A Crowdsourcing Approach for Quality Enhancement of eLearning Systems

by

LALIT Mohan Mohan, Padmapriya Raman, Venkatesh Choppella, Y.Raghu Babu Reddy

in

Innovations in Software Engineering Conference, ISEC

Report No: IIIT/TR/2017/-1

Centre for Software Engineering Research Lab
International Institute of Information Technology
Hyderabad - 500 032, INDIA
February 2017

A Crowdsourcing Approach for Quality Enhancement of eLearning Systems

Lalit Mohan S[*]
IIIT Hyderabad, India
lalit.mohan@research.iiit.ac.in

Priya Raman[†]
Virtual Labs, Hyderabad, India
priya@vlabs.ac.in

Venkatesh Choppella[‡]
IIIT Hyderabad, India
venkatesh.choppella@iiit.ac.in

YR Reddy[§]
IIIT Hyderabad, India
raghu.reddy@iiit.ac.in

ABSTRACT

In India, a large number of engineering undergraduates are adopting eLearning as it provides access to best faculty and reduces concerns on inadequate physical infrastructure at colleges. Virtual Labs is a Government of India eLearning initiative containing simulation and remote triggered labs for engineering students. Virtual Labs developed over a period of 6 years is used by more than a million undergraduate students across nine engineering disciplines. The software used for developing these experiments requires substantial effort for maintenance due to deprecation, compatibility, etc. We propose a targeted crowdsourcing approach for maintenance of Virtual Labs with sustainable quality. The targeted crowdsourcing involves the large number of engineering students who are also the major stakeholders of these labs. Our quality enhancement using crowdsourcing approach was validated for 14 labs and would be extended to 191 labs based on encouraging results.

CCS Concepts

•**eLearning Systems** → Crowdsourcing; •**Crowdsourcing** → Software Development; **Quality;**

Keywords

eLearning Systems, Crowdsourcing, Software Development, Quality

[*]Mr.Lalit Mohan works for Virtual Labs at IIIT-Hyderabad, an organization funded by MHRD, Government of India.

[†]Ms. Priya Raman is Project Manager at Virtual Labs.

[‡]Prof. Venkatesh Choppella is responsible for Integration activities of entire Virtual Labs.

[§]Prof. Raghu Reddy is Integration Coordinator of Virtual Labs at IIIT Hyderabad.

ISEC '17, February 05-07, 2017, Jaipur, India

DOI: http://dx.doi.org/10.1145/3021460.3021483

1. INTRODUCTION

Improved penetration of internet[1] and also increased digital literacy is leading to better adoption of crowdsourcing. Crowdsourcing is the process of obtaining information or getting tasks completed by crowd on internet [2]. This provides an opportunity for extending workforce beyond four walls. Micro tasks such as image tagging, language translation, Q&A, etc. are some of the most common crowdsourcing tasks. Some of the effort and knowledge intensive tasks such as software development, content development for massive open online courses (MOOCs), logo design, etc. are also benefiting in terms of reduced cost and faster completion time with large number of contributors.

Most of the crowdsourced eLearning courses [13] including MOOCs are video recordings or textual content with manual moderation / quality check. However, there could be courses delivered over the web with content that requires simulation or interfacing with hardware devices remotely. Development of such eLearning courses involves the usage of software and related development tools. The source code for such courses requires regular maintenance owing to software wear and tear. The wear and tear can be due to deprecation, compatibility issues among other evolution related issues. Software maintenance of these eLearning courses could be done with dedicated teams or crowdsourced workers. However, if the effort in fixing issues is relatively less and the frequency of issue occurrence is intermittent, having a dedicated team will not be economical. An open source software development approach could also be adopted if the course software is open to public. However, software developed / managed in open source environment have common standards and practices such as coding standards, continuous integration, etc. which may not be a necessary case in crowdsourced software [8] [12]. Also, effort size in open source environment for work items is relatively larger (more than a person day) with community members involvement to triage enhancements and fixes. Open source software have different license models to protect the interest of community and other stakeholders but eLearning courses developed using software are a form of content and using open source software licensing models is not appropriate. Hence, crowdsourcing of eLearning courses software maintenance would be appropriate and can adopt best practices of opensource software development.

[1]http://www.internetlivestats.com/

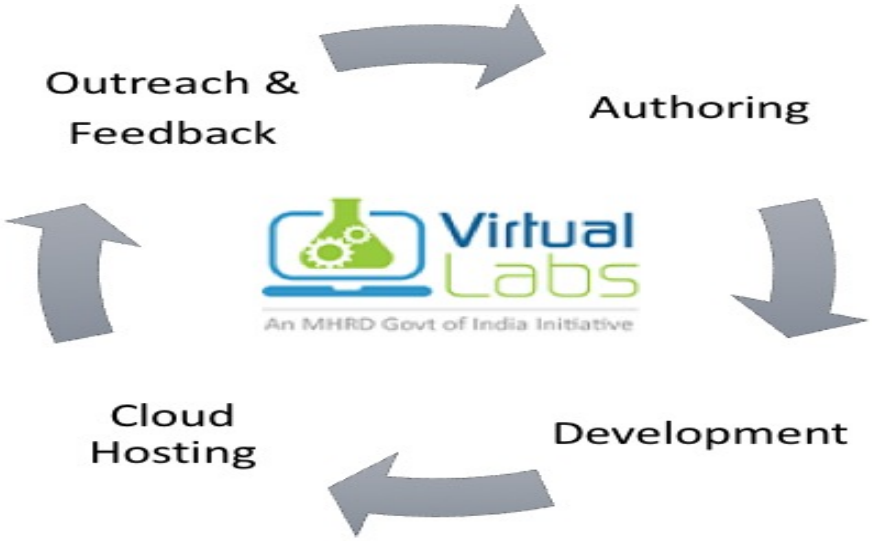

Figure 1: Life Cycle of a Lab

Motivation to participate in crowdsourcing could be intrinsic or extrinsic in nature [2]. Given that a lot of crowdsourced work is voluntary and not for monetary benefit, this can affect the quality of crowdsourced work. There could also be quality issues because of ignorance or lack of knowledge and thus quality check in crowdsourcing is critical. The current quality control processes in crowdsourcing use majority voting, peer-reviews, data mining, fault-tolerant sub-tasks, game theory and other hybrid modes [4]. In this paper, we try to address the following questions :

(a) Crowdsourcing of software development involves a task being assigned / identified by an owner and he / she evaluates the quality of accomplished task. If there are multiple stakeholders, can crowdsourcing of software development be possible with acceptable quality?

(b) What process should be adopted for ensuring quality of eLearning systems while using crowdsourcing for maintenance? This process includes identifying the task / work request type (issue, enhancement, etc) and the size (person days).

We demonstrate the quality enhancement of eLearning systems (Virtual Labs[2]) using crowdsourcing approach. The Virtual Labs is an eLearning initiative by Ministry of Human Resources Development (MHRD), Government of India. This initiative provides additional aides to undergraduate engineering course curriculum with lab components developed by India's top engineering institutes' faculty. As part of this initiative, 12 engineering institutes from India have developed 205 labs with 1,515 experiments for 9 engineering disciplines in Phase I that spanned from the year 2009-15. Each lab has an average of 8 experiments that are operate as a simulation or trigger a remote apparatus / instrumentation. The development process of these labs is shown in Figure 1. This process is followed for all labs that were built in Phase I stage of the project and also for any new labs that are being built in Phase II. However, most of the work in Phase II is maintenance of the labs and outreach program. Since 2015, the usage of lab experiments is greater than 5.8 million across locations.

- Authoring - The need of lab and its constituents (experiments) is envisioned by faculty (Lab Owner) of an engineering discipline. Faculty are expected to understand the curriculum that exists across colleges for the envisioned lab and prepare the storyboard for the lab and its experiments.
- Development - Faculty provides inputs to developers for developing simulation / remote triggered lab. Developers use available software libraries such as GWT[3], Javascript and other opensource software for building the lab. However, some of the developers used proprietary technologies such as Adobe Flash, Java3D, etc. and built 40+ labs in proprietary technologies. Most of the developers were students working for the respective faculty of the lab and they have transitioned out after graduation.
- Hosting - Release and Integration team of the engineering institutes are given the task of handling software compatibility and system configuration needs of the current version of the lab that were developed in Phase I. For any new Lab version to be hosted on the Virtual Labs servers, Lab owners and Integration coordinators are required to provide a sign-off. Currently, 90+ labs are hosted on public cloud (AWS) environment and the remaining labs are still hosted on college servers.
- Outreach and Feedback - Participating engineering institutes are given a yearly outreach target. As part of outreach program, workshops are conducted across India to explain the usefulness of Virtual Labs. The Feedback collected during these outreach programs had concerns of software issues such as compatibility, usability, availability, etc. To quickly assess the nature of concerns instead of going through details of a large

[2]http://vlabs.ac.in/

[3]http://www.gwtproject.org/

Figure 2: Word Cloud - Virtual Labs Feedback

number of users feedback, a word cloud as shown in Figure 2 was created from the feedback text after removal of stop words.

To address the identified concerns in outreach programs, we use a crowdsourced approach for quality enhancement of the labs. A pilot implementation of the proposed approach was done with crowd workers (students) from various engineering colleges in India for 14 Labs. Most of the engineering students are from lower ranked colleges and are not skillful in programming.

2. STATE OF THE ART

The section describes existing practices and the ongoing research for quality control in crowdsourcing. Alonso et al. [1] used crowd vote for evaluating the quality of responses. Mathew Lease [6] explained applicability of Machine Learning for automating quality control in crowdsourcing for more uncertain, diverse, specific, ongoing and rapid data. Afra et al. [11] used credibility based on past contributions and contributors mobility pattern for quality control. In a recent study, Bernstein et al. [16] discusses on reputation of crowd workers and importance of peer-reviews. Allahbakhsh et al. [**?**] reviewed existing quality control mechanisms and is concerned on the current hard wired mechanism and proposes customizable, multi-dimensional model, etc. as future directions of research for quality control in crowdsourcing. Some of the other related literature discusses usage of game theory such as multi-armed bandit, better task clarity, experience, etc. for evaluating quality of workers / tasks. However, Quality assessment for software development is more complex as it requires validating a working solution for given pre and post conditions.

Panos et al. [4] proposed an algorithmic approach and usage of gold data to identify the inherent quality of each worker. This research is applicable to software engineering as well, however, expecting crowd to fix 20-30 defects may not be always possible for building history on their performance. Jiangang et al. [18] suggested a ranking framework based on skill-matching, topic-based features to measure compatibility between tasks and developers, etc on TopCoder dataset. In our case, there is no skill database as crowd workers could be coming from engineering colleges with no Github understanding and also the problem solving skills required are different based on the request type. Zhenghui et al. [3] proposes a game theory model for competitive behaviour while developing software in crowdsourcing model. The approach would make sense if crowd workers are large in number, competitive and problems are algorithmic in nature. Wenjun et al. [17] uses min-max relationship from game theory to assess the software quality. Linus' Law [14] has a view that *'given enough eyeballs, all bugs are shallow'*, however, there is a cost associated in terms of money and effort vis-a-vis the required quality. As the saying goes *'You get what you pay for !!'*, Iren et al. [5] proposed a crowdsourcing cost model for efficiently utilizing quality assurance mechanisms. This model would be effective if there is direct remuneration to crowd. Tsai et al. [15] mentions four level roadmap for software crowdsourcing and also suggests tools that can be used for short and long term work. Tsai's four level approach is based on size of the task, available time for development, usage of existing toolset such as TopCoder, uTest, etc. for development process and number of developers involved in task completion.

Mao et al. [10] performed a detailed survey on the use of crowdsourcing in software engineering at various life cycle stages. The document also details some of the issues and open problems in task decomposition, planning and scheduling, motivation, remuneration and quality assurance. This survey paper also shares some of the existing quality evaluation methods such as rating / ranking, reporting spam, task pre-approval, skill filtering, etc. Leicht et al. [7] also discussed about challenges of task decomposition and organizational preparedness for software crowdsourcing in their literature survey.

Table 1: Issue Severity Categorization

Issue Type	Count
Severity 1 (S1)	294
Severity 2 (S2)	1443
Severity 3 (S3)	94

3. OUR APPROACH

Our approach is based on leveraging existing collaboration tools such as Github so that issue tracking, source code forking/branching, integration, deployment, etc. become integrated in software life cycle management. As shown in Figure 3, our approach includes identification of tasks (issues), using pull or push mechanism for assignment of issues and validation of completed issues before hosting / deployment.

Concerns expressed during outreach programs in the feedback forms are textual and are not specific to a specific experiment or a lab. A professional Quality Assurance (QA) team was deployed to obtain more specific information about concerns and the related issues were logged in Github [4] for each of the lab experiments alongwith steps for re-producing the issue. QA team identified 1831 issues with various severity levels for 37 labs as shown in Table 1. The purpose of severity categorization is to identify the serious of the concern and the related count. A timeline is also associated with severity for fixing the issues.

- Severity 1 (S1) : Indicates that the issue affects critical functionality or critical data. There is no workaround

[4]https://github.com/virtual-labs

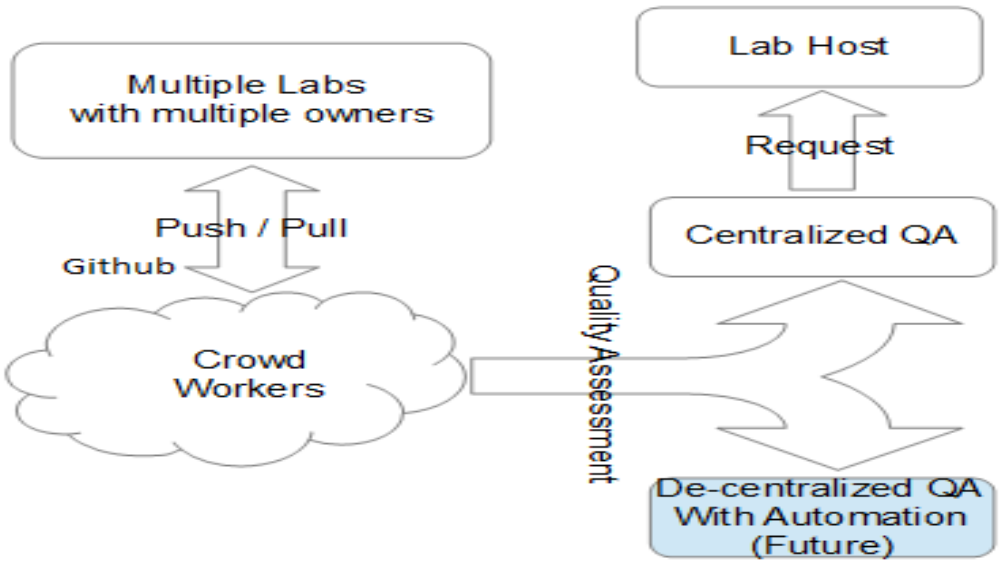

Figure 3: Crowdsourced Approach for Quality Enhancement of eLearning Systems

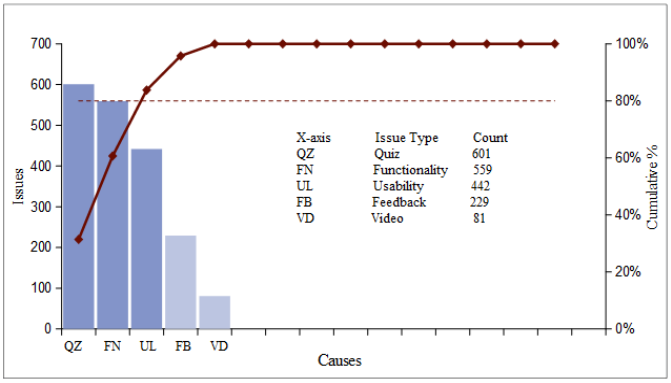

Figure 4: Virtual Lab Issues - Pareto Chart

to get to this functionality. Example: Prevention of user interaction, corruption of database, unfaithful to the semantics of interaction and redirection to the error page.

- Severity 2 (S2) : Indicates that the issue affects major functionality or major data. It could have a workaround but is not obvious and is difficult like broken links and a field view being inconsistent with its specifications. Example: In a form if there is a field which is editable but it is not allowing the user to edit it.
- Severity 3 (S3) : Indicates that the issue affects minor functionality or non-critical data. It could have an easy workaround. Example: Visual imperfections like spelling and grammar, alignment, inconsistent terminology, colour, shapes and fonts(css properties).

After analysis of issues as shown in Figure 4, following activities were performed to define the scope the work to be done by crowd workers. Issues related to lab availability were handled by deploying them in a public cloud environment. Issues related to plugins (Adobe and Java) are

Table 2: Labs Maturity level

Level 0	Unversioned
Level 1	Under version control
Level 2	Manually built on developer's machine
Level 3	Build process is automated
Level 4	Ready with deployment specification
Level 5	Life cycle management

handled by improving awareness in each of the lab sites by adding pre-requisites text. A parallel effort is currently in progress for removing the software dependencies by converting labs to Javascript from Adobe Flash via crowdsourcing and leveraging engineering institutes' expertise[5] [9]. From 1831 issues, 981 issues related to broken links, spelling mistakes, missing feature list, etc. are identified for resolution using the proposed crowdsourcing approach. To assess the readiness for crowdsourcing of issue fixes, a simple maturity model of labs was prepared to understand source code availability and readiness for deployment after fixes. The labs are categorized from Level 0 through Level 6 based on the availability of source code and build process for auto-deployment. The description of labs maturity level is given in Table 2.

The process for obtaining, fixing and deploying labs source code after fixes by crowd is depicted in Figure 5. As per the process,

1. Issues are logged by QA in Github are available for fixes by crowd workers. After issues are set to fixed status by crowd worker, a deployment request is initiated by the Lab Owner of the institute on the engineers-forum available on Virtual Labs Github repository. Lab Owner may club a group of issues for initiating the request. An alert in the form of an email notification is sent to the Release Engineer (RE).

[5]http://fossee.in/

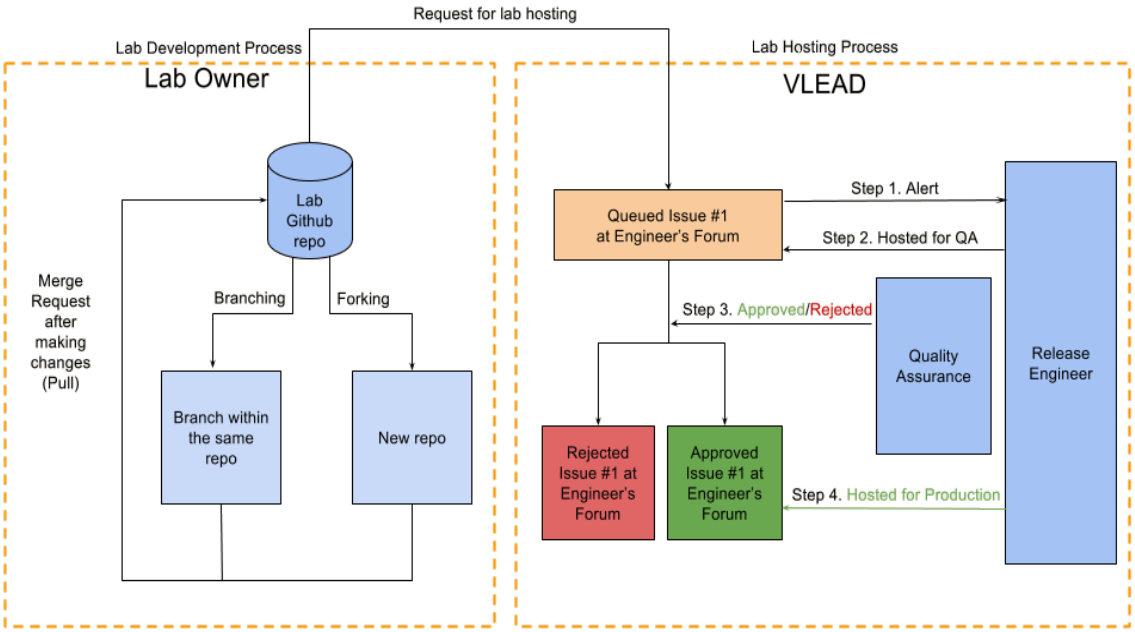

Figure 5: Process for Deployment after Fixing Issues

2. A deployment script prepared as part of lab maturity level is used by RE for deploying the lab in test environment. The test environment is similar to the production environment.

3. QA team for testing in test environment labels the fixed issues and the deploying request as 'Approved'. However, if a lab does not get the 'Approved' label, crowd worker gets inputs (on Github as issues) from the QA team regarding the reason for non-approval of the fix. The crowd workers can then go back and try to fix the issues. This process is repeated unless all the QA tests pass and the above mentioned 'Approved' condition is met based on the documented gating process as stated earlier.

4. After the QA team has finished testing and generated a test report, Integration coordinator and Lab Owner perform smoke testing to ensure identified issues are fixed and set the status as 'Pending for Host'.

5. RE executes the production deployment process for all 'Pending for Host' requests. Integration coordinator and Lab Owner perform final smoke testing and change the issue status to 'Resolved', otherwise the issue gets re-opened.

4. CASE STUDY AND RESULTS

To ensure effectiveness of the deployment process and the tools usage, issues in Problem Solving Lab of Virtual Labs was taken up as a case study. The steps involved in this issue fixing process was demonstrated to the lab owners and integration coordinators in a workshop. The workshop was attended by 10 of 12 (2 institutes could not make it) participating institutes from Feb 3rd to Feb 5th, 2016 at IIIT Hyderabad with the following objectives:

1. Understanding the process to be followed for issue resolution while taking the advantage of crowdsourcing.

2. Gather inputs to improve the process and extend the process for all labs.

Post workshop, feedback from the participants on the process and the workshop was gathered. The average feedback[6] score of the workshop is 4.54, 5 being the highest. Following are the feedback questions related to the workshop, the questions were kept simple to get instinctive responses from the participants.

- Q1: *Did the workshop meet the objective as specified before workshop?*
- Q2: *How would you rate the quality of the workshop documents?*
- Q3: *Was the pace of the workshop appropriate for understanding the process?*
- Q4: *What is your feedback on the integration process practiced during the workshop?*
- Q5: *Will you be able to use the process and conduct the workshop in your Institute? If NO, why?*
- Q6: *What improvements do you recommend for the workshop?*
- Q7: *Did you face any roadblocks during the workshop?*
- Q8: *Which task was the easiest to complete during the workshop?*
- Q9: *How was your overall experience of the workshop?*

[6]http://tinyurl.com/VirtualLabsWS

Table 3: Centralized vs Institute QA

Feature	Centralized	Institute
Lab identification	×	✓
Functional Testing	×	✓
System Testing	✓	✓
Issue Logging	✓	✓
Approval for Release	×	✓
Release Notes	✓	✓

Virtual Labs participating institutes reached out to various engineering college students through emails, outreach programs, interaction with college faculty, Vlabs-dev page, etc. for fixing the issues of 14 Labs. In the reach out activity, Virtual Labs usefulness was described and students were requested to volunteer for improving the quality of labs. For increased participation in this crowdsourced work, we believed that extrinsic motivation such as certificate of appreciation from a reputed institute like IIIT Hyderabad would be sufficient for now.

In a span of 3 months, 44 forks have been committed on Github by crowd workers for fixing the issues. Though there weren't tasks that could not be completed, some of the tasks (such as fixes in Problem Solving, Molecular Absorption Spectroscopy, etc.) went through more than one iteration before fix was approved by QA team. The iterations happened for border and negative test cases such as simulatenous double click of elements, reset options, etc. Post fixes, Lab owners and Integration coordinators reviewed and raised request for deploying fixed labs after validation in QA environment. This experience gave us the confidence that issues / enhancements that require less than a day software development effort can effectively leverage crowdsourcing. One Though some of the identified issues are with S1 and S2 severity category, they were not fixed with any prioritized timelines and were fixed with the other S3 issues.

After issues fixing and testing of 14 Labs using crowdsourcing approach, a Centralized and Institute specific QA approach was discussed as depicted in the Table 3. The purpose of this exercise was to identify the need of a professional QA team for final validation before deployment. The following table depicts the responsibilities of the institute and a proposed centralized QA team for each of the QA activities such as Functional Testing, System Testing, Issue Logging, Approval, etc. In this effort, the detailed roles and responsibilities[7] of other stakeholders in quality enhancement process was also identified.

As the usage feedback of labs improved (data available on Virtual Labs site) after fixes, opportunities for automating testing to identify broken links, spelling mistakes, refresh issues, etc. is being scoped. The automation scripts would be integrated into auto-deployment script to validate lab readiness and auto-generate a report to crowd workers and lab owner for failure / deployment based on severity criteria. This would improve the turn around time for testing and make the process scalable. The need and size of Centralized vs Institute specific QA is being re-looked based on automation scripts readiness. Also, a uniform UI 3.0[8] is proposed to remove inconsistencies and other usability issues of labs.

[7]http://vlabs-dev.vlabs.ac.in/labs/integration.htmltext-7

[8]http://vlabs-dev.vlabs.ac.in/labs/uniform-ui.html

5. THREATS TO VALIDITY

Following are some of the threats to establish our approach for all cases in crowdsourcing eLearning systems software development

- Students from engineering colleges contributed as crowd workers, however, the results may vary when other crowd workers contribute.
- In our approach, motivation factors based on the issue type and the crowd worker type was not identified.
- Fixes are QA'ed for identified failure conditions only with smoke testing for basic coverage, rigorous QA on crowdsourced may reveal different results including newly injected issues.
- We did not establish the tipping point of task effort for software development in crowdsourcing.
- We also did not validate possibilities and impact of crowdsourcing for software governed by different license model as we limited our scope to Creative Commons licensing model.
- There are research and industry reports stating that quality software requirements provide quality output. The accepted software quality characteristics [19] such as Completeness, Consistency, Correctness, etc. were not explored for its applicability in crowdsourcing of the current scope.

6. CONCLUSIONS

We establish that crowdsourcing of software development for eLearning systems is possible with multiple stake holders and certification of appreciation would be sufficient for crowd participation. As the quality of the labs improved with the proposed approach, we plan to extend the approach for the remaining 191 labs of Virtual Labs. As the process is streamlined and can be made scalable after automation, the crowd contributions can be extended beyond college students. The crowdsourcing model could also be extended for digitization of the physical feedback documents, enhancement (Flash to Javascript, responsive design) and other development activities of Virtual Labs or other eLearning systems. Using crowdsourcing for quality enhancement of eLearning systems reduces the overall maintenance cost and also provides a learning platform for budding programmers - engineering students in terms of maintaining others software, exposure to processes and software tooling. With improved contributions from crowd workers, the complexity of task can be also increased and the contributor credibility can be used for filtering some of the responses. The labs are being migrated to open edX platform to improve user experience. This migration would also streamline the issue reporting process during feedback collection and also aid involvement of crowd from issue identification stage.

7. REFERENCES

[1] O. Alonso, D. E. Rose, and B. Stewart. Crowdsourcing for relevance evaluation. In *ACM SigIR Forum*, volume 42, pages 9–15. ACM, 2008.

[2] J. Howe. The rise of crowdsourcing. *Wired magazine*, 14(6):1–4, 2006.

[3] Z. Hu and W. Wu. A game theoretic model of software crowdsourcing. In *Service Oriented System*

Engineering (SOSE), 2014 IEEE 8th International Symposium on, pages 446–453. IEEE, 2014.

[4] P. G. Ipeirotis, F. Provost, and J. Wang. Quality management on amazon mechanical turk. In *Proceedings of the ACM SIGKDD workshop on human computation*, pages 64–67. ACM, 2010.

[5] D. Iren and S. Bilgen. Cost models of quality assurance in crowdsourcing. In *Communications and Electronics (ICCE), 2014 IEEE Fifth International Conference on*, pages 504–509. IEEE, 2014.

[6] M. Lease. On quality control and machine learning in crowdsourcing. *Human Computation*, 11(11), 2011.

[7] N. Leicht, D. Durward, I. Blohm, and J. M. Leimeister. Crowdsourcing in software development: A state-of-the-art analysis. 2015.

[8] L. Machado, J. Kroll, R. Prikladnicki, C. R. de Souza, and E. Carmel. Software crowdsourcing challenges in the brazilian it industry.

[9] Y. Maheshwari and Y. R. Reddy. Transformation of flash files to html5 and javascript. In *Proceedings of the ASWEC 2015 24th Australasian Software Engineering Conference*, pages 23–27. ACM, 2015.

[10] K. Mao, L. Capra, M. Harman, and Y. Jia. A survey of the use of crowdsourcing in software engineering. *RN*, 15(01), 2015.

[11] A. J. Mashhadi and L. Capra. Quality control for real-time ubiquitous crowdsourcing. In *Proceedings of the 2nd international workshop on Ubiquitous crowdsouring*, pages 5–8. ACM, 2011.

[12] X. Peng, M. A. Babar, and C. Ebert. Collaborative software development platforms for crowdsourcing. *IEEE software*, 31(2):30–36, 2014.

[13] J. Prpić, J. Melton, A. Taeihagh, and T. Anderson. Moocs and crowdsourcing: Massive courses and massive resources. *Prpić, J., Melton, J., Taeihagh, A., & Anderson*, 2015.

[14] E. S. Raymond. *The Cathedral & the Bazaar: Musings on linux and open source by an accidental revolutionary.* " O'Reilly Media, Inc.", 2001.

[15] W.-T. Tsai, W. Wu, and M. N. Huhns. Cloud-based software crowdsourcing. *IEEE Internet Computing*, 18(3), 2014.

[16] M. E. Whiting, D. Gamage, S. Gaikwad, A. Gilbee, S. Goyal, A. Ballav, D. Majeti, N. Chhibber, A. Richmond-Fuller, F. Vargus, et al. Crowd guilds: Worker-led reputation and feedback on crowdsourcing platforms. *arXiv preprint arXiv:1611.01572*, 2016.

[17] W. Wu, W.-T. Tsai, and W. Li. An evaluation framework for software crowdsourcing. *Frontiers of Computer Science*, 7(5):694–709, 2013.

[18] J. Zhu, B. Shen, and F. Hu. A learning to rank framework for developer recommendation in software crowdsourcing. In *2015 Asia-Pacific Software Engineering Conference (APSEC)*, pages 285–292. IEEE, 2015.

[19] D. Zowghi and V. Gervasi. The three cs of requirements: consistency, completeness, and correctness. In *International Workshop on Requirements Engineering: Foundations for Software Quality, Essen, Germany: Essener Informatik Beitiage*, pages 155–164, 2002.