 방식이 비공개(closed) 개발 방식에 비해 개발 후기의 검수 단계에서 높은 효율성을 보인다는 점 또한 다수의 연구를 통해 입증되었다[6]. 이처럼 주요 개발 방식으로 자리잡은 오픈 소스 도메인에서 소셜 컴퓨팅을 적용하는 것이 개발 생산성이나 소프트웨어 품질에 어떠한 영향을 미치는지에 대해 알아보는 것은 매우 중요하다[7, 8]. 소셜 코딩 플랫폼에 대한 연구는 이 같은 맥락에서 반드시 수행되어야 할 필요성을 가진다.

그러나 앞서 언급하였듯 오픈 소스 결과물의 활용 확산에 의해 소셜 컴퓨팅 요소와 오픈 소스 개발 방식의 접목의 효과를 이해하는 것이 점점 중요해짐에도 불구하고, 소셜 코딩 플랫폼의 등장 자체가 최근에 발생한 일이라 아직 실증적인 데이터에 기반해 현상을 살펴본 연구는 매우 드물다. 이는 소셜 컴퓨팅의 효과를 실증적으로 확인하고자 하는 시도가 가지는 선천적인 어려움 에서 기인한다[9]. 소셜 컴퓨팅 요소 이용에 관여하는 대상의 수가 많아짐에 따라 기존의 현상들에 비해 대규모의 현상을 기록, 보관, 분석할 수 있어야 하며 협업 성과를 측정하기 위한 도구의 개발 또한 협업 맥락에 맞추어 탐색해야 한다[10]. 그러나 그럼에도 불구하고, 실증적인 행동 데이터에 의거해 현상을 파악하는 작업은 현상을 이해하고 응용하고자 하는 시도의 첫걸음으로서 큰 가치를 가진다[11]. 소셜 컴퓨팅과 온라인 협업 간의 접목이 가져올 효과에 대한 기존 연구가 부족한 실정인 만큼, 본 연구에서는 대표적인 소셜 코딩 플랫폼인 Github를 대상으로 웹 로그를 수집, 분석하는 실증적인 연구를 수행하였다.

요약하면, 본 연구의 목적은 Github와 같이 성공적인 소셜 코딩 플랫폼에서의 시스템 요소 이용 패턴과 영향을 파악하는 것으로, 이에 따른 연구 질문은 아래와 같다: 1) 소셜 코딩 플랫폼의 시스템 요소들은 기능적으로 어떻게 구분할 수 있는가? 2) 실제 프로젝트의 결과물과 시스템 요소 이용 간에는 어떠한 관계가 있는가?

이하 구성은 다음과 같다. 먼저 2장에서 기존 연구 동향에 대해 서술하고, 3장에서는 본 연구에서 적용한 방법론에 대해 설명한다. 4장에서는 각 방법론에 대한 결과를 기술한 후 5장에서 이에 대해 논하도록 하겠다.

2. 기존 연구 동향

소셜 컴퓨팅의 범위는 매우 광의적이기에 명확하게 규정하기 어렵다. 일반적으로는 "사회적 관계에 초점을 맞추거나 중간 매개체로서 이를 지원하는 모든 종류의 기술"을 소셜 컴퓨팅 기술이라 일컫는다[12]. 소셜 컴퓨팅의 아이디어는 1940년 Vannevar Bush 세미나에서 처음 등장하였으며[1], 긴 역사와 더불어 Facebook, Twitter와 같은 최근의 소셜 미디어 성공이 맞물려 그 어느 때보다도 빠르게 확산되고 있다. Parameswaran의 2007년 연구[5]에서는 소셜 컴퓨팅 플랫폼으로서 블로그, 위키피디아, SNS 등을 대표적인 예로 들고 있다. 그러나 이와 같은 빠른 확산에도 불구하고, 아직 소셜 컴퓨팅과 오픈 소스의 접목인 소셜 코딩에 대해서는 학계에서 연구된 바가 매우 드물다. 소수의 연구가 소셜 코딩의 개념을 다루고는 있으나 [7,8], 해당 연구에서는 소셜 코딩에 대한 아이디어 제시 이상의 실증적인 분석을 하지 않고 있다.

Begel, DeLine과 Zimmermann은 그들이 연구 [8]에서 개인들이 프로젝트 내부와 외부 경계를 넘나들며 자기 조직화를 한다는 아이디어를 제안하였다. 그들은 소프트웨어 개발 커뮤니티는 신 기술을 중심으로 하여 창발할 것이라고 하였으며, 이러한 개인들로 구성된 집단에서는 소셜 미디어를 활용해 성공적인 제품 라인을 고안하고, 디자인 하고, 개발하고, 도입하는 데에 레버리지(leverage) 효과를 줄 수 있을 것이라고 한다. 그들은 제안한 아이디어를 입증하기 위해 Tuckman과 Jensen이 제

시한 소프트웨어 개발 주기 모형[13]에 근거(형성, 합의, 규범 설정, 수행하기, 반성하고 반영하기)하여 각 단계에서 소셜 미디어가 할 수 있는 역할에 대해 서술하였다.

Storey와 동료들은 앞선 연구보다 더 구체적으로 소셜 미디어가 소프트웨어 개발에 있어서 적용될 수 있는 부분과 실제 도구들을 제안하였다[7]. 그들은 코드 편집기나 이슈 추적기 또한 소셜 미디어의 활용이라고 볼 수 있다고 서술하며, 소프트웨어 개발에 있어서 소셜 미디어를 활용하는 사례를 늘고 있으나 그에 따른 연구는 아직 미비한 실정이라고 주장한다. 따라서 소셜 미디어 활용에 따른 혜택, 위험과 제약을 밝히는 것을 목표로 하여 연구를 진행하였다. 저자들이 주장한 바에 의하면 개발과정에서 활용되는 소셜 미디어 요소는 wiki, 블로그, 마이크로 블로그, 태그(tagging), 피드(feed), 소셜 네트워킹, 매쉬업(mashup), 크라우드 소싱(crowdsourcing) 등이 있다. 이러한 소셜 미디어 요소들은 이를 통해 1) 요구사항을 통합하고, 검수 과정을 도우며, 최종 사용자 개발을 지원함으로써 커뮤니티와 최종 사용자 참여를 지원하고, 2) 과업 구체화와 비형식적 의사소통, 지식의 흐름과 인식을 촉진함으로써 프로젝트 조율과 관리를 용이하게 한다는 점에서 개발에 긍정적인 효과를 줄 수 있다고 해당 논문에서는 주장한다. 그러나 앞서 언급하였듯이 상기 두 논문은 저자들의 주장을 실증적으로 뒷받침하지 못했다는 점에서 한계를 가진다.

기존 연구는 아이디어 제시 위주라 객관성이 떨어진다는 점 외에도, 실증적인 데이터에 기반하지 않았기 때문에 시스템 요소 수준의 구체적인 함의를 제시하지 못한다는 한계를 가진다. 이를 극복하기 위해 본 연구에서는 실제 사용되는 시스템 요소와 실제 프로젝트 성과 관의 관계를 살펴봄으로써 보다 구체적인 결과와 시사점을 제공할 수 있다.

3. Github소개

Github는 2008년 초 설립된 오픈 소스 소프트웨어 개발을 위한 소스 코드 호스팅 웹사이트이다. 분산형 버전 관리 시스템(Version Control System)인 Git을 차용하고 있어 빠르고 분산된 공동 개발을 가능케 하며, 호스팅을 위한 저장 공간 제공뿐만이 아니라 지리적으로 떨어진 개발자들 간 협업이 가능하도록 다양한 시스템 요소를 제공하여 협업을 지원하기 때문에, 온라인 협업 플랫폼이라고 볼 수 있다.

Github가 다른 전통적인 플랫폼들과 차별화되는 점은, 스스로를 소셜 코딩 플랫폼으로서 주장하는 데에 있다[2]. 실제로도 플랫폼 설계 시 기존 플랫폼과 다르게 특정 프로젝트를 관리하거나 기여하는 데에 필요한 의사소통의 채널을 프로젝트 내부에서 거의 다 감당할 수 있도록 통합하였고 (issue, pull request 게시판), 프로젝트 외부적으로도 트위터와 유사한 일방향 관계 형성 기능(watch, follow)을 지원함으로써 개발자들 간, 또 개발자와 프로젝트 간에 손쉽게 관계가 형성되도록 하였다(그림 1).

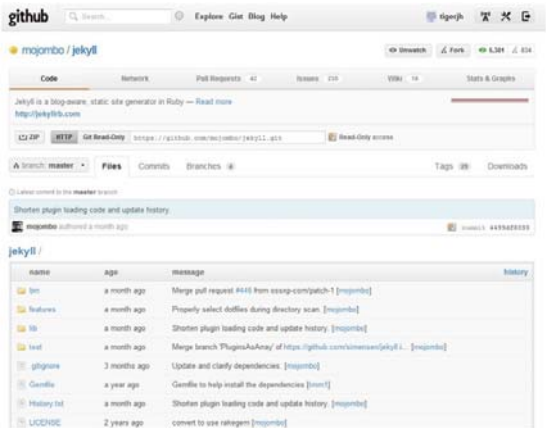


그림 1. Github의 프로젝트 화면 구성

특정 개발자 A가 다른 개발자를 follow하거나 다른 프로젝트를 watch할 경우, 해당 개발자와 프로젝트들의 모든 활동(activity)이 xml feed로서 개발자 A의 대쉬 보드에 전달된다(그림 2). 이러한 대쉬 보드는 개발자 A가 Github에 로그인하거나 접속하였을 때 보이는 기본 화면으로, 개발자는 손쉽게 다른 프로젝트나 개발자들의 작업 실황을 파악할 수 있다.

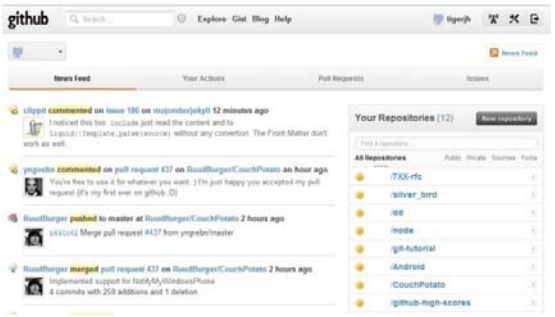


그림 2. Github 로그인 및 접속 시 보이는 기본 화면

4. 정성적 분석: 시스템 요소의 기능적 분류

4.1 시스템 요소 분석 방법

본 연구에서는 소셜 코딩 플랫폼인 Github와 Github가 제공하는 시스템 요소 및 기능을 더 체계적으로 이해하기 위해 주요 시스템 요소들을 기능에 따라 분류하고자 하였다. 2명의 연구자들이 2011년 4월부터 10월까지 약 6개월 간Github에서 실제 프로젝트 운영자와 참여자로서 참여 관찰하며 얻은 해당 플랫폼에 대한 지식과 실제 Github 이용자들인 7명의 전문가 인터뷰에 기반하여 해당 분석을 실시하였다. 인터뷰 참여자 중 6명은 전문 개발자였으며, 1명은 컴퓨터 과학 분야의 박사학위 소지자였다. 해당 참여자들의 Github 이용 연수는 평균 1.3년이며, 그 범위는 7.5개월에서 2년에 걸쳐 분포되어 있었다.

인터뷰 내용으로는 운영자 관점에서의Github 이용 경험과 참여자 관점에서의 Github 이용 경험, Github 외 다른 플랫폼의 이용 경험 및 비교, 소셜 컴퓨팅 시스템 요소들의 활용 목적 및 방법 등에 대해서 반구조화(semi-construced)된 인터뷰를 진행하였다. 진행 시간은 약 1시간 30분 가량이었으며, 모든 참여자에게는 금전적인 보상이 지급되었다.

이 밖에도 Github에 대한 외부 업체의 보고서 또는 Github 내 유명 프로젝트의 운영자가 직접 Github를 소개한 자료에 대해 문서 분석(document analysis) [14]을 실시함으로써 연구자의 주관이 지나치게 개입하지 않도록 방지하였다.

4.2 시스템 요소 분석 결과

분석 결과, Github가 제공하는 전체 시스템 요소 중 프로젝트 운영에 직접적으로 관여하는 요소로는 다음과 같은 것들이 있다: 1) 특정 프로젝트에 대한 정보(feed)와 알림을 받아볼 수 있는 Watch, 2) 특정 개발자에 대한 정보와 알림을 받아볼 수 있는 Follow, 3) 특정 프로젝트의 소스 코드를 임의로 수정, 추가 등 변경하고 싶을 때에 개인의 계정으로 해당 프로젝트의 코드를 복사(clone)해 갈 수 있는 Fork, 4) 특정 프로젝트 내 자유게시판 역할을 하는 Issue 게시판, 5) 특정 프로젝트에 기여하고자 하는 사람들이 자신이 수정하거나 새롭게 구성한 코드를 받아달라고 주인에게 요청하는 게시판인 Pull Request 게시판이 그것이다. 그 밖에 5) 오픈 소스 프로젝트들에게 매우 중요한 외부 참여를 의미하는 기여(commit) 횟수, 7) 참여자(contributor) 수가 있다.

위와 같은 7가지 중요 시스템 요소들은 크게 3 가지로 분류되었다. 이는 본 연구에 참여한 개발자들의 인터뷰 내용에 기반한 것으로, 각 요소 별 실제 인터뷰 내용은 이하와 같다(표 2). 표 3은 참가자들의 Github 경험에 대해 물었을 때 공통적으로 등장한 시스템 요소를 먼저 도출하고 이를 각 요소들의 이용 목적에 따라 분류한 것이다(표 3). 해당 분류는 Grounded theory에 의거하여 2명의 연구자에 의해 분석되었다[15]. 추가적으로 분석 결과의 타당성에 대하여 4명의 전문가 토의를 수행하였다.

표 1. 각 요소에 대한 인터뷰 내용

요소 분류	내용	참가자
Communication	협업하는데 중요한 게 Pull Request니까. 사실, Pull 받아서, Pull 서로서로 주고 받을 수 있잖아요? 서로서로 주고 받는다는 게 Pull 옆에 Issue 달리죠, Issue로 막 토론하다가 오케이 하면 발행되는 거고 아님 안 되는 거고 이렇게 하니까 되게 자유로운 거 같아요	P3
	이슈 트랙킹하는 것도 포함되어 있으면 그럼 거기다가 등록해놓으면 그 사람이 뭐 답변해주고, 다른 개발자들도 또 거기에 자기는 어떻게 해결했다, 라던지 자기는 이렇게 compile했다, 라던지 남겨두는 분도 있고요	P4
	제가 만든 소스를 Pull Request를 하는 것뿐만 아니라, 아까처럼 단순히 아이디어 제시를 해도 되요, 아 이런 아이디어 있는데 이슈로 등록하는 거죠, 그럼 이 사람들은 이슈를 보고 자기네들이 코딩을 합니다	P5
Code	Fork 같은 경우는.. 오픈 소스 프로젝트 뭐 카피해가지고 그 카피한 지점부터 다시 만드는 거에요. (…) fork를 했으면 commit할 의향이 있는 거에요. 뭔가 했을 거에요.	P1
	주요 개발자를 중심으로 참여 프로젝트들의 pull request와 commit을 주로 눈 여겨 봅니다	P2
	원래 프로젝트에서 할 수 없는 일들을 하기 위해서 프로젝트를 쪼갠 거죠, 새로운 뭔가를 하기 위해서, 그것이 포크죠.	P6
	Fork는 프로젝트에 직접 손을 대겠다고 생각하면 Fork를 하구요, (…)	P7
Inter-project 연결 지원	메일링 리스트로 작업하던 환경보다 하나의 타임라인에서 볼 수 있는 점은 프로젝트의 흐름을 파악하는데 도움이 되더군요	P2
	(각 프로젝트의 주요 시스템 요소를 들자면) 먼저, 포크나 와치 먼저 보고요 (…) Watch를 쓴 거는 처음에는 정말 그 프로젝트가 관심 있어서 본 게 Watching이죠	P3
	Watch는 프로젝트를 트렉킹하기 위해서, 언젠가 한 번 보기 위해서 하는 거고.	P4
	(watch나 follow가 feed로 전달되어 오는 시스템이) 내가 참여하고 있는 프로젝트를 관리하는데 있어서는 굉장히 중요한 기능이라고 생각하구요. (…) 다른 사람들의 프로젝트를 watch하는 것은 제가 패치를 쓰고 있는데 중간에서 패치가 계속 바뀌고 있는데 패치작업을 하면 사실 나중에 버그 작업을 해야 되고, 피곤한 일이 많이 생길 수 있잖아요. 그래서 제 때 제 때 변경점을 파악하고 나랑 비슷한 문제를 처리하고 있는 이슈 같은 것이 또 생기나 볼 수 있는 면에서 (중요하다고 생각합니다) (…) 전반적으로 어떤 식으로 그 사람이 최근에 작업 경향이 어떻게 흘러가고 있는지 주로 사람 위주로 작업 패턴을 파악하기 위해서 Follow를 활용하고 있습니다.	P6

표 2. 시스템 요소의 분류

	시스템 요소 및 행위 요소
Communication 관련	Issue, Pull Request
Code 기여 관련	Fork, Commit, Contribute
Inter-project 연결 지원	Watch, Follow

5. 정량적 분석: 시스템 요소 별로 프로젝트 성과에 미치는 영향에 대한 실증적 연구

5.1 샘플 선정 및 데이터 수집

Github 내 프로젝트들의 시스템 요소 이용과 결과물의 질 간의 관계를 알아보기 위해 다중 회귀 분석을 실시하였다. 전수조사를 위해 사이트 내 가장 많이 이용되는 언어인 JavaScript

[16]를 주로 사용하는 프로젝트 전체 (약 15만개 프로젝트)를 일차적인 분석 대상이자 단위로 보았다. 그 중, 본 연구의 목적인 소셜 컴퓨팅 요소가 협업에 미치는 영향을 알아보기 위해 프로젝트 주인(owner) 외에 최소 1인의 참여자가 기여한 기록이 있는 프로젝트를 분석의 대상으로 삼았다. 그 결과는 표 1과 같다. 덧붙여, 중요 시스템 요소로 분류된 pull request의 기록이 공개적으로 남아있는 2010년 9월 1일 이후 생성된 프로젝트들 수집하였다.

2011년 9월 기준 Javascript 프로젝트 전체의 수는 148,960개였고, 이 중 2010년 9월 1일 이후 생성된 프로젝트는 104,838개로 전수의 70.3%를 차지하였다. 오염 요소를 제거하기 위해 파생 프로젝트 (forked repository)는 배제하였다. 파생 프로젝트의 경우 파생 시점의 원 프로젝트 fork 횟수를 그대로 가져가기 때문에 분석의 신뢰도를 낮출 수 있기 때문이다. 본 작업을 수행한 후 남은 프로젝트 수는 49,643개로, 전수의 33.3%였다. 여기서 주인 이외에 최소한 1회의 기여를 의미하는 기준들(프로젝트를 watch하는 사람과 fork하는 사람 수가 2명 이상일 것, 프로젝트를 fork해 가서 수정한 후 pull request를 보낸 사람이 1명이라도 있을 것, pull request 중 실제로 프로젝트에 반영되어 해당 코드 작성자가 참여자contributor로 최소 1회 등록된 프로젝트일 것)을 적용하여 총 2,034개의 프로젝트를 얻었다. 마지막으로 중요 소셜 컴퓨팅 요소인 follower의 수가 미치는 영향을 고려하기 위해 개인 계정의 주인이 운영하는 프로젝트만 분석에 포함하였다. 최종적으로 선정된 프로젝트의 수는 1,695개이며, 이는 전수의 1.1%에 해당한다.

표 3. 분석 대상 프로젝트 선정

해당 프로젝트	개수	비율
JavaScript 총 Repo 수	148,960개	
2010년 8월 31일 이후 생성 Repository의 수	104,838개	70.3%
Forked repository 제외한 Core repository	49,643개	33.3%
Watcher < = 2	17,412개	11.7%
Forker < = 2	8,194개	5.5%
1회 이상의 Pull Request	2,783개	2%
2명 이상의 참여자	2,034개	1.4%
개인 계정의 주인이 운영하는 프로젝트	1,695개	1.1%

선정된 프로젝트들의 결과물 질(quality) 척도로서는 오픈소스 코드 평가 도구인 SONAR[17] 결과를 이용하였다. SONAR에 각 프로젝트의 소스 코드를 입력하고 평가하였을 때 나오는 결과인 Major Violation(MV)를 다중 회귀 분석의 종속 변수로 보았다. Major Violation 점수는 해당 프로그램의 얼마나 많은 코드들이 코드 개발 시 권장되는 지침(rule)을 위배하였는지를 가리킨다[18].

종합하면, 개별 프로젝트의 Watch, Follow, fork, commit, contributor, 총 issue 수, 총 pull request 수를 독립 변수로, SONAR 결과인 MV를 종속 변수로 하여 이를 총 1,695개 프로젝트 대상으로 실시하였다.

표 4. 다중 회귀 분석 결과

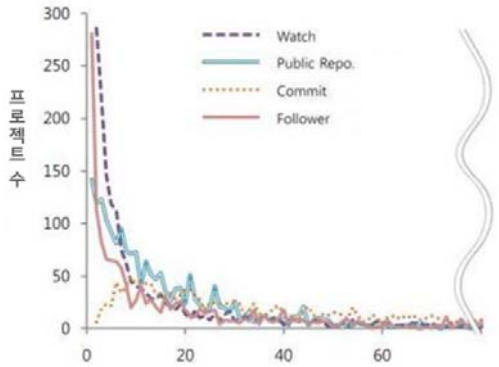
	Major Violation
수정된 R^2	.188
잔차 (β_0)	1.797
Watch	−.196**
Follow	−.104**
Fork	.194**
Commit	.453**
Contributor	−.031
Issue	−.019
Pull Request	−.049

*(p < 0.05, **:p<0.01)

5.2 샘플 프로젝트의 분포 양상

선정된 2,034개의 샘플 프로젝트들에 대해 중요 시스템 요소 및 행위 요소의 이용 정도를 조사한 결과, 그림 3과 같은 분포를 보였다.

그림 3에서 x축은 앞서 언급한 주요 시스템 요소의 이용 수를, y축은 프로젝트 수를 의미한다. 기존 오픈 소스 소프트웨어 개발 문헌들에서 프로젝트 성공의 중요한 요소로서 언급한 참여자 수의 분포[19]는 다른 플랫폼과 같이 Github에서도 변함없이 롱테일 분포를 그리는 것으로 나타났다. 이 밖에 다른 소셜 컴퓨팅 요소들의 이용 정도 또한 참여자 수의 분포와 같이 매우 오른쪽으로 왜곡된 그래프 형태를 보였으며, 이는 해당 플래폼 내 프로젝트들에 대한 참여 구조는 다른 플랫폼에 비해 Github에서 개선된 바가 없음을 의미한다.



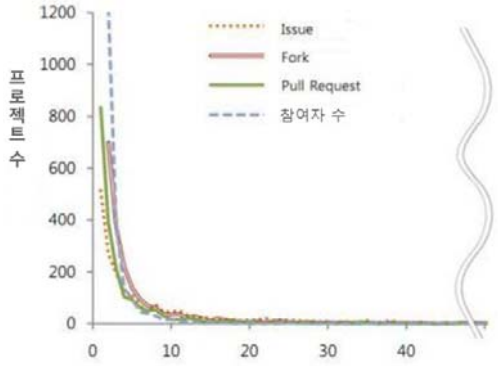


그림 3. 각 요소의 이용 분포

5.3 다중 회귀 분석 (Multiple Linear Regression)

다중 회귀 분석 결과는 표 4와 같다. 이하와 같은 조건들이 만족됨에 따라 전체 회귀 모형의 유효성이 검증되었다: Durbin- Watson 값이 2에 가까운 1.948, 공선성 검사 결과인 VIF 값이 10 이하, F test가 $p<0.01$ 수준에서 유의한 결과가 도출되었다($F = 51.831$) [20]. 시스템 요소 분석과 연결 지어 보았을 때, 프로젝트 간(inter-project) 연결을 지원하는 시스템 요소인 Watch와 Follow는 소스 코드의 violation와 부적 관계에 있었고, 코드 기여와 관련된 Fork, commit과 같은 요소는 violation와 정적 관계에 있었다. 그 밖의 요소들은 소스 코드의 violation과 유의미한 관계가 없는 것으로 밝혀졌다.

각 요소들에 대한 표준화된 베타 계수를 비교하여 그 관계 강도를 비교한 결과, commit 횟수가 가장 큰 관계성을 보였으며 (.453) watch (.196) 및 fork (.194)가 그 다음으로 큰 영향 관계에 있는 것으로 나타났다. 마지막으로 follow의 수는 .104 정도의 베타 계수를 가졌다. 기존 연구들에서 중요한 요소로 이야기되던 참여자(contributor)의 수는 최종 결과물과 유의한 관계를 가지지 않았다.

6. 논의

본 연구에서 발견된 흥미로운 사항은 프로젝트 내부 참여 지원 요소들(commit과 fork의 수)과 외부 프로젝트 간 참여 지원 요소(watch, follow 수)가 프로젝트 성과에 서로 반대되는 영향을 보였다는 점이다. 더 나은 결과물을 위해 더 많은 개발자들이 지원해야 한다는 오픈 소스 소프트웨어 개발 프로젝트들에 대한 전통적인 믿음 [21, 22]과는 반대로, 외부 프로젝트 간 참여를 지원하는 소셜 컴퓨팅 요소를 사용한 프로젝트들에서 더 적은 규율 위반 (violation)이 있었다. 이때 외부 프로젝트 간 참여 지원이라 함은, 해당 요소를 이용함으로써 다른 프로젝트에는 어떤 일이 발생하는지에 대한 인식(awareness)을 가능하게 하는 기능을 말한다. 더 많은 개발자가 코드를 기여할 수록, 소프트웨어는 그 크기 (LoC) 측면에서는 증가하지만 동시에 더 많은 규율 위반 또한 이루어지는 것을 살펴볼 수 있었다.

본 연구자들은 상기와 같이 관찰된 패턴이 Github의 소셜 컴퓨팅 이용의 결과라고 주장한다. 대부분의 오픈소스 소프트웨어 개발 플랫폼들은 특정 프로젝트 내부에서 협업을 하도록 지원하는 도구를 제공하는 데에 초점을 맞추었지만, 프로젝트 간의 상호작용과 노출까지 지원하지는 않았다[23]. Github에서는 소셜 컴퓨팅 요소 이용을 통해 각 개발 프로젝트 간의 투과성이 높다. 소셜 컴퓨팅 요소들이 해당 프로젝트의 전체 개발 과정을 단순히 공개할 뿐만 아니라 의도적으로 전달하고, 그럼으로써 해당 프로젝트에 직접적으로 참여하지 않은 개발자들에게도 높은 접근성을 제공하기 때문이다. 뿐만 아니라, Github의 느슨한 경계성(loosely bounded)은 개발자들로 하여금 더 자유롭게 프로젝트 간을 오가고, 과거 일반적이었던 프로젝트 '가입' 과정을 생략할 수 있도록 하였다[24].

이에, 우리는 전 오픈 소스 소프트웨어 커뮤니티에 걸쳐 개발자 활동 및 프로젝트 개발 과정의 가시성을 높여주는 소셜 컴퓨팅 요소를 이용함으로써 프로젝트들과 개발자들 간에 의도되지 않은 흥미로운 시너지를 창출할 수 있다고 보았다. 기존 오픈 소스 소프트웨어 개발 연구들이 프로젝트 내부에서 발생하는 활동들을 조율하는 사회적 상호작용에만 초점을 맞춘 반면, Github는 프로젝트 간 공동의 인식(awareness)를 촉진하는 사회적 시스템 요소를 제공한다. 이에 연구자들은 후자의 시스템 요소가 특정 개발자가 직접 운영하는 프로젝트를 넘어 다른 프로젝트들에 대한 사회적 인식을 증대시킬 것이라고 제안한다. 또한, 이렇게 증대된 사회적 인식이 오픈 소스 소프트웨어 개발 프로젝트들과 개발자들 간의 관계망에 대한 가시성을 증폭시킴으로써 프로젝트들의 성과(performance)에도 영향을 미칠 것이라고 주장한다. 이러한 효과는 Github 관련 웹 자료에서도 유사하게 제시된 바 있다[25, 26]. 저자들은 본 논문에서 주장한 기제가 타당하다고 믿으나, 오픈 소스 소프트웨어 개발 프로젝트들이 소셜 컴퓨팅 요소를 사용함으로써 어떤 수혜를 입는지에 대해 더 많은 연구가 필요하다는 점에 동의한다. 특히 프로젝트 간 공동의 인식을 지원하는 시스템 요소의 지원에 대해 추가적인 연구가 필요하다.

본 연구는 현재 오픈 소스 프로젝트 호스팅 사이트 중에서도 Social Coding이라는 기치 하에 유일하게 사회적 상호작용에 중점을 두고 있는 Github를 대상으로 사례 연구를 진행한 것이다. 따라서 본 연구에서 분석 대상으로 삼은 Github 플랫폼의 세부 기능을(watch, follow, fork, commit, pull, issue 등) 타 플랫폼을 대상으로 한 분석에 바로 적용하기에는 무리가 있다. 특히 watch, follow 요소는 Github의 독창적인 기능으로, 유사 기능을 제공하는 플랫폼은 매우 드물뿐만 아니라 Github 처럼 프로젝트 페이지 전면에 해당 요소를 노출시키는 경우는 없다. 그러나 세부 기능에 기반하여 도출한 3가지 분류(Code, Communication, Inter-project awareness)는 Github 사용자들의 이용 행태를 일

반화 하여 도출한 분류(categories)이므로 [15], 향후 오픈 소스 플랫폼에 대한 연구에도 적용 가능하다.

종합하면, 본 연구는 이하와 같은 이론적, 실용적 의의를 가진다. 이론적인 의의는 github라는 대표적인 소셜 코딩 플랫폼에서 소셜 컴퓨팅 요소와 프로젝트 결과물 간의 상관 관계를 실증적으로 파악했다는 점에 있다. 소셜 코딩이라는 소셜 컴퓨팅의 새로운 도메인을 실증적으로 연구함에 따라 이론적 확장을 이루었다고 볼 수 있다. 소셜 컴퓨팅 요소 중 프로젝트 간 연결 지원 요소가 프로젝트 결과물인 소스 코드의 복잡성을 낮춤으로써 긍정적인 기여를 한다는 점은 실용적, 이론적으로 모두 중요한 발견이다. 실용적 의의이자 HCI 관련 종사자들에게 제안할 수 있는 점으로는, 오픈 소스 개발 방식을 도입하고자 하는 개인 개발자 또는 기업에서 프로젝트의 성공을 위해 고려해야 할 요소로서 타 프로젝트와의 연계를 하는 것이 중요하다는 점을 실증적으로 밝혀냈다는 점이다. 비단 오픈 소스 개발뿐만 아니라 온라인에서 벌어지는 개방형 협업을 위한 플랫폼을 구성하는 데에도 소셜 컴퓨팅 요소의 도입을 긍정적으로 고려해 보아야 함을 본 연구의 결과로부터 도출할 수 있다. 또한, 본 연구에서는 소셜 컴퓨팅 요소가 협업 성과에 미치는 과정의 자세한 기제 및 다양한 플랫폼에서의 소셜 컴퓨팅 요소의 효과에 대한 연구 필요성을 제안함으로써 향후 HCI 연구의 주제를 제안하였다.

참고문헌

[1] Wang, F. Y., Carley, K. M., Zeng, D. and Mao, W. Social computing: From social informatics to social intelligence. Intelligent Systems. IEEE. 22(2):79-83. 2007.

[2] Github. http://www.github.com/

[3] Github now center of OSS development universe. http://www.itworld.com/software/170941/github-now-center-oss-development-universe.

[4] Raymond, E. The cathedral and the bazaar. Knowledge, Technology & Policy. 12(3):23-49. 1999.

[5] IDC's IT Forecaster, http://www.idc.com/itforecaster/itf20000808.stm 2000.

[6] Bonaccorsi, A. and Rossi, C. Why open source software can succeed. Research policy. 32(7):1243-58. 2003.

[7] Storey, M. A., Treude, C., van Deursen, A. and Cheng, L.T. In: The impact of social media on software engineering practices and tools. Proceedings of the FSE/SDP workshop on future of software engineering research; ACM; p. 359-64, 2010.

[8] Begel, A., DeLine, R. and Zimmermann, T. In: Social media for software engineering. Proceedings of the FSE/SDP workshop on future of software engineering research; ACM; p. 33-8, 2010.

[9] Parameswaran, M. and Whinston, A. B. Social computing: An overview. Communications of the Association for Information Systems. 19(1):37. 2007.

[10] Crowston, K., Annabi, H. and Howison, J. Defining open source software project success. Former Departments, Centers, Institutes and Projects. 4. 2003.

[11] Johnson, R. B. and Onwuegbuzie, A. J. Mixed methods research: A research paradigm whose time has come. Educational researcher. 33(7):14-26. 2004.

[12] Schuler, D. Social Computing," Comm. ACM. vol. 37. no. 1. 1994. pp. 28-29.

[13] Tuckman, B. W. and Jenson, M. A. C. Stages of small group development revisited. Group and Organizational Studies. 2(4):419-427. 1977.

[14] Hoepfl, M. C. Choosing qualitative research: A primer for technology education researchers. Journal of Technology Education: http://scholar.lib.vt.edu/ejournals/JTE/v9n1/hoepfl.html 1997.

[15] Glaser, B. G. and Strauss, A. L. The discovery of grounded theory: Strategies for Qualitative Research. Aldine de Gruyter; 1967.

[16] Top languages in Github, https://github.com/languages

[17] Sonar, www.sonarsource.org

[18] Kolawa, A. K., Aivazis, M. A. G., Hicken, W. T. and Strickland, B. R. Method and system for automatically checking computer source code quality based on rules. 1999.

[19] Lerner, J. and Tirole, J. Some simple economics of open source. The journal of industrial economics. 50(2):197-234. 2000.

[20] Aiken, L. S., West, S. G. and Pitts, S. C. Multiple linear regression. Handbook of psychology. 2003.

[21] Crowston, K., Annabi, H. and Howison, J. Defining open source software project success. Former Departments, Centers, Institutes and Projects. 4. 2000.

[22] Von Krogh, G. and Spaeth, S. The open source software phenomenon: Characteristics that promote research. The Journal of Strategic Information Systems. 6(3):236-53. 2007.

[23] Bagozzi, R. P. and Dholakia, U. M. Open source software user communities: A study of participation in linux user groups. Management

Science. 1099-115. 2006.

[24] Von Krogh, G., Spaeth, S. and Lakhani, K. R. Community, joining, and specialization in open source software innovation: A case study. Research Policy. 32(7):1217-41. 2003.

[25] Collaborative Github Workflow. http://www.eqqon.com/index.php/Collaborative_Github_Workflow 2011.

[26] Hamano, J. C. In: GIT - stupid content tracker. Proceedings of the linux symposium (ottawa, ontario, Canada). 2006.

최 주 희

2004년 3월~2010년 8월 연세대학교 문헌정보학과 졸업. 2010년 9월~2012년 8월 연세대학교 대학원 인지과학 협동과정 졸업(HCI 전공). 관심분야는 HCI, open innovation, Crowd sourcing 임

최 정 홍

2004년 3월~2011년 2월 아주 대학교 미디어학과 졸업. 2011년 3월~현재 연세 대학교 대학원 석사과정. 관심분야는 인간-컴퓨터 상호작용, Collective Innovation임

문 재 윤

1992년 3월~1996년 2월 연세대학교 경영학과 졸업. 1996년 3월~1998년 2월 연세대학교 대학원 경영학과 졸업(경영학석사). 1998년 9월~2004년 1월 New York University PhD (Information Systems 전공). 2004년 1월~2009년 8월 Hong Kong University of Science and Technology 조교수. 2009년 9월 ~현재 고려대학교 경영대학 부교수.