Association for Information Systems
AIS Electronic Library (AISeL)

ICEB 2017 Proceedings | International Conference on Electronic Business

Winter 12-4-2017

GitHub: Factors Influencing Project Activity Levels

Mohammad Azeez Alshomali
James Cook University, Australia, Mohammada.Abdulhassan@my.jcu.edu.au

John R. Hamilton
James Cook University, Australia, John.Hamilton@jcu.edu.au

Jason Holdsworth
James Cook University, Australia, Jason Holdsworth, James Cook University, Australia, jason.holdsworth@jcu.edu.au

SingWhat Tee
James Cook University, Australia, singwhat.tee@jcu.edu.au

Alshomali, M.A., Hamilton, J.R., Holdsworth, J. & Tee, S.W. (2017). GitHub: Factors influencing project activity levels. In *Proceedings of The 17th International Conference on Electronic Business* (pp. 116-124). ICEB, Dubai, UAE, December 4-8.

GitHub: Factors Influencing Project Activity Levels

Mohammad Azeez Alshomali, James Cook University, Australia, Mohammada.Abdulhassan@my.jcu.edu.au
John R. Hamilton, James Cook University, Australia, John.Hamilton@jcu.edu.au
Jason Holdsworth, James Cook University, Australia, Jason.Holdsworth@jcu.edu.au
SingWhat Tee, James Cook University, Australia, SingWhat.Tee@jcu.edu.au

ABSTRACT

Open source software projects typically extend the capabilities of their software by incorporating code contributions from a diverse cross-section of developers. This GitHub structural path modelling study captures the current top 100 JavaScript projects in operation for at least one year or more. It draws on three theories (information integration, planned behavior, and social translucence) to help frame its comparative path approach, and to show ways to speed the collaborative development of GitHub OSS projects. It shows a project's activity level increases with: (1) greater responder-group collaborative efforts, (2) increased numbers of major critical project version releases, and (3) the generation of further commits. However, the generation of additional forks negatively impacts overall project activity levels

Keywords: GitHub, open source, social media content, popularity, software repository, JavaScript

*Corresponding author

INTRODUCTION AND MOTIVATION

The on-line, open-source software development environment GitHub hosts, attracts, and builds collaborative social coding communities that have chosen to contribute into selected, but controlled, public (free) project repositories.
GitHub is currently the 'absolute dominant' data source for open source software (OSS) data mining research (Cosentino, Izquierdo & Cabot, 2017). It combines traditional capabilities including free hosting and version control with social features (Squire, 2014). Moreover, GitHub supports rapid software development, and has collaborative project features including bug-tracking, feature-requests, task-management and Wikis (Marlow, Dabbish & Herbsleb, 2013; Williams, 2012).

Researchers note that GitHub projects vary in their collaborative activities. Such variations depend on project commits (Yu *et al.*, 2014). Often, pull-requests (successful and unsuccessful) telling others of changes pushed into a GitHub repository, stimulate further activity to solve development issues. These often present through merged commits. Over-time, more pull-request merged commits add to the net project activity level within the GitHub repository ecosystem (Xavier & Macedo, 2014).
As a measure, the number of committers does not match each commit - since around 15% of committers are either non-collaborators, or committers who use alternate emails to lodge their commits (Kalliamvakou *et al.*, 2016). Although the number of committers do contribute to the project's activity level (Luo, Mao & Li, 2015), they also can vary in measurement accuracy, and so they are not used in this measurement study.

Over-time the number, and frequency of project version releases, also affects project activity levels. As a project nears a release its activity level first increases exponentially (like a bell-curve) towards the release date, and then rapidly drops after the release date (Cosentino, Izquierdo & Cabot, 2017). Thus, the number of releases alters, and cyclically affects, the project's activity level. Other GitHub studies gauge various aspects of project activity levels (Bissyande *et al.*, 2013; Borges, Hora & Valente, 2016; Capra *et al.*, 2011) (Mileva, 2012; Sajnani *et al.*, 2014; Tsay, Dabbish & Herbsleb, 2014; Zhu, Zhou & Mockus, 2014). Each approach first adopts some form of clustering, possibly including programming language, duration, size, and social connections. This clustering allows each resultant data set to be studied within a chosen modelling and/or coding and/or mathematical approach.

Popularity is gauged by (Aggarwal, Hindle & Stroulia, 2014; Borges, Hora & Valente, 2016; Ma *et al.*, 2016; Xavier & Macedo, 2014), and others against: (1) number of stars, (2) forks, (3) pull-requests and (4) watchers. In addition, popularity also relates to a project's activity level (Cosentino, Izquierdo & Cabot, 2017). To data previous studies do not provide a holistic view of the constructs affecting a project's activity level within the GitHub repository ecosystem. In this study's context, project activity level is the combined (holistic) level of all of the popularity contribution measures added into the GitHub project.

Hence, this study establishes a framework to capture the key contributors (constructs) and their relative GitHub project activity level relationships. Understanding the total effects of each of these key GitHub repository ecosystem contributors then allows a project creator, and their core team, to pursue ways to: (1) draw further OSS developers into this project, (2) induce higher project activity levels, and (3) shorten the time between project release versions.

GITHUB ECOSYSTEM

GitHub projects are diverse in: format, project-size, development-cycle-stage, release-count, change-frequency, and changeability. GitHub houses over 20M users and 57M repositories. It draws worldwide crowd-sourced coding contributors, each with unique individual levels of expertise, into an environment that allows adding value to its large number of ongoing software development projects (Tsay, Dabbish & Herbsleb, 2014).

Coding additions/deletions occur through a series of commits by repository collaborators that update a software codebase. Collaborating and external developers providing pull-request merged commits, are first reviewed and tested by other repository collaborators before their project code is merged into the main repository codebase. These collaborators are usually a core team of developers for this repository. Thus, the project's creator and its core team of collaborators, can be thought of as the ongoing guardians of repository quality (Yu *et al.*, 2014). The activeness of a repository's creator in handling pull-requests also influences the extent of pull-request activities by the ecosystem (Aggarwal, Hindle & Stroulia, 2014).

A pull request termed 'fork-pulls' is embedded within, and emanates from exiting repository forks. Fork-pulls can loop back into the fork with potential subset solutions. A visual scan suggests fork-pulls are generally numerically less than master projects pull-requests. Furthermore, fork-pulls tend to occur later into the project's development. Hence, this study does not focus on fork-pulls.

The project's creator and its core team of collaborators also house and organize the repository's source code documentation, including 'readme' files. Readme files are continually updated so coding contributors can select current problem areas aligned towards their coding capabilities (Zhu, Zhou & Mockus, 2014). The clarity of the source code, and its precision in documentation, encourages greater activity into the project, and small documentation improvements can deliver great benefits (Henderson, 2009).

Figure 1 presents the ecosystem of contributors to a GitHub software repository. This ecosystem supports and increases the capacities and capabilities of the project originator and their core teams.

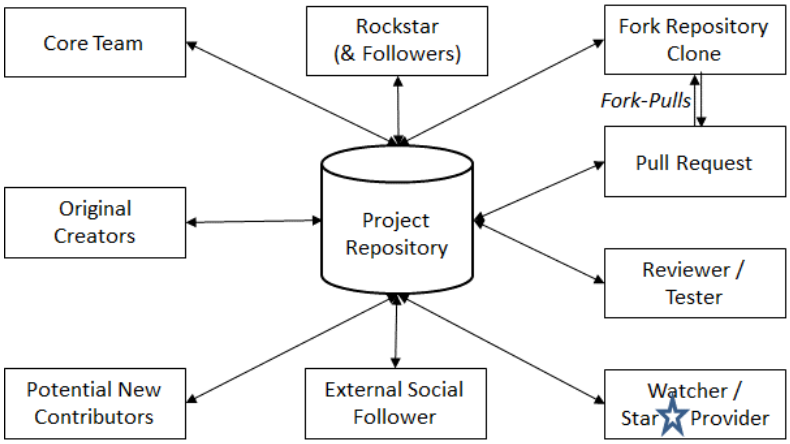

Figure 1: GitHub software development code contribution ecosystem.

Lee *et al.* (2013) see 'Rockstars' as star contributors whose popularity brings into a GitHub project ecosystem additional groups of skilled code-related followers. These additional groups often follow their Rockstar's focussed lead, and typically generate further Rockstar followers' pull-request activities within the project. This presence of a Rockstar group likely results in greater popularity along with enhanced project coding outcomes. A Rockstar is also a benchmark with easy project access (Ma *et al.*, 2016). Other individuals who generate high quality code or project contributions may also be recognized as Rockstar contributors.

The 'Fork-repository-clone' group is another indication of the project's popularity. The more forks a project has, the more likely the repository is recommended, and the higher the chance to increase the activity of potential code contributions into the project (Zhu, Zhou & Mockus, 2014). Forks sometimes generate strong changes in direction, new features, better implementation approaches, or even a different version of the existing project, whilst still visioning around the original project (Ma *et al.*, 2016).

'Reviewers/testers' discuss, assess, and recommend each contribution's merging (or rejection) within the project. When reviewers are specifically assigned the review or testing process becomes shorter and more effective (Yu *et al.*, 2014).

A 'Watcher/star-provider' receives notifications of any event (commits, pull-requests, and issues) arising within the project and on GitHub's social media (Ma *et al.*, 2016; Sheoran *et al.*, 2014). It is also common to see popular projects where coding

activities are seen to be successful as being 'starred' extensively, and experiencing higher commit frequencies (Cosentino, Izquierdo & Cabot, 2017). Watchers tend to contribute to popularity by their external activities on social media, and other digital community forums.

'External-social-followers' track the actions of other coding developers of good reputation (Luo, Mao & Li, 2015). Dabbish *et al.* (2012) note GitHub's External-social-follower and Tester/Watcher groups each contribute transparency into a project. They also bring additional social considerations, and their social actions can contribute towards the project's popularity.

'Potential-new-contributors' can be drawn into a GitHub project by: (1) current promotional activities, (2) social media, and/or Twitter, and/or Wiki awareness campaigns, (3) following others, (4) a desire to code, and (5) sourcing a personal area of interest.

The 'Project-repository' houses the software codebase along with various ongoing development streams (branches) as well as Wiki, readme, and other contributions. Many software develops a regard GitHub repository as a professional platform where to host their own projects or find other interesting open-source software projects (Wu *et al.*, 2014). Key GitHub programming languages are either web-focused (JavaScript, Ruby, PHP, CSS) or system-oriented (C, C++, Python). JavaScript, Java, and Python are the top three GitHub programming languages (Cosentino, Izquierdo & Cabot, 2017).

Github Projects

GitHub projects are diverse in: format, project-size, development-cycle-stage, release-count, change-frequency, change-degree, forks, watchers, and contributor-skills (Aggarwal, Hindle & Stroulia, 2014). Such potentially diverse project variations can also complicate project comparisons.

When comparing relationships within and around GitHub projects (Aggarwal, Hindle & Stroulia, 2014; Cosentino, Izquierdo & Cabot, 2017) further divide different projects. Their specific categories include: (1) popularity delivering higher/consistent documentation or (2) library projects needing less documentation. Over time, documentation quality improves especially in larger projects and as responders (reporters or assignees) become more experienced (Cosentino, Izquierdo & Cabot, 2017; Xavier & Macedo, 2014). Thus comparative longitudinal GitHub studies remain complex.

GITHUB MEASUREMENT CATEGORIES

Some of the measurement instruments available to GitHub researchers include:

- Project-type: GitHub projects range from major corporate software developments such as Adobe bracket, or Facebook that incorporate forks when overcoming issues and/or when speeding new release versions, through to small core creator / developer projects.
- Duration-of-project: Large GitHub projects tend to remain active, forked, retain interest and be long-term ongoing operations (Cosentino, Izquierdo & Cabot, 2017).
- Project-measures: GitHub measures commits, committers, software-releases, popularity-of-project, number-of-stars-provided, forks, watchers, followers, testers, and reviewers.
- Project-language: Key common GitHub software languages (discussed above) draw like-skilled programmers, and are more likely to retain project communities in excess of 40 (Cosentino, Izquierdo & Cabot, 2017).
- Readme files: 95% of popular GitHub projects have non-empty readme files (Tsay, Dabbish & Herbsleb, 2014).

GitHub popular projects typically engage forking. They also show clearer, more-consistent documentation advice (Aggarwal, Hindle & Stroulia, 2014), and useful documentation can draw-in other coding contributors (Hata *et al.*, 2015). This documentation may also be supported by testing mechanisms (Tsay, Dabbish & Herbsleb, 2014), Wikis (McDONALD *et al.*, 2014), Twitter (Singer, Figueira Filho & Storey, 2014), social media and websites (Jiang *et al.*, 2017).

GITHUB STUDY

Theoretical Basis

GitHub is recognized in (Wu *et al.*, 2014) study as a professional platform where software developers can: (1) host their own project, or (2) contribute towards other interesting projects, or (3) keep informed regarding what their peers are coding. GitHub's repository projects are typically not developed by individuals, but by a community of coders and associates working collaboratively. Hence, the more active the community project becomes, the quicker it progresses towards task completion (coding, documentation, and discussion).

The 'Theory of Social Translucence' suggests a clear awareness of a project and its design strategies, is advanced where a coherent behavior occurs through the visible sharing of each project collaborator's identity, contributions, and ongoing activities. These behavioral actions also occur within a GitHub project's community (Dabbish *et al.*, 2012). This transparency extends out to GitHub followers, watchers, and stars-provided, and it reaches into social supporting areas including Facebook, websites, Twitter, and Wikis (Aggarwal, Hindle & Stroulia, 2014). Thus, progress is likely quicker where coherent behavior is permeated across a GitHub project's community.

GitHub behaviorally ties developers into a project via 'Information Integration Theory' - which draws on the 'Theory of Planned

Behavior' (Ajzen, 1991). It links GitHub responder attitudes and their subjective norms (Ajzen, 1991) through their common beliefs and behaviors. These intentionally drive the overall behavioral (strength-of-belief) within their behaviorally-controlled project's domain. Thus, for projects of around the same size, same programming language and similar degree of complexity, the GitHub project's responder-tracking-measures likely related to its resultant activity level and to its overall development time.

Hence, this longitudinal study considers the GitHub project's activities as those contributing towards advancing the source code towards project solution. It poses the research questions:

- RQ1: do collaborative responder-group efforts drive GitHub JavaScript project activity?
- RQ2: do less time-to-release version completions drive GitHub JavaScript project activity

This study's approach incorporates the above theory and literature considerations. Cosentino *et al.* (2017) adds to this literature assessment - contributing that project activities levels are generated via multifaceted contributions that encompass the above literature. As almost all popular GitHub projects possess active readme files (Tsay, Dabbish & Herbsleb, 2014). This study also asks:

- RQ3: do more changes-to-documentation drive GitHub JavaScript project activity?

Study Considerations

To reduce GitHub's vast array of projects into a manageable and comparable scale, this study adopts a convenience approach. It focuses on longer-term and substantive software developments. Projects are included if they have been operational, and active, for at least one year's duration. Only JavaScript most popular projects are considered.

This study assesses GitHub project activity levels. It captures Rockstars and their followers through their inconsistent, but selective pull and fork contributions into each specifically-chosen project. It recognises that forks, pulls watchers and stars-provided are contributors towards a project's activity level. It compiles these contributors with the other construct intermediary measures defined above (commits, project-version-releases). It builds a GitHub path model that delivers a measurable project activity level outcomes. It determines each responder group contributing to the project's activity level through Figure 2's GitHub JavaScript framework model. These independent GitHub responder contributions offer the overall total effects pathways that contribute towards the project's overall activity level (Xavier & Macedo, 2014).

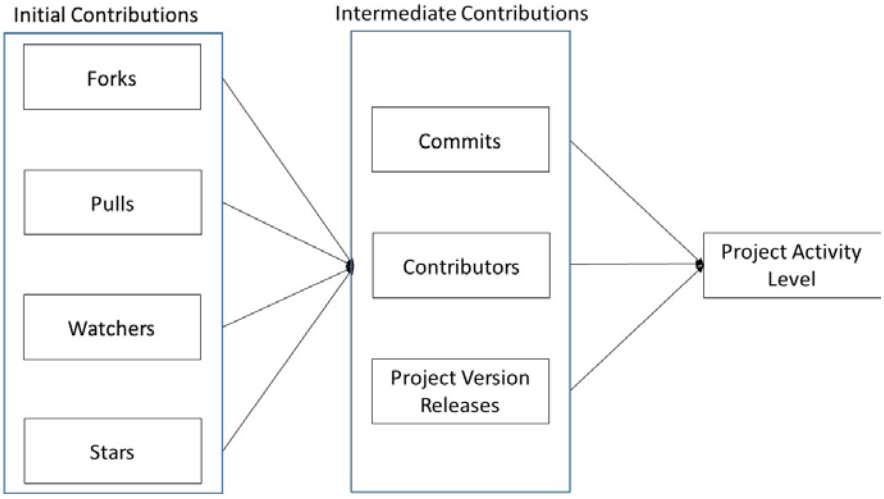

Figure 2: GitHub JavaScript framework model.

Study Approach

This study assesses Figure 2 for the current top 100 GitHub JavaScript projects as gauged against the level of forking. The most popular projects are typically extensively forked, and usually well-starred (Berry, 2015). Each fork copies the original project repository (https://github.com/popular/watched). Being outside the original project repository each fork: (1) allows free code experimentation, (2) develops proposed project changes, (3) generally feeds back to the original project repository, or (4) sometimes becomes a development initiation point for a new project idea (or code). Thus the level of forking offers a well-used approach to define top GitHub projects.

The study's structural path model combines the influence findings of such past studies. It models their relative input measurement effects against project activity level. Aggarwal *et al.* (2014) argue that small increases to existing documentation helps the growth of project activity levels, and helps reduce the project time to reach a product-life-cycle midpoint. Thus standardized total effects can be gauged against project activity levels.

METHODOLOGY

Data Capture

GitHub provides a web-based API for querying the raw information and statistics about GitHub repositories over time. The data extraction method used by this study is presented in Appendix 1. Table 2 illustrates data extraction collects the Figure 2 construct measures. Commits include source code and documentation addition/deletion counts. Contributors are summed counts of project forks, pulls, watchers and stars-provided. Product-version-releases are numerical counts into the project. Project activity level is the number of lines added or removed from repository files. These construct measures are exported to SPSS23.0 for statistical assessment and modelled via structural path analysis in AMOS 23.0.

Data Analysis

Table 1 shows extracted entries for the current top 100 GitHub JavaScript projects totalled 840Mb. All data set entries relate to project information contributions. Commits is the number of accepted changes.

Table 1: GitHub JavaScript data (top 100 projects).

Project Actions	Top Projects JavaScript	Project Actions	Top Projects JavaScript	Project Actions	Top Projects JavaScript)
Commits	336,181	Pulls	5,223	Additions	180,982,063
Contributors	19,762	Watchers	92,536	Deletions	95,764,624
Releases	7,167	Issues Open	26,117	Activity	279,876,894
Stars (provided)	1,800,049	Issues Closed	192,503	Owner	100 titles
Forks	454,640	Total Updates	276,746,687	Owner Repos	100 named

The committers are the individual entries contributing to the commit. Contributors are the different individuals adding something to the project. Stars-provided are the recognition starring provided by individuals in recognition of their general support for the project and its targets. A fork is a split from a project and it differs from its comparative pull request. A fork often works in response to a variant in thought (or approach) and a fork can sometimes heads off in a different direction taking some of the project's team resources with it. Pulls requests are rare as they represent attempts to re-work or adapt the code-base. Releases are also small in number. They occur when the project work is at a deliverable stage. Watchers observe progress of a project. They may or may not then contribute at a later time to the project.

Popularity has been used in the past by others to measure project activity level. However, popularity engages inconsistent models, and popularity is derived as a dependent regression. Hence, this study adopts to measure GitHub contributions as its measure of overall popularity, and terms this measure the 'project activity level.' The project activity level is a large term capturing all information around the lines of code added or deleted. data from GitHub's top 100 projects is summarized below in Table 1. The project activity level structural path model is developed in AMOS23.0 from Table 1's data set. It is displayed as Figure 3.

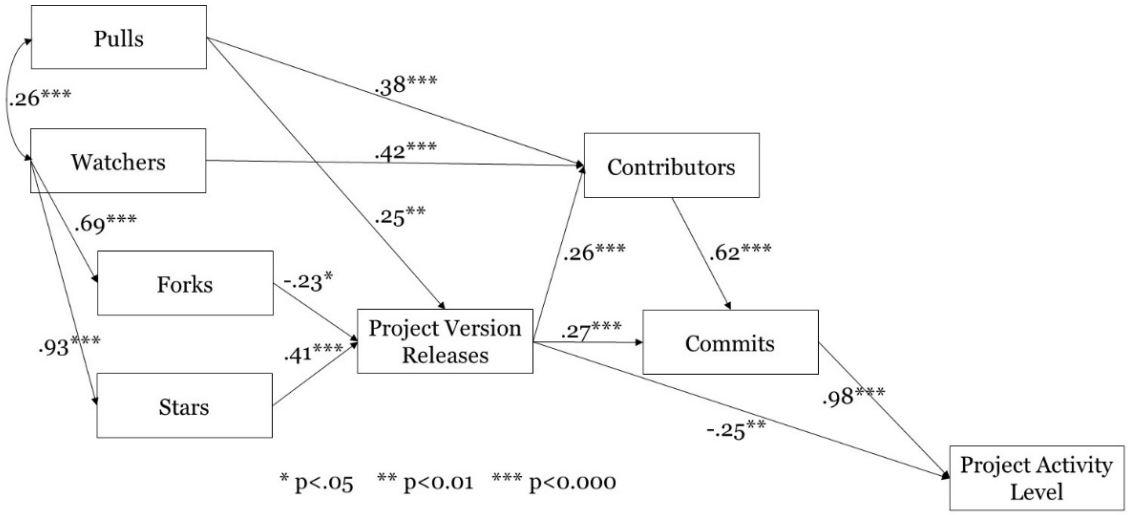

Figure 3: GitHub JavaScript project activity level structural path model.

The project activity level structural path model (Figure 3) shows excellent fit ($\chi 2/df = 20.01/15 = 1.33$; $p < 0.171$; TLI = 0.983; CFI = 0.991; GFI = 0.952; AGFI = 0.885) (Cunningham, 2008; Hair *et al.*, 2010). The GFI-AGFI difference being just above 0.06, and the RMSEA (0.059) being above 0.05, both indicate there remains some minor fit improvement – such as engaging a larger case study of top GitHub projects (Kline, 2015), or using an even tighter outlier (kutosis/mahalanobis-distance) removal consideration (Cunningham, 2008; Hair *et al.*, 2010).

Three outliers (cases 25, 31, 37) were removed - leaving a final data set of 97. Ideally the data set of projects should exceed 150-160 (Hair *et al.*, 2010; Muthén & Muthén, 2002). However structural models retain meaning within sample sizes of 100–150 case range (Anderson & Gerbing, 1988; Ding, Velicer & Harlow, 1995; Tabachnick & Fidell, 2012; Tinsley & Tinsley, 1987), and with 10 times as many cases as parameters, this data set of 97 remains acceptable (Kline, 2015). The final data set contains no notable mahalanobis outlier distance (SD) gaps through to the mean. Also, all construct skews are acceptable (< 3), but with vast differences in project scope and scale, multivariate kurtosis is large (113) but acceptable (Hair *et al.*, 2010). Thus the model is suitable for structural path analysis.

Watchers are highly correlated with the stars-provided to the project, and strongly correlated with the degree of forking. The level of forking exerts negative influences towards project version releases as these draw potential project contributors into non-core development tasks. The number of project version releases exerts cyclical peak-and-trough influences that negatively directly impact the overall project activity, but positively impact into commits and contributors. Commits and project level activity move in-line with each other, but they are different constructs. Contributors (via commits) exert an intermediate effect on project level activity.

DISCUSSION

Figure 3's project activity level structural path model is summarized in Table 2's standardized total effects. Table 2 shows the key effects in generating Project Activity Levels. The major contributors are the pull requests, the watchers, the number-of-releases, the total different contributors, and the commits. The commits lodged directly mirror the Project Activity Levels. Thus, five levers can be used by the project creators and their core team leaders when seeking to speed their project's software development processes. The number of stars-provided to the project make a lesser contribution.

The forks actually work against the project's progress by generating very minor negative total effects into the project's activity level. They sometimes dilute the focus of the project's software development strategies. Here, a fork may generate new ideas, create a new project, and then draw some original project developers off into this new software development direction, thus retarding the original project's activity level.

Table 2: Standardized Total Effects for current top (97/100) GitHub JavaScript Projects.

	Pulls	Watchers	Forks	Stars (provided)	Releases	Contributors	Commits
Forks	0	0.69	0	0	0	0	0
Stars (provided)	0	0.93	0	0	0	0	0
Releases	0.25	0.22	-0.23	0.41	0	0	0
Contributors	0.45	0.48	-0.06	0.11	0.26	0	0
Commits	0.34	0.35	-0.10	0.18	0.43	0.62	0
Activity	0.22	0.22	-0.08	0.14	0.35	0.35	0.98

Figure 3 shows collaborative responder-group efforts do drive GitHub JavaScript project activities as a multi-pronged approach (RQ1). Research question RQ2 is supported as major (critical) version releases do positively affect project activity levels. Multiple intermittent and minor version releases exert less GitHub JavaScript project activity levels because they often involve slight improvements, and only require minimal activity level contributions. The remaining research RQ3 question is supported when a code commit's documentation is lodged. Here, more commits also brings more changes to documentation, and as a GitHub JavaScript project's activity level rises, additional documentation emerges as a continual project requirement.

IMPLICATIONS

Theoretical Implications

This GitHub study follows responder behavioral patterns, in particular Information Integration Theory, and the Theory of Social Translucence. This framework allows behavioral activities to be gauged collectively and measured against each project's overall activity level. This allows a new way to compare projects and to understand projects once the masking features such as: size, programming-language, degree-of-complexity and longevity are removed.

Extensions to this study can map each project responder's/collaborator's identity, contributions, and ongoing activities through to GitHub repository followers, watchers and stars-provided into their social interaction domains including Facebook, websites, Twitter, and Wikis (Aggarwal, Hindle & Stroulia, 2014). Here, interpretations of value by understanding social network site consumer engagements (Hamilton & Tee, 2013) can be incorporated to extend the behavioral understanding of GitHub's social and external responders.

Practical Implications

The activity level of JavaScript project responders is measured using repository-collated measures. These behavioral measures first include pull-requests and project watchers which results in subsequent commit changes, or watchers changing roles and then generating new pull-requests against the project's repository. Pull-requests impact on project contributions and on project version releases, and with commits positively influence on project activity levels. Commit changes are generally clarified

through comments with linkages into project documentation. If accepted, these commits may also appear as ongoing documentation updates. Table 3 shows five constructs can be leveraged to jointly (or individually) deliver, faster project software-development processes.

A lead focus for the repository creator and the core team of collaborators is to generate additional commits. Here, commits can be encouraged by cross-promotional strategies including: (1) encouraging pull-requesters to respond and to generate multiple commits, (2) promoting the starring of the ongoing value of the project's development on Facebook, Twitter, and web media, and also converting social media watchers into pull requesters, and (3) engaging developer forums, Wikis, conferences and across other social connectivity avenues directly targeted towards encouraging more pull requests and follow-up commits.

Social media sites can also add transaction-related project information via inclusions of community 'fan-pages.' Fan-pages help to build stronger communities, provided they show usefulness, economic value, and are suitably branded. Here promotions and/or other consumer benefits can be incentivized (Hamilton & Tee, 2013). In addition, to further highlight and draw developer traffic, fan-pages news can be linked to HackerNews and GitHub Explore (Borges, Hora & Valente, 2016). Ultimately the key internal approach is to generate very-rapidly reviewing and incorporating decisions across all commits.

A second behavioral approach is to recognize committers by crediting their contributions against their personal email. This is achievable by recognizing, ranking, and promoting each contribution as enhancing: performance and/or quality and/or service and/or economic value and/or emotional perception (Hamilton, John R & Tee, 2015). These value recognition triggers are rewards to the respondent committer, and they likely positively affect the committer's satisfaction and ongoing loyalty (Hamilton, John R. & Tee, 2015). This recognition approach behaviorally encourages the committer to pursue further opportunities of benefit to a GitHub project. It also enhances their personal profile, and it promotes more project activity.
This study considers a collation of contribution constructs that have sometimes been used as popularity measures as exhibiting behaviors harmonious with the project. It shows all of these contribute directly or indirectly as build components of the project's activity level.

This study recognizes that increasing contributors is a complex task. For example, the project's activity level is cyclic, peaking around each version release (Borges, Hora & Valente, 2016). More major and critical project releases drive activity levels. These release-date developer traffic hypes can be enhanced with boosts via continual social media project-related achievement postings (Cassidy & Hamilton, 2016) - provided quality public communications are delivered.

FUTURE RESEARCH

Measurement Aspects

To further validate the project activity level GitHub JavaScript structural path model two additional studies are suggested (1) random sampling across the full suite of JavaScript projects, and (2) re-testing against each key GitHub programming language.

The refinement of the pull request counts is another measurement consideration. Pull-requests occur because of internal commits for review as well as and via forked versions of a repository. Some fork-pull-requests loop back into the originating repository. Hence, it may be useful to categorise pull-requests, and also to consider longitudinally if fork-pulls do actually occur later during project development. This research is underway.

There remains a need to create and deploy APIs that monitor project activity levels over time. This can expose where open source software development offers maximum improvement for the GitHub project under consideration.
Commits offer detailed content analysis that can be mined to elucidate where, and how, the documentation of substantive top performing programming language projects can be improved. The relevance of which commit(s) provide most benefit is another area waiting development.

Theoretical Aspects

GitHub studies can be theory-based, and/or behaviorally-based, and/or translucently-based, and/or values-based. They can also be linked via social networks and web media through into other consumer marketing and retailing approaches - typically focusing on consumer motivation, consumption and gratification aspects (Hamilton, John R & Tee, 2015).

Managerial Aspects

The project activity level model is applicable for GitHub JavaScript project creators. It can be astutely managed to generate high project level activities. It can be interpreted through Table 2's total effects and Figure 3's path strengths towards better targeting, and harnessing of a project's reach, and engagement, across relevant software development communities.

Learning how to extract pertinent information from responder review comments is often useful to a repository originator seeking to improve ongoing project deliverables. Approaches to understanding big data vary, but Bello-Orgaz *et al.* (2016) describe big data social capture approaches are of use when considering GitHub's watchers.

Projects can be more closely managed by developing text capture routines to extract responder key words from GitHub

documentation. For example value(s)-related words epitomising behaviors can include motivation (intentions to act) towards engaging/actioning, consumptive actions being undertaken, and gratification reflections of actions delivered. This data can then be real-time analysed, thus keeping GitHub repository originators behaviorally attuned to individuals and to their core collaborators.

CONCLUSION

The current top 100 GitHub JavaScript projects are assessed across various pathways of project contribution. Data captured from these popular projects differs in: format, project-size, development-cycle-stage, change-frequency, change-degree, forks, watchers, and contributor-skills, and has remained difficult to interpret. Three theories (Information Integration, Planned Behavior, and Social Translucence) frame this study's comparative project activity level structural path model approach.

A JavaScript project's activity level can be enhance with increased responder-group collaborative efforts, with more frequent major project version releases, and with greater numbers of commit project additions and/or deletions. The generation of additional forks delivers a minor net negative impact on project activity levels. Hence, repository originators and their core team of collaborators should ensure forking contributions remain under close monitoring and assessment.

REFERENCES

[1] Aggarwal, K., Hindle, A. & Stroulia, E. (2014). Co-evolution of project documentation and popularity within Github. In *Proceedings of the 11th Working Conference on Mining Software Repositories - MSR 2014* (pp. 360-363).

[2] Ajzen, I. (1991). The theory of planned behavior. *Organizational Behavior and Human Decision Processes*, *50*(2), 179-211.

[3] Anderson, J.C. & Gerbing, D.W. (1988). Structural equation modeling in practice: A review and recommended two-step approach. *Psychological Bulletin*, *103*(3), 411-423.

[4] Bello-Orgaz, G., Jung, J.J. & Camacho, D. (2016). Social big data: Recent achievements and new challenges. *Information Fusion*, *28*, 45-59.

[5] Berry, R. (2015). What are the most popular GitHub repositories of all time?. *Quora*. Retrieved from https://www.quora.com (3 September 2017).

[6] Bissyande, T.F., Thung, F., Lo, D., Jiang, L. & Reveillere, L. (2013). Popularity, interoperability, and impact of programming languages in 100,000 open source projects. In *Proceedings of the International Computer Software and Applications Conference* (pp. 303-312).

[7] Borges, H., Hora, A. & Valente, M.T. (2016). Understanding the factors that impact the popularity of GitHub repositories. In *Proceedings of the 2016 IEEE International Conference on Software Maintenance and Evolution - ICSME 2016* (pp. 334-344).

[8] Capra, E., Francalanci, C., Merlo, F. & Rossi-Lamastra, C. (2011). Firms' involvement in Open Source projects: A trade-off between software structural quality and popularity. *Journal of Systems and Software*, *84*(1), 144-161.

[9] Cassidy, L. J., & Hamilton, J. (2016). Website benchmarking: an abridged WAM study. *Benchmarking: An International Journal*, *23*(7), 2061-2079.

[10] Cosentino, V., Izquierdo, J.L.C. & Cabot, J. (2017). A systematic mapping study of software development with GitHub, *IEEE Access*, *5*, 7173-7192.

[11] Cunningham, E. (2008). *A Practical Guide to Structure Equation Modeling Using AMOS*. Melbourne: Streams Statsline,.

[12] Dabbish, L., Stuart, C., Tsay, J. & Herbsleb, J. (2012). Social coding in GitHub: transparency and collaboration in an open software repository. In *Proceedings of the ACM 2012 Conference on Computer Supported Cooperative Work* (pp. 1277-1286).

[13] Ding, L., Velicer, W.F. & Harlow, L.L. (1995). Effects of estimation methods, number of indicators per factor, and improper solutions on structural equation modeling fit indices. *Structural Equation Modeling: A Multidisciplinary Journal*, *2*(2), 119-143.

[14] Hair, J.F., Anderson, R.E., Tatham, R.L. & Black, W.C. (2010), *Multivariate Data Analysis* (7th ed.). Uppersaddle River, New Jersey: Pearson Education International.

[15] Hamilton, J.R. & Tee, S. (2013). Understanding social network site consumer engagements. In *Proceedings of the 24th Australasian Conference on Information Systems*. Melbourne, VIC, Australia, 4-6 December.

[16] Hamilton, J.R. & Tee, S. (2015). Engaging technologies-savvy consumers with the internet of things. In *Proceedings of the 15th International Conference on Electronic Business - ICEB* 2015 (pp. 242-246).

[17] Hamilton, J.R. & Tee, S. (2015). Expectations-to-value: Connecting customers with business offerings. *International Journal of Internet Marketing and Advertising*, 9(2), 121-140.

[18] Hata, H., Todo, T., Onoue, S. & Matsumoto, K. (2015). Characteristics of sustainable OSS projects: A theoretical and empirical study. In *Proceedings of the 8th International Workshop on Cooperative and Human Aspects of Software*

Engineering - CHASE 2015 (pp. 15-21).

[19] Henderson, S. (2009). How do people manage their documents?: An empirical investigation into personal document management practices among knowledge workers. (Doctoral dissertation, The University of Auckland, NZ)..

[20] Jiang, J., Lo, D., He, J., Xia, X., Kochhar, P.S. & Zhang, L. (2017). Why and how developers fork what from whom in GitHub. *Empirical Software Engineering*, *22*(1), 547-578.

[21] Kalliamvakou, E., Gousios, G., Blincoe, K., Singer, L., German, D.M. & Damian, D. (2016). An in-depth study of the promises and perils of mining GitHub. *Empirical Software Engineering*, *21*(5), 2035-2071.

[22] Kline, R.B. (2015). *Principles and Practice of Structural Equation Modeling*, New York: The Guilford Press.

[23] Lee, M. J., Ferwerda, B., Choi, J., Hahn, J., Moon, J. Y., & Kim, J. (2013, April). GitHub developers use rockstars to overcome overflow of news. In *CHI'13 Extended Abstracts on Human Factors in Computing Systems* (pp. 133-138). ACM.

[24] Luo, Z., Mao, X. & Li, A. (2015). An exploratory research of GitHub based on graph model. In *Ninth International Conference on Frontier of Computer Science and Technology - FCST* (pp. 96-103). IEEE.

[25] Ma, W., Chen, L., Zhou, Y. & Xu, B. (2016). What are the dominant projects in the GitHub python ecosystem? In *Proceedings of 3rd International Conference on Trustworthy Systems and Their Applications - TSA 2016* (pp. 87-95).

[26] Marlow, J., Dabbish, L. & Herbsleb, J. (2013). Impression formation in online peer production : Activity traces and personal profiles in GitHub. In *Proceedings of the 16th ACM Conference on Computer Supported Cooperative Work* (pp. 117-128).

[27] Mcdonald, N., Blincoe, K., Petakovic, E. & Goggins, S. (2014). Modeling distributed collaboration on Github. *Advances in Complex Systems*, *17*(7/8), Paper 1450024.

[28] Mileva, Y.M. (2012). Mining the evolution of software component usage. (Doctoral dissertation, Saarland University, Germany). Retrieved from https://publikationen.sulb.uni-saarland.de/bitstream/20.500.11880/26438/1/thesis.pdf (6 July 2017).

[29] Muthén, L.K. & Muthén, B.O. (2002). How to use a Monte Carlo study to decide on sample size and determine power. *Structural Equation Modeling: A Multidisciplinary Journal*, *9*(4), 599-620.

[30] Sajnani, H., Saini, V., Ossher, J. & Lopes, C. V. (2014). Is popularity a measure of quality? An analysis of maven components. In *Proceedings of the 30th International Conference on Software Maintenance and Evolution - ICSME 2014* (pp. 231-240).

[31] Sheoran, J., Blincoe, K., Kalliamvakou, E., Damian, D. & Ell, J. (2014). Understanding “watchers” on GitHub. In *Proceedings of the 11th Working Conference on Mining Software Repositories - MSR 2014* (pp. 336-339).

[32] Singer, L., Figueira Filho, F. & Storey, M.-A. (2014). Software engineering at the speed of light: how developers stay current using twitter. In *Proceedings of the 36th International Conference on Software Engineering - ICSE 2014* (pp. 211-221).

[33] Squire, M. (2014). Forge++: The changing landscape of FLOSS development. In *Proceedings of the Annual Hawaii International Conference on System Sciences* (pp. 3266-3275).

[34] Tabachnick, B.G. & Fidell, L.S. (2012), *Using Multivariate Statistics* (6th ed.). New York: Harper and Row.

[35] Tinsley, H.E. & Tinsley, D.J. (1987). Uses of factor analysis in counseling psychology research.. *Journal of Counseling Psychology*, *34*(4), 414-424.

[36] Tsay, J., Dabbish, L. & Herbsleb, J. (2014). Let’s talk about it: evaluating contributions through discussion in GitHub. In *Proceedings of the 22nd ACM SIGSOFT International Symposium on Foundations of Software Engineering - FSE 2014* (pp. 144-154).

[37] Williams, A. (2012), *GitHub pours energies into enterprise - Raises $100 million from power VC Andreessen Horowitz*, *Tech Crunch*. Retrieved from https://techcrunch.com (6 July 2017).

[38] Wu, Y., Kropczynski, J., Shih, P.C. & Carroll, J.M. (2014). Exploring the ecosystem of software developers on GitHub and other platforms. In *Proceedings of The Companion Publication of the 17th ACM Conference on Computer Supported Cooperative Work & Social Computing* (pp. 265-268). ACM.

[39] Xavier, J. & Macedo, A. (2014). Understanding the popularity of reporters and assignees in the Github. *Proceedings of the 26th International Conference on Software Engineering and Knowledge Engineering* (pp. 484-489). SEKE, Vancouver, Canada from July 1- 3.

[40] Yu, Y., Wang, H., Yin, G. & Ling, C.X. (2014). Reviewer recommender of pull-requests in GitHub. In *Proceedings of the 30th International Conference on Software Maintenance and Evolution* (pp. 609-612). ICSME.

[41] Zhu, J., Zhou, M. & Mockus, A. (2014). The relationship between folder use and the number of forks : A case study on github repositories. In *ESEM ’14*. Torino, Italy. Retrieved from http://mockiene.com/papers/folder-short.pdf (6 July 2017).