Hamilton, J.R., Tee, S.W., Holdsworth, J. & Alshomali, M.A. (2017). Analysing big data projects using Github and JavaScript repositories. In *Proceedings of The 17th International Conference on Electronic Business* (pp. 47-52). ICEB, Dubai, UAE, December 4-8.

Analysing Big Data Projects Using Github and JavaScript Repositories

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

ABSTRACT

GitHub open source software developers remain in short supply. Successful GitHub projects offer multiple pathways for developers to contribute into their repositories. This study's GitHub JavaScript big data is path modelled to provide understanding of the different significant developer contribution pathways towards raising the project's activity level. Its significant pathways offer the project's creator benchmark decision making capabilities that can be used to trigger faster project software development through to its next completion point. This approach has behavioural consumptive value connotations that may provide a future pathway towards tapping big data sources and to also delivering real business values.

Keywords: Open source software (OSS), GitHub software development, popularity, activity level, value, consumption.

*Corresponding author

INTRODUCTION

The online open source software version control system GitHub (Firestine, 2017) is large. It currently enlists around twenty million developers who have delivered over one hundred million pull requests (Firestine, 2017). Github's repositories (around fifty seven million (Firestine, 2017) outnumber the available open source software developers by nearly three-to-one. Hence open source developers remain in short supply. Also new software projects often need skilled external open source software developer support to quickly advance (and complete) their coding requirements (GitHub, 2017b). Thus, this study considers first how developers contribute towards a project, and second what may trigger the speeding of project's software development through to its next completion point.

GITHUB

GitHub is the faster growing online software version control system (Yu et al., 2014). GitHub allows the professional developers of companies like Facebook and Google to enlist global help, and to rapidly solve complex software coding. Such substantive GitHub projects house a conglomeration software development components each held within their specific project's big data repository. GitHub Repositories extend beyond such major project developments. They also capture projects involving independent developers, or teams of developers seeking specific open source assistance to solve specific software project tasks. GitHub repositories also encompass individual repositories where specific student projects seek individual open source assistance.

A GitHub project creator forms a repository, This GitHub repository houses the source code, and a master (production development) branch. The project creator and a core development team use GitHub, social media, other web/mobile sites, and personal connections to promote their project. They encourage open source developers to fork the project (copy and adapt the master branch) and to work independently. Theses open source software developers then make direct changes to the repository content and feedback (pull request) changes that can advance the master branch code and/or contribute in some indirect supportive way into this project's repository.

Over time each repository modifies as: developer contributions add and/or delete content, explanations are added, code is tested by others, and the original creator and core team accept pull-requests as commits (content changes) into the master branch. Thus the repository grows as a collection of: source code, a master (production development) branch, other branches (including experimental, developmental, internal clones, and test site components), support materials (including text description, audio, video and other multi-dimensional clarifications), ad hoc contributions and information advice updates.

From time-to-time GitHub's repository ecosystem may suffer developer and knowledge losses - which can retard the project's software development. For example, a specific developer may choose to externally clone the repository's master branch and create a unique (and/or alternate) software development pathway outside the original repository's ecosystem. This loss of developer capabilities likely negatively impacts the project's development, and may move other developers to follow this unique

alternative development pathway.

GITHUB AND RESEARCH

The GitHub REST API can be used by researchers to slowly capture select comparison data from several GitHub repositories (GitHub, 2017d; Gitstar., 2017; Octoverse., 2017). The REST API generates only specific GitHub ecosystem data concerning each repository (Borges, Hora & Valente, 2017; Cosentino, Luis & Cabot, 2016; Onoue, Hata & Matsumoto, 2013). Such GitHub ecosystem data can then be collated, statistically modelled, and assessed to expose each open source developer groups' ongoing contributions towards the repository's popularity (Aggarwal, Hindle & Stroulia, 2014; Bissyande et al., 2013; Blincoe et al., 2016; Borges, Hora & Valente, 2017; Cosentino, Izquierdo & Cabot, 2017).

Researchers measure popularity differently. Hence popularity remains an inconsistent GitHub measure with different studies including differing: convenience measures, different models, and/or different regression equations (Borges et al., 2015; Sajnani et al., 2014; Xavier & Macedo, 2014). Further, popularity as the overall suite of contribution into a GitHub ecosystem is really a proxy for the project's activity level (Dabbish et al., 2012).

The project's activity level is a repository performance benchmark. It can then be used by the GitHub project creator as a decision making tool, and can also be used to benchmark against competition. The project creator (and the core team of supporting developers) can then focus their benchmark decision making onto further ways of drawing new potential GitHub contributors (such additional developers, advisors, testers, and watchers) into their repository's ecosystem, and into convincing them to provide enhancing ecosystem contributions. This approach is also behavioral, and where the developer ecosystem's responses lift the project's consumption, its values deliverance processes are also enhanced (Bradlow et al., 2017). Thus a consumptive-values approach may provide a future pathway towards tapping big data sources and tying these into pathways for delivering business values.

GITHUB CONTRIBUTOR GROUPS

A GitHub repository had many contributor groups. These can be loosely classified into three developer consideration sections. First are those interested in the GitHub project including: watchers (receive notifications of content changes), stars (indicate a liking for the project), and contributors (request to contribute into the project). Second are those who actively participate to the project including: commits (incremental changes provided as pull requests/documentation), forks (independent development solution approaches), and version releases (meeting project milestones). Lastly are those executing changes including: open issues (identified problems in the code), closed issues (problems fixed in the code), open pulls (problems still not fixed from contributing developers or fork commit problems still under development), and closed pulls (pull requests adopted into the master branch) [20]. We model these measurable project activity level contributors as Figure 1.

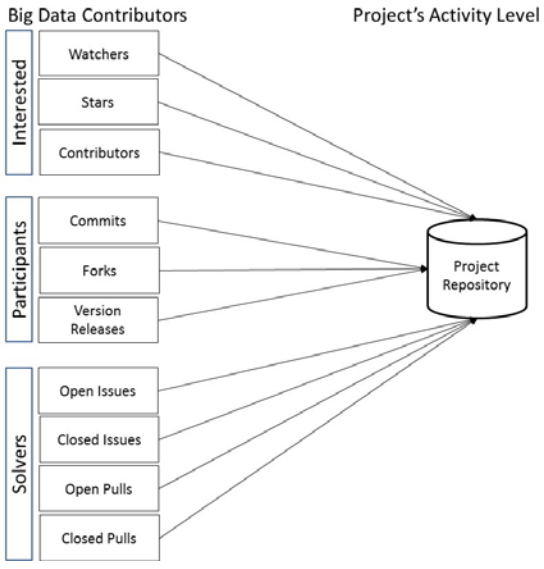

Figure 1: GitHub big data contributors to project's activity level into the repository.

Hence the purpose of this paper is to present a path model approach capable of investigating (and interpreting) drivers of a GitHub repository's project activity level. We suggest our path model approach also offers direction for future work using more sophisticated statistical techniques. For example the text mining of GitHub's Figure 1 construct blocks for values related measures may indicate measurement pathways between GitHub and a business' consumptive-values constructs. This may initiate improved repository search and capture algorithms.

METHODOLOGY

Past studies have seldom extracted and compared big data from GitHub to study Java, JavaScript and Python project activity levels. This study investigates the most common programming language in GitHub – namely JavaScript, as a preliminary study before repeating the approach using Java and Python, Extraction utilizes the REST API provided by GitHub (GitHub, 2017c). This process is slow and the approach used captures the top 160 GitHub projects (as defined by number of forks) for each language, with each big data 24/7 download requiring several days. Data captured is then collated and checked to remove anomalies. Anomolies must be individually checked. For example one of the top projects in Python is still listed as shadowsocks - an internet firewall repository. But this is now inactive and it is blocked by the Chinese government (Shadowsocks, 2017). There are anomalies in JavaScript projects too, but they are not quite as dramatic as this Python example.

By regressing each contributing construct in Figure 1 their relative contributions towards activity level may be gauged. A better more complex approach is assessing each contributing construct within a full path model (Figures2 and 3), and then investigating both their relative path strengths of significance, and their total effects onto a project's activity level. Again outlier removal using AMOS24.0 is engaged to remove non-conforming project repositories (like Python's shadowsocks).

GITHUB EXPERIMENTAL DESIGN

Figure 1 has interested observers - that discover the project, and through their actions (watching, staring or contributing) can potentially be drawn to add into the project as participants and solvers - who add/remove/copy/reflect-on the project's content through its repository contributions and changes. Hence, a regressive path model may follow a logical progression as shown in Figure 2.

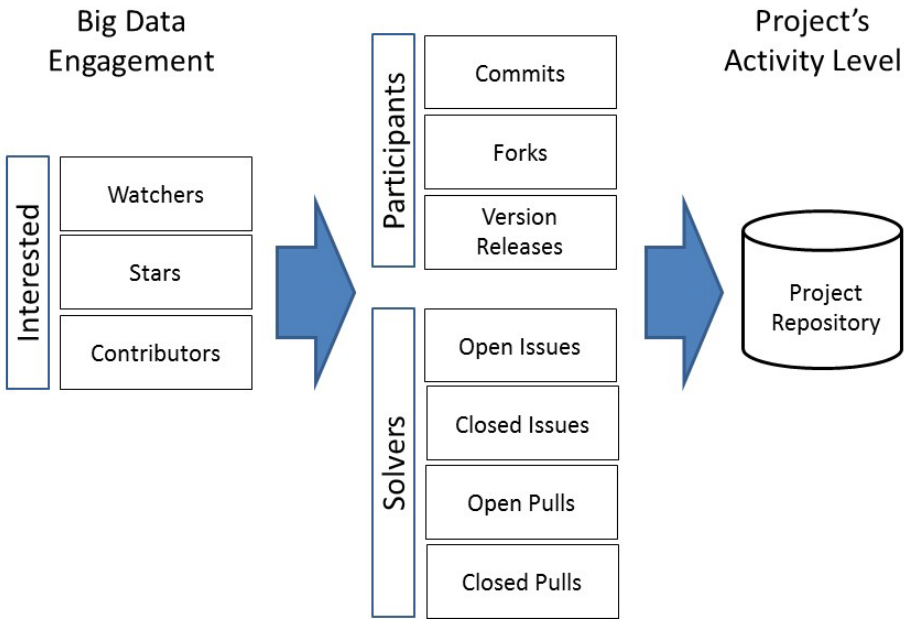

Figure 2: Proposed GitHub path model structure

RESULTS

In this exploratory study we include pulls as a combined pulls-open and pulls-closed value, and hence we do not include the Figure 2 issues-open and the issues-closed constructs. Our analysis of the top 160 JavaScript projects eliminates six outliers and leaves the Figure 3 path model with all paths significant at $p < 0.05$.

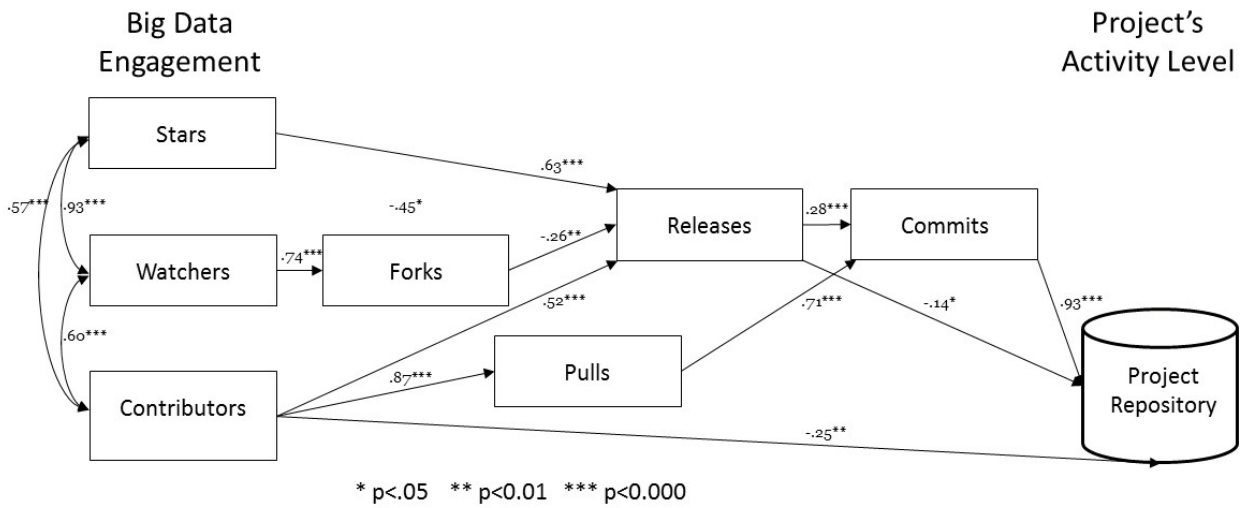

Figure 3: GitHub path model (JavaScript Top 160 projects)

This exploratory path model study delivers excellent fit. Its Figure 3 exploratory fit indices are as follows: $\chi 2/df = 1.68$, $p = 0.052$, CFI = 0.99, TLI = 0.98, GFI = 0.96, AGFI = 0.90, RMSEA = 0.066 (Hair et al., 2010). Hence, the full study comparing 480 of the top GitHub projects - deploying Java, JavaScript and Python projects, is now being conducted by the authors.

DISCUSSION

The path model approach is regression based. It is applicable where models (such as Figure 3) are not too complex (Grapentine, 2000). It identifies the most important and least important constructs. However, it assumes data collection measures are made without random measurement error. This feature can disguise multicollinearity effects (Grapentine, 2000; Hair et al., 2010). In this study, we control for these multicollinearity effects by our research design by only engaging one software language at a time, by only engaging the top GitHub specific projects, by only using the projects that are current, and by only using projects with a continual repository longevity exceeding 3 years. The Figure 3 path model approach supports the theoretical flow of GitHub contributions.

Table 1: Standardized Total Effects for GitHub path model (JavaScript Top 160 projects)

	Contributors	Watchers	Stars	Forks	Pulls	Releases	Commits
Forks		0.74					
Pulls	0.87						
Releases	0.52	-0.64	0.63	-0.26			
Commits	0.76	-0.18	0.17	-0.07	0.71	0.28	
Project Activity	0.39	-0.08	0.07	-0.03	0.66	0.12	0.93

Table 1 provides Figure 3's standardized total effects. This table shows commits are key direct drivers of the project's activity level, other contributors are indirect drivers of the project's activity level. Pulls and commits are the strongest drivers of the project's activity level. Contributors also generate positive project activity level additions or deletions. This suggests creating high levels of pull requests should be a prime target consideration for project creator's core team of developers.

Stars, watchers and forks provide little effect on the project's activity level. Some forks divert some developer activity into other directions that occasionally do not feed back into the project, and watchers, outside contributing an effect onto forks, provide a minor negative towards the project's activity level.

This study offers a big data direction for future work. It allows for the deployment of more sophisticated statistical comparison techniques. It is logically developed. It offers further indications around the internal and broad relationships that likely exist between GitHub's big data and models linking through to business/consumer consumption, and how these may be connected using improved repository search algorithms to release business value.

CONCLUSION

With around twenty six million developers, and around fifty seven million projects, a GitHub project's competition to win open source developers remains highly competitive. There are multiple roles through which a GitHub developer can participate in a project and contribute to its repository (Figure 1).

This GitHub study engages a staged behavioral approach. The sequencing of three developer consideration sections is considered in line with Hamilton and Tee (2013) three stage behavioral approach using the project's overall activity level as the outcomes driver. This delivers a new approach to investigate project developments, and to understand projects against varying sizes, software languages, degrees-of-complexity and varying longevities.

The project's activity level of JavaScript created developer and other contributions is successfully measured through repository-collated addition and deletion measures. Pulls, commits and contributors are the strongest OSS drivers of the project's activity level – suggesting the pursuit of OSSDs in an ongoing requirement in the rapid progression of any OSS project's completion, stage completion or release.

GitHub's JavaScript big data is path modelled to provide understanding of the different significant developer contribution pathways towards raising the project's activity level (Figures 2 and 3). The significant pathways offer the project's creator decision making capabilities that can be used to trigger faster project software development through to its next completion point.

The project's activity level is a performance benchmark that is now available to the GitHub project creator. This approach can benchmark against competition. This approach is also behavioral and where the developer ecosystem's responses lift the project's consumption, its values deliverance processes are also enhanced. Thus a consumptive-values approach may provide a future pathway towards tapping big data sources and to also delivering real business values.

Although a path model approach is applied, and not structural equation modeling, all factors affecting multicollinearity effects are not captured in this research design. Hence this study holds some weaknesses. It remains impossible to get a perfect snapshot of real-time data as it takes many days to access and combine the GitHub data using the REST API. The selection of the top 160 JavaScript repositories (and sample size) is guided first by each repository being operable for over three years, and second by the top 160 projects being selected based on their level of forking - a common, and widely-accepted approach (an alternative is the project selection based on watchers).

The GitHub project's activity level deployed herein can be further investigated by (1) randomly sampling all GitHub JavaScript projects, by (2) monitoring and comparing the differences in releases, pulls, forks and commits, and by (3) using API's to monitor and show where OSSD can offer further improvement opportunities.

REFERENCES

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

[2] Bissyandé, T. F., Thung, F., Lo, D., Jiang, L., & Réveillere, L. (2013, July). Popularity, interoperability, and impact of programming languages in 100,000 open source projects. In *IEEE 37th Annual Computer Software and Applications Conference (COMPSAC)* (pp. 303-312). IEEE.

[3] Blincoe, K., Sheoran, J., Goggins, S., Petakovic, E. & Damian, D. (2016). Understanding the popular users: Following, affiliation influence and leadership on GitHub. *Information and Software Technology*, *70*, 30-39.

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

[5] Borges, H., Valente, M. T., Hora, A., & Coelho, J. (2015). On the popularity of GitHub applications: A preliminary note. arXiv preprint arXiv:1507.00604.

[6] Bradlow, E.T., Gangwar, M., Kopalle, P. & Voleti, S. (2017). The role of big data and predictive analytics in retailing. *Journal of Retailing*, *93*(1), 79-95.

[7] Cosentino, V., Izquierdo, J. L. C., & Cabot, J. (2017). A Systematic Mapping Study of Software Development with GitHub. *IEEE Access*, *5*, 7173-7192.

[8] Cosentino, V., Izquierdo, J. L. C., & Cabot, J. (2016, May). Findings from GitHub: methods, datasets and limitations. In *IEEE/ACM 13th Working Conference on Mining Software Repositories* (MSR) (pp. 137-141). IEEE.

[9] Dabbish, L., Stuart, C., Tsay, J., & Herbsleb, J. (2012, February). 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). ACM.

[10] Firestine, B. (2017). Celebrating nine years of GitHub with an anniversary sale. GitHub, Inc. Retrieved from https://github.com/blog/2345-celebrating-nine-years-of-github-with-an-anniversary-sale (1 October 2017).

[11] GitHub, I. (2017a). A small place to discover languages in GitHub. GitHub, Inc. http://githut.info/ (1 October 2017).

[12] GitHub, I. (2017b). Bring GitHub to work GitHub, Inc. Retrieved from https://github.com/business (1 October 2017).

[13] GitHub, I. (2017c). GitHub REST API v3. GitHub, Inc. Retrieved from https://developer.github.com/v3/ (1 October 2017).

[14] GitHub, I. (2017d). Trending in open source. GitHub, Inc. Retrieved from https://github.com/trending (1 October 2017).

[15] Gitstar. (2017). Gitstar ranking. Gitstar. Retrieved from https://gitstar-ranking.com/ (1 October 2017).

[16] Grapentine, T. (2000). Path analysis vs. structural equation modeling. *Marketing Research*, *12*(3), 13-29.

[17] 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.

[18] 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, December 4-6. Retrieved from https://researchonline.jcu.edu.au/31031/1/402.pdf (1 October 2017).

[19] Octoverse. (2017), *The state of the Octoverse 2016.* Octoverse. Retrieved from https://octoverse.github.com/ (1 October 2017).

[20] Onoue, S., Hata, H. & Matsumoto, K.I. (2013). A study of the characteristics of developers' activities in GitHub. *In Software Engineering Conference (APSEC)* (Vol. 2, pp. 7-12). IEEE, Bangkok, Thailand, December 2-5.

[21] Sajnani, H., Saini, V., Ossher, J. & Lopes, C. V. (2014). Is popularity a measure of quality? An analysis of maven components. In *IEEE International Conference on Software Maintenance and Evolution (ICSME)* (pp. 231-240). IEEE. Victoria, British Columbia, Canada, September 29 - October 3.

[22] Shadowsocks. (2017). Meet Shadowsocks, the underground tool that China's coders use to blast through the Great Firewall, Shadowsocks. Retrieved from https://qz.com/1072701/meet-shadowsocks-the-underground-tool-that-chinas-coders-use-to-blast-through%0A-the-great-firewall/%0A (19 September 2017).

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

[24] Yu, Y., Yin, G., Wang, H. & Wang, T. (2014). Exploring the Patterns of Social Behavior in GitHub. In *Proceedings of the 1st International Workshop on Crowd-based Software Development Methods and Technologies* (pp. 31-36). ACM: Hong Kong, China, November 16-22.